
ScanCode-Toolkit

AboutCode.org authors and contributors

Mar 19, 2024

CONTENTS

1	Are you new to Scancode-Toolkit?	2
1.1	Table of Contents	2
1.2	Try ScanCode Toolkit	3
1.3	Installing ScanCode	3
1.4	Learn more about ScanCode Toolkit	5
1.5	Contribute	5
2	Getting Started	8
2.1	Getting Started	8
3	Command Line Options Reference	18
3.1	Command Line Interface Reference	18
4	Tutorials	124
4.1	Basic Tutorials	124
5	How-To Documents	140
5.1	How-To Guides	140
6	Contribute To ScanCode	148
6.1	Contribute	148
7	Plugins Documentation	168
7.1	Plugins	168
8	Miscellaneous Documents	176
8.1	Miscellaneous	176
9	Reference Documents	179
9.1	Reference Docs	179
10	Indices and Tables	217
10.1	Something Missing?	217

Welcome to ScanCode Toolkit Documentation!

If you are new to ScanCode Toolkit, start here:

ARE YOU NEW TO SCANCODE-TOOLKIT?

This is the perfect place to start, if you are new to ScanCode-Toolkit. Have a quick look at the table of contents below, as these are the main sections you might need help on. These sections have extensive links to other important documentation pages, and make sure you go through them all.

1.1 Table of Contents

1. *Try ScanCode Toolkit*
 - *Before you start using ScanCode*
 - *Installing ScanCode*
 - *Scan a Codebase*
 - *Use ScanCode Better*
 - *All Tutorials/How-Tos*
 - *ScanCode Versions*
2. *Learn more about ScanCode Toolkit*
 - *CLI Reference*
 - *How Scancode Works*
 - *Plugins*
3. *Contribute*
 - *General Information*
 - *Contribute Code*
 - *Good First Issues*
 - *Add new Functionality/Enhancement to ScanCode*
 - *Update our Documentation*
 - *Participate in GSoC/GSoD*

1.2 Try ScanCode Toolkit

This section is about using the ScanCode Toolkit, i.e. Performing a scan on a codebase/files to determine their license, copyrights and other information, according to your requirements.

1. The [Scan a Codebase](#) section helps you with configuring your virtual environment, installing Scancode and performing a basic scan, and subsequently visualize the results.
2. The [Use ScanCode Better](#) section helps you customize the scan according to your requirements, and better understand the advanced features you can use.
3. The [All Tutorials/How-Tos](#) is essentially an exhaustive list of all Tutorials and How To's with a brief description on what they help you to achieve.

1.3 Installing ScanCode

Scancode-Toolkit can be installed in 3 different methods.

1. The [Scan a Codebase](#) section helps you with configuring and installing ScanCode and performing a basic scan, and then visualizing the results.
2. The [Use ScanCode Better](#) section helps you customize the scan according to your requirements, and better understand advanced features.
3. The [All Tutorials/How-Tos](#) is an exhaustive directory of all Tutorials and How To's with a brief description.

1.3.1 Before you start using ScanCode

1. You need to make sure [Prerequisites](#) are installed, and a `virtualenv` is created.

Installation as an Application: Downloading Releases Installation via Docker: Installation as a library: via pip Installation from Source Code: Git Clone

1. Now you can either follow the instructions for the recommended [Installation as an Application: Downloading Releases](#) method, or run `pip install scancode-toolkit[full]` like that in the [Installation as a library: via pip](#) documentation. Alternatively, you can also [Installation from Source Code: Git Clone](#).
2. Run `scancode -h` to make sure Scancode was installed properly. If this shows any Error, refer the [Common Installation Errors Issue](#) for common errors.

Note: Refer [Quickstart](#) to make sure you are using the scan command correctly.

Note: For Windows, Refer to [Installation on Windows 10](#) for installing easily using Releases.

1.3.2 Scan a Codebase

Once you are all set up with Scancode Toolkit, i.e. Running `scancode -h` shows the *Help text*, you can start scanning files or a codebase.

1. Refer *Quickstart* for commonly used scan commands, and commonly used *Output Formats*. (The recommended output format is JSON)
2. Refer *this section* for Extractcode Options.
3. *How to Run a Scan* is a sample tutorial for absolute beginners, to walk them through the process of running a scan. Follow this tutorial and perform a scan on the `sample` folder distributed with ScanCode, or any file/folder of your choice. Avoid advanced options, and just follow the basic instructions.
4. ScanCode generates output files with scan results. You can visualize JSON result files using *Scancode Workbench*. Follow this tutorial *How to Visualize Scan results* to visualize the scan results.

1.3.3 Use ScanCode Better

1. Go through all the options in the page *All Available Options*, to know about Scancode Command Line options. You can then modify the Scan according to your requirements.

1.3.4 All Tutorials/How-Tos

The Tutorials are:

1. *How to Run a Scan*
2. *How to Visualize Scan results*
3. *How to set what will be detected in Scan*
4. *How To Extract Archives*
5. *How to specify Scancode Output Format*
6. *Add A Post-Scan Plugin*

The How-To's are:

1. *How To Add a New License for Detection*
2. *How to Add New License Rules for Enhanced Detection*

1.3.5 ScanCode Versions

1. You can see all Scancode Toolkit versions on the *GitHub release page*.
 2. Read the *CHANGELOG* for more information on specific releases.
 3. If you want to use/test a specific version of Scancode Toolkit, you can follow the instructions in *Installation from Source Code: Git Clone* docs.
-

1.4 Learn more about ScanCode Toolkit

Here we give an introduction on the ScanCode Toolkit Documentation Sections that can help you to learn more about ScanCode Toolkit.

1.4.1 CLI Reference

This section contains a complete guide to ScanCode Toolkit Command Line options, i.e. What the command-line options are, how different options affect the scan and outputs, how to use these options and examples of their use cases.

Now this section has three types of pages:

1. The *Synopsis* page and the *How to Run a Scan* page as summaries.
2. An exhaustive list of all Command Line Options at *All Available Options*
3. All the other pages detailing the *Type of Options*

Note that the page for one type of options also has a short list of all the options detailed on that page in the beginning. The *All Available Options* page just has all of them together, and also the extractcode options.

1.4.2 How Scancode Works

This section has documentation on *How does ScanCode detect licenses?*.

1.4.3 Plugins

Plugins are an integral part of ScanCode Toolkit in the sense they are used to easily extend Scancode capabilities, and developers can code their own plugins according to their requirements.

This section has documentation on:

1. The *Plugin Architecture*
 2. The *License Policy Plugin*
 3. All *Plugin Tutorials*
-

1.5 Contribute

If you are looking to Contribute to Scancode Toolkit, this is where you start.

1.5.1 General Information

1. Also refer the *Contribution* page here.
2. For more Project Ideas, refer *Contributor Project Ideas (old)*.
3. Before committing your work, make sure you have read this post on *Writing good Commit Messages*.

1.5.2 Contribute Code

If you haven't contributed to Scancode Toolkit refer *Good First Issues*.

To determine where to contribute, you can refer:

1. ScanCode Toolkit tracks issues via the [GitHub Issue tracker](#)
2. Broad [milestones](#) for upcoming versions are also maintained.

And documentation related to contributing code can be referred at *Contributing to Code Development*.

1.5.3 Good First Issues

A [good first issue](#) means it's recommended for people who haven't contributed to Scancode Toolkit before.

1.5.4 Add new Functionality/Enhancement to ScanCode

There are two main paths you can follow to add a new functionality to Scancode. They are:

1. Add the functionality to Scancode itself.
2. Add plugins if the functionality is very much application dependent.

Refer [enhancement issues](#) for the first type of enhancements. If you want to add a plugin to implement the functionality, refer all the *Plugin Tutorials*.

1.5.5 Update our Documentation

Maintaining a comprehensive, accurate, updated and effective documentation is very important as that directly affects the acceptability of Scancode Toolkit.

To contribute to Scancode Toolkit Documentation, first refer the *Contributing to the Documentation* section.

The sections in this page cover the following:

1. *Setup Local Build*
2. *Share Document Improvements*
3. *Continuous Integration* system for the Documentation
4. *Style Checks Using Doc8*
5. *Interspinx*
6. *Style Conventions for the Documentaion*

You can contribute to the following Open Issues on documentation.

1. [First Timers Only Issues List](#)
2. [Documentation Inconsistencies Tracker](#)
3. [ScanCode Toolkit Documentation Roadmap](#)
4. [Issues with label Documentation](#)

Note: Refer *Something Missing?* to report Documentation Errors or to request Improvements.

Also, consider contributing to other Aboutcode Project Documentations, as they need more support.

1.5.6 Participate in GSoC/GSoD

If you want to participate in any of the two programs:

- [Google Summer of Code](#)
- [Google Season of Docs](#)

Then:

1. Keep an eye out for Application Timelines.
2. Solve multiple of these *Good First Issues* to demonstrate your skills, and improve your chances of selection.
3. Refer to the Projects Ideas List for details on tentative projects.
 - [GSoC2023](#)
4. Remain active in Element and talk with the organization mentors well ahead of the deadlines.
5. Select projects according to your skills and finalize project proposals.
6. Discuss your proposals extensively with corresponding mentors.
7. Apply for the Programs well before the Deadline.

Here's a list of more Documentation Pages:

- A *Synopsis* of ScanCode Command Line Options
 - Tutorials on *How to Run a Scan* and *How to Visualize Scan results*
 - An exhaustive List of *All Available Options*
 - Documentation on *Contributing to Code Development*
 - Documentation on *Plugin Architecture*
 - *FAQ*
-

GETTING STARTED

2.1 Getting Started

2.1.1 Home

ScanCode does scan code to detect packages and dependencies, licenses, copyrights and more.

Why ScanCode?

Discovering the origin and license for a software component is important, but it is often much harder to accomplish than it should be because:

- A typical software project may reuse tens or thousands of third-party software components
- Software authors do not always provide copyright and license information
- Copyright and license information that is provided may be hard to find and interpret

ScanCode tries to address these issues by offering:

- A simple command line approach that runs on Windows, Linux, and macOS
- A comprehensive code scanner that can detect origin and license information in codebase files, including binaries
- A comprehensive set of package manifests and lockfile parsers to report direct and pinned dependencies
- Your choice of JSON or other output formats (YAML, SPDX, HTML, CSV) for integration with other tools
- Well-tested, easy to hack, and well-documented code
- A plugin system for easily adding new Functionality to Scans.
- Extensive documentation and support.
- We release of the code and reference data under permissive licenses (Apache 2.0 and CC-BY-4.0)
- ScanCode.io to assemble scripted and specialised code analysis pipelines with a web-based analysis server
- ScanCode workbench for desktop-based scans visualization

ScanCode is recognized as the industry leading engine for license and copyright detection and used as the basis of several open source compliance efforts in open source projects and companies. It's detection engine is embedded in the most advanced open source and commercial tools available today for Software Composition Analysis.

What does ScanCode Toolkit do?

ScanCode detects and normalizes origin, dependencies, licensing and other related information in your code:

- by parsing package manifests and dependencies lock files to a normalized metadata model and assigning each an identifying [Package URL](#),
- by detecting license tags, notices and texts in text and binaries using the world most comprehensive database of licenses texts and notices and a unique combination of techniques,
- by recognizing copyright statements using an advanced natural language parsing grammar and detecting other origin clues (such as emails, urls, and authors)

Using this data you can:

- Discover the origin and license of the open source and third-party software components that you use,
- Discover direct dependent packages and indirect pinned or locked dependencies,
- Assemble a software component Inventory of your codebase, and report the data using standard SBOM formats,
- Use this data as the input to:
 - open source license compliance obligations such as attribution and redistribution.
 - open source package vulnerability detection.

How does it work?

Given a code directory, ScanCode will “scan code”:

- Extract files from any archive using a [universal archive extractor](#)
- Collect an inventory of the code files and classify the code using file types
- Extract texts from binary files as needed
- Use an extensible rules engine to detect open source license text, notices tags, mentions and license expressions with over 31,000 detection rules.
- Use a specialized natural language parser and grammar to capture copyright statements
- Identify packaged code and collect metadata from packages by parsing the manifest and lockfiles (and in some cases also the installed databases for system packages) for these package types: .ABOUT, Alpine Linux apk as packages or installed, Android apk, Autotools, Bazel, JS Bower, Buck, Msft Cab, Rust Cargo, Chef, Chrome, PHP Composer, Conda, Perl CPAN, R CRAN, Debian deb as packages or installed, Apple dmg, Java EAR, FreeBSD, Ruby Gem, Go modules, Haxe, InstallShield, iOS ipa, ISO disk images, Apache IVY, Java JAR, JBoss SAR, Maven, JS Meteor, Mozilla Extension, Msft MSI, JS npm, NSIS Installer, NuGet, Ocaml OPAM, CocoaPods, Dart Pub, Python PyPI wheel and related, structured README, RPMs as packages or installed, Shell archive, Squashfs, Java WAR, Msft Update Manifest, and Windows Executable.
- Report the results in the formats of your choice (JSON, YAML, CSV, SPDX, etc.) for integration with other tools

ScanCode is written in Python and also uses other open source packages.

Alternative?

There are several utilities that do some of what ScanCode does - for instance you can grep files for copyright and license text. This may work well for simple cases - e.g. at the single whole license text files and well structured copyright statements, but we created ScanCode for ourselves because this approach does not help you to see the recurring patterns of licenses and other origin history clues at scale.

You can consider other tools such as:

- FOSSology (open source, written in C, Linux only, GPL-licensed)

History

ScanCode was originally created by nexB to support our software audit consulting services. We have used and continuously enhanced the underlying toolkit for over 12 years. We decided to release ScanCode as open source software to give software development teams the opportunity to perform as much of the software audit function as they like on their own.

Thank you for giving ScanCode a try!

Other Important Documentation

1. *Type of Options*
2. *How to Run a Scan*
3. *Basic Tutorials*
4. *How-To Guides*
5. *Reference Docs*
6. *Contributing to Code Development*
7. *Contributing to the Documentation*
8. *Plugin Architecture*
9. *FAQ*
10. *Support*

2.1.2 Comprehensive Installation

The recommended way to install ScanCode is using app archives:

- *Installation as an Application: Downloading Releases*

The recommended method is to download the latest application release as an application and then configure and use directly. No knowledge of pip/git or other developer tools is necessary. You only need to install Python then download and extract the ScanCode application archive to run ScanCode. For standard usage that's all you need.

For advanced usage and experienced users, you can also use any of these mode:

- *Installation via Docker:*

An alternative to installing the latest Scancode Toolkit release natively is to build a Docker image from the included Dockerfile. The only prerequisite is a working Docker installation.

- *Installation from Source Code: Git Clone*

You can clone the git source code repository and then run the configure script to configure and install ScanCode for local and development usage.

- *Installation as a library: via pip*

To use ScanCode as a library in your application, you can install it via `pip`. This is recommended for developers or users familiar with Python that want to embed ScanCode as a library.

Before Installing

- ScanCode requires a Python version 3.7, 3.8, 3.9 or 3.10 and is tested on Linux, macOS, and Windows. It should work fine on FreeBSD.

System Requirements

- Hardware : ScanCode will run best with a modern X86 64 bits processor and at least 8GB of RAM and 2GB of disk space. These are minimum requirements.
- Supported operating systems: ScanCode should run on these 64-bit OSes running X86_64 processors:
 1. Linux: on recent 64-bit Linux distributions,
 2. Mac: on recent x86 64-bit macOS (10.15 and up, including 11 and 12), Use the X86 emulation mode on Apple ARM M1 CPUs. (Note that *pip install* does not work on ARM CPUs)
 3. Windows: on Windows 10 and up,
 4. FreeBSD.

Prerequisites

ScanCode needs a Python 3.7+ interpreter; We support all Python versions from 3.7 to 3.10. The default version for the application archives is Python 3.8

- **On Linux:**

Use your package manager to install python3.

For Ubuntu, it is `sudo apt install python3-dev`

- On Ubuntu 16, 18, 20 and 22 run:

```
sudo apt install python-dev bzip2 xz-utils zlib1g libxml2-dev libxslt1-
dev libpopt0
```

- On Debian and Debian-based distros run:

```
sudo apt-get install python3-dev libbz2-1.0 xz-utils zlib1g libxml2-dev
libxslt1-dev libpopt0
```

- On RPM-based distros run:

```
sudo yum install python3.8-devel zlib bzip2-libs xz-libs libxml2-devel
libxslt-devel libpopt0
```

- On Fedora 22 and later run:

```
sudo dnf install python3.8-devel xz-libs zlib libxml2-devel libxslt-
↳devel bzip2-libs libpopt0
```

If these packages are not available from your package manager, you must compile them from sources.

- **On Mac:**

The default Python 3 provided with macOS is 3.8. Alternatively you can download and install Python 3.8 from <https://www.python.org/>

- **On Windows:**

Download and install Python 3.8 from <https://www.python.org/>

Note: 64-bit Python interpreters (x86-64) are the only interpreters supported by ScanCode on all operating systems which means only 64-bit Windows is supported.

See the *Installation on Windows 10* section for more installation details.

Installation as an Application: Downloading Releases

Get the ScanCode Toolkit tarball archive of a specific version and your operating system by going to the [project releases page](#)

For example, Version 30.0.1 archive can be obtained from [Toolkit release 30.0.1](#) under assets options.

Note: ScanCode app archives come with packaged with all required dependencies except for Python that has to be downloaded and installed separately. On more recent versions of Ubuntu, you will have to install Python 3.8 manually. One possibility is to use the Deadsnakes PPA (Personal Package Archive) which is a project that provides older Python version builds for Debian and Ubuntu and is available at <https://github.com/deadsnakes/> and <https://launchpad.net/~deadsnakes/+archive/ubuntu/ppa>

```
sudo apt-get update && sudo apt-get upgrade
sudo add-apt-repository ppa:deadsnakes/ppa --yes
sudo apt-get install python3.8 python3.8-distutils
```

Installation on Linux and Mac

Download the archive for your operating system and extract the archive from command line:

```
tar -xvf scandcode-toolkit-30.0.1_py38-linux.tar.gz
```

Or, on Linux, right click and select “Extract Here”.

Check whether the *Prerequisites* are installed. Open a terminal in the extracted directory and run:

```
./scancode --help
```

This will configure ScanCode and display the command line *Help text*.

Installation on Windows 10

- Download the latest ScanCode release zip file for Windows from the latest version at <https://github.com/nexB/scancode-toolkit/releases/>
- In the File Explorer, select the downloaded ScanCode zip and right-click.
- In the pop-up menu select 'Extract All...'
- In the pop-up window 'Extract Compressed (Zipped) Folders' use the default options to extract.
- Once the extraction is complete, a new File Explorer window will pop up.
- In this Explorer window, select the new folder that was created and right-click.

