

---

# **reuse Documentation**

***Release 1.0.0***

**Free Software Foundation Europe**

**May 19, 2022**



# CONTENTS

<b>1</b>	<b>reuse</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Example demo . . . . .	2
1.3	Install . . . . .	2
1.4	Usage . . . . .	3
1.5	Maintainers . . . . .	4
1.6	Contribute . . . . .	5
1.7	License . . . . .	5
<b>2</b>	<b>Usage</b>	<b>7</b>
2.1	Implementation details . . . . .	7
2.2	addheader . . . . .	7
2.3	lint . . . . .	10
<b>3</b>	<b>Helper scripts</b>	<b>15</b>
3.1	Starting point of the codebase . . . . .	15
3.2	Authors . . . . .	16
3.3	Add headers . . . . .	17
<b>4</b>	<b>Credits</b>	<b>19</b>
4.1	Development Lead . . . . .	19
4.2	Contributors . . . . .	19
4.3	Translators . . . . .	20
<b>5</b>	<b>Change log</b>	<b>21</b>
5.1	1.0.0 - 2022-05-19 . . . . .	21
5.2	0.14.0 - 2021-12-27 . . . . .	23
5.3	0.13.0 - 2021-06-11 . . . . .	24
5.4	0.12.1 - 2020-12-17 . . . . .	25
5.5	0.12.0 - 2020-12-16 . . . . .	25
5.6	0.11.1 - 2020-06-08 . . . . .	26
5.7	0.11.0 - 2020-05-25 . . . . .	26
5.8	0.10.1 - 2020-05-14 . . . . .	27
5.9	0.10.0 - 2020-04-24 . . . . .	27
5.10	0.9.0 - 2020-04-21 . . . . .	28
5.11	0.8.1 - 2020-02-22 . . . . .	28
5.12	0.8.0 - 2020-01-20 . . . . .	29
5.13	0.7.0 - 2019-11-28 . . . . .	30
5.14	0.6.0 - 2019-11-19 . . . . .	30
5.15	0.5.2 - 2019-10-27 . . . . .	31

5.16	0.5.1 - 2019-10-24 [YANKED]	31
5.17	0.5.0 - 2019-08-29	31
5.18	0.4.1 - 2019-08-07	32
5.19	0.4.0 - 2019-08-07	32
5.20	0.3.4 - 2019-04-15	33
5.21	0.3.3 - 2018-07-15	34
5.22	0.3.2 - 2018-07-15	34
5.23	0.3.1 - 2018-07-14	34
5.24	0.3.0 - 2018-05-16	34
5.25	0.2.0 - 2018-04-17	34
5.26	0.1.1 - 2017-12-14	35
5.27	0.1.0 - 2017-12-14	35
5.28	0.0.4 - 2017-11-06	36
5.29	0.0.3 - 2017-11-06	36
5.30	0.0.2 - 2017-11-03	36
<b>6</b>	<b>reuse</b>	<b>39</b>
6.1	reuse package	39
<b>7</b>	<b>Indices and tables</b>	<b>47</b>
	<b>Python Module Index</b>	<b>49</b>
	<b>Index</b>	<b>51</b>

reuse is a tool for compliance with the REUSE recommendations.

- Documentation: <https://reuse.readthedocs.io> and <https://reuse.software>
- Source code: <https://github.com/fsfe/reuse-tool>
- PyPI: <https://pypi.python.org/pypi/reuse>
- REUSE: 3.0
- Python: 3.6+

## 1.1 Background

Copyright and licensing is difficult, especially when reusing software from different projects that are released under various different licenses. REUSE was started by the Free Software Foundation Europe (FSFE) to provide a set of recommendations to make licensing your Free Software projects easier. Not only do these recommendations make it easier for you to declare the licenses under which your works are released, but they also make it easier for a computer to understand how your project is licensed.

As a short summary, the recommendations are threefold:

1. Choose and provide licenses
2. Add copyright and licensing information to each file
3. Confirm REUSE compliance

You are recommended to read [our tutorial](#) for a step-by-step guide through these three steps. The [FAQ](#) covers basic questions about licensing, copyright, and more complex use cases. Advanced users and integrators will find the [full specification](#) helpful.

This tool exists to facilitate the developer in complying with the above recommendations.

There are [other tools](#) that have a lot more features and functionality surrounding the analysis and inspection of copyright and licenses in software projects. The REUSE helper tool, on the other hand, is solely designed to be a simple tool to assist in compliance with the REUSE recommendations.

## 1.2 Example demo

In this screencast, we are going to follow the [tutorial](#), making the [REUSE example repository](#) compliant.

## 1.3 Install

### 1.3.1 Installation via package managers (Recommended)

There are packages available for easy install on some operating systems. You are welcome to help us package this tool for more distributions!

- Arch Linux: [reuse](#)
- Debian: [reuse](#)
- GNU Guix: [reuse](#)
- Fedora: [reuse](#)
- NixOS: [reuse](#)
- openSUSE: [reuse](#)
- VoidLinux: [reuse](#)

An automatically generated list can be found at [repology.org](https://repology.org), without any guarantee for completeness.

### 1.3.2 Install and run via pipx (Recommended)

The following one-liner both installs and runs this tool from [PyPI](#) via [pipx](#):

```
pipx run reuse lint
```

[pipx](#) automatically isolates [reuse](#) into its own Python virtualenv, which means that it won't interfere with other Python packages, and other Python packages won't interfere with it.

If you want to be able to use [reuse](#) without prepending it with `pipx run` every time, install it globally like so:

```
pipx install reuse
```

[reuse](#) will then be available in `~/.local/bin`, which must be added to your `$PATH`.

### 1.3.3 Installation via pip

To install [reuse](#), you need to have the following pieces of software on your computer:

- Python 3.6+
- `pip`

You then only need to run the following command:

```
pip3 install --user reuse
```

After this, make sure that `~/.local/bin` is in your `$PATH`. On Windows, the required path for your environment may look like `%USERPROFILE%\AppData\Roaming\Python\Python39\Scripts`, depending on the Python version you have installed.

To update reuse, run this command:

```
pip3 install --user --upgrade reuse
```

For full functionality, the following pieces of software are recommended:

- Git
- Mercurial 4.3+

### 1.3.4 Installation from source

You can also install this tool from the source code, but we recommend the methods above for easier and more stable updates. Please make sure the requirements for the installation via pip are present on your machine.

```
python3 setup.py install
```

## 1.4 Usage

First, read the [REUSE tutorial](#). In a nutshell:

1. Put your licenses in the `LICENSES/` directory.
2. Add a comment header to each file that says `SPDX-License-Identifier: GPL-3.0-or-later`, and `SPDX-FileCopyrightText: $YEAR $NAME`. You can be flexible with the format, just make sure that the line starts with `SPDX-FileCopyrightText:`.
3. Verify your work using this tool.

Example of header:

```
# SPDX-FileCopyrightText: 2017 Free Software Foundation Europe e.V. <https://fsfe.org>
#
# SPDX-License-Identifier: CC-BY-SA-4.0
```

To check against the recommendations, use `reuse lint`:

```
~/Projects/reuse-tool $ reuse lint
[...]

Congratulations! Your project is compliant with version 3.0 of the REUSE Specification :-
↪)
```

This tool can do various more things, detailed in the documentation. Here a short summary:

- `addheader` — Add copyright and/or licensing information to the header of a file.
- `download` — Download the specified license into the `LICENSES/` directory.
- `init` — Set up the project for REUSE compliance.
- `lint` — Verify the project for REUSE compliance.
- `spdx` — Generate an SPDX Document of all files in the project.

- `supported-licenses` — Prints all licenses supported by REUSE.

### 1.4.1 Run in Docker

The `fsfe/reuse` Docker image is available on [Docker Hub](#). With it, you can easily include REUSE in CI/CD processes. This way, you can check for REUSE compliance for each build. In our [resources for developers](#) you can learn how to integrate the REUSE tool in Drone, Travis, GitHub, or GitLab CI.

You can run the helper tool simply by providing the command you want to run (e.g., `lint`, `spdx`). The image's working directory is `/data` by default. So if you want to lint a project that is in your current working directory, you can mount it on the container's `/data` directory, and tell the tool to lint. That looks a little like this:

```
docker run --rm --volume $(pwd):/data fsfe/reuse lint
```

You can also provide additional arguments, like so:

```
docker run --rm --volume $(pwd):/data fsfe/reuse --include-submodules spdx -o out.spdx
```

There are a number of tags available:

- `latest` is the most recent stable release.
- `dev` follows the `master` branch of this repository. Up-to-date, but potentially unstable.
- `latest-extra` has a few extra packages installed, currently `openssh-client`.
- `latest-debian` is based on `python:slim`. It is larger, but may be better suited for license compliance.

