



» The Open Compliance Program

## Code Janitor Tool

Overview and Discussion

.....  
By Stew Benedict and Jeff Licquia, The Linux Foundation

A White Paper By The Linux Foundation  
<http://www.linuxfoundation.org>

# Introduction

Over the past few months, the Linux Foundation has published a number of papers on the topic of free and open source software licensing compliance, in preparation for its Open Compliance Program. This paper, the fifth in the series, covers the Code Janitor tool, an open source tool designed to assist in evaluating source code being released for compliance with open source software licenses.

The Code Janitor tool is intended to be used as part of a total end-to-end compliance management process, as described in previous papers in the series. To sum up, such a process might contain:

- A Bill of Materials analysis tool, to list what is being distributed and track when that list changes;
- A Source Code and License Identification tool, for collecting source code and identifying its licensing;
- A Dependency Checker tool, for detecting dependencies and highlighting legal compatibility problems in those dependencies;
- A Software Inventory tool, for maintaining information about source code, dependencies, and licensing over time ;
- A Source Code Peer Review tool, for reviewing the quality of code and finding problems early ;
- A Binary Analysis tool, for detecting potential licensing issues in binary-only components.

These tools fit into a complete compliance process designed to ensure that source code releases are done properly and when necessary, as illustrated in Figure 1.

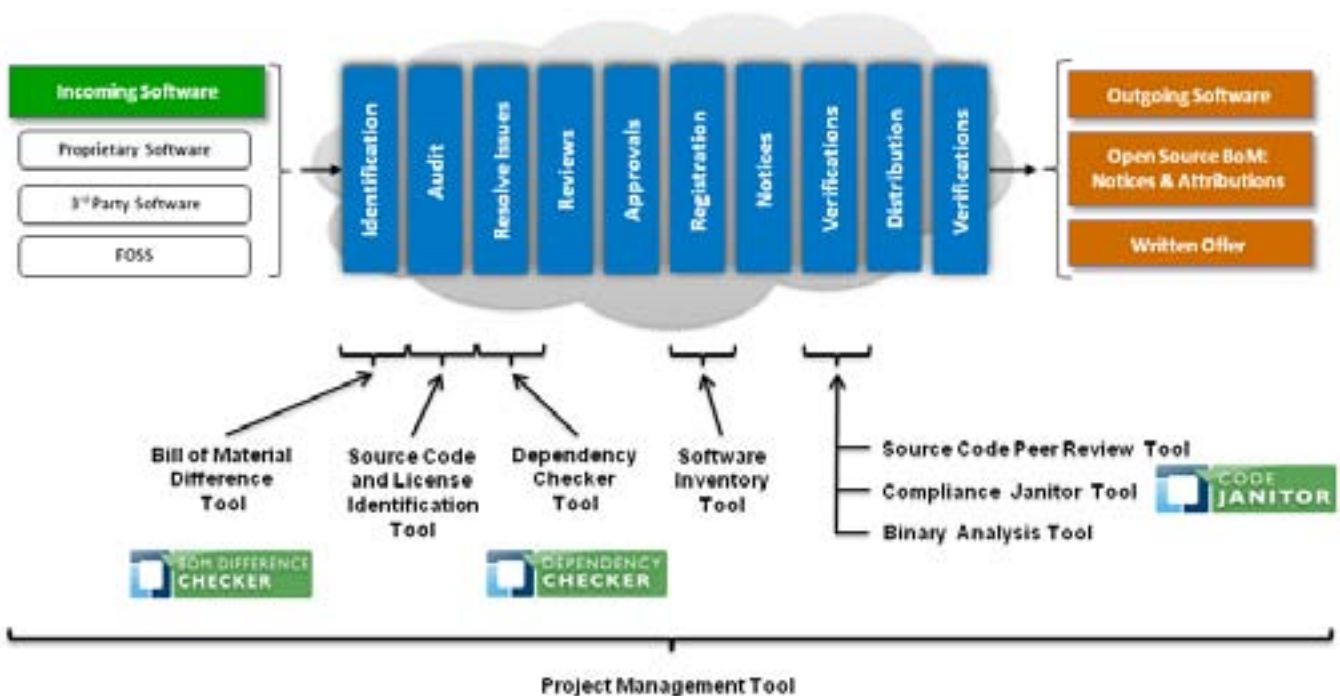


Figure 1. Examples of tools used in compliance management

As part of that process, the Code Janitor tool can help to verify that code is suitable for release by searching for keywords that might indicate problems.