Note: On Windows 10, double-click the new folder, select one of the files inside the folder (e.g., 'setup.py'), and right-click.

- In the pop-up menu select 'Properties'.
- In the pop-up window 'Properties', select the Location value. Copy this to the clipboard and close the 'Properties' window.
- Press the start menu button, click the search box or search icon in the taskbar.
- In the search box type:

```
cmd
```

- Select 'cmd.exe' or 'Command Prompt' listed in the search results.
- A new 'Command Prompt' pops up.
- In this window (aka a 'command prompt'), type 'cd' followed by a space and then Right-click in this window and select Paste. This will paste the path you copied before and is where you extracted ScanCode:

```
cd path/to/extracted/ScanCode
```

- Press Enter.
- This will change the current location of your command prompt to the root directory where ScanCode is installed.
- Then type:

```
scancode -h
```

- Press enter. This first command will configure your ScanCode installation. Several messages are displayed followed by the ScanCode command help.
- The installation is complete.

Un-installation

- Delete the directory in which you extracted ScanCode.
 - Delete any temporary files created in your system temp and user temp directory under a ScanCode-prefixed directory such as `.scancode-tk` or `.cache/scancode-tk`.
-

Installation via Docker:

You can install ScanCode Toolkit by building a Docker image from the included Dockerfile. The prerequisite is a working [docker installation](#).

Download the ScanCode-Toolkit Source Code

- `git clone https://github.com/nexB/scancode-toolkit` to get the latest (*Installation from Source Code: Git Clone*) source code.

Build the Docker image

Run the `docker build` source code checkout directory.:

```
cd scancode-toolkit
docker build --tag scancode-toolkit --tag scancode-toolkit:$(git describe --tags) .
```

Run using Docker

The docker image will forward all arguments it receives directly to the `scancode` command.

Display help:

```
docker run scancode-toolkit --help
```

Mount current working directory as `"/project"` and run a scan on a file name `apache-2.0.LICENSE` directory. The JSON results will be in `scan-result.json`:

```
docker run -v $PWD:/project scancode-toolkit -clipecu --json-pp /project/scan-result.
↪ json /project/apache-2.0.LICENSE
```

This will mount your current working from the host into `/project` in the container and then scan the contents. The output `result.json` will be written back to your current working directory on the host.

Note that the parameters *before* `scancode-toolkit` are used for docker, those after will be forwarded to `scancode`.

Installation from Source Code: Git Clone

You can download the ScanCode Toolkit Source Code and build from it yourself. This is what you would want to do it if:

- You are developing ScanCode or adding new patches or want to run tests.
- You want to test or run a specific version/checkpoint/branch from the version control.

Download the ScanCode-Toolkit Source Code

Run the following once you have [Git](#) installed:

```
git clone https://github.com/nexB/scancode-toolkit.git
cd scancode-toolkit
```

Configure the build

ScanCode use a configure scripts to create an isolated virtual environment, install required packaged dependencies.

On Linux/Mac:

- Open a terminal
- cd to the clone directory
- run `./configure`
- run `source venv/bin/activate`

On Windows:

- open a command prompt
- cd to the clone directory
- run `configure`
- run `venv\Scripts\activate`

Now you are ready to use the freshly configured scancode-toolkit.

Note: For use in development, run instead `configure --dev`. If your face issues while configuring a previous version, `configure --clean` to clean and reset your enviroment. You will need to run `configure` again.

Installation as a library: via pip

ScanCode can be installed from the public PyPI repository using `pip` which is the standard Python package management tool.

Note: Note that *pip* installation method does work on ARM chips, i.e. Linux/macOS on Apple M1 chips, as some non-native dependencies do not have pre-built wheels for ARM (like `py-ahocorasick`, `intbitset`). See [System Requirements](#) for more information. See related issues for more info:

- [Fallback pure-python deps](#)
- [pip install failing on M1](#)

The steps are:

1. Create a Python virtual environment:

```
/usr/bin/python3 -m venv venv
```

For more information on Python virtualenv, visit this [page](#).

1. Activate the virtual environment you just created:

```
source venv/bin/activate
```

2. Run `pip` to install the latest versions of base utilities:

```
pip install --upgrade pip setuptools wheel
```

3. Install the latest version of ScanCode:

```
pip install scancode-toolkit
```

Note: For advanced usage, `scancode-toolkit-mini` is an alternative package with no default dependencies on pre-built binaries. This may come handy for some special use cases such as packaging for a Linux or FreeBSD distro.

To uninstall, run:

```
pip uninstall scancode-toolkit
```

Command Invocation Variations

These are the commands to invoke ScanCode based on:

- your installation methods
- your operating systems

The two forms of commands are:

- Use the `scancode` command directly, typically on Windows or in an activated virtualenv:

```
scancode [OPTIONS] <OUTPUT FORMAT OPTION(s)> <SCAN INPUT>
```

- Use a path to the scancode command, typically with an application installation

`path/to/scancode [OPTIONS] <OUTPUT FORMAT OPTION(s)> <SCAN INPUT>`

These variations are summed up in the following table:

Installation Methods	Appli- ca- tion In- stall	Pip Install	Install from Source Code
Linux	path: <i>./scan- code</i>	direct: scan- code	path: <i>./scancode</i> or direct: <i>scancode</i>
Mac	path: <i>./scan- code</i>	direct: scan- code	path: <i>./scancode</i> or direct: <i>scancode</i>
Windows	path: <i>scan- code</i>	direct: scan- code	path: <i>scancode</i> or direct: <i>scancode</i>

COMMAND LINE OPTIONS REFERENCE

Reference documents describe the Command Line options, and application concepts in depth.

3.1 Command Line Interface Reference

3.1.1 Synopsis

ScanCode detects licenses, copyrights, package manifests and direct dependencies and more, both in source code and binary files, by scanning the files. This page introduces you to the ScanCode Toolkit Command Line Interface in the following sections:

- Installation
- Quickstart
- Type of Options
- Output Formats
- Other Important Documentation

Installation

Scancode-Toolkit installation can be done by downloading ScanCode as an application, which is recommended generally. For users who wish to use ScanCode as a library, it can be installed via `pip`, the default Python Package Manager. Refer the following sections for detailed Instructions on the each of the Installation Methods.

- *Installation as an Application: Downloading Releases*
- *Installation as a library: via pip*
- *Installation from Source Code: Git Clone*

Quickstart

The basic command to perform a scan, in case of a download and configure installation (on Linux/macOS) is:

```
path/to/scancode [OPTIONS] <OUTPUT FORMAT OPTION(s)> <SCAN INPUT>
```

The basic usage, if Scancode is installed from pip, or in Windows:

```
scancode [OPTIONS] <OUTPUT FORMAT OPTION(s)> <SCAN INPUT>
```

Here Scancode scans the <SCAN INPUT> file or directory for license, origin and packages and saves results to FILE(s) using one or more output format option. Error and progress are printed to stdout.

To scan the `samples` directory distributed with ScanCode-Toolkit, the command will be:

```
scancode -clpieu --json-pp path/to/output.json path/to/samples
```

Note: The <OUTPUT FORMAT OPTION(s)> includes both the output option and output file name. For example in the command `scancode -clpieu --json-pp output.json samples`, `--json-pp output.json` is <OUTPUT FORMAT OPTION(s)>.

Warning: There isn't a "Default" output option in Versions 3.x onwards, you have to specify <OUTPUT FORMAT OPTION(s)> explicitly.

Alternatively, in case of download and configure installations, where `path/to/scancode` is used (the path from root of file system) we can go into the scancode directory (like `scancode-toolkit-3.1.1`) and then use `./scancode`. The same applies for input and output options. To scan a folder `samples` inside ScanCode directory, and output to a file `output.json` in the same directory, the command will be:

```
./scancode -clpieu --json-pp output.json samples
```

While a scan using absolute paths from the file system root will look like:

```
home/ayansm/software/scancode-toolkit-3.1.1/scancode -clpieu --json-pp home/ayansm/scan_
↪scan_results/output.json home/ayansm/codebases/samples/
```

Commands similar to `scancode -clpi --json-pp output.json samples` will be used as examples throughout the documentation.

- Here we are inside the `virtualenv` where ScanCode-Toolkit is configured.
- And the default `samples` folder is being scanned, which is distributed by default with ScanCode-Toolkit.

Type of Options

ScanCode Toolkit Command Line options can be divided into these major sections:

- *All “Basic” Scan Options*
- *Extractcode Options*
- *All “Core” Scan Options*
- *Controlling Scancode Output and Filters*
- *Pre-Scan Options*
- *Post-Scan Options*

Refer the individual pages which are linked to above, for detailed discussions on the Command Line Options listed under each section.

Output Formats

The output file format is set by using the various output options. The recommended output format is JSON. If `--json` is used, the entire file being in one line, without whitespace characters.

The following example scans will show you how to run a scan with each of the result formats. For the scans, we will use the `samples` directory provided with the ScanCode Toolkit.

Tip: You can also output to `stdout` instead of a file. For more information refer *Print to stdout (Terminal)*.

JSON file output

Scan the `samples` directory and save the scan to a JSON file (pretty-printed)::

```
scancode -clpieu --json-pp output.json samples
```

A sample JSON output file structure will look like:

```
{
  "headers": [
    {
      "tool_name": "scancode-toolkit",
      "tool_version": "3.1.1",
      "options": {
        "input": [
          "samples/"
        ],
        "--copyright": true,
        "--email": true,
        "--info": true,
        "--json-pp": "output.json",
        "--license": true,
        "--package": true,
        "--url": true
      }
    },
  ],
}
```

(continues on next page)

(continued from previous page)

```

    "notice": "Generated with ScanCode and provided on an \"AS IS\" BASIS, WITHOUT
    ↳ WARRANTIES\\NOR CONDITIONS OF ANY KIND, either express or implied. No content created
    ↳ from\\nScanCode should be considered or used as legal advice. Consult an Attorney\\nfor
    ↳ any legal advice.\\nScanCode is a free software code scanning tool from nexB Inc. and
    ↳ others.\\nVisit https://github.com/nexB/scancode-toolkit/ for support and download.",
    "start_timestamp": "2019-10-19T191117.292858",
    "end_timestamp": "2019-10-19T191219.743133",
    "message": null,
    "errors": [],
    "extra_data": {
        "files_count": 36
    }
  },
  "files": [
    {
      "path": "samples",
      "type": "directory",
      ...
      ...
      ...
      "scan_errors": []
    },
    {
      "path": "samples/README",
      "type": "file",
      "name": "README",
      "base_name": "README",
      "extension": "",
      "size": 236,
      "date": "2019-02-12",
      "sha1": "2e07e32c52d607204fad196052d70e3d18fb8636",
      "md5": "effc6856ef85a9250fb1a470792b3f38",
      "mime_type": "text/plain",
      "file_type": "ASCII text",
      "programming_language": null,
      "is_binary": false,
      "is_text": true,
      "is_archive": false,
      "is_media": false,
      "is_source": false,
      "is_script": false,
      "licenses": [],
      "license_expressions": [],
      "copyrights": [],
      "holders": [],
      "authors": [],
      "packages": [],
      "emails": [],
      "urls": [],
      "files_count": 0,
      "dirs_count": 0,
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    "size_count": 0,
    "scan_errors": []
  },
  ...
  ...
  {
    "path": "samples/zlib/iostream2/zstream_test.cpp",
    "type": "file",
    "name": "zstream_test.cpp",
    "base_name": "zstream_test",
    "extension": ".cpp",
    "size": 711,
    "date": "2019-02-12",
    ...
    ...
    ...
    "scan_errors": []
  }
]
}

```

A sample JSON output for an individual file will look like:

```

{
  "path": "samples/zlib/iostream2/zstream.h",
  "type": "file",
  "name": "zstream.h",
  "base_name": "zstream",
  "extension": ".h",
  "size": 9283,
  "date": "2019-02-12",
  "sha1": "fca4540d490fff36bb90fd801cf9cd8fc695bb17",
  "md5": "a980b61c1e8be68d5cdb1236ba6b43e7",
  "mime_type": "text/x-c++",
  "file_type": "C++ source, ASCII text",
  "programming_language": "C++",
  "is_binary": false,
  "is_text": true,
  "is_archive": false,
  "is_media": false,
  "is_source": true,
  "is_script": false,
  "licenses": [
    {
      "key": "mit-old-style",
      "score": 100.0,
      "name": "MIT Old Style",
      "short_name": "MIT Old Style",
      "category": "Permissive",
      "is_exception": false,
      "is_unknown": false,
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    "owner": "MIT",
    "homepage_url": "http://fedoraproject.org/wiki/Licensing:MIT#Old_Style",
    "text_url": "http://fedoraproject.org/wiki/Licensing:MIT#Old_Style",
    "reference_url": "https://enterprise.dejacode.com/urn/urn:dje:license:mit-old-style
↪",
    "spdx_license_key": null,
    "spdx_url": null,
    "start_line": 9,
    "end_line": 15,
    "matched_rule": {
        "identifier": "mit-old-style_cmr-no_1.RULE",
        "license_expression": "mit-old-style",
        "licenses": [
            "mit-old-style"
        ],
        "is_license_text": true,
        "is_license_notice": false,
        "is_license_reference": false,
        "is_license_tag": false,
        "matcher": "2-aho",
        "rule_length": 71,
        "matched_length": 71,
        "match_coverage": 100.0,
        "rule_relevance": 100
    }
},
"license_expressions": [
    "mit-old-style"
],
"copyrights": [
    {
        "copyright": "Copyright (c) 1997 Christian Michelsen Research AS Advanced Computing
↪",
        "start_line": 3,
        "end_line": 5
    }
],
"holders": [
    {
        "holder": "Christian Michelsen Research AS Advanced Computing",
        "start_line": 3,
        "end_line": 5
    }
],
"authors": [],
"packages": [],
"emails": [],
"urls": [
    {
        "url": "http://www.cmr.no/",
        "start_line": 7,

```

(continues on next page)

(continued from previous page)

```

    "end_line": 7
  }
],
"files_count": 0,
"dirs_count": 0,
"size_count": 0,
"scan_errors": []
},

```

Static HTML output

Scan the `samples` directory for licenses and copyrights and save the scan results to an HTML file. When the scan is done, open `samples.html` in your web browser.

```
scancode -clpieu --html output.html samples
```

location	start	end	what	value
<code>samples/arch/zlib.tar.gz-extract/zlib-1.2.8/adler32.c</code>	1	3	copyright	Copyright (c) 1995-2011 Mark Adler
<code>samples/arch/zlib.tar.gz-extract/zlib-1.2.8/adler32.c</code>	3	3	license	zlib
<code>samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zlib.h</code>	4	4	copyright	Copyright (c) 1995-2013 Jean-loup Gailly and Mark Adler
<code>samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zlib.h</code>	6	20	license	zlib
<code>samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zutil.h</code>	1	3	copyright	Copyright (c) 1995-2013 Jean-loup Gailly.
<code>samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zutil.h</code>	3	3	license	zlib
<code>samples/JGroups/EULA</code>	3	108	license	jboss-eula
<code>samples/JGroups/EULA</code>	104	104	copyright	Copyright 2006 Red Hat, Inc.
<code>samples/JGroups/LICENSE</code>	1	502	license	lgpl-2.1-plus
<code>samples/JGroups/LICENSE</code>	4	7	copyright	Copyright (c) 1991, 1999 Free Software Foundation, Inc.
<code>samples/JGroups/LICENSE</code>	426	433	copyright	copyrighted by the Free Software Foundation
<code>samples/JGroups/licenses/apache-1.1.txt</code>	2	56	license	apache-1.1
<code>samples/JGroups/licenses/apache-1.1.txt</code>	4	5	copyright	Copyright (c) 2000 The Apache Software Foundation.
<code>samples/JGroups/licenses/apache-2.0.txt</code>	2	202	license	apache-2.0
<code>samples/JGroups/licenses/bouncycastle.txt</code>	5	5	copyright	Copyright (c) 2000 - 2006 The Legion Of The Bouncy Castle
<code>samples/JGroups/licenses/bouncycastle.txt</code>	7	18	license	mit
<code>samples/JGroups/licenses/cpl-1.0.txt</code>	1	1	license	cpl-1.0
<code>samples/JGroups/licenses/lgpl.txt</code>	1	502	license	lgpl-2.1-plus
<code>samples/JGroups/licenses/lgpl.txt</code>	4	7	copyright	Copyright (c) 1991, 1999 Free Software Foundation, Inc.
<code>samples/JGroups/licenses/lgpl.txt</code>	426	433	copyright	copyrighted by the Free Software Foundation
<code>samples/JGroups/src/FixedMembershipToken.java</code>	2	5	copyright	Copyright 2005, JBoss Inc.

Package Information

location	type	packaging	primary_language
<code>samples/arch/zlib.tar.gz</code>	plain tarball	archive	None

Licenses

key	short_name	category	owner	dejacode_url	homepage_url
apache-1.1	Apache 1.1	Attribution	Apache Software Foundation	https://enterprise.dejacode.com/license_library/Demo/apache-1.1/	http://www.apache.org/licenses/
apache-2.0	Apache 2.0	Attribution	Apache Software Foundation	https://enterprise.dejacode.com/license_library/Demo/apache-2.0/	http://www.apache.org/licenses/
boost-1.0	Boost 1.0	Attribution	Boost	https://enterprise.dejacode.com/license_library/Demo/boost-1.0/	http://www.boost.org/users/license.html
cc-by-2.5	CC-BY-2.5	Attribution	Creative Commons	https://enterprise.dejacode.com/license_library/Demo/cc-by-2.5/	http://creativecommons.org/licenses/by/2.5/
cmr-no	CMR License	Attribution	CMR - Christian Michelsen Research AS	https://enterprise.dejacode.com/license_library/Demo/cmr-no/	
cpl-1.0	CPL 1.0	Copyright Limited	IBM	https://enterprise.dejacode.com/license_library/Demo/cpl-1.0/	http://www.eclipse.org/legal/cpl-v10.html
gpl-2.0-plus-ada	GPL 2.0 or later with Ada exception	Copyright Limited	Dmitriy Anisimkov	https://enterprise.dejacode.com/license_library/Demo/gpl-2.0-plus-ada/	
jboss-eula	JBoss EULA	Proprietary Free	JBoss Community	https://enterprise.dejacode.com/license_library/Demo/jboss-eula/	
lgpl-2.1-plus	LGPL 2.1 or later	Copyright Limited	Free Software Foundation (FSF)	https://enterprise.dejacode.com/license_library/Demo/lgpl-2.1-plus/	http://www.gnu.org/licenses/old-licenses/lgpl-2.1-standalone.html

Other Important Documentation

1. *Type of Options*
2. *How to Run a Scan*
3. *Basic Tutorials*
4. *How-To Guides*
5. *Reference Docs*
6. *Contributing to Code Development*
7. *Contributing to the Documentation*
8. *Plugin Architecture*
9. *FAQ*
10. *Support*

3.1.2 Getting Help from the Command Line

ScanCode-Toolkit Command Line Interface can help you to search for specific options or use cases from the command line itself. These are two options are `--help` and `--examples`, and are very helpful if you need a quick glance of the options or use cases. Or it can be useful when you can't access, the more elaborate online documentation.