### 1.4.2 Run as pre-commit hook

You can automatically run `reuse lint` on every commit as a pre-commit hook for Git. This uses `pre-commit`. Once you [have it installed](#), add this to the `.pre-commit-config.yaml` in your repository:

```
repos:
- repo: https://github.com/fsfe/reuse-tool
  rev: latest
  hooks:
  - id: reuse
```

Then run `pre-commit install`. Now, every time you commit, `reuse lint` is run in the background, and will prevent your commit from going through if there was an error.

## 1.5 Maintainers

- Carmen Bianca Bakker - [carmenbianca@fsfe.org](mailto:carmenbianca@fsfe.org)
- Max Mehl - [max.mehl@fsfe.org](mailto:max.mehl@fsfe.org)



## 1.6 Contribute

Any pull requests or suggestions are welcome at <https://github.com/fsfe/reuse-tool> or via e-mail to one of the maintainers. General inquiries can be sent to [reuse@lists.fsfe.org](mailto:reuse@lists.fsfe.org).

Interaction within this project is covered by the [FSFE's Code of Conduct](#).

Starting local development is very simple, just execute the following commands:

```
git clone git@github.com:fsfe/reuse-tool.git
cd reuse-tool/
python3 -mvenv venv
source venv/bin/activate
make develop
```

You need to run `make develop` at least once to set up the virtualenv.

Next, run `make help` to see the available interactions.

## 1.7 License

This work is licensed under multiple licences. Because keeping this section up-to-date is challenging, here is a brief summary as of April 2020:

- All original source code is licensed under GPL-3.0-or-later.
- All documentation is licensed under CC-BY-SA-4.0.
- Some configuration and data files are licensed under CC0-1.0.
- Some code borrowed from [spdx/tool-python](#) is licensed under Apache-2.0.

For more accurate information, check the individual files.



The *overview* documents some basic usage on how to use this tool. It is highly recommended to read the overview first, and you might not even need to read this chapter. This chapter covers details that might not be immediately obvious when using the tool. This chapter does not cover *everything*, assuming that the user is helped enough by `reuse --help` and `reuse <subcommand> --help`.

## 2.1 Implementation details

This section covers implementation details that are true for the entire tool.

When searching for copyright and licensing tags inside of files, the tool does not strictly limit itself to the header comment as prescribed by the specification. It searches the first 4 kibibytes of the file. This makes sure that the tool can parse any type of plain-text file, even if the comment style is not recognised.

If a file is found to have an unparsable tag, that file is not parsed at all. This is [a bug](#).

The tool does not verify the correctness of copyright notices. If it finds any line containing ‘©’, ‘Copyright’, or ‘SPDX-FileCopyrightText:’, then the tag and everything following it is considered a valid copyright notice, even if the copyright notice is not compliant with the specification.

When running the tool, the root of the project is automatically found if the working directory is inside a VCS repository. Otherwise, it treats the working directory as the root of the project. You can override the root of the project with the `--root` optional argument.

Git submodules are automatically ignored unless `--include-submodules` is passed as optional argument.

Symbolically links and files that are zero-sized are automatically ignored.

## 2.2 addheader

`addheader` makes it possible to semi-automatically add copyright and licensing information into the header of a file. This is useful especially in scenarios where you want to add a copyright holder or license to a lot of files without having to manually edit the header of each file.

**Warning:** You should be cautious with using `addheader` in automated processes. While nothing is stopping you from using it in your release script, you should make sure that the information it adds is actually reflective of reality. This is best verified manually.

The basic usage is `reuse addheader --copyright="Jane Doe" --license=MIT my_file.py`. This will add the following header to the file (assuming that the current year is 2019):

```
# SPDX-FileCopyrightText: 2019 Jane Doe
#
# SPDX-License-Identifier: MIT
```

You can use as many `--copyright` and `--license` arguments, so long as there is at least one such argument.

The REUSE header is placed at the very top of the file. If a different REUSE header already existed—at the top or elsewhere—its tags are copied, and the header is replaced in-place.

With the argument `--copyright-style` it is possible to change the default `SPDX-FileCopyrightText` to one of the following style:

<code>spdx:</code>	<code>SPDX-FileCopyrightText: &lt;year&gt; &lt;statement&gt;</code>
<code>spdx-symbol:</code>	<code>SPDX-FileCopyrightText: © &lt;year&gt; &lt;statement&gt;</code>
<code>string:</code>	<code>Copyright &lt;year&gt; &lt;statement&gt;</code>
<code>string-c:</code>	<code>Copyright (C) &lt;year&gt; &lt;statement&gt;</code>
<code>string-symbol:</code>	<code>Copyright © &lt;year&gt; &lt;statement&gt;</code>
<code>symbol:</code>	<code>© &lt;year&gt; &lt;statement&gt;</code>

Shebangs are always preserved at the top of the file.

## 2.2.1 Merging Statements

When the tool parses copyright headers, *reuse* can be configured to automatically merge copyright lines based on the statement element. This effectively transforms multiple lines with a single year into a single line with a range.

Starting with the following header,

```
# SPDX-FileCopyrightText: 2016 Jane Doe
# SPDX-FileCopyrightText: 2018 John Doe
#
# SPDX-License-Identifier: GPL-2.0
```

The standard tool options would produce the following

```
$ reuse addheader --year 2018 --license GPL-2.0 --copyright="Jane Doe" file.py
```

```
# SPDX-FileCopyrightText: 2016 Jane Doe
# SPDX-FileCopyrightText: 2018 John Doe
# SPDX-FileCopyrightText: 2018 Jane Doe
#
# SPDX-License-Identifier: GPL-2.0
```

Running the same command with the `--merge-copyrights` option will instead produce the following

```
# SPDX-FileCopyrightText: 2016 - 2018 Jane Doe
# SPDX-FileCopyrightText: 2018 John Doe
#
# SPDX-License-Identifier: GPL-2.0
```

### 2.2.2 Comment styles

The tool normally tries to auto-detect the comment style to use from the file extension of a file, and use that comment style. If the tool is unable to detect the comment style, or if it detects the wrong style, you can override the style using `--style`. The supported styles are:

- AppleScript
- ASPX
- BibTex
- C
- CSS
- Haskell
- HTML
- Jinja
- JSX
- ML
- Python
- TeX

If your comment style is not supported or a file extension is not correctly detected, please [open an issue](#).

Normally, the tool uses a single-line comment style when it is available (e.g., `//` is used instead of `/* */` for C comment styles). If no single-line comment style is available, a multi-line style is used. You can force a certain comment style using `-single-line` or `-multi-line`.

### 2.2.3 Templates

When the tool adds a header to a file, it normally first lists all copyright statements alphabetically, adds a single empty line, and then lists all SPDX License Expressions alphabetically. That is all that the header contains. It is possible to change this behaviour, and use a custom type of header that contains extra text. This is done through Jinja2 templates.

The default template is:

```
{% for copyright_line in copyright_lines %}
{{ copyright_line }}
{% endfor %}

{% for expression in spdx_expressions %}
SPDX-License-Identifier: {{ expression }}
{% endfor %}
```

Templates are automatically commented by the tool, depending on the detected or specified comment style.

You can create your own Jinja2 templates and place them in `.reuse/templates/`. If you create the template `mytemplate.jinja2`, you can use it with `reuse addheader --copyright="Jane Doe" --template=mytemplate foo.py`.

Inside of the template, you have access to the following variables:

- `copyright_lines` — a list of copyright notices (string).
- `spdx_expressions` — a list of SPDX License Expressions (string).

In the future, more variables will be added.

In some cases, you might want to do custom comment formatting. In those cases, you can pre-format your header as a comment. When doing so, suffix your template with `.commented.jinja2`.