# Motivation for the Code Janitor

Often, compliance with open source licenses will require the release of source code written internally by a company. This may have been the plan from the beginning, or the release might have been dictated by other factors well after the project's start.

As projects progress, programmers make choices every day that could impact the company as a whole, should the code they write be revealed to the public. Some examples include:

- Disparaging comments about individuals or other companies could be embarrassing, or even open the company to legal liability.
- Profanity in comments, or profanely-named identifiers. Since the advent of Google Code Search, a number of top companies have had their code scrutinized and been embarrassed by the result.
- Unauthorized "borrowing" of source code from other projects, which can lead to copyright problems when the code is discovered.
- Accidental disclosure of internal processes, secret projects, or other proprietary information.
- Removal of information that could identify individual authors of code, should the company wish to keep that information confidential.

These problems can be solved with a thorough review of source code before release. However, when projects run into the millions of lines, a complete independent review can take a long time. Searching for keywords can speed up the process considerably, by directing manual attention at only suspicious parts of the code.

Many tools exist for doing this kind of search, but none is tailored to the specific needs of a compliance audit. The classic tools for this job in the Linux space are the "grep" family of command-line utilities; these are powerful, yet require a familiarity with the UNIX shell that is standard fare for Linux developers, but perhaps more than a typical legal professional may be used to.

Further, most search tools make no finer distinctions between results; either something is found, or it isn't. Dividing results into sections can often result in an entirely new search, as the raw results of one search are split by multiple new searches, each requiring its own configuration.

It seems clear, therefore, that a new tool is called for, with the following features:

- A user interface designed for non-professional computer users.
- Keyword and key-phrase support.
- Category support, for classes of keywords that fit the same goal.
- The ability to generate reports suitable for sharing, embedding, and archiving.

## License

The license of the Code Janitor is as follows:

Copyright (c) 2010 Linux Foundation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following

conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Using the Code Janitor

The Code Janitor uses a HTML-based user interface, designed for viewing in a Web browser. It can be installed on a user's own workstation, or on a separate server.

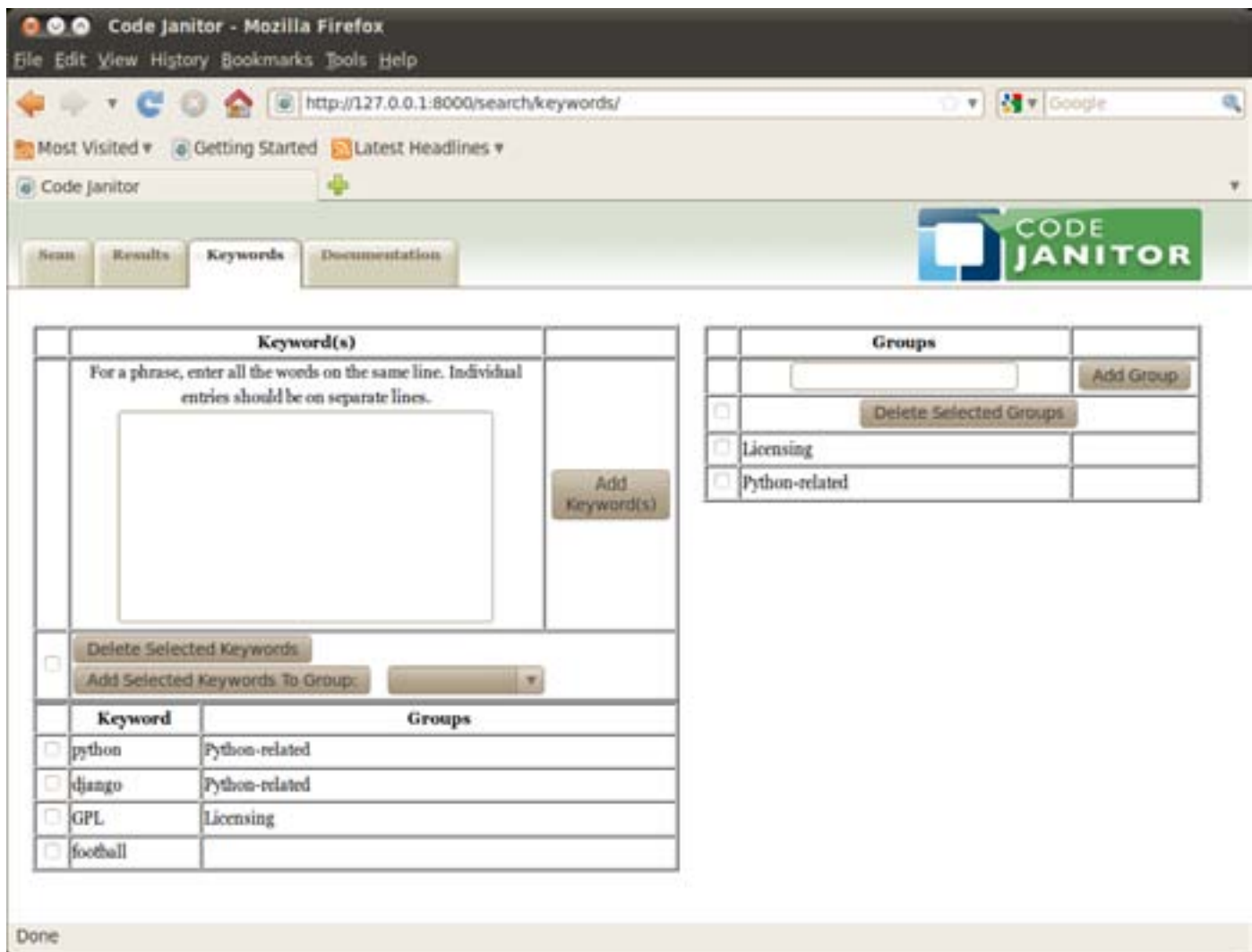
The GUI is arranged with "tabs" for the major functional areas. Most interaction with the Code Janitor will be on the Scan tab, although when the Code Janitor is run for the first time, you should start with the Keywords tab to define the keywords to search for.