All Documentation/Help Options

-h, --help	Show the <i>Help text</i> and exit.
--examples	Show the <i>Command Examples Text</i> and exit.
-A, --about	Show information about ScanCode and licensing and exit.
-V, --version	Show the version and exit.
--list-packages	Show the list of supported package types and exit.
--plugins	Show the list of available ScanCode plugins and exit.
--print-options	Show the list of selected options and exit.

Help text

The Scancode-Toolkit Command Line Interface has a Help option displaying all the options. It also displays basic usage, and some simple examples. The command line option for this is `--help`.

Tip: You can also use the shorter `-h` option, which does the same.

To see the help text from the Terminal, execute the following command:

```
$ scancode --help
```

The Following Help Text is displayed, i.e. This is the help text for Scancode Version 32.0.0:

```
Usage: scancode [OPTIONS] <OUTPUT FORMAT OPTION(s)> <input>...

scan the <input> file or directory for license, origin and packages and save
results to FILE(s) using one or more output format option.

Error and progress are printed to stderr.

Options:

primary scans:
  -l, --license      Scan <input> for licenses.
  -p, --package      Scan <input> for application package and dependency
                    manifests, lockfiles and related data.
  --system-package  Scan <input> for installed system package databases.
  -c, --copyright    Scan <input> for copyrights.

other scans:
  -i, --info         Scan <input> for file information (size, checksums, etc).
  --generated        Classify automatically generated code files with a flag.
```

(continues on next page)

(continued from previous page)

```
-e, --email  Scan <input> for emails.
-u, --url    Scan <input> for urls.
```

scan options:

```
--license-diagnostics  In license detections, include diagnostic details
                        to figure out the license detection post
                        processing steps applied.
--license-score INTEGER Do not return license matches with a score lower
                        than this score. A number between 0 and 100.
                        [default: 0]
--license-text          Include the detected licenses matched text.
--license-text-diagnostics In the matched license text, include diagnostic
                        highlights surrounding with square brackets []
                        words that are not matched.
--license-url-template TEXT Set the template URL used for the license
                        reference URLs. Curly braces ({} ) are replaced by
                        the license key. [default: https://scancode-
                        licensedb.aboutcode.org/{}]
--max-email INT         Report only up to INT emails found in a file. Use
                        0 for no limit. [default: 50]
--max-url INT           Report only up to INT urls found in a file. Use 0
                        for no limit. [default: 50]
--unknown-licenses      [EXPERIMENTAL] Detect unknown licenses.
```

output formats:

```
--json FILE            Write scan output as compact JSON to FILE.
--json-pp FILE         Write scan output as pretty-printed JSON to FILE.
--json-lines FILE      Write scan output as JSON Lines to FILE.
--yaml FILE            Write scan output as YAML to FILE.
--csv FILE             [DEPRECATED] Write scan output as CSV to FILE. The
--csv option is deprecated and will be replaced by new
                        CSV and tabular output formats in the next ScanCode
                        release. Visit https://github.com/nexB/scancode-
                        toolkit/issues/3043 to provide inputs and feedback.
--html FILE            Write scan output as HTML to FILE.
--custom-output FILE   Write scan output to FILE formatted with the custom
                        Jinja template file.
--debian FILE          Write scan output in machine-readable Debian copyright
                        format to FILE.
--custom-template FILE Use this Jinja template FILE as a custom template.
--cyclonedx FILE       Write scan output in CycloneDX JSON format to FILE.
--cyclonedx-xml FILE   Write scan output in CycloneDX XML format to FILE.
--spdx-rdf FILE        Write scan output as SPDX RDF to FILE.
--spdx-tv FILE         Write scan output as SPDX Tag/Value to FILE.
--html-app FILE        (DEPRECATED: use the ScanCode Workbench app instead)
                        Write scan output as a mini HTML application to FILE.
```

output filters:

```
--ignore-author <pattern> Ignore a file (and all its findings) if an
                        author contains a match to the <pattern>
                        regular expression. Note that this will ignore
                        a file even if it has other findings such as a
```

(continues on next page)

(continued from previous page)

```

                                license or errors.
--ignore-copyright-holder <pattern> Ignore a file (and all its findings) if a
                                copyright holder contains a match to the
                                <pattern> regular expression. Note that this
                                will ignore a file even if it has other
                                scanned data such as a license or errors.
--only-findings                Only return files or directories with findings
                                for the requested scans. Files and directories
                                without findings are omitted (file information
                                is not treated as findings).

output control:
--full-root    Report full, absolute paths.
--strip-root   Strip the root directory segment of all paths. The default is to
                always include the last directory segment of the scanned path
                such that all paths have a common root directory.

pre-scan:
--ignore <pattern>          Ignore files matching <pattern>.
--include <pattern>        Include files matching <pattern>.
--classify           Classify files with flags indicating whether the file is a
                        legal, readme, test or similar file.
--facet <facet>=<pattern> Add the <facet> to files with a path matching
                        <pattern>.

post-scan:
--consolidate          Group resources by Packages or license and copyright
                        holder and return those groupings as a list of
                        consolidated packages and a list of consolidated
                        components. This requires the scan to have/be run
                        with the copyright, license, and package options
                        active
--filter-clues          Filter redundant duplicated clues already contained
                        in detected license and copyright texts and notices.
--license-clarity-score Compute a summary license clarity score at the
                        codebase level.
--license-policy FILE   Load a License Policy file and apply it to the scan
                        at the Resource level.
--license-references    Return reference data for all licenses and license
                        rules present in detections.
--mark-source           Set the "is_source" to true for directories that
                        contain over 90% of source files as children and
                        descendants. Count the number of source files in a
                        directory as a new source_file_counts attribute
--summary              Summarize scans by providing declared origin
                        information and other detected origin info at the
                        codebase attribute level.
--tallies              Compute tallies for license, copyright and other
                        scans at the codebase level.
--tallies-by-facet     Compute tallies for license, copyright and other
                        scans and group the results by facet.
--tallies-key-files    Compute tallies for license, copyright and other

```

(continues on next page)

(continued from previous page)

```

scans for key, top-level files. Key files are top-
level codebase files such as COPYING, README and
package manifests as reported by the --classify
option "is_legal", "is_readme", "is_manifest" and
"is_top_level" flags.
--tallies-with-details Compute tallies of license, copyright and other scans
at the codebase level, keeping intermediate details
at the file and directory level.

core:
--timeout <seconds> Stop an unfinished file scan after a timeout in
seconds. [default: 120 seconds]
-n, --processes INT Set the number of parallel processes to use. Disable
parallel processing if 0. Also disable threading if
-1. [default: 1]
-q, --quiet Do not print summary or progress.
-v, --verbose Print progress as file-by-file path instead of a
progress bar. Print verbose scan counters.
--from-json Load codebase from one or more <input> JSON scan
file(s).
--max-in-memory INTEGER Maximum number of files and directories scan details
kept in memory during a scan. Additional files and
directories scan details above this number are cached
on-disk rather than in memory. Use 0 to use unlimited
memory and disable on-disk caching. Use -1 to use
only on-disk caching. [default: 10000]
--max-depth INTEGER Maximum nesting depth of subdirectories to scan.
Descend at most INTEGER levels of directories below
and including the starting directory. Use 0 for no
scan depth limit.

documentation:
-h, --help Show this message and exit.
-A, --about Show information about ScanCode and licensing and exit.
-V, --version Show the version and exit.
--examples Show command examples and exit.
--list-packages Show the list of supported package manifest parsers and exit.
--plugins Show the list of available ScanCode plugins and exit.
--print-options Show the list of selected options and exit.

```

Examples (use --examples for more):

Scan the 'samples' directory for licenses and copyrights.
 Save scan results to the 'scancode_result.json' JSON file:

```
scancode --license --copyright --json-pp scancode_result.json samples
```

Scan the 'samples' directory for licenses and package manifests. Print scan
 results on screen as pretty-formatted JSON (using the special '-' FILE to print
 to on screen/to stdout):

```
scancode --json-pp - --license --package samples
```

(continues on next page)

(continued from previous page)

Note: when you run ScanCode, a progress bar is displayed with a counter of the number of files processed. Use `--verbose` to display file-by-file progress.

Command Examples Text

The Scancode-Toolkit Command Line Interface has an `--examples` option which displays some basic examples (more than the basic synopsis in `--help`). These examples include the following aspects of code scanning:

- Scanning Single File/Directory
- Output Scan results to stdout (as JSON) or HTML/JSON file
- Scanning for only Copyrights/Licenses
- Ignoring Files
- Using GLOB Patterns to Scan Multiple Files
- Using Verbose Mode

The command line option for displaying these basic examples is `--examples`.

To see the help text from the Terminal, execute the following command:

```
$ scancode --examples
```

The Following Text is displayed, i.e. This is the examples for Scancode Version 3.1.1

Scancode command lines examples:

(Note for Windows: use `'\'` back slash instead of `'/'` forward slash for paths.)

Scan a single file for copyrights. Print scan results to stdout as pretty JSON:

```
scancode --copyright samples/zlib/zlib.h --json-pp -
```

Scan a single file for licenses, print verbose progress to stderr as each file is scanned. Save scan to a JSON file:

```
scancode --license --verbose samples/zlib/zlib.h --json licenses.json
```

Scan a directory explicitly for licenses and copyrights. Redirect JSON scan results to a file:

```
scancode --license --copyright samples/zlib/ --json - > scan.json
```

Scan a directory while ignoring a single file. Scan for license, copyright and package manifests. Use four parallel processes. Print scan results to stdout as pretty formatted JSON.

```
scancode -lc --package --ignore README --processes 4 --json-pp - samples/
```

Scan a directory while ignoring all files with `.txt` extension. Print scan results to stdout as pretty formatted JSON.

(continues on next page)

(continued from previous page)

It is recommended to use quotes around glob patterns to prevent pattern expansion by the shell:

```
scancode --json-pp - --ignore "*.txt" samples/
```

Special characters supported in GLOB pattern:

- * matches everything
- ? matches any single character
- [seq] matches any character in seq
- [!seq] matches any character not in seq

For a literal match, wrap the meta-characters in brackets.

For example, '[?]' matches the character '?'.

For details on GLOB patterns see [https://en.wikipedia.org/wiki/Glob_\(programming\)](https://en.wikipedia.org/wiki/Glob_(programming)).

Note: Glob patterns cannot be applied to path as strings.

For example, this will not ignore "samples/JGroups/licenses".

```
scancode --json - --ignore "samples*licenses" samples/
```

Scan a directory while ignoring multiple files (or glob patterns).

Print the scan results to stdout as JSON:

```
scancode --json - --ignore README --ignore "*.txt" samples/
```

Scan a directory for licenses and copyrights. Save scan results to an HTML file:

```
scancode --license --copyright --html scancode_result.html samples/zlib
```

To extract archives, see the 'extractcode' command instead.

Plugins Help Text

The command line option for displaying all the plugins is:

- --plugins

To see the help text from the Terminal, execute the following command:

```
$ scancode --plugins
```

Note: Plugins that are shown by using --plugins include the following:

1. Post-Scan Plugins
2. Pre-Scan Plugins
3. Output Options
4. Output Control
5. Basic Scan Options

The Following Text is displayed, i.e. This is the available plugins for Scancode Version 31.2.1

```

-----
Plugin: scancode_output:csv  class: formattedcode.output_csv:CsvOutput
  codebase_attributes:
  resource_attributes:
  sort_order: 100
  required_plugins:
  options:
    help_group: output formats, name: csv: --csv
    help: [DEPRECATED] Write scan output as CSV to FILE. The --csv option is
    ↪ deprecated and will be replaced by new CSV and tabular output formats in the next
    ↪ ScanCode release. Visit https://github.com/nexB/scancode-toolkit/issues/3043 to
    ↪ provide inputs and feedback.
  doc: None
-----

Plugin: scancode_output:cyclonedx  class: formattedcode.output_
    ↪ cyclonedx:CycloneDxJsonOutput
  codebase_attributes:
  resource_attributes:
  sort_order: 100
  required_plugins:
  options:
    help_group: output formats, name: output_cyclonedx_json: --cyclonedx
    help: Write scan output in CycloneDX JSON format to FILE.
  doc:
    Output plugin to write scan results in CycloneDX JSON format.
    For additional information on the format,
    please see https://cyclonedx.org/specification/overview/
-----

Plugin: scancode_output:cyclonedx-xml  class: formattedcode.output_
    ↪ cyclonedx:CycloneDxXmlOutput
  codebase_attributes:
  resource_attributes:
  sort_order: 100
  required_plugins:
  options:
    help_group: output formats, name: output_cyclonedx_xml: --cyclonedx-xml
    help: Write scan output in CycloneDX XML format to FILE.
  doc:
    Output plugin to write scan results in CycloneDX XML format.
    For additional information on the format,
    please see https://cyclonedx.org/specification/overview/
-----

Plugin: scancode_output:debian  class: formattedcode.output_debian:DebianCopyrightOutput
  codebase_attributes:
  resource_attributes:

```

(continues on next page)

(continued from previous page)

```

sort_order: 100
required_plugins:
options:
  help_group: output formats, name: output_debian: --debian
  help: Write scan output in machine-readable Debian copyright format to FILE.
doc: None
-----
Plugin: scancode_output:html  class: formattedcode.output_html:HtmlOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: html: --html
  help: Write scan output as HTML to FILE.
doc: None
-----
Plugin: scancode_output:html-app  class: formattedcode.output_html:HtmlAppOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: html_app: --html-app
  help: (DEPRECATED: use the ScanCode Workbench app instead)
  Write scan output as a mini HTML application to FILE.
doc:
  Write scan output as a mini HTML application.
-----
Plugin: scancode_output:json  class: formattedcode.output_json:JsonCompactOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: output_json: --json
  help: Write scan output as compact JSON to FILE.
doc: None
-----
Plugin: scancode_output:json-pp  class: formattedcode.output_json:JsonPrettyOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: output_json_pp: --json-pp
  help: Write scan output as pretty-printed JSON to FILE.

```

(continues on next page)

(continued from previous page)

doc: **None**

```

-----
Plugin: scancode_output:jsonlines  class: formattedcode.output_jsonlines:JsonLinesOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: output_json_lines: --json-lines
  help: Write scan output as JSON Lines to FILE.
doc: None

```

```

-----
Plugin: scancode_output:spdx-rdf  class: formattedcode.output_spdx:SpdxRdfOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: spdx_rdf: --spdx-rdf
  help: Write scan output as SPDX RDF to FILE.
doc: None

```

```

-----
Plugin: scancode_output:spdx-tv  class: formattedcode.output_spdx:SpdxTvOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: spdx_tv: --spdx-tv
  help: Write scan output as SPDX Tag/Value to FILE.
doc: None

```

```

-----
Plugin: scancode_output:template  class: formattedcode.output_html:CustomTemplateOutput
codebase_attributes:
resource_attributes:
sort_order: 100
required_plugins:
options:
  help_group: output formats, name: custom_output: --custom-output
  help: Write scan output to FILE formatted with the custom Jinja template file.
  help_group: output formats, name: custom_template: --custom-template
  help: Use this Jinja template FILE as a custom template.
doc: None

```

```

-----
Plugin: scancode_output:yaml  class: formattedcode.output_yaml:YamlOutput
codebase_attributes:
resource_attributes:

```

(continues on next page)

(continued from previous page)

```

sort_order: 100
required_plugins:
options:
  help_group: output formats, name: output_yaml: --yaml
  help: Write scan output as YAML to FILE.
doc: None
-----
Plugin: scancode_output_filter:ignore-copyrights  class: cluecode.plugin_ignore_
↳ copyrights:IgnoreCopyrights
  codebase_attributes:
  resource_attributes:
  sort_order: 100
  required_plugins:
  options:
    help_group: output filters, name: ignore_copyright_holder: --ignore-copyright-holder
    help: Ignore a file (and all its findings) if a copyright holder contains a match_
↳ to the <pattern> regular expression. Note that this will ignore a file even if it has_
↳ other scanned data such as a license or errors.
    help_group: output filters, name: ignore_author: --ignore-author
    help: Ignore a file (and all its findings) if an author contains a match to the
↳ <pattern> regular expression. Note that this will ignore a file even if it has other_
↳ findings such as a license or errors.
  doc:
    Filter findings that match given copyright holder or author patterns.
    Has no effect unless the --copyright scan is requested.
-----
Plugin: scancode_output_filter:only-findings  class: scancode.plugin_only_
↳ findings:OnlyFindings
  codebase_attributes:
  resource_attributes:
  sort_order: 100
  required_plugins:
  options:
    help_group: output filters, name: only_findings: --only-findings
    help: Only return files or directories with findings for the requested scans_
↳ Files and directories without findings are omitted (file information is not treated as_
↳ findings).
  doc:
    Filter files or directories without scan findings for the requested scans.
-----
Plugin: scancode_post_scan:consolidate  class: summarycode.plugin_
↳ consolidate:Consolidator
  codebase_attributes: consolidated_components, consolidated_packages
  resource_attributes: consolidated_to
  sort_order: 10
  required_plugins:
  options:

```

(continues on next page)

(continued from previous page)

```

    help_group: post-scan, name: consolidate: --consolidate
    help: Group resources by Packages or license and copyright holder and return those.
↳ groupings as a list of consolidated packages and a list of consolidated components.
↳ This requires the scan to have/be run with the copyright, license, and package options.
↳ active
doc:
    A ScanCode post-scan plugin to return consolidated components and consolidated
    packages for different types of codebase summarization.

    A consolidated component is a group of Resources that have the same origin.
    Currently, a ConsolidatedComponent is created for each detected copyright holder
    in a codebase and contains resources that have that particular copyright holder.

    A consolidated package is a detected package in the scanned codebase that has
    been enhanced with data about other licenses and holders found within it.

    If a Resource is part of a consolidated component or consolidated package, then
    the identifier of the consolidated component or consolidated package it is part
    of is in the Resource's ``consolidated_to`` field.

-----
Plugin: scancode_post_scan:filter-clues  class: cluecode.plugin_filter_
↳ clues:RedundantCluesFilter
  codebase_attributes:
  resource_attributes:
  sort_order: 1
  required_plugins:
  options:
    help_group: post-scan, name: filter_clues: --filter-clues
    help: Filter redundant duplicated clues already contained in detected license and
↳ copyright texts and notices.
  doc:
    Filter redundant clues (copyrights, authors, emails, and urls) that are
    already contained in a matched license text.

-----
Plugin: scancode_post_scan:license-clarity-score  class: summarycode.
↳ score:LicenseClarityScore
  codebase_attributes: summary
  resource_attributes:
  sort_order: 5
  required_plugins:
  options:
    help_group: post-scan, name: license_clarity_score: --license-clarity-score
    help: Compute a summary license clarity score at the codebase level.
  doc:
    Compute a License clarity score at the codebase level.

-----