An example of a custom template with manual commenting is:

```
/*
{% for copyright_line in copyright_lines %}
* {{ copyright_line }}
{% endfor %}
{% if copyright_lines and spdx_expressions %}
*
{% endif %}
{% for expression in spdx_expressions %}
* SPDX-License-Identifier: {{ expression }}
{% endfor %}
{% if "GPL-3.0-or-later" in spdx_expressions %}
*
* This program is free software: you can redistribute it and/or modify it under
* the terms of the GNU General Public License as published by the Free Software
* Foundation, either version 3 of the License, or (at your option) any later
* version.
*
* This program is distributed in the hope that it will be useful, but WITHOUT
* ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
* FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License along with
* this program. If not, see <https://www.gnu.org/licenses/>.
{% endif %}
*/
```

## 2.3 lint

`lint` is the main component of the tool. Summarily, it verifies whether the project is compliant with [the REUSE Specification](#). Its main goal is to find all files that do not have copyright and licensing information in their headers, but it also checks a few other things.

The STDOUT output of `reuse lint` is valid Markdown. Occasionally some logging will be printed to STDERR, which is not valid Markdown.

You may prevent STDOUT output by using the `-quiet` option. This can be useful if you are only interested in the exit code.

This is some example output of `reuse lint`:

```
# BAD LICENSES

'bad-license' found in:
* LICENSES/bad-license.txt

# UNUSED LICENSES
```

(continues on next page)

(continued from previous page)

```

The following licenses are not used:
* bad-license

# MISSING COPYRIGHT AND LICENSING INFORMATION

The following files have no copyright and licensing information:
* no-information.txt

# SUMMARY

* Bad licenses: bad-license
* Deprecated licenses:
* Licenses without file extension:
* Missing licenses:
* Unused licenses: bad-license
* Used licenses: Apache-2.0, CC-BY-SA-4.0, CC0-1.0, GPL-3.0-or-later
* Read errors: 0
* Files with copyright information: 57 / 58
* Files with license information: 57 / 58

Unfortunately, your project is not compliant with version 3.0 of the REUSE Specification.
↪ :- (

```

### 2.3.1 Criteria

These are the criteria that the linter checks against:

#### Bad licenses

Licenses that are found in LICENSES/ that are not found in the SPDX License List or do not start with LicenseRef- are bad licenses.

#### Deprecated licenses

If a license has an SPDX License Identifier that has been deprecated by SPDX, the license will show up here.

#### Licenses without file extension

These are licenses whose file names are a valid SPDX License Identifier, but which do not have a file extension.

## Missing licenses

If a license is referred to in a comment header, but the license is not found in the LICENSES/ directory, then that license is missing.

## Unused licenses

Conversely, if a license is found in the LICENSES/ directory but is not referred to in any comment header, then that license is unused.

## Read errors

Not technically a criterion, but files that cannot be read by the operating system are read errors, and need to be fixed.

## Files with copyright and license information

Every file needs to have copyright and licensing information associated with it. The REUSE Specification details several ways of doing it. By and large, these are the methods:

- Placing tags in the header of the file.
- Placing tags in a .license file adjacent to the file.
- Putting the information in the DEP5 file.

If a file is found that does not have copyright and/or license information associated with it, then the project is not compliant.

### 2.3.2 Ignoring parts of a file

You can easily ignore parts of a file that will always cause problems for `reuse lint`. Suppose you have the following bash script:

```
#!/usr/bin/env bash
# SPDX-FileCopyrightText: 2021 John Doe
#
# SPDX-License-Identifier: CC0-1.0

echo "SPDX-FileCopyrightText: $(date +%Y) Jane Doe" > file.txt
echo "SPDX-License-Identifier: MIT" > file.txt

exit 0
```

This will lead to the following error message despite the file having the correct licensing info in the header:

```
$ reuse lint
reuse._util - ERROR - Could not parse 'MIT' > file.txt'
reuse.project - ERROR - 'foobar.sh' holds an SPDX expression that cannot be parsed,
↳ skipping the file
# MISSING COPYRIGHT AND LICENSING INFORMATION
```

The following files have no copyright and licensing information:

(continues on next page)



(continued from previous page)

```
* foobar.sh  
[...]
```

To avoid this error message, you can simply amend the file as follows:

```
#!/usr/bin/env bash  
# SPDX-FileCopyrightText: 2021 John Doe  
#  
# SPDX-License-Identifier: CC0-1.0  
  
# REUSE-IgnoreStart  
echo "SPDX-FileCopyrightText: $(date +%Y) Jane Doe" > file.txt  
echo "SPDX-License-Identifier: MIT" > file.txt  
# REUSE-IgnoreEnd  
  
exit 0
```

Now, *reuse lint* will not report any problems with this file anymore.



## HELPER SCRIPTS

This section contains scripts and snippets to help with the usage of REUSE. REUSE already has a lot of functionality builtin, but there are still cases where REUSE could use support from some external scripts. A typical example is adding SPDX headers based on the information in the version control system. This collection of scripts and snippets offers help for such situations.

**Warning:** These scripts help you run REUSE against your codebase. Automatically extracted information might not resemble the truth. The correctness of these scripts is not guaranteed. Use with caution and at your own risk.

### 3.1 Starting point of the codebase

The first code contribution can be a worthwhile date to include in the copyright annotation.

#### 3.1.1 First commit

Git log can show summaries of all commits. At the moment of writing git does not allow selecting just one commit, so `head` is used to restrict the output.

```
$ git log --reverse --all | head -n 3
commit cdcea0887e0a85149f93e734b647301a16dd893e
Author: Carmen Bianca Bakker <carmenbianca@fsfe.org>
Date:   Tue Oct 10 18:27:11 2017 +0200
```

#### 3.1.2 Year of first commit

With a custom format just the year of the first commit can be displayed. This output is convenient for use in larger scripts.

```
$ git log --reverse --date="format:%Y" --format="format:%cd" | head -n 1
2017
```

## 3.2 Authors

Unless the authors have signed away their copyright to a company or the project, the authors are also the copyright holders of their contributions. So in a lot of cases it is valuable to know the original authors in order to explicitly state these copyright holders.

### 3.2.1 Commit authors

Some examples on how to get information about the authors from the codebase.

#### Based on commit order

Print out the authors known to git in chronological order. Awk is used to filter out subsequent duplicate entries so each entry is shown only for the first appearance.

```
$ git log --reverse --all --format="%aN <%aE>" | awk '!seen[$0]++'
Carmen Bianca Bakker <carmenbianca@fsfe.org>
carmenbianca <carmenbianca@fsfe.org>
Carmen Bianca Bakker <carmen@carmenbianca.eu>
...
```

The same command, but now without the email addresses.

```
$ git log --reverse --all --format="%aN" | awk '!seen[$0]++'
Carmen Bianca Bakker
carmenbianca
Sebastian Schuberth
...
```

#### Sorted by name

All authors as known to the version control system, simply sorted by name.

```
$ git log --all --format="%aN <%aE>" | sort | uniq
Adam Spiers <reuse-tool@adamspiers.org>
Ajinkya Patil <ajinkyarangnathpatil@gmail.com>
Alvar <8402811+oxzi@users.noreply.github.com>
...
```

The same command, but now without the email addresses.

```
$ git log --all --format="%aN" | sort | uniq
Adam Spiers
Ajinkya Patil
Alvar
...
```

### 3.2.2 Authors in commit trailers like sign-off

A sign-off annotation in a commit also contains author details that can be as valuable.

```
$ git log --all | grep -i 'Signed-off-by\|Co-authored-by' | sort | uniq
Co-authored-by: Ethel Morgan <eth@ethulhu.co.uk>
Co-authored-by: max.mehl <max.mehl@fsfe.org>
Co-authored-by: Max Mehl <max.mehl@fsfe.org>
Signed-off-by: Carmen Bianca Bakker <carmenbianca.bakker@liferay.com>
...
```

## 3.3 Add headers

A common use-case is to add headers to existing, modified or newly written code.

### 3.3.1 Add headers to staged files based on git settings

This script helps you add your copyright headers right before committing the code you wrote.

The list of files staged in git can be retrieved using `git diff --name-only --cached`, which is the basis to apply the `reuse addheader` command to.

Git user and email address are available through `git config --get user.name` and `git config --get user.email`.

REUSE already sets the current year, so there is no need to set that explicitly.

These elements can be combined into a single command:

```
$ git diff --name-only --cached | xargs reuse addheader -c "$(git config --get user.
↪name) <$(git config --get user.email)>"
```

### Copyright

This page is licensed under the [Creative Commons Attribution-ShareAlike 4.0 International license](#). Examples, recipes, and other code in the documentation are additionally licensed under the [Creative Commons Zero v1.0 Universal License](#).