To illustrate usage of the tool, consider an example of a project to search source code for unauthorized use of GPL code, references to Python, and a miscellaneous, uncategorized keyword.

### Keywords

The **Keywords** tab allows you to enter keywords to scan for. The large text entry area is where you add new keywords. Words all on one line are considered a phrase, and words entered on separate lines will be different keyword records.

For our example, we'll add some words and phrases that we want to look for, and group them into GPL and Profanity categories. Figure 1 shows the Keywords tab layout after some keywords have been entered.



In our example, we would enter "python", "django", and "GPL" as keywords. Clicking the Add Keywords button causes all three keywords to appear.

You can also create groups that can be used to arrange the results in the final report, or if you want to only scan for a subset of keywords. Type the name for the new group into the text area immediately below the Groups label, and then click Add Group to add the group. For our example, type "Python-related" and click the Add Group button, then do the same with "Licensing".

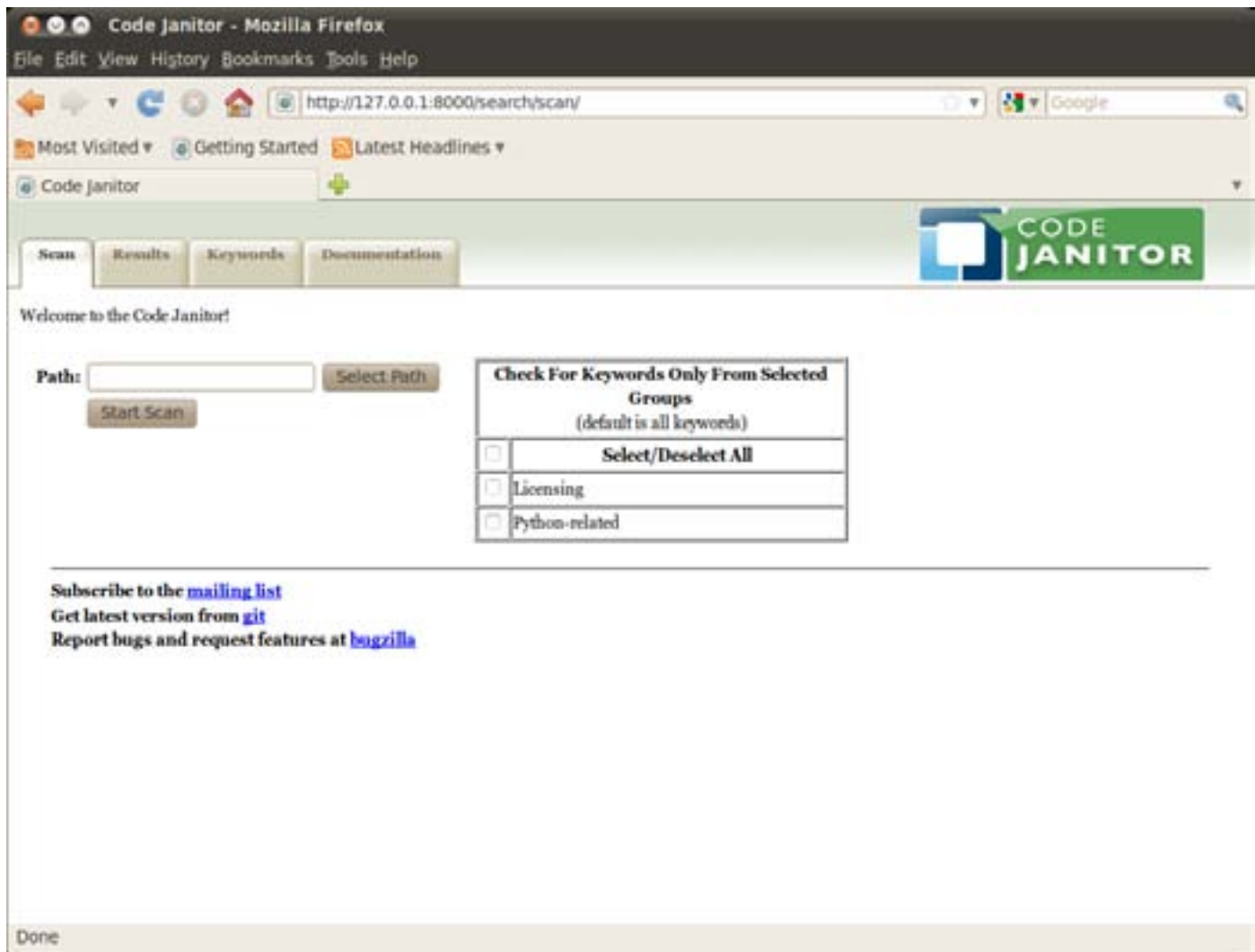
Once the groups have been defined, you can add keywords to them. For our example, click the checkboxes next to "python" and "django", and then use the drop-down bar next to the Add Selected Keywords to Group button to select "Python-related". Clicking the button will cause the group names to appear besides the keywords. Clicking the checkbox by "GPL" and selecting "Licensing" in the drop-down before clicking Add Selected Keywords To Group finishes classifying. Keywords can also stand on their own. Let's say that a last-minute request comes in to find sports references in the code. Adding "football" does not cause it to be added to a group automatically.

Groups can be deleted by selecting the checkbox next to the desired groups and clicking Delete Selected Groups. Deleting a group will also remove keyword associations, but will not delete the keywords themselves; that can be done by selecting the checkbox next to the keyword entry and clicking on Delete Selected Keywords.