```

(continues on next page)

(continued from previous page)

```

Plugin: scancode_post_scan:license-policy  class: licensedcode.plugin_license_
↪policy:LicensePolicy
  codebase_attributes:
  resource_attributes: license_policy
  sort_order: 9
  required_plugins:
  options:
    help_group: post-scan, name: license_policy: --license-policy
    help: Load a License Policy file and apply it to the scan at the Resource level.
  doc:
    Add the "license_policy" attribute to a resource if it contains a
    detected license key that is found in the license_policy.yml file

-----
Plugin: scancode_post_scan:license-references  class: licensedcode.licenses_
↪reference:LicenseReference
  codebase_attributes: license_references, license_rule_references
  resource_attributes:
  sort_order: 1000
  required_plugins:
  options:
    help_group: post-scan, name: license_references: --license-references
    help: Return reference data for all licenses and license rules present in_
↪detections.
  doc:
    Add license and rule reference data to a scan.

-----
Plugin: scancode_post_scan:mark-source  class: scancode.plugin_mark_source:MarkSource
  codebase_attributes:
  resource_attributes: source_count
  sort_order: 8
  required_plugins:
  options:
    help_group: post-scan, name: mark_source: --mark-source
    help: Set the "is_source" to true for directories that contain over 90% of source_
↪files as children and descendants. Count the number of source files in a directory as_
↪a new source_file_counts attribute
  doc:
    Set the "is_source" flag to true for directories that contain
    over 90% of source files as direct children.
    Has no effect unless the --info scan is requested.

-----
Plugin: scancode_post_scan:summary  class: summarycode.summarizer:ScanSummary
  codebase_attributes: summary
  resource_attributes:
  sort_order: 2
  required_plugins:

```

(continues on next page)

(continued from previous page)

```

options:
  help_group: post-scan, name: summary: --summary
  help: Summarize scans by providing declared origin information and other detected_
↳origin info at the codebase attribute level.
doc:
  Summarize a scan at the codebase level.

-----

Plugin: scancode_post_scan:tallies class: summarycode.tallies:Tallies
codebase_attributes: tallies
resource_attributes:
sort_order: 15
required_plugins:
options:
  help_group: post-scan, name: tallies: --tallies
  help: Compute tallies for license, copyright and other scans at the codebase level.
doc:
  Compute tallies for license, copyright and other scans at the codebase level

-----

Plugin: scancode_post_scan:tallies-by-facet class: summarycode.tallies:FacetTallies
codebase_attributes: tallies_by_facet
resource_attributes:
sort_order: 200
required_plugins:
options:
  help_group: post-scan, name: tallies_by_facet: --tallies-by-facet
  help: Compute tallies for license, copyright and other scans and group the results_
↳by facet.
doc:
  Compute tallies for a scan at the codebase level, grouping by facets.

-----

Plugin: scancode_post_scan:tallies-key-files class: summarycode.tallies:KeyFilesTallies
codebase_attributes: tallies_of_key_files
resource_attributes:
sort_order: 150
required_plugins:
options:
  help_group: post-scan, name: tallies_key_files: --tallies-key-files
  help: Compute tallies for license, copyright and other scans for key, top-level_
↳files. Key files are top-level codebase files such as COPYING, README and package_
↳manifests as reported by the --classify option "is_legal", "is_readme", "is_manifest"_
↳and "is_top_level" flags.
doc:
  Compute tallies of a scan at the codebase level for only key files.

-----

```

(continues on next page)

(continued from previous page)

```

Plugin: scancode_post_scan:tallies-with-details class: summarycode.
↳tallies:TalliesWithDetails
  codebase_attributes: tallies
  resource_attributes: tallies
  sort_order: 100
  required_plugins:
  options:
    help_group: post-scan, name: tallies_with_details: --tallies-with-details
    help: Compute tallies of license, copyright and other scans at the codebase level,
↳keeping intermediate details at the file and directory level.
  doc:
    Compute tallies of different scan attributes of a scan at the codebase level and
    keep file and directory details.

    The scan attributes that are tallied are:
    - detected_license_expression
    - copyrights
    - holders
    - authors
    - programming_language
    - packages
-----

Plugin: scancode_pre_scan:classify class: summarycode.classify_plugin:FileClassifier
  codebase_attributes:
  resource_attributes: is_legal, is_manifest, is_readme, is_top_level, is_key_file
  sort_order: 30
  required_plugins:
  options:
    help_group: pre-scan, name: classify: --classify
    help: Classify files with flags telling if the file is a legal, or readme or test,
↳file, etc.
  doc:
    Classify a file such as a COPYING file or a package manifest with a flag.
-----

Plugin: scancode_pre_scan:facet class: summarycode.facet:AddFacet
  codebase_attributes:
  resource_attributes: facets
  sort_order: 20
  required_plugins:
  options:
    help_group: pre-scan, name: facet: --facet
    help: Add the <facet> to files with a path matching <pattern>.
  doc:
    Assign one or more "facet" to each file (and NOT to directories). Facets are
    a way to qualify that some part of the scanned code may be core code vs.
    test vs. data, etc.

```

(continues on next page)

(continued from previous page)

```

-----
Plugin: scancode_pre_scan:ignore  class: scancode.plugin_ignore:ProcessIgnore
  codebase_attributes:
  resource_attributes:
  sort_order: 100
  required_plugins:
  options:
    help_group: pre-scan, name: ignore: --ignore
    help: Ignore files matching <pattern>.
    help_group: pre-scan, name: include: --include
    help: Include files matching <pattern>.
  doc:
    Include or ignore files matching patterns.

```

```

-----
Plugin: scancode_scan:copyrights  class: cluecode.plugin_copyright:CopyrightScanner
  codebase_attributes:
  resource_attributes: copyrights, holders, authors
  sort_order: 6
  required_plugins:
  options:
    help_group: primary scans, name: copyright: -c, --copyright
    help: Scan <input> for copyrights.
  doc:
    Scan a Resource for copyrights.

```

```

-----
Plugin: scancode_scan:emails  class: cluecode.plugin_email:EmailScanner
  codebase_attributes:
  resource_attributes: emails
  sort_order: 7
  required_plugins:
  options:
    help_group: other scans, name: email: -e, --email
    help: Scan <input> for emails.
    help_group: scan options, name: max_email: --max-email
    help: Report only up to INT emails found in a file. Use 0 for no limit.
  doc:
    Scan a Resource for emails.

```

```

-----
Plugin: scancode_scan:generated  class: summarycode.generated:GeneratedCodeDetector
  codebase_attributes:
  resource_attributes: is_generated
  sort_order: 50
  required_plugins:
  options:
    help_group: other scans, name: generated: --generated
    help: Classify automatically generated code files with a flag.

```

(continues on next page)

(continued from previous page)

```

doc:
    Tag a file as generated.

-----

Plugin: scancode_scan:info  class: scancode.plugin_info:InfoScanner
codebase_attributes:
resource_attributes: date, sha1, md5, sha256, mime_type, file_type, programming_
↪ language, is_binary, is_text, is_archive, is_media, is_source, is_script
sort_order: 0
required_plugins:
options:
    help_group: other scans, name: info: -i, --info
    help: Scan <input> for file information (size, checksums, etc).
doc:
    Scan a file Resource for miscellaneous information such as mime/filetype and
    basic checksums.

-----

Plugin: scancode_scan:licenses  class: licensedcode.plugin_license:LicenseScanner
codebase_attributes: license_detections
resource_attributes: detected_license_expression, detected_license_expression_spdx, ↪
↪ license_detections, license_clues, percentage_of_license_text
sort_order: 4
required_plugins:
options:
    help_group: primary scans, name: license: -l, --license
    help: Scan <input> for licenses.
    help_group: scan options, name: license_score: --license-score
    help: Do not return license matches with a score lower than this score. A number ↪
↪ between 0 and 100.
    help_group: scan options, name: license_text: --license-text
    help: Include the detected licenses matched text.
    help_group: scan options, name: license_text_diagnostics: --license-text-diagnostics
    help: In the matched license text, include diagnostic highlights surrounding with ↪
↪ square brackets [] words that are not matched.
    help_group: scan options, name: license_diagnostics: --license-diagnostics
    help: In license detections, include diagnostic details to figure out the license ↪
↪ detection post processing steps applied.
    help_group: scan options, name: license_url_template: --license-url-template
    help: Set the template URL used for the license reference URLs. Curly braces ({}). ↪
↪ are replaced by the license key.
    help_group: scan options, name: unknown_licenses: --unknown-licenses
    help: [EXPERIMENTAL] Detect unknown licenses.
doc:
    Scan a Resource for licenses.

-----

Plugin: scancode_scan:packages  class: packagedcode.plugin_package:PackageScanner
codebase_attributes: packages, dependencies

```

(continues on next page)

(continued from previous page)

```

resource_attributes: package_data, for_packages
sort_order: 3
required_plugins: scan:licenses
options:
  help_group: primary scans, name: package: -p, --package
  help: Scan <input> for application package and dependency manifests, lockfiles and
↳ related data.
  help_group: primary scans, name: system_package: --system-package
  help: Scan <input> for installed system package databases.
  help_group: documentation, name: list_packages: --list-packages
  help: Show the list of supported package manifest parsers and exit.
doc:
  Scan a Resource for Package data and report these as "package_data" at the
  file level. Then create "packages" from these "package_data" at the top
  level.

-----
Plugin: scancode_scan:urls  class: cluecode.plugin_url:UrlScanner
codebase_attributes:
resource_attributes: urls
sort_order: 8
required_plugins:
options:
  help_group: other scans, name: url: -u, --url
  help: Scan <input> for urls.
  help_group: scan options, name: max_url: --max-url
  help: Report only up to INT urls found in a file. Use 0 for no limit.
doc:
  Scan a Resource for URLs.

```

--list-packages Option

This shows all the types of packages that can be scanned using Scancode. These are located in packagedcode i.e. Code used to parse various package formats.

See the [Supported package manifests and package datafiles](#) page for more details and documentation automatically generated using this data.

--print-options Option

This option prints the options selected for one specific scan command.

If we run this command:

```

scancode -clpieu --json-pp sample.json samples --classify --tallies --tallies-with-
↳ details --print-options

```

The output will be:

```
Options:
  classify: True
  copyright: True
  email: True
  info: True
  license: True
  list_packages: None
  output_json_pp: <unopened file 'sample.json' wb>
  package: True
  reindex_licenses: None
  tallies: True
  tallies_with_details: True
  url: True
```

3.1.3 All Available Options

This section contains an exhaustive list of all Scancode options, arranged in various sections. The sections are as follows:

- Basic Scan Options
- Core Scan Options
- Output Formats
- Controlling Output and Filters
- Pre-Scan Options
- Post-Scan Options

There's also another section for `extractcode` options.

The order of the sections and all their options is the same as in the *Help text*, available in the command line.

All “Basic” Scan Options

Option lists are two-column lists of command-line options and descriptions, documenting a program's options. For example:

-c, --copyright	Scan <input> for copyrights.
	Sub-Options:
	<ul style="list-style-type: none"> • --consolidate
-l, --license	Scan <input> for licenses.
	Sub-Options:
	<ul style="list-style-type: none"> • --license-references • --license-text • --license-text-diagnostics • --license-diagnostics • --license-url-template TEXT • --license-score INT

	<ul style="list-style-type: none"> • <code>--license-clarity-score</code> • <code>--consolidate</code> • <code>--unknown-licenses</code>
-p, --package	Scan <input> for packages. Sub-Options: <ul style="list-style-type: none"> • <code>--consolidate</code>
--system-package	Scan <input> for installed system package databases.
--package-only	Scan <input> for system and application only for package metadata, without license/ copyright detection and package assembly.
-e, --email	Scan <input> for emails. Sub-Options: <ul style="list-style-type: none"> • <code>--max-email INT</code>
-u, --url	Scan <input> for urls. Sub-Options: <ul style="list-style-type: none"> • <code>--max-url INT</code>
-i, --info	Scan for and include information such as: <ul style="list-style-type: none"> • Size, • Type, • Date, • Programming language, • sha1 and md5 hashes, • binary/text/archive/media/source/script flags • Additional options through more CLI options Sub-Options: <ul style="list-style-type: none"> • <code>--mark-source</code>

Note: Unlike previous 2.x versions, `-c`, `-l`, and `-p` are not default. If any combination of these options are used, ScanCode performs only that specific task, and not the others. `scancode -l` scans only for licenses, and doesn't scan for copyright/packages/general information/emails/urls. The only notable exception: a `--package` scan also has license information for package manifests and top-level packages, which are derived regardless of `--license` option being used.

Note: These options, i.e. `-c`, `-l`, `-p`, `-e`, `-u`, and `-i` can be used together. As in, instead of `scancode -c -i -p`, you can write `scancode -cip` and it will be the same.

--generated	Classify automatically generated code files with a flag.
--max-email INT	Report only up to INT emails found in a file. Use 0 for no limit. [Default: 50] Sub-Option of: <code>--email</code>

-
- max-url INT** Report only up to INT urls found in a file. Use 0 for no limit. [Default: 50]
Sub-Option of: `--url`
- license-score INTEGER** Do not return license matches with scores lower than this score. A number between 0 and 100. [Default: 0] Here, a bigger number means a better match, i.e. Setting a higher license score translates to a higher threshold (with equal or smaller number of matches).
Sub-Option of: `--license`
- license-text** Include the matched text for the detected licenses in the output report.
Sub-Option of: `--license`
Sub-Options:
- `--license-text-diagnostics`
- license-url-template TEXT** Set the template URL used for the license reference URLs.
In a template URL, curly braces ({}) are replaced by the license key. [Default: default: <https://scancode-licensedb.aboutcode.org/{}>]
Sub-Option of: `--license`
- license-text-diagnostics** In the matched license text, include diagnostic highlights surrounding with square brackets [] words that are not matched.
Sub-Option of: `--license` and `--license-text`
- license-diagnostics** In license detections, include diagnostic details to figure out the license detection post processing steps applied.
Sub-Option of: `--license`
- unknown-licenses** [EXPERIMENTAL] Detect unknown licenses.
Sub-Option of: `--license`
-

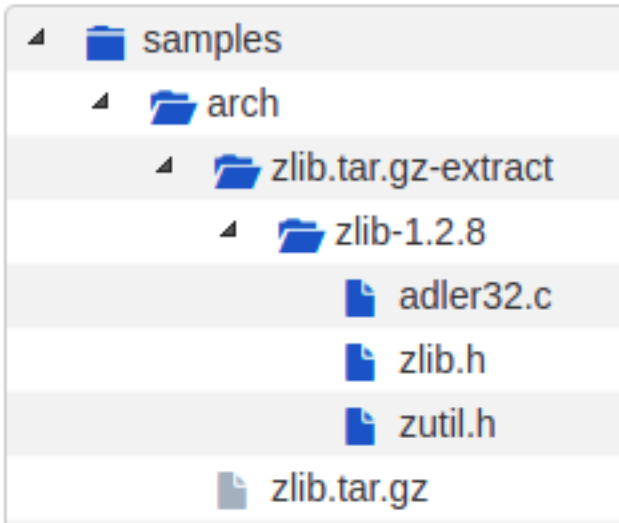
All Extractcode Options

This is intended to be used as an input preparation step, before running the scan. Archives found in an extracted archive are extracted **recursively** by default. Extraction is done in-place in a directory named ‘-extract’ side-by-side with an archive.

To extract the packages in the `samples` directory

```
extractcode samples
```

This extracts the `zlib.tar.gz` package:



--shallow	Do not extract recursively nested archives (e.g. Not archives in archives).
--verbose	Print verbose file-by-file progress messages.
--quiet	Do not print any summary or progress message.
-h, --help	Show the extractcode help message and exit.
--about	Show information about ScanCode and licensing and exit.
--version	Show the version and exit.

scancode-reindex-licenses Usage

Usage: scancode-reindex-licenses [OPTIONS]

Reindex scancode licenses and exit

Options

--all-languages	[EXPERIMENTAL] Rebuild the license index including texts all languages (and not only English) and exit.
--only-builtin	Rebuild the license index excluding any additional license directory or additional license plugins which were added previously, i.e. with only builtin scancode license and rules.
--additional-directory DIR	Include this directory with additional custom licenses and license rules in the license detection index.
--load-dump	Load all license and rules from their respective files and then dump them back to those same files.
-h, --help	Shows the options and explanations.

All “Core” Scan Options

- n, --processes INTEGER** Scan <input> using n parallel processes. [Default: 1]
- v, --verbose** Print verbose file-by-file progress messages.
- q, --quiet** Do not print summary or progress messages.
- timeout FLOAT** Stop scanning a file if scanning takes longer than a timeout in seconds. [Default: 120]
- from-json** Load codebase from one or more existing JSON scans.
- max-in-memory INTEGER** Maximum number of files and directories scan details kept in memory during a scan. Additional files and directories scan details above this number are cached on-disk rather than in memory. Use 0 to use unlimited memory and disable on-disk caching. Use -1 to use only on-disk caching. [Default: 10000]
- max-depth INTEGER** Descend at most INTEGER levels of directories including and below the starting point. INTEGER must be positive or zero for no limit. [Default: 0]

All Scan Output Options

- json FILE** Write scan output as compact JSON to FILE.
- json-pp FILE** Write scan output as pretty-printed JSON to FILE. This is one of the recommended output formats and contains all the data scancode can show along with the YAML output format.
- json-lines FILE** Write scan output as JSON Lines to FILE.
- yaml FILE** Write scan output as YAML to FILE. This is one of the recommended output formats and contains all the data scancode can show along with the JSON output format.
- csv FILE** DEPRECATED: Write scan output as CSV to FILE. This option is deprecated and will be replaced by new CSV and tabular output formats in the next ScanCode release. Visit this issue for details, and to provide input and feedback: <https://github.com/nexB/scancode-toolkit/issues/3043>
- html FILE** Write scan output as HTML to FILE.
- custom-output** Write scan output to FILE formatted with the custom Jinja template file.
Mandatory Sub-option:
 - **--custom-template FILE**
- custom-template FILE** Use this Jinja template FILE as a custom template.
Sub-Option of: **--custom-output**
- debian FILE** Write scan output in machine-readable Debian copyright format to FILE.
- spdx-rdf FILE** Write scan output as SPDX RDF to FILE.
- spdx-tv FILE** Write scan output as SPDX Tag/Value to FILE.
- html-app FILE** [DEPRECATED] Use `scancode-workbench` instead. Write scan output as a mini HTML application to FILE.