**CREDITS**

## 4.1 Development Lead

- Carmen Bianca Bakker <carmenbianca@fsfe.org>

## 4.2 Contributors

- Sebastian Schuberth <schuberth@fsfe.org>
- Max Mehl <max.mehl@fsfe.org>
- Matija Šuklje <matija@suklje.name>
- Greg Kroah-Hartman
- Basil Peace
- Keith Maxwell
- Stefan Bakker <s.bakker777@gmail.com>
- Kirill Elagin <kirelagin@gmail.com>
- John Mulligan <jmulligan@redhat.com>
- Tuomas Siipola <tuomas@zpl.fi>
- Diego Elio Pettenò <flameeyes@flameeyes.com>
- David Alfonso
- Maximilian Dolling <mdolling@gfz-potsdam.de>
- yoctocell
- Tuomas Siipola
- Gri-ffin <gameshunter359@gmail.com>
- Nico Rikken <nico.rikken@fsfe.org>
- Florian Snow <florian@familysnow.net>

## 4.3 Translators

- Dutch:
  - André Ockers <ao@fsfe.org>
  - Carmen Bianca Bakker <carmenbianca@fsfe.org>
- French:
  - OliBug <act-zebug@riseup.net>
  - Vincent Lequertier <vincent@fsfe.org>
- Galician:
  - pd <euklade@gmail.com>
- German:
  - Max Mehl <max.mehl@fsfe.org>
  - Thomas Doczkal <doczkal@fsfe.org>
- Esperanto:
  - Carmen Bianca Bakker <carmenbianca@fsfe.org>
  - Tirifto <tirifto@posteo.cz>
- Italian:
  - Luca Bonissi <lucabon@fsfe.org>
- Portuguese:
  - José Vieira <jvieira33@sapo.pt>
- Spanish:
  - flow <adolflow@sindominio.net>
  - pd <euklade@gmail.com>
  - Roberto Bauglir <bauglir@fsfe.org>
- Turkish:
  - T. E. Kalayci <tekrei@member.fsf.org>



## CHANGE LOG

This change log follows the [Keep a Changelog](#) spec. Every release contains the following sections:

- Added for new features.
- Changed for changes in existing functionality.
- Deprecated for soon-to-be removed features.
- Removed for now removed features.
- Fixed for any bug fixes.
- Security in case of vulnerabilities.

The versions follow [semantic versioning](#).

### 5.1 1.0.0 - 2022-05-19

A major release! Do not worry, no breaking changes but a development team (@carmenbianca, @floriansnow, @linozen, @mxmehl and @nicorikken) that is confident enough to declare the REUSE helper tool stable, and a bunch of long-awaited features!

Apart from smaller changes under the hood and typical maintenance tasks, the main additions are new flags to the `addheader` subcommand that ease recursive and automatic operations, the ability to ignore areas of a file that contain strings that may falsely be detected as copyright or license statements, and the option to merge copyright lines. The tool now also has better handling of some edge cases with copyright and license identifiers.

We would like to thank the many contributors to this release, among them @ajinkyapatil8190, @aspier, @ferdnyc, @Gri-ffin, @hexagonrecursion, @hoijui, @Jakelyst, @Liambeguin, @rex4539, @robinkrahl, @rpavlik, @siiptuo, @thbde and @ventosus.

#### 5.1.1 Added

- Extend [tool documentation](#) with scripts to help using this tool and automating some steps that are not built into the tool itself. (#500)
- Recommendations for installation/run methods: package managers and pipx (#457)
- Docker images for AArch64 (#478)
- Added the ability to ignore parts of a file when running `reuse lint`. Simply add `REUSE-IgnoreStart` and `REUSE-IgnoreEnd` as comments and all lines between the two will be ignored by the next run of `reuse lint`. (#463)
- [Meson subprojects](#) are now ignored by default. (#496)

- More file types are recognised:
  - sbt build files (`.sbt`)
  - Vimscript files (`.vim`)
- Added `--skip-existing` flag to `addheader` in order to skip files that already contain SPDX information. This may be useful for only adding SPDX information to newly created files. (#480)
- Added `--recursive` flag to `addheader`. (#469)
- Preserve shebang for more script files:
  - V-Lang (#432)
- Ignore all SPDX files with their typical formats and extensions. (#494)
- Add support for merging copyright lines based on copyright statement, transforming multiple lines with a single year into a single line with a range. (#328)

### 5.1.2 Changed

- Use `setuptools` instead of the deprecated `distutils` which will be removed with Python 3.12. (#451)
- `addheader --explicit-license` renamed to `--force-dot-license`. (#476)
- Dockerfiles for reuse-tool are now in a separate subdirectory `docker`. (#499)
- Updated SPDX license list to 3.17. (#513)
- The copyright detection mechanism now silently accepts the following strings: `Copyright(c)` and `Copyright(C)`. (#440)

### 5.1.3 Deprecated

- Deprecated `--explicit-license` in favour of `--force-dot-license`. `--explicit-license` will remain useable (although undocumented) for the foreseeable future. (#476)

### 5.1.4 Removed

- `JsxCommentStyle` in favor of using `CCommentStyle` directly (see section Fixed). (#406)

### 5.1.5 Fixed

- Better support for unary “+” operator in license identifiers. For example, if `Apache-1.0+` appears as a declared license, it should not be identified as missing, bad, or unused if `LICENSES/Apache-1.0.txt` exists. It is, however, identified separately as a used license. (#123)
- When `addheader` creates a `.license` file, that file now has a newline at the end. (#477)
- Cleaned up internal string manipulation. (#477)
- JSX (`.jsx` and `.tsx`) actually uses C comment syntax as JSX blocks never stand at the beginning of the file where the licensing info needs to go. (#406)

## 5.2 0.14.0 - 2021-12-27

Happy holidays! This is mainly a maintenance release fixing some subcommands and adding loads of supported file types and file names. However, you can also enjoy the `supported-licenses` subcommand and the `--quiet` flag for linting as well as better suggestions for license identifiers. Thanks to everyone who contributed!

### 5.2.1 Added

- `supported-licenses` command that lists all licenses supported by REUSE (#401)
- `--quiet` switch to the `lint` command (#402)
- Better suggestions for faulty SPDX license identifiers in `download` and `init` (#416)
- Python 3.10 support declared
- More file types are recognised:
  - Apache FreeMarker Template Language (`.ftl`)
  - AsciiDoc (`.adoc`, `.asc`, `.asciidoc`)
  - Bibliography (`.csl`)
  - C++ (`.cc` and `.hh`)
  - GraphQL (`.graphql`)
  - Handlebars (`.hbs`)
  - Markdown-linter config (`.mdlrc`)
  - MS Office (`.doc`, `.xls`, `.pptx` and many more)
  - Nimble (`.nim.cfg`, `.nimble`)
  - Open Document Format (`.odt`, `.ods`, `.fodp` and many more)
  - Perl plain old documentation (`.pod`)
  - Portable document format (`.pdf`)
  - Protobuf files (`.proto`)
  - Soy templates (`.soy`)
  - SuperCollider (`.sc`, `.scsyndef`)
  - Turtle/RDF (`.ttl`)
  - V-Lang (`.v`, `.vsh`)
  - Vue.js (`.vue`)
- More file names are recognised:
  - Doxygen (`Doxyfile`)
  - ESLint (`.eslintignore` and `.eslintrc`)
  - Meson options file (`meson_options.txt`)
  - NPM ignore (`.npmignore`)
  - Podman container files (`Containerfile`)
  - SuperCollider (`archive.sctxar`)

- Yarn package manager (`.yarn.lock` and `.yarnrc`)

## 5.2.2 Changed

- Updated SPDX license list to 3.15

## 5.2.3 Fixed

- Fix Extensible Stylesheet Language (`.xsl`) to use HTML comment syntax
- Allow creating `.license` file for write-protected files (#347) (#418)
- Do not break XML files special first line (#378)
- Make `download` subcommand work correctly outside of project root and with `--root` (#430)

## 5.3 0.13.0 - 2021-06-11

### 5.3.1 Added

- `addheader` recognises file types that specifically require `.license` files instead of headers using `UncommentableCommentStyle`. (#189)
- `.hgtags` is ignored. (#227)
- `spdx-symbol` added to possible copyright styles. (#350)
- `addheader` ignores case when matching file extensions and names. (#359)
- Provide `latest-debian` as Docker Hub tag, created by `Dockerfile-debian`. (#321)
- More file types are recognised:
  - Javascript modules (`.mjs`)
  - Jupyter Notebook (`.ipynb`)
  - Scalable Vector Graphics (`.svg`)
  - JSON (`.json`)
  - Comma-separated values (`.csv`)
  - Racket (`.rkt`)
  - Org-mode (`.org`)
  - LaTeX package files (`.sty`)
  - `devicetree` (`.dts`, `.dtsi`)
  - Bitbake (`.bb`, `.bbappend`, `.bbclass`)
  - XML schemas (`.xsd`)
  - OpenSCAD (`.scad`)
- More file names are recognised:
  - Bash configuration (`.bashrc`)
  - `Coverage.py` (`.coveragerc`)