## Scan

The Scan tab is the home page of the system and where you initiate a test run. If there are no keywords defined, you will be directed to go to the Keywords tab and add keywords.

Once you have defined keywords, you can either type in a file or directory path or select one using the path browser. If you have groups defined, you will also be presented with the option to only scan for certain groups of keywords. Once you press Start Scan, the display will change to show a status screen of the test and will redirect to the test results when complete.



In our example, we would select the path to our project and click Start Scan. We could, if we

wished, limit the search to just Python-related keywords or just Licensing keywords.

## Results

When the search is done, you will be taken to the results page for the search, which will look much like Figure 3:

File/Path	Line Number(s)	Keyword(s) Found
views.py	1	django
	4	django
ooREADME	19	django
	20	django
	22	django
	24	django
urls.pyc	4	django
urls.py	1	django
	2	django
	4	django
	14	django
	16	django
	18	django
manage.py	1	python
	2	django
	7	django
janitor.sqlite	34	django
	40	django
	41	django

Figure 4: Search report

For each file found in the project, the report lists all keywords, as well as which lines the keywords were found in. The report can be printed using the browser's printing functionality; most browsers support printing to a file as well as to a printer.

The results for each search done are saved, and can be retrieved at any time by clicking on the Results tab. Clicking on the link for a given search result will display a report for that search. Each search result in the result list has a checkbox next to it, which is used for deleting searches. Click the checkbox next to the result(s) you wish to delete, and then click Delete Selected Searches. The checkbox next to the Delete Selected Searches button is used to select or deselect

all searches.

## Installation

The Code Janitor is provided as both an RPM and Debian package. It depends on the Django Web application framework.