- cyclonedx FILE** Write scan output as a CycloneDx 1.3 BOM in pretty-printed JSON format to FILE
- cyclonedx-xml FILE** Write scan output as a CycloneDx 1.3 BOM in pretty-printed XML format to FILE

Warning: The html-app feature has been deprecated and you should use Scancode Workbench instead to visualize scan results. The official Repository [link](#). Also refer *How to Visualize Scan results*.

All “Output Control” Scan Options

- strip-root** Strip the root directory segment of all paths.
- full-root** Report full, absolute paths.

Note: The options `--strip-root` and `--full-root` can't be used together, i.e. Any one option may be used in a single scan.

Note: The default is to always include the last directory segment of the scanned path such that all paths have a common root directory.

- ignore-author <pattern>** Ignore a file (and all its findings) if an author contains a match to the <pattern> regular expression.
- ignore-copyright-holder <pattern>** Ignore a file (and all its findings) if a copyright holder contains a match to the <pattern> regular expression.

Note: Note that this both the options `--ignore-author` and `--ignore-copyright-holder` will ignore a file even if it has other scanned data such as a license or errors.

- only-findings** Only return files or directories with findings for the requested scans. Files and directories without findings are omitted (file information is not treated as findings).

All “Pre-Scan” Options

- ignore <pattern>** Ignore files matching <pattern>.
- include <pattern>** Include files matching <pattern>.
- classify** Classify files with flags telling if the file is a legal, or readme or test file, etc.
- Sub-Options:
- `--license-clarity-score`
 - `--tallies-key-files`

--facet <facet_pattern> Here <facet_pattern> represents <facet>=<pattern>. Add the <facet> to files with a path matching <pattern>.

Sub-Options:

- --tallies-by-facet

All “Post-Scan” Options

--mark-source Set the “is_source” flag to true for directories that contain over 90% of source files as direct children and descendants. Count the number of source files in a directory as a new “source_file_counts” attribute

Sub-Option of: --url

--consolidate Group resources by Packages or license and copyright holder and return those groupings as a list of consolidated packages and a list of consolidated components. The --consolidate option will be deprecated in a future version of scancode-toolkit as top level packages now provide improved consolidated data.

Sub-Option of: --copyright, --license and --packages.

--filter-clues Filter redundant duplicated clues already contained in detected licenses, copyright texts and notices.

--license-clarity-score Compute a summary license clarity score at the codebase level.

Sub-Option of: --classify.

--license-policy FILE Load a License Policy file and apply it to the scan at the Resource level.

--summary Summarize scans by providing declared origin information and other detected info at the codebase attribute level.

--tallies Summarize license, copyright and other scans at the codebase level with occurrence counts.

Sub-Options:

- --tallies-by-facet
- --tallies-key-files
- --tallies-with-details

--tallies-by-facet Summarize license, copyright and other scans and group the results by facet.

Sub-Option of: --tallies and --facet.

--tallies-key-files Summarize license, copyright and other scans for key, top-level files, with occurrence counts. Key files are top-level codebase files such as COPYING, README and package manifests as reported by the --classify option: “is_legal”, “is_readme”, “is_manifest” and “is_top_level” flags.

Sub-Option of: --classify and --summary.

--tallies-with-details Summarize license, copyright and other scans at the codebase level with occurrence counts, while also keeping intermediate details at the file and directory level.

3.1.4 How to Run a Scan

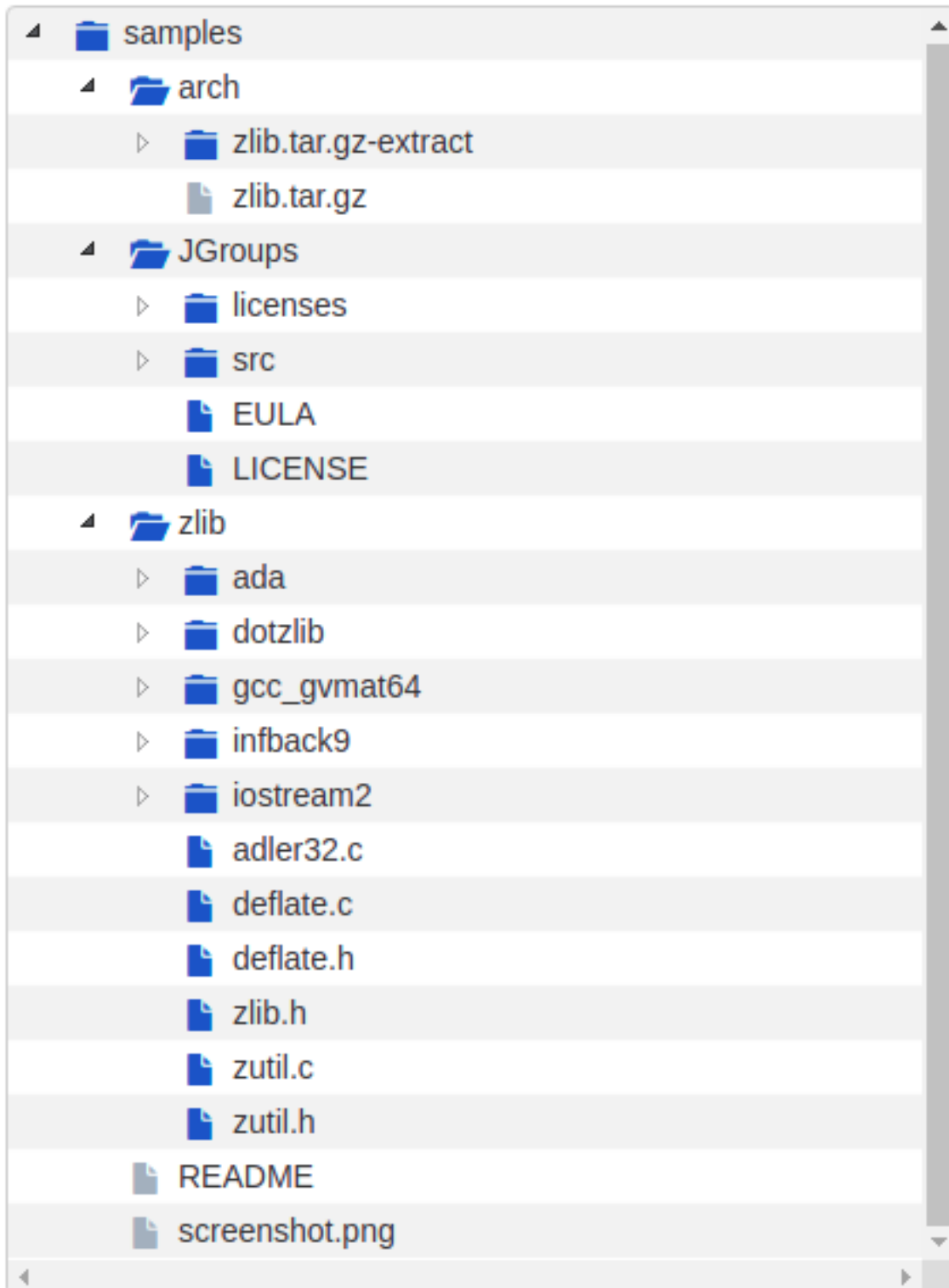
In this simple tutorial example, we perform a basic scan on the `samples` directory distributed by default with Scancode.

Prerequisites

Refer to the *Comprehensive Installation* installation guide.

Looking into Files

As mentioned previously, we are going to perform the scan on the `samples` directory distributed by default with Scancode Toolkit. Here's the directory structure and respective files:



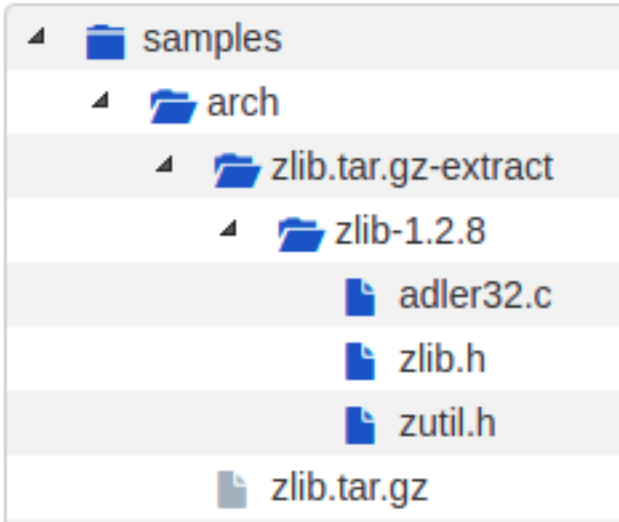
We notice here that the sample files contain a package `zlib.tar.gz`. So we have to extract the archive before running the scan, to also scan the files inside this package.

Performing Extraction

To extract the packages inside `samples` directory:

```
extractcode samples
```

This extracts the `zlib.tar.gz` package:



Note: Use the `--shallow` option to prevent recursive extraction of nested archives.

Deciding Scan Options

These are some common scan options you should consider using before you start the actual scan, according to your requirements.

1. The basic scan options, i.e. `-c` or `--copyright`, `-l` or `--license`, `-p` or `--package`, `-e` or `--email`, `-u` or `--url`, and `-i` or `--info` can be selected according to your requirements. If you do not need one specific type of information (say, licenses), consider removing it because the more options you scan for, the longer it will take for the scan to complete.
2. `--license-score` `INTEGER` is to be set if license matching accuracy is desired (Default is 0, and increasing this means a more accurate match). Also, using `--license-text` includes the matched text to the result.
3. `-n` `INTEGER` option can be used to speed up the scan using multiple parallel processes.
4. `--timeout` `FLOAT` option can be used to skip files taking a long time to scan.
5. `--ignore` `<pattern>` can be used to skip certain group of files.
6. `<OUTPUT FORMAT OPTION(s)>` is also a very important decision when you want to use the output for specific tasks/have requirements. Here we are using `json` as ScanCode Workbench imports `json` files only.

For the complete list of options, refer [All Available Options](#).

Running The Scan

Now, run the scan with the options decided:

```
scancode -clpeui -n 2 --ignore "*.java" --json-pp sample.json samples
```

A Progress report is shown:

```
Setup plugins...
Collect file inventory...
Scan files for: info, licenses, copyrights, packages, emails, urls with 2 process(es)...
[#####] 29
Scanning done.
Summary:      info, licenses, copyrights, packages, emails, urls with 2 process(es)
Errors count: 0
Scan Speed:   1.09 files/sec. 40.67 KB/sec.
Initial counts: 49 resource(s): 36 file(s) and 13 directorie(s)
Final counts:  42 resource(s): 29 file(s) and 13 directorie(s) for 1.06 MB
Timings:
  scan_start: 2019-09-24T203514.573671
  scan_end:   2019-09-24T203545.649805
  setup_scan:licenses: 4.30s
  setup: 4.30s
  scan: 26.62s
  total: 31.14s
Removing temporary files...done.
```

Other Important Documentation

1. *Type of Options*
2. *How to Run a Scan*
3. *Basic Tutorials*
4. *How-To Guides*
5. *Reference Docs*
6. *Contributing to Code Development*
7. *Contributing to the Documentation*
8. *Plugin Architecture*
9. *FAQ*
10. *Support*

3.1.5 Other available CLIs

scancode-reindex-licenses Usage

Usage: `scancode-reindex-licenses [OPTIONS]`

Reindex scancode licenses and exit

Options

--all-languages	[EXPERIMENTAL] Rebuild the license index including texts all languages (and not only English) and exit.
--only-builtin	Rebuild the license index excluding any additional license directory or additional license plugins which were added previously, i.e. with only builtin scancode license and rules.
--additional-directory DIR	Include this directory with additional custom licenses and license rules in the license detection index.
--load-dump	Load all license and rules from their respective files and then dump them back to those same files.
-h, --help	Shows the options and explanations.

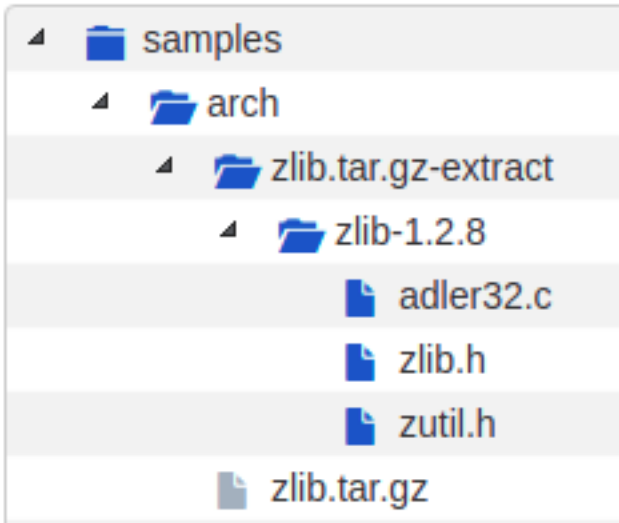
All Extractcode Options

This is intended to be used as an input preparation step, before running the scan. Archives found in an extracted archive are extracted **recursively** by default. Extraction is done in-place in a directory named ‘-extract’ side-by-side with an archive.

To extract the packages in the `samples` directory

```
extractcode samples
```

This extracts the `zlib.tar.gz` package:



--shallow	Do not extract recursively nested archives (e.g. Not archives in archives).
--verbose	Print verbose file-by-file progress messages.
--quiet	Do not print any summary or progress message.
-h, --help	Show the extractcode help message and exit.
--about	Show information about ScanCode and licensing and exit.
--version	Show the version and exit.

scancode-reindex-licenses command

ScanCode maintains a license index to search for and detect licenses. When Scancode is configured for the first time, a license index is built and used in every scan thereafter.

This `scancode-reindex-licenses` command rebuilds the license index. Running this command displays the following message to the terminal:

```
Checking and rebuilding the license index...
```

This has several CLI options as follows:

--additional-directory Option:

The `--additional-directory` option allows the user to include additional directories of licenses to use in license detection.

This command only needs to be run once for each set of additional directories, in all subsequent runs of Scancode with the same directories all the licenses in the directories will be cached and used in License detection. But reindexing removes these directories, if they aren't reintroduced as additional directories.

The directory structure should look something like this:

```

additional_license_directory/
├── licenses/
│   ├── example-installed-1.LICENSE
│   └── example-installed-1.yaml
└── rules/
    ├── example-installed-1.RULE
    └── example-installed-1.yaml

```

Here is an example of reindexing the license cache using the `--additional-directory` PATH option with a single directory:

```

scancode-reindex-licenses --additional-directory tests/licensedcode/data/additional_
↪ licenses/additional_dir/

```

You can also include multiple directories like so:

```

scancode-reindex-licenses --additional-directory /home/user/external_licenses/external1 -
↪ -additional-directory /home/user/external_licenses/external2

```

If you want to continue running scans with `/home/user/external_licenses/external1` and `/home/user/external_licenses/external2`, you can simply run scans after the command above reindexing with those directories and they will be included.

```

scancode -l --license-text --json-pp output.json samples

```

However, if you wanted to run a scan with a new set of directories, such as `home/user/external_licenses/external1` and `home/user/external_licenses/external3`, you would need to reindex the license index with those directories as parameters:

```

scancode --additional-directory /home/user/external_licenses/external1 --additional-
↪ directory /home/user/external_licenses/external3

```

Note: Adding licenses/rules from an additional directory is not permanent. Another reindexing without the additional directory option would just use the builtin scancode licenses and rules, and will not have these additional licenses/rules anymore.

Note: You can also install external licenses through a plugin for better reproducibility and distribution of those license/rules for use in conjunction with scancode-toolkit licenses. See [How to install a plugin containing external licenses and/or rules](#)

--only-builtin Option:

Rebuild the license index excluding any additional license directory or additional license plugins which were added previously, i.e. with only builtin scancode license and rules.

This is applicable when there are additional license plugins installed already and you want to reindex the licenses without these licenses from the additional plugins.

Note: Running the `--only-builtin` command won't get rid of the installed license plugins, it would just reindex without the licenses from these plugins for once. Another reindex afterwards without this option would bring back the

licenses from the plugins again in the index.

--all-languages Option:

Rebuild the license index including texts all languages (and not only English) and exit. This is an EXPERIMENTAL option.

--load-dump Option

Load all licenses and rules from their respective files and then dump them to their respective files. This is done to make small formatting changes across all licenses and rules, to be consistent across them.

3.1.6 Basic Options

All “Basic” Scan Options

Option lists are two-column lists of command-line options and descriptions, documenting a program’s options. For example:

-c, --copyright	Scan <input> for copyrights. Sub-Options: <ul style="list-style-type: none"> • --consolidate
-l, --license	Scan <input> for licenses. Sub-Options: <ul style="list-style-type: none"> • --license-references • --license-text • --license-text-diagnostics • --license-diagnostics • --license-url-template TEXT • --license-score INT • --license-clarity-score • --consolidate • --unknown-licenses
-p, --package	Scan <input> for packages. Sub-Options: <ul style="list-style-type: none"> • --consolidate
--system-package	Scan <input> for installed system package databases.
--package-only	Scan <input> for system and application only for package metadata, without license/ copyright detection and package assembly.

-e, --email	Scan <input> for emails. Sub-Options: <ul style="list-style-type: none"> • <code>--max-email INT</code>
-u, --url	Scan <input> for urls. Sub-Options: <ul style="list-style-type: none"> • <code>--max-url INT</code>
-i, --info	Scan for and include information such as: <ul style="list-style-type: none"> • Size, • Type, • Date, • Programming language, • sha1 and md5 hashes, • binary/text/archive/media/source/script flags • Additional options through more CLI options Sub-Options: <ul style="list-style-type: none"> • <code>--mark-source</code>

Note: Unlike previous 2.x versions, `-c`, `-l`, and `-p` are not default. If any combination of these options are used, ScanCode performs only that specific task, and not the others. `scancode -l` scans only for licenses, and doesn't scan for copyright/packages/general information/emails/urls. The only notable exception: a `--package` scan also has license information for package manifests and top-level packages, which are derived regardless of `--license` option being used.

Note: These options, i.e. `-c`, `-l`, `-p`, `-e`, `-u`, and `-i` can be used together. As in, instead of `scancode -c -i -p`, you can write `scancode -cip` and it will be the same.