- Jenkins (`Jenkinsfile`)
- SonarScanner (`sonar-project.properties`)
- Gradle (`gradle-wrapper.properties`, `gradlew`)

### 5.3.2 Changed

- Bump alpine Docker base image to 3.13. (#369)

### 5.3.3 Fixed

- Fixed a regression where unused licenses were not at all detected. (#285)
- Declared dependency on `python-debian` != 0.1.39 on Windows. This version does not import on Windows. (#310)
- `MANIFEST.in` is now recognised instead of the incorrect `Manifest.in` by `addheader`. (#306)
- `addheader` now checks whether a file is both readable and writeable instead of only writeable. (#241)
- `addheader` now preserves line endings. (#308)
- `download` does no longer fail when both `--output` and `--all` are used. (#326)
- Catch erroneous SPDX expressions. (#331)
- Updated SPDX license list to 3.13.

## 5.4 0.12.1 - 2020-12-17

### 5.4.1 Fixed

- Bumped versions of requirements. (#288)

## 5.5 0.12.0 - 2020-12-16

This release was delayed due to the absence of the lead developer (Carmen, me, the person writing these release notes). Many thanks to Max Mehl for coordinating the project in my absence. Many thanks also to the contributors who sent in pull requests, in reverse chronological order: Olaf Meeuwissen, Mikko Piuola, Wolfgang Traylor, Paul Spooren, Robert Cohn, ethulhu, pukkamustard, and Diego Elio Pettenò.

### 5.5.1 Added

- Separate Docker image with additional executables installed (`fsfe/reuse:latest-extra`) (#238)
- Allow different styles of copyright lines: SPDX (default), String, String (C), String ©, and © (#248)
- Convenience function to update resources (SPDX license list and exceptions) (#268)
- More file types are recognised:
  - ClojureScript (`.cljc`, `.cljs`)
  - Fortran (`.F`, `.F90`, `.f90`, `.f95`, `.f03`, `.f`, `.for`)

- Makefile (.mk)
- PlantUML (.iuml, .plantuml, .pu, .puml)
- R (.R, .Renvirom, .Rprofile)
- ReStructured Text (.rst)
- RMarkdown (.Rmd)
- Scheme (.scm)
- TypeScript (.ts)
- TypeScript JSX (.tsx)
- Windows Batch (.bat)
- More file names are recognised:
  - .dockerignore
  - Gemfile
  - go.mod
  - meson.build
  - Rakefile

### 5.5.2 Changed

- Use UTF-8 explicitly when reading files (#242)

### 5.5.3 Fixed

- Updated license list to 3.11.

## 5.6 0.11.1 - 2020-06-08

### 5.6.1 Fixed

- Similar to CAL-1.0 and CAL-1.0-Combined-Work-Exception, SHL-2.1 is now ignored because it contains an SPDX tag within itself.

## 5.7 0.11.0 - 2020-05-25

### 5.7.1 Added

- Added `--skip-unrecognised` flag to `addheader` in order to skip files with unrecognised comment styles instead of aborting without processing any file.

### 5.7.2 Changed

- Always write the output files encoded in UTF-8, explicitly. This is already the default on most Unix systems, but it was not on Windows.
- All symlinks and 0-sized files in projects are now ignored.

### 5.7.3 Fixed

- The licenses CAL-1.0 and CAL-1.0-Combined-Work-Exception contain an SPDX tag within themselves. Files that are named after these licenses are now ignored.
- Fixed a bug where `addheader` wouldn't properly apply the template on `.license` files if the `.license` file was non-empty, but did not contain valid SPDX tags.

## 5.8 0.10.1 - 2020-05-14

### 5.8.1 Fixed

- Updated license list to 3.8-106-g4cfec76.

## 5.9 0.10.0 - 2020-04-24

### 5.9.1 Added

- Add support for `autoconf` comment style (listed as `m4`).
- More file types are recognised:
  - Cython (`.pyx`, `.pxd`)
  - Sass and SCSS (`.sass`, `.scss`)
  - XSL (`.xsl`)
  - Mailmap (`.mailmap`)
- Added `--single-line` and `--multi-line` flags to `addheader`. These flags force a certain comment style.

### 5.9.2 Changed

- The Docker image has an `entrypoint` now. In effect, this means running:  

```
docker run -v $(pwd):/data fsfe/reuse lint
```

  
instead of  

```
docker run -v $(pwd):/data fsfe/reuse reuse lint.
```

## 5.10 0.9.0 - 2020-04-21

### 5.10.1 Added

- Added support for Mercurial 4.3+.
- A pre-commit hook has been added.
- When an incorrect SPDX identifier is forwarded to `download` or `init`, the tool now suggests what you might have meant.

### 5.10.2 Changed

- Under the hood, a lot of code that has to do with Git and Mercurial was moved into its own module.
- The Docker image has been changed such that it now automatically runs `reuse lint` on the `/data` directory unless something else is specified by the user.

### 5.10.3 Fixed

- Fixed a bug with `addheader --explicit-license` that would result in `file.license.license` if `file.license` already existed.
- Fixed a Windows-only bug to do with calling subprocesses.
- Fixed a rare bug that would trigger when a directory is both ignored and contains a `.git` file.

## 5.11 0.8.1 - 2020-02-22

### 5.11.1 Added

- Support Jinja (Jinja2) comment style.
- Support all multi-line comment endings when parsing for SPDX information.

### 5.11.2 Fixed

- Improvements to German translation by Thomas Doczkal.
- No longer remove newlines at the end of files when using `addheader`.
- There can now be a tab as whitespace after `SPDX-License-Identifier` and `SPDX-FileCopyrightText`.



## 5.12 0.8.0 - 2020-01-20

### 5.12.1 Added

- Implemented `--root` argument to specify the root of the project without heuristics.
- The linter will complain about licenses without file extensions.
- Deprecated licenses are now recognised. `lint` will complain about deprecated licenses.
- ProjectReport generation (`lint`, `spdx`) now uses Python multiprocessing, more commonly called multi-threading outside of Python. This has a significant speedup of approximately 300% in testing. Because of overhead, performance increase is not exactly linear.
- For setups where multiprocessing is unsupported or unwanted, `--no-multiprocessing` is added as flag.
- `addheader` now recognises many more extensions. Too many to list here.
- `addheader` now also recognises full filenames such as `Makefile` and `.gitignore`.
- Added BibTex comment style.
- Updated translations:
  - Dutch (André Ockers, Carmen Bianca Bakker)
  - French (OliBug, Vincent Lequertier)
  - Galician (pd)
  - German (Max Mehl)
  - Esperanto (Carmen Bianca Bakker)
  - Portuguese (José Vieira)
  - Spanish (Roberto Bauglir)
  - Turkish (T. E. Kalayci)

### 5.12.2 Changed

- The linter output has been very slightly re-ordered to be more internally consistent.
- `reuse --version` now prints a version with a Git hash on development versions. Towards that end, the tool now depends on `setuptools-scm` during setup. It is not a runtime dependency.

### 5.12.3 Removed

- `lint` no longer accepts path arguments. Where previously one could do `reuse lint SUBDIRECTORY`, this is no longer possible. When linting, you must always lint the entire project. To change the project's root, use `--root`.
- `FileReportInfo` has been removed. `FileReport` is used instead.

### 5.12.4 Fixed

- A license that does not have a file extension, but whose full name is a valid SPDX License Identifier, is now correctly identified as such. The linter will complain about them, however.
- If the linter detects a license as being a bad license, that license can now also be detected as being missing.
- Performance of `project.all_files()` has been improved by quite a lot.
- Files with CRLF line endings are now better supported.