The Debian package depends on `python-django`. It may be installed via the usual methods of installing packages on your Debian-based distribution; all of them should support `dpkg`:

```
dpkg -i code-janitor_0.0.2-1_all.deb
```

Since different RPM-based distributions name their Django packages differently, every attempt has been made to depend on at least one of the most popular names, so the following should work:

```
rpm -Uvh code-janitor-0.0.2-1.noarch.rpm
```

If your distribution names the package differently, or if you download and install Django via some other means, you will need to install the package without dependency checking, using the `--nodeps` (for RPM) or `--force-depends` (for Debian) options.

### From Git

You can also check the project out from our version control (git) and run it in place:

```
git clone http://git.linuxfoundation.org/janitor.git
cd janitor
```

Alternatively, you can download the latest tarball from the git web page by clicking on the snapshot link in the upper right-hand part of the page. Once you have the tarball, unpack it:

```
tar -xf dep-checker-[long number].tar.gz
cd dep-checker
```

After either of these two steps, you will need to create the database and the README documentation (this latter step will require `w3m` to be installed):

```
make
```

## Running the Tool

Once the tool is installed, it can be started. If the tool was installed via a package, run it as follows:

```
/opt/linuxfoundation/bin/code-janitor.py start
```

If it was installed from a tarball or from git, run it as follows from the root of the project directory:

```
./code-janitor.py start
```

This will start the Code Janitor, and cause a Web browser to start, pointed at the embedded Code Janitor web server.

## Stopping the Tool

When you're done with the tool, you can stop the tool with the `code-janitor.py` command as before, but using `stop` instead of `start`:

```
/opt/linuxfoundation/bin/code-janitor.py stop
./code-janitor.py stop
```

Use the command which corresponds to the command used to start the tool.

## Getting Help

For more advanced options, or to figure out where something is confusing, look at the [Documentation](#) tab. It contains the full online documentation for the tool.

## Availability and Participation

- You can view the latest source code for the project online, at <http://git.linuxfoundation.org/janitor.git>, or check the source code out using standard git commands:  

```
git clone http://git.linuxfoundation.org/janitor.git
```
- To file bugs or request features, go to <http://bugs.linuxfoundation.org/> and select the Compliance product.
- To subscribe to the mailing list or view the archives, go to <https://lists.linux-foundation.org/mailman/listinfo/code-janitor-dev>.

## Conclusion

The Code Janitor tool provides compliance auditors with a way to find trouble spots within source code before its release. As an Open Source tool, the Linux Foundation invites anyone interested to use and modify it, to contribute changes back, and to provide other feedback about its usefulness.

## About the Authors

Stew Benedict is a Member of Technical Staff at the Linux Foundation. He previously worked as a Distribution Developer for Mandrake/Mandriva, working on PPC/IA64 ports and various security initiatives. Prior to that, he held several roles with an automotive parts manufacturer in Cleveland, Ohio, and with GE Lighting as part of their Electronic Products group. He has written several Linux and Solaris articles for Linux Journal and TechRepublic.

Jeff Licquia is a Member of Technical Staff at the Linux Foundation. Previously, he worked at Progeny Linux Systems, and was the project lead for the Progeny Debian distribution. He is also a member of the Debian project, and has contributed to several books on Linux software development. Over his career, he has worked on a diverse set of problems, from industrial automation to system administration, and on platforms ranging from MS-DOS and Ultrix to Windows and Linux.

# About the Open Compliance Program

The Linux Foundation's Open Compliance Program is the industry's only neutral, comprehensive software compliance initiative. By marshaling the resources of its members and leaders in the compliance community, the Linux Foundation brings together the individuals, companies and legal entities needed to expand the use of open source software while decreasing legal costs and FUD. The Open Compliance Program offers comprehensive training and informational materials, open source tools, an online community (FOSSBazaar), a best practices checklist, a rapid alert directory of company's compliance officers and a standard to help companies uniformly tag and report software used in their products. The Open Compliance Program is led by experts in the compliance industry and backed by such organizations as the Adobe, AMD, ARM Limited, Cisco Systems, Google, HP, IBM, Intel, Motorola, NEC, Novell, Samsung, Software Freedom Law Center, Sony Electronics and many more. More information can be found at <http://www.linuxfoundation.org/programs/legal/compliance>.

The Linux Foundation promotes, protects and standardizes Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about The Linux Foundation, the Open Compliance Program or our other initiatives please visit us at <http://www.linuxfoundation.org/>.