--generated	Classify automatically generated code files with a flag.
--max-email INT	Report only up to INT emails found in a file. Use 0 for no limit. [Default: 50] Sub-Option of: <code>--email</code>
--max-url INT	Report only up to INT urls found in a file. Use 0 for no limit. [Default: 50] Sub-Option of: <code>--url</code>
--license-score INTEGER	Do not return license matches with scores lower than this score. A number between 0 and 100. [Default: 0] Here, a bigger number means a better match, i.e. Setting a higher license score translates to a higher threshold (with equal or smaller number of matches). Sub-Option of: <code>--license</code>
--license-text	Include the matched text for the detected licenses in the output report. Sub-Option of: <code>--license</code> Sub-Options:

- `--license-text-diagnostics`

`--license-url-template TEXT` Set the template URL used for the license reference URLs.

In a template URL, curly braces (`{}`) are replaced by the license key. [Default: default: <https://scancode-licensedb.aboutcode.org/{ }>]

Sub-Option of: `--license`

`--license-text-diagnostics` In the matched license text, include diagnostic highlights surrounding with square brackets `[]` words that are not matched.

Sub-Option of: `--license` and `--license-text`

`--license-diagnostics` In license detections, include diagnostic details to figure out the license detection post processing steps applied.

Sub-Option of: `--license`

`--unknown-licenses` [EXPERIMENTAL] Detect unknown licenses.

Sub-Option of: `--license`

`--copyright` Option

The `--copyright` option detects copyright statements in files.

It adds the following resource-level attributes:

1. **copyrights:** This is a data mapping with the following attributes: **copyright** containing the whole copyright value, with **start_line** and **end_line** containing the line numbers in the file where this copyright value was detected.
2. **holders:** This is a data mapping with the following attributes: **holder** containing the whole copyright holder value, with **start_line** and **end_line** containing the line numbers in the file where this copyright value was detected.
3. **authors:** This is a data mapping with the following attributes: **author** containing the whole copyright author value, with **start_line** and **end_line** containing the line numbers in the file where this copyright value was detected.

Example:

```
#
# Copyright (c) 2010 Patrick McHardy All rights reserved.
# Authors:      Patrick McHardy <kaber@trash.net>
```

The above lines when scanned for copyrights generates the following results for the discussed attributes:

```
{
  "copyrights": [
    {
      "copyright": "Copyright (c) 2010 Patrick McHardy",
      "start_line": 2,
      "end_line": 2
    }
  ],
  "holders": [
```

(continues on next page)

(continued from previous page)

```

    {
      "holder": "Patrick McHardy",
      "start_line": 2,
      "end_line": 2
    }
  ],
  "authors": [
    {
      "author": "Patrick McHardy <kaber@trash.net>",
      "start_line": 3,
      "end_line": 3
    }
  ],
}

```

--license Option

The `--license` option detects various kinds of license texts, notices, tags, references and other specialized license declarations like the SPDX license identifier in files.

It adds the following attributes to the file data:

1. `license_detections`: This has a mapping of license detection data with the license expression, detection log and license matches. And the license matches contain the license expression for the match, score, more details for the license detected and the rule detected, along with the match text optionally.
2. `license_clues`: This is a list of license matches, same as matches in `license_detections`. These are mere license clues and not perfect detections.
3. `detected_license_expression`: This is a scancode license expression string.
4. `detected_license_expression_spdx`: This is the SPDX version of `detected_license_expression`.
5. `percentage_of_license_text`: This has a percentage number which denotes what percentage of the resource scanned has legalese words.

Example:

```
License: Apache-2.0
```

If we run license detection (with `--license-text`) on the above text we get the following result for the resource attributes added by the license detection:

```

{
  "path": "apache-2.0.txt",
  "type": "file",
  "detected_license_expression": "apache-2.0",
  "detected_license_expression_spdx": "Apache-2.0",
  "license_detections": [
    {
      "license_expression": "apache-2.0",

```

(continues on next page)

(continued from previous page)

```

    "matches": [
      {
        "score": 100.0,
        "start_line": 1,
        "end_line": 1,
        "matched_length": 4,
        "match_coverage": 100.0,
        "matcher": "1-hash",
        "license_expression": "apache-2.0",
        "rule_identifier": "apache-2.0_65.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/
→develop/src/licensedcode/data/rules/apache-2.0_65.RULE",
        "matched_text": "License: Apache-2.0"
      }
    ],
    "identifier": "apache_2_0-ec759ae0-ea5a-f138-793e-388520e080c0"
  },
  "license_clues": [],
  "percentage_of_license_text": 100.0,
  "scan_errors": []
}

```

We also have top level unique license detections with the same identifier referencing all occurrences of this license detection and counts:

```

{
  "license_detections": [
    {
      "identifier": "apache_2_0-ec759ae0-ea5a-f138-793e-388520e080c0",
      "license_expression": "apache-2.0",
      "detection_count": 1
    }
  ]
}

```

--package Option

The `--package` option detects various package manifests, lockfiles and package-like data and then assembles codebase level packages and dependencies from these package data detected at files. Also tags files if they are part of the packages.

It adds the following attributes to the file data:

1. `package_data`: This is a mapping of package data parsed and retrieved from the file, with the fields for the package URL, license detections, copyrights, dependencies, and the various URLs.
2. `for_packages`: This is a list of strings pointing to the packages that the files is a part of. The string is basically a packageURL with an UUID as a qualifier.

It adds the following attributes to the top-level in results:

1. **packages:** This is a mapping of package data with all the attributes present in file level `package_data` with the following extra attributes: `package_uid`, `datafile_paths` and `datasource_ids`.
2. **dependencies:** This is a mapping of dependency data from all the lockfiles or package manifests in the scan.

Example:

The following scan result was generated from scanning a package manifest:

```
{
  "dependencies": [
    {
      "purl": "pkg:bower/get-size",
      "extracted_requirement": "~1.2.2",
      "scope": "dependencies",
      "is_runtime": true,
      "is_optional": false,
      "is_resolved": false,
      "resolved_package": {},
      "extra_data": {},
      "dependency_uid": "pkg:bower/get-size?uuid=fixed-uid-done-for-
→testing-5642512d1758",
      "for_package_uid": "pkg:bower/blue-leaf?uuid=fixed-uid-done-for-
→testing-5642512d1758",
      "datafile_path": "bower.json",
      "datasource_id": "bower_json"
    }
  ],
  "packages": [
    {
      "type": "bower",
      "namespace": null,
      "name": "blue-leaf",
      "version": null,
      "qualifiers": {},
      "subpath": null,
      "primary_language": null,
      "description": "Physics-like animations for pretty particles",
      "release_date": null,
      "parties": [
        {
          "type": null,
          "role": "author",
          "name": "Betty Beta <bbeta@example.com>",
          "email": null,
          "url": null
        }
      ],
      "keywords": [
        "motion",
        "physics",
        "particles"
      ],
    }
  ],
}
```

(continues on next page)

(continued from previous page)

```

    "homepage_url": null,
    "download_url": null,
    "size": null,
    "sha1": null,
    "md5": null,
    "sha256": null,
    "sha512": null,
    "bug_tracking_url": null,
    "code_view_url": null,
    "vcs_url": null,
    "copyright": null,
    "declared_license_expression": "mit",
    "declared_license_expression_spdx": "MIT",
    "license_detections": [
      {
        "license_expression": "mit",
        "matches": [
          {
            "score": 100.0,
            "start_line": 1,
            "end_line": 1,
            "matched_length": 1,
            "match_coverage": 100.0,
            "matcher": "1-spdx-id",
            "license_expression": "mit",
            "rule_identifier": "spdx-license-identifier: mit",
            "rule_url": null,
            "rule_relevance": 100,
            "matched_text": "MIT"
          }
        ],
        "identifier": "apache_2_0-ec759abc-ea5a-2a38-793e-
↪ 312340e080c0"
      }
    ],
    "other_license_expression": null,
    "other_license_expression_spdx": null,
    "other_license_detections": [],
    "extracted_license_statement": "MIT",
    "notice_text": null,
    "source_packages": [],
    "extra_data": {},
    "repository_homepage_url": null,
    "repository_download_url": null,
    "api_data_url": null,
    "package_uid": "pkg:bower/blue-leaf?uuid=fixed-uid-done-for-
↪ testing-5642512d1758",
    "datafile_paths": [
      "bower.json"
    ],
    "datasource_ids": [
      "bower_json"
    ]

```

(continues on next page)

(continued from previous page)

```

    ],
    "purl": "pkg:bower/blue-leaf"
  }
],
"files": [
  {
    "path": "bower.json",
    "type": "file",
    "package_data": [
      {
        "type": "bower",
        "namespace": null,
        "name": "blue-leaf",
        "version": null,
        "qualifiers": {},
        "subpath": null,
        "primary_language": null,
        "description": "Physics-like animations for pretty_
↪particles",
        "release_date": null,
        "parties": [
          {
            "type": null,
            "role": "author",
            "name": "Betty Beta <bbeta@example.com>",
            "email": null,
            "url": null
          }
        ],
        "keywords": [
          "motion",
          "physics",
          "particles"
        ],
        "homepage_url": null,
        "download_url": null,
        "size": null,
        "sha1": null,
        "md5": null,
        "sha256": null,
        "sha512": null,
        "bug_tracking_url": null,
        "code_view_url": null,
        "vcs_url": null,
        "copyright": null,
        "declared_license_expression": "mit",
        "declared_license_expression_spdx": "MIT",
        "license_detections": [
          {
            "license_expression": "mit",
            "matches": [

```

(continues on next page)

(continued from previous page)

```

        "score": 100.0,
        "start_line": 1,
        "end_line": 1,
        "matched_length": 1,
        "match_coverage": 100.0,
        "matcher": "1-spdx-id",
        "license_expression": "mit",
        "rule_identifier": "spdx-license-
→ identifier: mit",
        "rule_url": null,
        "rule_relevance": 100,
        "matched_text": "MIT"
    },
    ],
    "identifier": "apache_2_0-ec759abc-ea5a-2a38-793e-
→ 312340e080c0"
    },
    ],
    "other_license_expression": null,
    "other_license_expression_spdx": null,
    "other_license_detections": [],
    "extracted_license_statement": "MIT",
    "notice_text": null,
    "source_packages": [],
    "file_references": [],
    "extra_data": {},
    "dependencies": [
        {
            "purl": "pkg:bower/get-size",
            "extracted_requirement": "~1.2.2",
            "scope": "dependencies",
            "is_runtime": true,
            "is_optional": false,
            "is_resolved": false,
            "resolved_package": {},
            "extra_data": {}
        }
    ],
    "repository_homepage_url": null,
    "repository_download_url": null,
    "api_data_url": null,
    "datasource_id": "bower_json",
    "purl": "pkg:bower/blue-leaf"
    },
    ],
    "for_packages": [
        "pkg:bower/blue-leaf?uuid=fixed-uid-done-for-testing-
→ 5642512d1758"
    ],
    "scan_errors": []
    }
]

```

(continues on next page)

(continued from previous page)

```
}
```

--info Option

The `--info` option obtains miscellaneous information about the file being scanned such as mime/filetype, checksums, programming language, and various boolean flags.

It adds the following attributes to the file data:

1. `date`: last modified data of the file.
2. `sha1`, `md5` and `sha256`: file checksums of various algorithms.
3. `mime_type` and `file_type`: basic file type and mime type/subtype information obtained from lib-magic.
4. `programming_language`: programming language based on extensions.
5. `is_binary`, `is_text`, `is_archive`, `is_media`, `is_source`, and `is_script`: various boolean flags with misc. information about the file.

--email Option

The `--email` option detects and reports email addresses present in scanned files.

It adds the `emails` attribute to the file data with the following attributes: `email` with the actual email that was present in the file, `start_line` and `end_line` to be able to locate where the email was detected in the file.

--url Option

The `--url` option detects and reports URLs present in scanned files.

It adds the `urls` attribute to the file data with the following attributes: `url` with the actual URL that was present in the file, `start_line` and `end_line` to be able to locate where the URL was detected in the file.

--generated Option

The `--generated` option classifies automatically generated code files with a flag.

An example of using `--generated` in a scan:

```
scancode -clpieu --json-pp output.json samples --generated
```

In the results, for each file the following attribute is added with its corresponding `true/false` value

```
"is_generated": true
```

Classification of a file being generated or not is done based on the first few lines having usually encountered generated keywords.

--max-email Option

Dependency

The option `--max-email` is a sub-option of and requires the option `--email`.

If in the files that are scanned, in individual files, there are a lot of emails (i.e lists) which are unnecessary and clutter the scan results, `--max-email` option can be used to report emails only up to a limit in individual files.

Some important INTEGER values of the `--max-email` INTEGER option:

- 0 - No limit, include all emails.
- 50 - Default.

An example usage:

```
scancode -clpieu --json-pp output.json samples --max-email 5
```

This only reports 5 email addresses per file and ignores the rest.

--max-url Option

Dependency

The option `--max-url` is a sub-option of and requires the option `--url`.

If in the files that are scanned, in individual files, there are a lot of links to other websites (i.e url lists) which are unnecessary and clutter the scan results, `--max-url` option can be used to report urls only up to a limit in individual files.

Some important INTEGER values of the `--max-url` INTEGER option:

- 0 - No limit, include all urls.
- 50 - Default.

An example usage:

```
scancode -clpieu --json-pp output.json samples --max-url 10
```

This only reports 10 urls per file and ignores the rest.

--license-score Option

Dependency

The option `--license-score` is a sub-option of and requires the option `--license`.

License matching strictness, i.e. How closely matched licenses are detected in a scan, can be modified by using this `--license-score` option.

Some important INTEGER values of the `--license-score` INTEGER option:

- **0** - Default and Lowest Value, All matches are reported.
- **100** - Highest Value, Only licenses with a much better match are reported

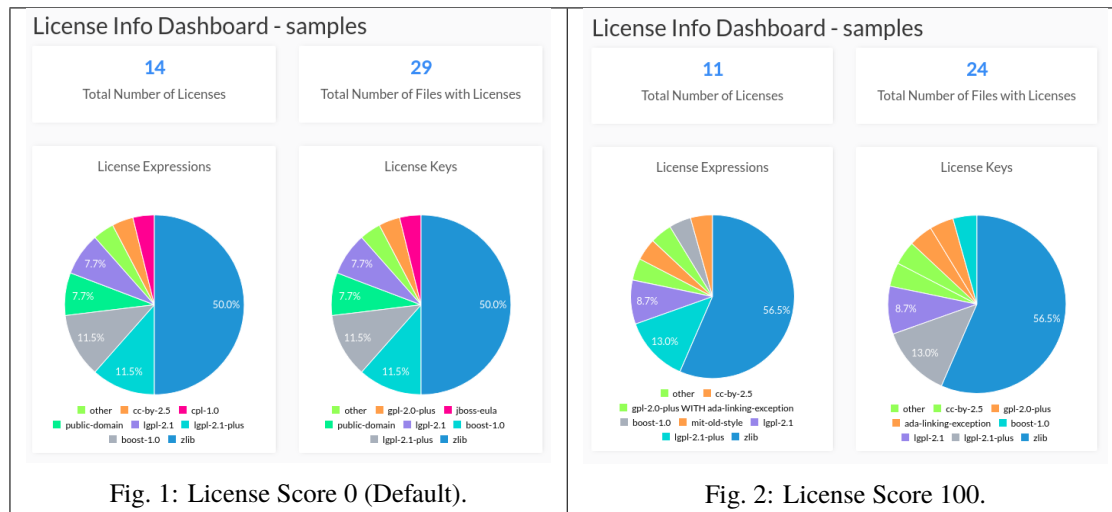
Here, a bigger number means a better match, i.e. Setting a higher license score translates to a higher threshold for matching licenses (with equal or less number of license matches).

An example usage:

```
scancode -clpieu --json-pp output.json samples --license-score 70
```

Here's the license results on setting the integer value to 100, Vs. the default value 0. This is visualized using ScanCode workbench in the License Info Dashboard.

Table 1: License scan results of Samples Directory.



--license-text Option

Dependency

The option `--license-text` is a sub-option of and requires the option `--license`.

Sub-Option

The option `--license-text-diagnostics` is a sub-option of `--license-text`.

With the `--license-text` option, the scan results attribute “matched text” includes the matched text for the detected license.

An example Scan:

```
scancode -cplieu --json-pp output.json samples --license-text
```

An example matched text included in the results is as follows:

```
"matched_text":
"  This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.
Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:
1. The origin of this software must not be misrepresented; you must not
claim that you wrote the original software. If you use this software
in a product, an acknowledgment in the product documentation would be
appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly          Mark Adler
jloup@gzip.org            madler@alumni.caltech.edu"
```

- The file in which this license was detected: `samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zlib.h`
- License name: “ZLIB License”

--license-url-template Option

Dependency

The option `--license-url-template` is a sub-option of and requires the option `--license`.

The `--license-url-template` option sets the template URL used for the license reference URLs.

The default template URL is : `[https://scancode-licensedb.aboutcode.org/{}]` In a template URL, curly braces ({}) are replaced by the license key.

So, by default the license reference URL points to the LicenseDB page for that license.

A scan example using the `--license-url-template TEXT` option

```
scancode -clpieu --json-pp output.json samples --license-url-template https://
↪github.com/nexB/scancode-toolkit/tree/develop/src/licensedcode/data/licenses/
↪{}.LICENSE
```

In a normal scan, reference url for “ZLIB License” is as follows:

```
"reference_url": "https://scancode-licensedb.aboutcode.org/zlib",
```

After using the option in the following manner:

```
`--license-url-template https://github.com/nexB/scancode-toolkit/tree/develop/
↪src/licensedcode/data/licenses/{}.LICENSE`
```

the reference URL changes to this `zlib.LICENSE` file:

```
"reference_url": "https://github.com/nexB/scancode-toolkit/blob/develop/src/
↪licensedcode/data/licenses/zlib.LICENSE",
```

The reference URL changes for all detected licenses in the scan, across the scan result file.

--license-text-diagnostics Option

Dependency

The option `--license-text-diagnostics` is a sub-option of and requires the options `--license` and `--license-text`.

In the matched license text, include diagnostic highlights surrounding with square brackets [] words that are not matched.

In a normal scan, whole lines of text are included in the matched license text, including parts that are possibly unmatched.

An example Scan:

```
scancode -cplieu --json-pp output.json samples --license-text --license-text-
↪diagnostics
```


Running a scan on the samples directory with `--license-text --license-text-diagnostics` options, causes the following difference in the scan result of the file `samples/JGroups/licenses/bouncycastle.txt`.