## 5.13 0.7.0 - 2019-11-28

### 5.13.1 Changed

- The program's package name on PyPI has been changed from `fsfe-reuse` to `reuse`. `fsfe-reuse==1.0.0` has been created as an alias that depends on `reuse`. `fsfe-reuse` will not receive any more updates, but will still host the old versions.
- For users of `fsfe-reuse`, this means:
  - If you depend on `fsfe-reuse` or `fsfe-reuse>=0.X.Y` in your `requirements.txt`, you will get the latest version of `reuse` when you install `fsfe-reuse`. You may like to change the name to `reuse` explicitly, but this is not strictly necessary.
  - If you depend on `fsfe-reuse==0.X.Y`, then you will keep getting that version. When you bump the version you depend on, you will need to change the name to `reuse`.
  - If you depend on `fsfe-reuse>=0.X.Y<1.0.0`, then 0.6.0 will be the latest version you receive. In order to get a later version, you will need to change the name to `reuse`.

## 5.14 0.6.0 - 2019-11-19

### 5.14.1 Added

- `--include-submodules` is added to also include submodules when linting et cetera.
- `addheader` now also recognises the following extensions:
  - `.kt`
  - `.xml`
  - `.yaml`
  - `.yml`

### 5.14.2 Changed

- Made the workaround for `MachineReadableFormatError` introduced in 0.5.2 more generic.
- Improved shebang detection in `addheader`.
- For `addheader`, the SPDX comment block now need not be the first thing in the file. It will find the SPDX comment block and deal with it in-place.
- Git submodules are now ignored by default.
- `addheader --explicit-license` now no longer breaks on unsupported filetypes.

## 5.15 0.5.2 - 2019-10-27

### 5.15.1 Added

- `python3 -m reuse` now works.

### 5.15.2 Changed

- Updated license list to 3.6-2-g2a14810.

### 5.15.3 Fixed

- Performance of `reuse lint` improved by at least a factor of 2. It no longer does any checksums on files behind the scenes.
- Also handle `MachineReadableFormatError` when parsing DEP5 files. Tries to import that error. If the import is unsuccessful, it is handled.

## 5.16 0.5.1 - 2019-10-24 [YANKED]

This release was replaced by 0.5.2 due to importing `MachineReadableFormatError`, which is not a backwards-compatible change.

## 5.17 0.5.0 - 2019-08-29

### 5.17.1 Added

- TeX and ML comment styles added.
- Added `--year` and `--exclude-year` to `reuse addheader`.
- Added `--template` to `reuse addheader`.
- Added `--explicit-license` to `reuse addheader`.
- `binaryornot` added as new dependency.
- Greatly improved the usage documentation.

### 5.17.2 Changed

- `reuse addheader` now automatically adds the current year to the copyright notice.
- `reuse addheader` preserves the original header below the new header if it did not contain any SPDX information.
- `reuse addheader` now correctly handles `.license` files.
- Bad licenses are no longer resolved to LicenseRef-Unknown. They are instead resolved to the stem of the path. This reduces the magic in the code base.
- `.gitkeep` files are now ignored by the tool.
- Changed Lisp's comment character from `';;'` to `';'`.

## 5.18 0.4.1 - 2019-08-07

### 5.18.1 Added

- `--all` argument help to `reuse download`, which downloads all detected missing licenses.

### 5.18.2 Fixed

- When using `reuse addheader` on a file that contains a shebang, the shebang is preserved.
- Copyright lines in `reuse spdx` are now sorted.
- Some publicly visible TODOs were patched away.

## 5.19 0.4.0 - 2019-08-07

This release is a major overhaul and refactoring of the tool. Its primary focus is improved usability and speed, as well as adhering to version 3.0 of the REUSE Specification.

### 5.19.1 Added

- `reuse addheader` has been added as a way to automatically add copyright statements and license identifiers to the headers of files. It is currently not complete.
- `reuse init` has been added as a way to initialise a REUSE project. Its functionality is currently scarce, but should improve in the future.

### 5.19.2 Changed

- `reuse lint` now provides a helpful summary instead of merely spitting out non-compliant files.
- `reuse compile` is now `reuse spdx`.
- In addition to `Copyright` and `@`, copyright lines can be marked with the tag `SPDX-FileCopyrightText:`. This is the new recommended default.
- Project no longer depends on `pygit2`.
- The list of SPDX licenses has been updated.
- `Valid-License-Identifier` is no longer used, and licenses and exceptions can now only live inside of the `LICENSES/` directory.

### 5.19.3 Removed

- Removed `--ignore-debian`.
- Removed `--spdx-mandatory`, `--copyright-mandatory`, `--ignore-missing` arguments from `reuse lint`.
- Remove `reuse license`.
- GPL-3.0 and GPL-3.0+ (and all other similar GPL licenses) are no longer detected as SPDX identifiers. Use GPL-3.0-only and GPL-3.0-or-later instead.

### 5.19.4 Fixed

- Scanning a Git directory is a lot faster now.
- Scanning binary files is a lot faster now.

## 5.20 0.3.4 - 2019-04-15

This release should be a short-lived one. A new (slightly backwards-incompatible) version is in the works.

### 5.20.1 Added

- Copyrights can now start with `@` in addition to `Copyright`. The former is now recommended, but they are functionally similar.

### 5.20.2 Changed

- The source code of reuse is now formatted with `black`.
- The repository has been moved from <https://git.fsfe.org/reuse/reuse> to <https://gitlab.com/reuse/reuse>.

## 5.21 0.3.3 - 2018-07-15

### 5.21.1 Fixed

- Any files with the suffix `.spdx` are no longer considered licenses.

## 5.22 0.3.2 - 2018-07-15

### 5.22.1 Fixed

- The documentation now builds under Python 3.7.

## 5.23 0.3.1 - 2018-07-14

### 5.23.1 Fixed

- When using reuse from a child directory using `pygit2`, correctly find the root.

## 5.24 0.3.0 - 2018-05-16

### 5.24.1 Changed

- The output of `reuse compile` is now deterministic. The files, copyright lines and SPDX expressions are sorted alphabetically.

### 5.24.2 Fixed

- When a GPL license could not be found, the correct `-only` or `-or-later` extension is now used in the warning message, rather than a bare `GPL-3.0`.
- If you have a license listed as `SPDX-Valid-License: GPL-3.0-or-later`, this now correctly matches corresponding SPDX identifiers. Still it is recommended to use `SPDX-Valid-License: GPL-3.0` instead.

## 5.25 0.2.0 - 2018-04-17

### 5.25.1 Added

- Internationalisation support added. Initial support for:
  - English.
  - Dutch.
  - Esperanto.
  - Spanish.

### 5.25.2 Fixed

- The license list of SPDX 3.0 has deprecated `GPL-3.0` and `GPL-3.0+` et al in favour of `GPL-3.0-only` and `GPL-3.0-or-later`. The program has been amended to accommodate sufficiently for those licenses.

### 5.25.3 Changed

- `Project.reuse_info_of` now extracts, combines and returns information both from the file itself and from `debian/copyright`.
- `ReuseInfo` now holds sets instead of lists.
  - As a result of this, `ReuseInfo` will not hold duplicates of copyright lines or SPDX expressions.
- `click` removed as dependency. Good old `argparse` from the library is used instead.

## 5.26 0.1.1 - 2017-12-14

### 5.26.1 Changed

- The `reuse --help` text has been tidied up a little bit.

### 5.26.2 Fixed

- Release date in change log fixed.
- The PyPI homepage now gets `reStructuredText` instead of `Markdown`.

## 5.27 0.1.0 - 2017-12-14

### 5.27.1 Added

- Successfully parse old-style C and HTML comments now.
- Added `reuse compile`, which creates an SPDX bill of materials.
- Added `--ignore-missing` to `reuse lint`.
- Allow to specify multiple paths to `reuse lint`.
- `chardet` added as dependency.
- `pygit2` added as soft dependency. `reuse` remains usable without it, but the performance with `pygit2` is significantly better. Because `pygit2` has a non-Python dependency (`libgit2`), it must be installed independently by the user. In the future, when `reuse` is packaged natively, this will not be an issue.

### 5.27.2 Changed

- Updated to version 2.0 of the REUSE recommendations. The most important change is that `License-Filename` is no longer used. Instead, the filename is deducted from `SPDX-License-Identifier`. This change is **NOT** backwards compatible.
- The conditions for linting have changed. A file is now non-compliant when:
  - The license associated with the file could not be found.
  - There is no SPDX expression associated with the file.
  - There is no copyright notice associated with the file.
- Only read the first 4 KiB (by default) from code files rather than the entire file when searching for SPDX tags. This speeds up the tool a bit.
- `Project.reuse_info_of` no longer raises an exception. Instead, it returns an empty `ReuseInfo` object when no reuse information is found.
- Logging is a lot prettier now. Only output entries from the `reuse` module.