Without Diagnostics:

```
"matched_text":
"License Copyright (c) 2000 - 2006 The Legion Of The Bouncy Castle
(http://www.bouncycastle.org) 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
```

With Diagnostics on:

```
"matched_text":
"License [Copyright] ([c]) [2000] - [2006] [The] [Legion] [Of] [The] [Bouncy]
↪[Castle]
([http]://[www].[bouncycastle].[org]) 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,
```

--license-diagnostics Option

Dependency

The option `--license-diagnostics` is a sub-option of and requires the option `--license`

On using the `--license-diagnostics` option on a license scan there is the `detection_log` attribute added to license detections with diagnostics information about the license detection post-processing steps which are used to create license detections from license matches.

Consider the following text:

```
## License
All code, unless stated otherwise, is dual-licensed under
[`WTFPL`](http://www.wtfpl.net/txt/copying/) and
[`MIT`](https://opensource.org/licenses/MIT).
```

If we run a license scan with the `--license-diagnostics` option enabled, we have the following license detection results:

```
{
  "path": "README.md",
  "type": "file",
  "detected_license_expression": "wtfpl-2.0 AND mit",
  "detected_license_expression_spdx": "WTFPL AND MIT",
  "license_detections": [
```

(continues on next page)

(continued from previous page)

```

{
  "license_expression": "wtfpl-2.0 AND mit",
  "matches": [
    {
      "score": 100.0,
      "start_line": 43,
      "end_line": 43,
      "matched_length": 3,
      "match_coverage": 100.0,
      "matcher": "2-aho",
      "license_expression": "unknown-license-reference",
      "rule_identifier": "lead-in_unknown_30.RULE",
      "rule_relevance": 100,
      "rule_url": "https://github.com/nexB/scancode-toolkit/tree/
↳develop/src/licensedcode/data/rules/lead-in_unknown_30.RULE",
      "matched_text": "dual-licensed under [`
    },
    {
      "score": 50.0,
      "start_line": 43,
      "end_line": 43,
      "matched_length": 1,
      "match_coverage": 100.0,
      "matcher": "2-aho",
      "license_expression": "wtfpl-2.0",
      "rule_identifier": "spdx_license_id_wtfpl_for_wtfpl-2.0.
↳RULE",
      "rule_relevance": 50,
      "rule_url": "https://github.com/nexB/scancode-toolkit/tree/
↳develop/src/licensedcode/data/rules/spdx_license_id_wtfpl_for_wtfpl-2.0.RULE
↳",
      "matched_text": "WTFPL"
    },
    {
      "score": 100.0,
      "start_line": 43,
      "end_line": 43,
      "matched_length": 3,
      "match_coverage": 100.0,
      "matcher": "2-aho",
      "license_expression": "wtfpl-2.0",
      "rule_identifier": "wtfpl-2.0_27.RULE",
      "rule_relevance": 100,
      "rule_url": "https://github.com/nexB/scancode-toolkit/tree/
↳develop/src/licensedcode/data/rules/wtfpl-2.0_27.RULE",
      "matched_text": "www.wtfpl.net/"
    },
    {
      "score": 100.0,
      "start_line": 43,
      "end_line": 43,
      "matched_length": 6,

```

(continues on next page)

(continued from previous page)

```

        "match_coverage": 100.0,
        "matcher": "2-aho",
        "license_expression": "mit",
        "rule_identifier": "mit_64.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/
→develop/src/licensedcode/data/rules/mit_64.RULE",
        "matched_text": "MIT`](https://opensource.org/licenses/
→MIT)."
    }
],
    "detection_log": [
        "unknown-intro-followed-by-match"
    ],
    "identifier": "wtfpl_2_0_and_mit-e5642b07-705c-9730-80ab-
→f5ed0565be28"
    }
],
    "license_clues": [],
    "percentage_of_license_text": 8.18,
    "scan_errors": []
}

```

Here from the "detection_log": ["unknown-intro-followed-by-match"] added diagnostics information we learn that there was an unknown intro license match, followed by proper detections, so we conclude the unknown intro to be an introduction to the following license and hence conclude the license from the license matches after the unknown detection.

3.1.7 Core Options

All “Core” Scan Options

- n, --processes INTEGER** Scan <input> using n parallel processes. [Default: 1]
- v, --verbose** Print verbose file-by-file progress messages.
- q, --quiet** Do not print summary or progress messages.
- timeout FLOAT** Stop scanning a file if scanning takes longer than a timeout in seconds. [Default: 120]
- from-json** Load codebase from one or more existing JSON scans.
- max-in-memory INTEGER** Maximum number of files and directories scan details kept in memory during a scan. Additional files and directories scan details above this number are cached on-disk rather than in memory. Use 0 to use unlimited memory and disable on-disk caching. Use -1 to use only on-disk caching. [Default: 10000]
- max-depth INTEGER** Descend at most INTEGER levels of directories including and below the starting point. INTEGER must be positive or zero for no limit. [Default: 0]

Comparing Progress Message Options

Default Progress Message:

```
Scanning files for: infos, licenses, copyrights, packages, emails, urls with 1
↳process(es)...
Building license detection index...Done.
Scanning files...
[#####] 43
Scanning done.
Scan statistics: 43 files scanned in 33s.
Scan options:    infos, licenses, copyrights, packages, emails, urls with 1
↳process(es).
Scanning speed:  1.4 files per sec.
Scanning time:   30s.
Indexing time:   2s.
Saving results.
```

Progress Message with ``--verbose``:

```
Scanning files for: infos, licenses, copyrights, packages, emails, urls with 1
↳process(es)...
Building license detection index...Done.
Scanning files...
Scanned: screenshot.png
Scanned: README
...
Scanned: zlib/dotzlib/ChecksumImpl.cs
Scanned: zlib/dotzlib/readme.txt
Scanned: zlib/gcc_gvmt64/gvmt64.S
Scanned: zlib/ada/zlib.ads
Scanned: zlib/infbck9/infbck9.c
Scanned: zlib/infbck9/infbck9.h
Scanned: arch/zlib.tar.gz
Scanning done.
Scan statistics: 43 files scanned in 29s.
Scan options:    infos, licenses, copyrights, packages, emails, urls with 1
↳process(es).
Scanning speed:  1.58 files per sec.
Scanning time:   27s.
Indexing time:   2s.
Saving results.
```

So, with `--verbose` enables, progress messages for individual files are shown.

With the `--quiet` option enabled, nothing is printed on the Command Line.

--timeout Option

This option sets scan timeout for **each file** (and not the entire scan). If some file scan exceeds the specified timeout, that file isn't scanned anymore and the next file scanning starts. This helps avoiding very large/long files, and saves time.

Also the number (timeout in seconds) to be followed by this option can be a floating point number, i.e. 1.5467.

--from-json Option

If you want to input scan results from a .json file, and run a scan again on those same files, with some other options/output format, you can do so using the `--from-json` option.

An example scan command using `--from-json`:

```
scancode --from-json sample.json --json-pp sample_2.json --classify
```

This inputs the scan results from `sample.json`, runs the post-scan plugin `--classify` and outputs the results for this scan to `sample_2.json`.

--max-in-memory Option

During a scan, as individual files are scanned, the scan details for those files are kept on memory till the scan is completed. Then after the scan is completed, they are written in the specified output format.

Now, if the scan involves a very large number of files, they might not fit in the memory during the scan. For this reason, disk-caching can be used for some/all of the files.

Some important INTEGER values of the `--max-in-memory` INTEGER option:

- **0** - Unlimited Memory, store all the file/directory scan results on memory
- **-1** - Use only Disk-Caching, store all the file/directory scan results on disk
- **10000** - Default, store 10,000 file/directory scan results on memory and the rest on disk

An example usage:

```
scancode -clieu --json-pp sample.json samples --max-in-memory -1
```

--max_depth Option

Normally, the scan takes place upto the maximum level of nesting of directories possible. But using the `--max-depth` option, you can specify the maximum level of directories to scan, including and below the root location. This can reduce the time taken for the scan when deeper directories are not relevant.

Note that the `--max-depth` option will be ignored if you are scanning from a JSON file using the `--from-json` option. In that case, the original depth is used.

An example usage:

```
scancode -clieu --json-pp results.json samples --max-depth 3
```

This would scan the file `samples/levelone/leveltwo/file` but ignore `samples/levelone/leveltwo/levelthree/file`

3.1.8 Scancode Output Formats

Scan results generated by Scancode are available in different formats, to be specified by the following options.

All Scan Output Options

--json FILE	Write scan output as compact JSON to FILE.
--json-pp FILE	Write scan output as pretty-printed JSON to FILE. This is one of the recommended output formats and contains all the data scancode can show along with the YAML output format.
--json-lines FILE	Write scan output as JSON Lines to FILE.
--yaml FILE	Write scan output as YAML to FILE. This is one of the recommended output formats and contains all the data scancode can show along with the JSON output format.
--csv FILE	DEPRECATED: Write scan output as CSV to FILE. This option is deprecated and will be replaced by new CSV and tabular output formats in the next ScanCode release. Visit this issue for details, and to provide input and feedback: https://github.com/nexB/scancode-toolkit/issues/3043
--html FILE	Write scan output as HTML to FILE.
--custom-output	Write scan output to FILE formatted with the custom Jinja template file. Mandatory Sub-option: <ul style="list-style-type: none"> --custom-template FILE
--custom-template FILE	Use this Jinja template FILE as a custom template. Sub-Option of: --custom-output
--debian FILE	Write scan output in machine-readable Debian copyright format to FILE.
--spdx-rdf FILE	Write scan output as SPDX RDF to FILE.
--spdx-tv FILE	Write scan output as SPDX Tag/Value to FILE.
--html-app FILE	[DEPRECATED] Use <code>scancode-workbench</code> instead. Write scan output as a mini HTML application to FILE.
--cyclonedx FILE	Write scan output as a CycloneDx 1.3 BOM in pretty-printed JSON format to FILE
--cyclonedx-xml FILE	Write scan output as a CycloneDx 1.3 BOM in pretty-printed XML format to FILE

Warning: The `html-app` feature has been deprecated and you should use Scancode Workbench instead to visualize scan results. The official Repository [link](#). Also refer [How to Visualize Scan results](#).

Note: You can Output Scan Results in two different file formats simultaneously in one Scan. An example - `scancode -clpieu --json-pp output.json --html output.html samples`.

Note: All the examples and snippets that follows has been generated by scanning the `samples` folder distributed with `scancode-toolkit`.

Print to stdout (Terminal)

If you want to format the output in JSON and print it at stdout, you can replace the JSON filename with a “-”, like `--json-pp -` instead of `--json-pp output.json`.

The following command will output the scan results in JSON format to stdout (In the Terminal):

```
./scancode -clpieu --json-pp - samples/
```

--json FILE

Among the ScanCode Output Formats, `json` is the most important one, and is recommended over others. Scancode Workbench and other applications that use Scancode Result data as input accept only the `json` format.

The following code performs a scan on the `samples` directory, and publishes the results in `json` format:

```
scancode -clpieu --json output.json samples
```

Note: The default `json` format prints the whole report without line breaks/spaces/indentations, which can be ugly to look at.

The entire JSON file is structured in the following manner:

At first some general information on the scan, what options were used, the number of files etc. And then all the files follow.

```
{
  "headers": [
    {
      "tool_name": "scancode-toolkit",
      "tool_version": "3.1.1",
      "options": {
        "input": [
          "samples/"
        ],
        "--copyright": true,
        "--email": true,
        "--info": true,
```

(continues on next page)

```
(venv-scan3.1.1) ayansm@AyanUbuntuDell:~/Desktop/GSoD/scancode-toolkit-versions/scancode-toolkit-3.1.1$ ./scancode -clpieu --json - samples
Setup plugins...
Collect file inventory...
Scan files for: info, licenses, copyrights, packages, emails, urls with 1 process(es)...
[#####] 36
{"headers":{"tool_name":"scancode-toolkit","tool_version":"3.1.1","options":{"input":["samples"],"--copyright":true,"--email":true,"--info":true,"--json":"","--license":true,"--package":true,"--url":true},"notice":"Generated with ScanCode and provided on an \"AS IS\" BASIS, WITHOUT WARRANTIES\\NOR CONDITIONS OF ANY KIND, either express or implied. No content created from\\nScanCode should be considered or used as legal advice. Consult an Attorney\\nfor any legal advice.\\nScanCode is a free software code scanning tool from nexB Inc. and others.\\nVisit https://github.com/nexB/scancode-toolkit/ for support and download."},"start_timestamp":"2019-10-20T013633.417138","end_timestamp":"2019-10-20T013742.659168","message":null,"errors":[],"extra_data":{"files_count":36},"files":[{"path":"samples","type":"directory","name":"samples","base_name":"samples","extension":"","size":0,"date":null,"sha1":null,"md5":null,"mime_type":null,"file_type":null,"programming_language":null,"is_binary":false,"is_text":false,"is_archive":false,"is_media":false,"is_source":false,"is_script":false,"licenses":[],"license_expressions":[],"copyrights":[],"holders":[],"authors":[],"packages":[],"emails":[],"urls":[],"files_count":36,"dirs_count":12,"size_count":1260700,"scan_errors":[]},{path:"samples/README","type":"file","name":"README","base_name":"README","extension":"","size":236,"date":"2019-02-12","sha1":"2e07e32c52d607204fad196052d70e3d18fb8636","md5":"effc6856ef85a9250fb1a470792b3f38","mime_type":"text/plain","file_type":"ASCII text","programming_language":null,"is_binary":false,"is_text":true,"is_archive":false,"is_media":false,"is_source":false,"is_script":false,"licenses":[],"license_expressions":[],"copyrights":[],"holders":[],"authors":[],"packages":[],"emails":[],"urls":["http://zlib.net/zlib-1.2.8.tar.gz","start_line":4,"end_line":4]},"files_count":0,"dirs_count":0,"size_count":0,"scan_errors":[]},{path:"samples/screenshot.png","type":"file","name":"screenshot.png","base_name":"screenshot","extension":".png","size":622754,"date":"2019-02-12","sha1":"01ff4b1de0bc6c75c9ccade6c80c1082d6976d4","md5":"b6ef5a90777147423c98b42a6a25e57a","mime_type":"image/png","file_type":"PNG image data, 2880 x 1666, 8-bit/color RGB, non-interlaced","programming_language":null,"is_binary":true,"is_text":false,"is_archive":false,"is_media":true,"is_source":false,"is_script":false,"licenses":[],"license_expressions":[],"copyrights":[],"holders":[],"authors":[],"packages":[],"emails":[],"urls":["http://master.dl.sourceforge.net/project/javagroups/JGroups/2.10.0.GA/JGroups-2.10.0.GA.src.zip","start_line":4,"end_line":4]},"files_count":0,"dirs_count":0,"size_count":0,"scan_errors":[]},{path:"samples/arch/zlib.tar.gz","type":"file","name":"zlib.tar.gz","base_name":"zlib","extension":".tar.gz","size":28103,"date":"2019-02-12","sha1":"576f0ccfe534d7f5ff5d6400078d3c6586de3abd","md5":"20b2370751abfc08bb3556c1d8114b5a","mime_type":"application/x-gzip","file_type":"gzip compressed data, last modified: Wed Jul 15 14:38:19 2015, from Unix","programming_language":null,"is_binary":true,"is_text":false,"is_archive":true,"is_media":false,"is_source":false,"is_script":false,"licenses":[],"license_expressions":[],"copyrights":[],"holders":[],"authors":["zlib","packages":["zlib","emails":["zlib","urls":["zlib","files_count":4,"dirs_count":2,"size_count":127720,"scan_errors":[]},{path:"samples/arch/zlib.tar.gz","type":"file","name":"zlib.tar.gz","base_name":"zlib","extension":".tar.gz","size":28103,"date":"2019-02-12","sha1":"576f0ccfe534d7f5ff5d6400078d3c6586de3abd","md5":"20b2370751abfc08bb3556c1d8114b5a","mime_type":"application/x-gzip","file_type":"gzip compressed data, last modified: Wed Jul 15 14:38:19 2015, from Unix","programming_language":null,"is_binary":true,"is_text":false,"is_archive":true,"is_media":false,"is_source":false,"is_script":false,"licenses":["zlib","license_expressions":["zlib","copyrights":["zlib","holders":["zlib","authors":["zlib","packages":["zlib","emails":["zlib","urls":["zlib","files_count":3,"dirs_count":1,"size_count":99617,"scan_errors":[]},{path:"samples/arch/zlib.tar.gz-extract/zlib-1.2.8","type":"directory","name":"zlib-1.2.8","base_name":"zlib-1.2.8","extension":"","size":0,"date":null,"sha1":null,"md5":null,"mime_type":null,"file_type":null,"programming_language":null,"is_binary":null,"is_text":null,"is_archive":null,"is_media":null,"is_source":null,"is_script":null,"licenses":null,"license_expressions":null,"copyrights":null,"holders":null,"authors":null,"packages":null,"emails":null,"urls":null,"files_count":0,"dirs_count":0,"size_count":0,"scan_errors":[]}]}}]}
```

(continued from previous page)

```

"--json-pp": "output.json",
"--license": true,
"--package": true,
"--url": true
},
"notice": "Generated with ScanCode and provided on an \"AS IS\" BASIS,
WITHOUT WARRANTIES\\NOR CONDITIONS OF ANY KIND, either express or implied. No
content created from\\nScanCode should be considered or used as legal advice.
Consult an Attorney\\nfor any legal advice.\\nScanCode is a free software code
scanning tool from nexB Inc. and others.\\nVisit https://github.com/nexB/
scancode-toolkit/ for support and download.",
"start_timestamp": "2019-10-19T191117.292858",
"end_timestamp": "2019-10-19T191219.743133",
"message": null,
"errors": [],
"extra_data": {
  "files_count": 36
}
},
"files": [
{
  "path": "samples",
  "type": "directory",
  ...
  "scan_errors": []
},
{
  "path": "samples/README",
  "type": "file",
  "name": "README",
  "base_name": "README",

```

(continues on next page)

(continued from previous page)

```

    "extension": "",
    "size": 236,
    "date": "2019-02-12",
    "sha1": "2e07e32c52d607204fad196052d70e3d18fb8636",
    "md5": "effc6856ef85a9250fb1a470792b3f38",
    "mime_type": "text/plain",
    "file_type": "ASCII text",
    "programming_language": null,
    "is_binary": false,
    "is_text": true,
    "is_archive": false,
    "is_media": false,
    "is_source": false,
    "is_script": false,
    "license_detections": [],
    "detected_license_expression": None,
    "detected_license_expression_spdx": None,
    "copyrights": [],
    "holders": [],
    "authors": [],
    "package_data": [],
    "for_packages": [],
    "emails": [],
    "urls": [],
    "files_count": 0,
    "dirs_count": 0,
    "size_count": 0,
    "scan_errors": []
  },
  {...},
  ...
]
}

```

--json-pp FILE

`json-pp` stands for JSON Pretty-Print format. In the previous format, i.e. Simple `json`, the whole output is printed in one line, which isn't well suited for getting information if you're looking at the file itself (or printing at stdout). So this option formats the output results in `json` but in a properly spaced and indented manner, and is easy to look at.