### 5.27.3 Fixed

- `reuse --ignore-debian compile` now works as expected.
- The tool no longer breaks when reading a file that has a non-UTF-8 encoding. Instead, `chardet` is used to detect the encoding before reading the file. If a file still has errors during decoding, those errors are silently ignored and replaced.

## 5.28 0.0.4 - 2017-11-06

### 5.28.1 Fixed

- Removed dependency on `os.PathLike` so that Python 3.5 is actually supported

## 5.29 0.0.3 - 2017-11-06

### 5.29.1 Fixed

- Fixed the link to PyPI in the README.

## 5.30 0.0.2 - 2017-11-03

This is a very early development release aimed at distributing the program as soon as possible. Because this is the first release, the changelog is a little empty beyond “created the program”.

The program can do roughly the following:

- Detect the license of a given file through one of three methods (in order of precedence):
  - Information embedded in the `.license` file.



- Information embedded in its header.
  - Information from the global debian/copyright file.
- Find and report all files in a project tree of which the license could not be found.
- Ignore files ignored by Git.
- Do some logging into STDERR.



## 6.1 reuse package

### 6.1.1 Submodules

#### reuse.download module

Functions for downloading license files from spdx/license-data-list.

`reuse.download.add_arguments(parser)`

Add arguments to parser.

**Return type** `None`

`reuse.download.download_license(spx_identifier)`

Download the license text from the SPDX repository.

**Parameters** `spx_identifier` (`str`) – SPDX identifier of the license.

**Raises** `requests.RequestException` – if the license could not be downloaded.

**Return type** `str`

**Returns** The license text.

`reuse.download.put_license_in_file(spx_identifier, destination)`

Download a license and put it in the destination file.

This function exists solely for convenience.

**Parameters**

- `spx_identifier` (`str`) – SPDX License Identifier of the license.
- `destination` (`PathLike`) – Where to put the license.

**Raises**

- `requests.RequestException` – if the license could not be downloaded.
- `FileExistsError` – if the license file already exists.

**Return type** `None`

`reuse.download.run(args, project, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Download license and place it in the LICENSES/ directory.

**Return type** `int`

## reuse.header module

Functions for manipulating the comment headers of files.

**exception** `reuse.header.MissingSpxInfo`

Bases: `Exception`

Some SPDX information is missing from the result.

`reuse.header.add_arguments(parser)`

Add arguments to parser.

**Return type** `None`

`reuse.header.create_header(spx_info, header=None, template=None, template_is_commented=False, style=None, force_multi=False, merge_copyrights=False)`

Create a header containing *spx\_info*. *header* is an optional argument containing a header which should be modified to include *spx\_info*. If *header* is not given, a brand new header is created.

*template*, *template\_is\_commented*, and *style* determine what the header will look like, and whether it will be commented or not.

**Raises**

- **CommentCreateError** – if a comment could not be created.
- **MissingSpxInfo** – if the generated comment is missing SPDX information.

**Return type** `str`

`reuse.header.find_and_replace_header(text, spdx_info, template=None, template_is_commented=False, style=None, force_multi=False, merge_copyrights=False)`

Find the first SPDX comment block in *text*. That comment block is replaced by a new comment block containing *spx\_info*. It is formatted as according to *template*. The template is normally uncommented, but if it is already commented, *template\_is\_commented* should be `True`.

If both *style* and *template\_is\_commented* are provided, *style* is only used to find the header comment.

If the comment block already contained some SPDX information, that information is merged into *spx\_info*.

If no header exists, one is simply created.

*text* is returned with a new header.

**Raises**

- **CommentCreateError** – if a comment could not be created.
- **MissingSpxInfo** – if the generated comment is missing SPDX information.

**Return type** `str`

`reuse.header.run(args, project, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Add headers to files.

**Return type** `int`

## reuse.init module

Functions for REUSE-ifying a project.

`reuse.init.add_arguments(parser)`  
Add arguments to parser.

`reuse.init.prompt_licenses(out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
Prompt the user for a list of licenses.

**Return type** `List[str]`

`reuse.init.run(args, project, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
List all non-compliant files.

## reuse.lint module

All linting happens here. The linting here is nothing more than reading the reports and printing some conclusions.

`reuse.lint.add_arguments(parser)`  
Add arguments to parser.

`reuse.lint.lint(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
Lint the entire project.

**Return type** `bool`

`reuse.lint.lint_bad_licenses(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Lint for bad licenses. Bad licenses are licenses that are not in the SPDX License List or do not start with LicenseRef-.

**Return type** `Iterable[str]`

`reuse.lint.lint_deprecated_licenses(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Lint for deprecated licenses.

**Return type** `Iterable[str]`

`reuse.lint.lint_files_without_copyright_and_licensing(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Lint for files that do not have copyright or licensing information.

**Return type** `Iterable[str]`

`reuse.lint.lint_licenses_without_extension(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Lint for licenses without extensions.

**Return type** `Iterable[str]`

`reuse.lint.lint_missing_licenses(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Lint for missing licenses. A license is missing when it is referenced in a file, but cannot be found.

**Return type** `Iterable[str]`

`reuse.lint.lint_read_errors(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Lint for read errors.

**Return type** `Iterable[str]`

`reuse.lint.lint_summary(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
Print a summary for linting.

**Return type** `None`

`reuse.lint.lint_unused_licenses(report, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
Lint for unused licenses.

**Return type** `Iterable[str]`

`reuse.lint.run(args, project, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
List all non-compliant files.

## reuse.project module

Module that contains the central Project class.

**class** `reuse.project.Project(root, include_submodules=False, include_meson_subprojects=False)`  
Bases: `object`

Simple object that holds the project's root, which is necessary for many interactions.

**all\_files**(*directory=None*)

Yield all files in *directory* and its subdirectories.

The files that are not yielded are:

- Files ignored by VCS (e.g., see `.gitignore`)
- Files/directories matching `IGNORE_*_PATTERNS`.

**Return type** `Iterator[Path]`

**relative\_from\_root**(*path*)

If the project root is `/tmp/project`, and *path* is `/tmp/project/src/file`, then return `src/file`.

**Return type** `Path`

**property root:** `pathlib.Path`

Path to the root of the project.

**Return type** `Path`

**spdx\_info\_of**(*path*)

Return SPDX info of *path*.

This function will return any SPDX information that it can find, both from within the file and from the `.reuse/dep5` file.

**Return type** `SpdxInfo`

`reuse.project.create_project()`

Create a project object. Try to find the project root from CWD, otherwise treat CWD as root.

**Return type** `Project`

## reuse.report module

Module that contains reports about files and projects for linting.

**class** reuse.report.**FileReport**(*name, path, do\_checksum=True*)

Bases: `object`

Object that holds a linting report about a single file. Importantly, it also contains SPDX File information in `spdxfile`.

**classmethod** **generate**(*project, path, do\_checksum=True*)

Generate a FileReport from a path in a Project.

Return type `FileReport`

**to\_dict**()

Turn the report into a json-like dictionary.

**class** reuse.report.**ProjectReport**(*do\_checksum=True*)

Bases: `object`

Object that holds linting report about the project.

**bill\_of\_materials**()

Generate a bill of materials from the project.

See <https://spdx.org/specifications>.

Return type `str`

**property** **files\_without\_copyright**: `Iterable[os.PathLike]`

Iterable of paths that have no copyright information.

Return type `Iterable[PathLike]`

**property** **files\_without\_licenses**: `Iterable[os.PathLike]`

Iterable of paths that have no license information.

Return type `Iterable[PathLike]`

**classmethod** **generate**(*project, do\_checksum=True, multiprocessing=True*)

Generate a ProjectReport from a Project.

Return type `ProjectReport`

**to\_dict**()

Turn the report into a json-like dictionary.

**property** **unused\_licenses**: `Set[str]`

Set of license identifiers that are not found in any file report.

Return type `Set[str]`

**property** **used\_licenses**: `Set[str]`

Set of license identifiers that are found in file reports.

Return type `Set[str]`

## reuse.spdx module

Compilation of the SPDX Document.

`reuse.spdx.add_arguments(parser)`  
Add arguments to the parser.

**Return type** `None`

`reuse.spdx.run(args, project, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
Print the project's bill of materials.

**Return type** `int`

## reuse.supported\_licenses module

supported-licenses command handler

`reuse.supported_licenses.add_arguments(parser)`  
Add arguments to the parser.

**Return type** `None`

`reuse.supported_licenses.run(args, project, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`  
Print the supported SPDX licenses list

## reuse.vcs module

This module deals with version control systems.

**class** `reuse.vcs.VCSStrategy(project)`  
Bases: `abc.ABC`

Strategy pattern for version control systems.

**abstract classmethod** `find_root(cwd=None)`  
Try to find the root of the project from `cwd`. If none is found, return `None`.

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `Optional[Path]`

**abstract classmethod** `in_repo(directory)`  
Is `directory` inside of the VCS repository?

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `bool`

**abstract** `is_ignored(path)`  
Is `path` ignored by the VCS?

**Return type** `bool`

**class** `reuse.vcs.VCSStrategyGit(project)`  
Bases: `reuse.vcs.VCSStrategy`

Strategy that is used for Git.

**classmethod** `find_root(cwd=None)`  
Try to find the root of the project from `cwd`. If none is found, return `None`.



**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `Optional[Path]`

**classmethod** `in_repo(directory)`

Is *directory* inside of the VCS repository?

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `bool`

**is\_ignored(path)**

Is *path* ignored by the VCS?

**Return type** `bool`

**class** `reuse.vcs.VCSStrategyHg(project)`

Bases: `reuse.vcs.VCSStrategy`

Strategy that is used for Mercurial.

**classmethod** `find_root(cwd=None)`

Try to find the root of the project from *cwd*. If none is found, return None.

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `Optional[Path]`

**classmethod** `in_repo(directory)`

Is *directory* inside of the VCS repository?

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `bool`

**is\_ignored(path)**

Is *path* ignored by the VCS?

**Return type** `bool`

**class** `reuse.vcs.VCSStrategyNone(project)`

Bases: `reuse.vcs.VCSStrategy`

Strategy that is used when there is no VCS.

**classmethod** `find_root(cwd=None)`

Try to find the root of the project from *cwd*. If none is found, return None.

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `Optional[Path]`

**classmethod** `in_repo(directory)`

Is *directory* inside of the VCS repository?

**Raises** `NotADirectoryError` – if directory is not a directory.

**Return type** `bool`

**is\_ignored(path)**

Is *path* ignored by the VCS?

**Return type** `bool`

`reuse.vcs.find_root(cwd=None)`

Try to find the root of the project from *cwd*. If none is found, return None.

**Raises** `NotADirectoryError` – if directory is not a directory.

Return type `Optional[Path]`

### 6.1.2 Module contents

reuse is a tool for compliance with the REUSE recommendations.

**exception** `reuse.IdentifierNotFound`

Bases: `reuse.ReuseException`

Could not find SPDX identifier for license file.

**exception** `reuse.ReuseException`

Bases: `Exception`

Base exception.

**class** `reuse.SpdxInfo(spdx_expressions, copyright_lines)`

Bases: `tuple`

Simple structure for holding SPDX information.

The two iterables MUST be sets.

**property** `copyright_lines`

Alias for field number 1

**property** `spdx_expressions`

Alias for field number 0

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### r

- `reuse`, [46](#)
- `reuse.download`, [39](#)
- `reuse.header`, [40](#)
- `reuse.init`, [41](#)
- `reuse.lint`, [41](#)
- `reuse.project`, [42](#)
- `reuse.report`, [43](#)
- `reuse.spdx`, [44](#)
- `reuse.supported_licenses`, [44](#)
- `reuse.vcs`, [44](#)



## INDEX

### A

`add_arguments()` (in module `reuse.download`), 39  
`add_arguments()` (in module `reuse.header`), 40  
`add_arguments()` (in module `reuse.init`), 41  
`add_arguments()` (in module `reuse.lint`), 41  
`add_arguments()` (in module `reuse.spdx`), 44  
`add_arguments()` (in module `reuse.supported_licenses`), 44  
`all_files()` (`reuse.project.Project` method), 42

### B

`bill_of_materials()` (`reuse.report.ProjectReport` method), 43

### C

`copyright_lines` (`reuse.SpdxInfo` property), 46  
`create_header()` (in module `reuse.header`), 40  
`create_project()` (in module `reuse.project`), 42

### D

`download_license()` (in module `reuse.download`), 39

### F

`FileReport` (class in `reuse.report`), 43  
`files_without_copyright` (`reuse.report.ProjectReport` property), 43  
`files_without_licenses` (`reuse.report.ProjectReport` property), 43  
`find_and_replace_header()` (in module `reuse.header`), 40  
`find_root()` (in module `reuse.vcs`), 45  
`find_root()` (`reuse.vcs.VCSStrategy` class method), 44  
`find_root()` (`reuse.vcs.VCSStrategyGit` class method), 44  
`find_root()` (`reuse.vcs.VCSStrategyHg` class method), 45  
`find_root()` (`reuse.vcs.VCSStrategyNone` class method), 45

### G

`generate()` (`reuse.report.FileReport` class method), 43

`generate()` (`reuse.report.ProjectReport` class method), 43

### I

`IdentifierNotFound`, 46  
`in_repo()` (`reuse.vcs.VCSStrategy` class method), 44  
`in_repo()` (`reuse.vcs.VCSStrategyGit` class method), 45  
`in_repo()` (`reuse.vcs.VCSStrategyHg` class method), 45  
`in_repo()` (`reuse.vcs.VCSStrategyNone` class method), 45  
`is_ignored()` (`reuse.vcs.VCSStrategy` method), 44  
`is_ignored()` (`reuse.vcs.VCSStrategyGit` method), 45  
`is_ignored()` (`reuse.vcs.VCSStrategyHg` method), 45  
`is_ignored()` (`reuse.vcs.VCSStrategyNone` method), 45

### L

`lint()` (in module `reuse.lint`), 41  
`lint_bad_licenses()` (in module `reuse.lint`), 41  
`lint_deprecated_licenses()` (in module `reuse.lint`), 41  
`lint_files_without_copyright_and_licensing()` (in module `reuse.lint`), 41  
`lint_licenses_without_extension()` (in module `reuse.lint`), 41  
`lint_missing_licenses()` (in module `reuse.lint`), 41  
`lint_read_errors()` (in module `reuse.lint`), 41  
`lint_summary()` (in module `reuse.lint`), 42  
`lint_unused_licenses()` (in module `reuse.lint`), 42

### M

`MissingSpdxInfo`, 40  
module  
    `reuse`, 46  
    `reuse.download`, 39  
    `reuse.header`, 40  
    `reuse.init`, 41  
    `reuse.lint`, 41  
    `reuse.project`, 42  
    `reuse.report`, 43  
    `reuse.spdx`, 44  
    `reuse.supported_licenses`, 44  
    `reuse.vcs`, 44

## P

`Project` (class in `reuse.project`), 42  
`ProjectReport` (class in `reuse.report`), 43  
`prompt_licenses()` (in module `reuse.init`), 41  
`put_license_in_file()` (in module `reuse.download`), 39

## R

`relative_from_root()` (`reuse.project.Project` method), 42  
`reuse`  
    module, 46  
`reuse.download`  
    module, 39  
`reuse.header`  
    module, 40  
`reuse.init`  
    module, 41  
`reuse.lint`  
    module, 41  
`reuse.project`  
    module, 42  
`reuse.report`  
    module, 43  
`reuse.spdx`  
    module, 44  
`reuse.supported_licenses`  
    module, 44  
`reuse.vcs`  
    module, 44  
`ReuseException`, 46  
`root` (`reuse.project.Project` property), 42  
`run()` (in module `reuse.download`), 39  
`run()` (in module `reuse.header`), 40  
`run()` (in module `reuse.init`), 41  
`run()` (in module `reuse.lint`), 42  
`run()` (in module `reuse.spdx`), 44  
`run()` (in module `reuse.supported_licenses`), 44

## S

`spdx_expressions` (`reuse.SpdxInfo` property), 46  
`spdx_info_of()` (`reuse.project.Project` method), 42  
`SpdxInfo` (class in `reuse`), 46

## T

`to_dict()` (`reuse.report.FileReport` method), 43  
`to_dict()` (`reuse.report.ProjectReport` method), 43

## U

`unused_licenses` (`reuse.report.ProjectReport` property), 43  
`used_licenses` (`reuse.report.ProjectReport` property), 43

## V

`VCSStrategy` (class in `reuse.vcs`), 44  
`VCSStrategyGit` (class in `reuse.vcs`), 44  
`VCSStrategyHg` (class in `reuse.vcs`), 45  
`VCSStrategyNone` (class in `reuse.vcs`), 45