The following code performs a scan on the samples directory, and publishes the results in `json-pp` format:

```
scancode -clpieu --json-pp output.json samples
```

A sample JSON output for an individual file will look like:

```

{
  "path": "samples/zlib/iostream2/zstream.h",
  "type": "file",

```

(continues on next page)

(continued from previous page)

```

"name": "zstream.h",
"base_name": "zstream",
"extension": ".h",
"size": 9283,
"date": "2019-02-12",
"sha1": "fca4540d490fff36bb90fd801cf9cd8fc695bb17",
"md5": "a980b61c1e8be68d5cdb1236ba6b43e7",
"mime_type": "text/x-c++",
"file_type": "C++ source, ASCII text",
"programming_language": "C++",
"is_binary": false,
"is_text": true,
"is_archive": false,
"is_media": false,
"is_source": true,
"is_script": false,
"license_detections": [
  "license-expression": "mit-old-style",
  "matches": [
    {
      "license_expression": "mit-old-style",
      "score": 100.0,
      "rule_identifier": "mit-old-style_cmr-no_1.RULE",
      "matcher": "2-aho",
      "rule_length": 71,
      "matched_length": 71,
      "match_coverage": 100.0,
      "rule_relevance": 100
    }
  ]
},
"identifier": "mit-old-style-ec759ae0-1234-f138-793e-356789e080c0",
],
"detected_license_expressions": "mit-old-style",
"detected_license_expressions_spdx": "LicenseRef-scancode-mit-old-style",
"copyrights": [
  {
    "value": "Copyright (c) 1997 Christian Michelsen Research AS Advanced_
↪ Computing",
    "start_line": 3,
    "end_line": 5
  }
],
"holders": [
  {
    "value": "Christian Michelsen Research AS Advanced Computing",
    "start_line": 3,
    "end_line": 5
  }
],
"authors": [],
"package_data": [],
"emails": [],

```

(continues on next page)

(continued from previous page)

```

"urls": [
  {
    "url": "http://www.cmr.no/",
    "start_line": 7,
    "end_line": 7
  }
],
"files_count": 0,
"dirs_count": 0,
"size_count": 0,
"scan_errors": []
},

```

This is the recommended Output option for Scancode Toolkit.

--json-lines FILE

ScanCode also has a --json-lines format option, where each report of a file scanned is formatted in one line.

The following code performs a scan on the samples directory, and publishes the results in json-lines format:

```
scancode -clpieu --json-lines output.json samples
```

Here is a sample line from a report generated by the jsonlines format:

```

{"files":[{"path":"samples/zlib/ada", "licenses":[], "copyrights":[], "packages":
→":[]}]}
```

The header information is also formatted in one line (i.e. The First Line of the file).

The whole Output file looks like:

```

{"headers":[{"tool_name":"scancode-toolkit", "tool_version":"3.1.1", "options":{"
→input":["samples/"], "--copyright":true, "--email":true, "--info":true, "--json-
→lines":"output.json", "--license":true, "--package":true, "--url":true}, "notice
→":"Generated with ScanCode and provided on an \"AS IS\" BASIS, WITHOUT
→WARRANTIES\\NOR CONDITIONS OF ANY KIND, either express or implied. No content
→created from\\nScanCode should be considered or used as legal advice. Consult
→an Attorney\\nfor any legal advice.\\nScanCode is a free software code
→scanning tool from nexB Inc. and others.\\nVisit https://github.com/nexB/
→scancode-toolkit/ for support and download.", "start_timestamp":"2019-10-
→19T210920.143831", "end_timestamp":"2019-10-19T211052.048182", "message":null,
→"errors":[], "extra_data":{"files_count":36}}]}
{"files":[{"path":"samples" ... "scan_errors":[]}]}
```

(continues on next page)

(continued from previous page)

```

{"files":[{"path":"samples/arch/zlib.tar.gz-extract/zlib-1.2.8", ... "scan_
↪errors":[]}]}}
{"files":[{"path":"samples/arch/zlib.tar.gz-extract/zlib-1.2.8/adler32.c", ...
↪"scan_errors":[]}]}}
{"files":[{"path":"samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zlib.h", ...
↪"scan_errors":[]}]}}
{"files":[{"path":"samples/arch/zlib.tar.gz-extract/zlib-1.2.8/zutil.h", ...
↪"scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/EULA", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/LICENSE", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/licenses", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/licenses/apache-1.1.txt", ... "scan_errors
↪":[]}]}}
{"files":[{"path":"samples/JGroups/licenses/apache-2.0.txt", ... "scan_errors
↪":[]}]}}
{"files":[{"path":"samples/JGroups/licenses/bouncycastle.txt", ... "scan_errors
↪":[]}]}}
{"files":[{"path":"samples/JGroups/licenses/cpl-1.0.txt", ... "scan_errors":[]
↪}}
{"files":[{"path":"samples/JGroups/licenses/lgpl.txt", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/src", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/src/FixedMembershipToken.java", ... "scan_
↪errors":[]}]}}
{"files":[{"path":"samples/JGroups/src/GuardedBy.java", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/JGroups/src/ImmutableReference.java", ... "scan_
↪errors":[]}]}}
{"files":[{"path":"samples/JGroups/src/RATE_LIMITER.java", ... "scan_errors
↪":[]}]}}
{"files":[{"path":"samples/JGroups/src/RouterStub.java", ... "scan_errors":[]
↪}}
{"files":[{"path":"samples/JGroups/src/RouterStubManager.java", ... "scan_
↪errors":[]}]}}
{"files":[{"path":"samples/JGroups/src/S3_PING.java", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/adler32.c", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/deflate.c", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/deflate.h", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/zlib.h", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/zutil.c", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/zutil.h", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/ada", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/ada/zlib.ads", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/dotzlib", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/dotzlib/AssemblyInfo.cs", ... "scan_errors":[]
↪}}
{"files":[{"path":"samples/zlib/dotzlib/ChecksumImpl.cs", ... "scan_errors":[]
↪}}
{"files":[{"path":"samples/zlib/dotzlib/LICENSE_1_0.txt", ... "scan_errors":[]
↪}}
{"files":[{"path":"samples/zlib/dotzlib/readme.txt", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/gcc_gvmt64" ... "scan_errors":[]}]}}

```

(continues on next page)

(continued from previous page)

```

{"files":[{"path":"samples/zlib/gcc_gvmat64/gvmat64.S" ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/infbck9", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/infbck9/infbck9.c", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/infbck9/infbck9.h", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/iostream2", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/iostream2/zstream.h", ... "scan_errors":[]}]}}
{"files":[{"path":"samples/zlib/iostream2/zstream_test.cpp", ... "scan_errors
→":[]}]}}

```

Note: This jsonlines format also omits other file information like type, name, date, extension, sha1 and md5 hashes, programming language etc.

Comparing Different json Output Formats

Default --json Output:

```

sample-def-lic.json x
1 {"scancode_notice":"Generated with ScanCode and provided on an \"AS IS\" BASIS, WIT
2

```

--json-pp Output:

--json-lines Output:

--spdx-rdf FILE

SPDX stands for “Software Package and Data Exchange” and is an open standard for communicating software bill of material information (including components, licenses, copyrights, and security references).

The following code performs a scan on the samples directory, and publishes the results in **spdx-rdf** format:

```
scancode -clpieu --spdx-rdf output.spdx samples
```

Learn more about SPDX specifications [here](#) and in this [GitHub repository](#).

Here the file is structured as a dictionary of named properties and classes using W3C’s [RDF Technology](#).

```
{
  "path": "samples/JGroups/licenses/apache-1.1.txt",
  "type": "file",
  "name": "apache-1.1.txt",
  "base_name": "apache-1.1",
  "extension": ".txt",
  "date": "2019-09-18",
  "size": 2937,
  "sha1": "186d9195787fcbf2e5401b966159395640e06d11",
  "md5": "8c909d7735f28f4fdb0128ee57fb430e",
  "files_count": null,
  "mime_type": "text/plain",
  "file_type": "ASCII text, with CRLF line terminators",
  "programming_language": null,
  "is_binary": false,
  "is_text": true,
  "is_archive": false,
  "is_media": false,
  "is_source": false,
  "is_script": false,
  "scan_errors": [],
  "copyrights": [
    {
      "statements": [
        "Copyright (c) 2000 The Apache Software Foundation."
      ],
      "holders": [
        "The Apache Software Foundation."
      ],
      "authors": [],
      "start_line": 4,
      "end_line": 5
    },
    {
      "statements": [],
      "holders": [],
      "authors": [
        "the Apache Software Foundation"
      ],
      "start_line": 20,
      "end_line": 23
    }
  ],
  "packages": []
}
```

```
{
  "header": {
    "scancode_notice": "Generated with ScanCode and provided on an \"AS IS\" BASIS, WITHOUT WARRANTIES\\NOR CONDITIONS OF"
  },
  "files": [
    {
      "path": "samples/screenshot.png",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/README",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/zlib",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/arch",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/EULA",
      "scan_errors": [],
      "licenses": [
        {
          "key": "jboss-eula",
          "score": 100.0,
          "short_name": "JBoss EUL"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/LICENSE",
      "scan_errors": [],
      "licenses": [
        {
          "key": "lgpl-2.1-plus",
          "score": 100.0,
          "short_name": "LGP"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/licenses",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/licenses/lgpl.txt",
      "scan_errors": [],
      "licenses": [
        {
          "key": "lgpl-2.1-plus",
          "score": 100.0,
          "short_name": "LGPL"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/licenses/bouncycastle.txt",
      "scan_errors": [],
      "licenses": [
        {
          "key": "mit",
          "score": 100.0,
          "short_name": "MIT"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/licenses/apache-1.1.txt",
      "scan_errors": [],
      "licenses": [
        {
          "key": "apache-1.1",
          "score": 100.0,
          "short_name": "Apache 1.1"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/licenses/apache-2.0.txt",
      "scan_errors": [],
      "licenses": [
        {
          "key": "apache-2.0",
          "score": 100.0,
          "short_name": "Apache 2.0"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/licenses/cpl-1.0.txt",
      "scan_errors": [],
      "licenses": [
        {
          "key": "cpl-1.0",
          "score": 99.94,
          "short_name": "CPL 1.0"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/RouterStub.java",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/FixedMembershipToken.java",
      "scan_errors": [],
      "licenses": [
        {
          "key": "lgpl-2.1-plus",
          "score": 100.0,
          "short_name": "LGPL 2.1+
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/RATE_LIMITER.java",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [
        {
          "statements": [],
          "holder": "JBoss Inc.",
          "score": 100.0,
          "short_name": "JBoss Inc."
        }
      ],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/S3_PING.java",
      "scan_errors": [],
      "licenses": [
        {
          "key": "public-domain",
          "score": 10.0,
          "short_name": "Public Domain"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/GuardedBy.java",
      "scan_errors": [],
      "licenses": [
        {
          "key": "cc-by-2.5",
          "score": 70.0,
          "short_name": "CC BY 2.5"
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/ImmutableReference.java",
      "scan_errors": [],
      "licenses": [
        {
          "key": "lgpl-2.1-plus",
          "score": 100.0,
          "short_name": "LGPL 2.1+
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/JGroups/src/RouterStubManager.java",
      "scan_errors": [],
      "licenses": [
        {
          "key": "lgpl-2.1-plus",
          "score": 100.0,
          "short_name": "LGPL 2.1+
        }
      ],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/zlib/zutil.h",
      "scan_errors": [],
      "licenses": [
        {
          "key": "zlib",
          "score": 60.0,
          "short_name": "ZLIB License",
          "copyright_text": "Copyright (c) 1991, 1999 Free Software Foundation, Inc."
        }
      ],
      "copyrights": [
        {
          "statements": [
            "copyrighted by the Free Software Foundation"
          ],
          "holder": "Free Software Foundation",
          "score": 100.0,
          "short_name": "Free Software Foundation"
        }
      ],
      "packages": []
    },
    {
      "path": "samples/zlib/deflate.c",
      "scan_errors": [],
      "licenses": [
        {
          "key": "zlib",
          "score": 60.0,
          "short_name": "ZLIB License",
          "copyright_text": "Copyright (c) 1991, 1999 Free Software Foundation, Inc."
        }
      ],
      "copyrights": [
        {
          "statements": [
            "copyrighted by the Free Software Foundation"
          ],
          "holder": "Free Software Foundation",
          "score": 100.0,
          "short_name": "Free Software Foundation"
        }
      ],
      "packages": []
    },
    {
      "path": "samples/zlib/zlib.h",
      "scan_errors": [],
      "licenses": [
        {
          "key": "zlib",
          "score": 100.0,
          "short_name": "ZLIB License",
          "copyright_text": "Copyright (c) 1991, 1999 Free Software Foundation, Inc."
        }
      ],
      "copyrights": [
        {
          "statements": [
            "copyrighted by the Free Software Foundation"
          ],
          "holder": "Free Software Foundation",
          "score": 100.0,
          "short_name": "Free Software Foundation"
        }
      ],
      "packages": []
    },
    {
      "path": "samples/zlib/zutil.c",
      "scan_errors": [],
      "licenses": [
        {
          "key": "zlib",
          "score": 60.0,
          "short_name": "ZLIB License",
          "copyright_text": "Copyright (c) 1991, 1999 Free Software Foundation, Inc."
        }
      ],
      "copyrights": [
        {
          "statements": [
            "copyrighted by the Free Software Foundation"
          ],
          "holder": "Free Software Foundation",
          "score": 100.0,
          "short_name": "Free Software Foundation"
        }
      ],
      "packages": []
    },
    {
      "path": "samples/zlib/deflate.h",
      "scan_errors": [],
      "licenses": [
        {
          "key": "zlib",
          "score": 60.0,
          "short_name": "ZLIB License",
          "copyright_text": "Copyright (c) 1991, 1999 Free Software Foundation, Inc."
        }
      ],
      "copyrights": [
        {
          "statements": [
            "copyrighted by the Free Software Foundation"
          ],
          "holder": "Free Software Foundation",
          "score": 100.0,
          "short_name": "Free Software Foundation"
        }
      ],
      "packages": []
    },
    {
      "path": "samples/zlib/adler32.c",
      "scan_errors": [],
      "licenses": [
        {
          "key": "zlib",
          "score": 60.0,
          "short_name": "ZLIB License",
          "copyright_text": "Copyright (c) 1991, 1999 Free Software Foundation, Inc."
        }
      ],
      "copyrights": [
        {
          "statements": [
            "copyrighted by the Free Software Foundation"
          ],
          "holder": "Free Software Foundation",
          "score": 100.0,
          "short_name": "Free Software Foundation"
        }
      ],
      "packages": []
    },
    {
      "path": "samples/zlib/iostream2",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/zlib/dotzlib",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/zlib/gcc_gvmt64",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    },
    {
      "path": "samples/zlib/ada",
      "scan_errors": [],
      "licenses": [],
      "copyrights": [],
      "packages": []
    }
  ]
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:ns1="http://spdx.org/rdf/terms#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
  <ns1:SpdxDocument rdf:about="http://www.spdx.org/tools#SPDXANALYSIS">
    <ns1:referencesFile>
      <ns1:File rdf:nodeID="Ncb42239258db4eddbed44a0ef350c606">
        <ns1:licenseInfoInFile rdf:resource="http://spdx.org/licenses/LGPL-2.1+/">
        <ns1:checksum>
          <ns1:Checksum rdf:nodeID="Nbde3d8e53e734e1b81632bb15ce0bdea">
            <ns1:checksumValue>8f1a637d2e2ed1bdb9eb01a7dccb5c12cc0557e1</ns1:checksumValue>
            <ns1:algorithm>SHA1</ns1:algorithm>
          </ns1:Checksum>
        </ns1:checksum>
        <ns1:licenseConcluded rdf:resource="http://spdx.org/rdf/terms#noassertion"/>
        <ns1:copyrightText>Copyright (c) 1991, 1999 Free Software Foundation, Inc.
copyrighted by the Free Software Foundation
</ns1:copyrightText>
        <ns1:fileName>./samples/JGroups/licenses/lgpl.txt</ns1:fileName>
      </ns1:File>
    </ns1:referencesFile>
    <ns1:referencesFile>
      <ns1:File rdf:nodeID="Nd9ed3bcccc894c2b8579dbf7261932b8">
        <ns1:copyrightText>Copyright 2005, JBoss Inc.
</ns1:copyrightText>
      </ns1:File>
    </ns1:referencesFile>
  </ns1:SpdxDocument>
</rdf:RDF>
```

--spdx-tv FILE

This format is another SPDX variant, with the output file being structured in the following manner:

The following code performs a scan on the samples directory, and publishes the results in `spdx-tv` format:

```
scancode -clpieu --spdx-tv output.spdx samples
```

A SPDX-TV file starts with:

```
# Document Information

SPDXVersion: SPDX-2.1
DataLicense: CC0-1.0
DocumentComment: <text>Generated with ScanCode and provided on an "AS IS"
↳BASIS, WITHOUT WARRANTIES
OR CONDITIONS OF ANY KIND, either express or implied. No content created from
ScanCode should be considered or used as legal advice. Consult an Attorney
for any legal advice.
ScanCode is a free software code scanning tool from nexB Inc. and others.
Visit https://github.com/nexB/scancode-toolkit/ for support and download.</
↳text>

# Creation Info

Creator: Tool: ScanCode 2.2.1
Created: 2019-09-22T21:55:04Z
```

After a section titled `#Packages`, a list follows.

Each File information is listed under a `#File` title, for each of the files.

- FileName
- LicenseConcluded
- FileCopyrightText
- FileChecksum
- LicenseInfoInFile

An example goes as follows:

After the files section, there's a section for licenses under a `#Licences` title, with the following information for each license:

- LicenseID
- LicenseComment
- ExtractedText

Here's an example:


```
# Package

PackageName: samples
PackageDownloadLocation: NOASSERTION
PackageVerificationCode: ba63ad293ba27f95c8aa32ab097dc99895f35078
PackageLicenseDeclared: NOASSERTION
PackageLicenseConcluded: NOASSERTION
PackageLicenseInfoFromFiles: Apache-1.1
PackageLicenseInfoFromFiles: Apache-2.0
PackageLicenseInfoFromFiles: BSL-1.0
PackageLicenseInfoFromFiles: CC-BY-2.5
PackageLicenseInfoFromFiles: CPL-1.0
PackageLicenseInfoFromFiles: LGPL-2.1+
PackageLicenseInfoFromFiles: MIT
PackageLicenseInfoFromFiles: Zlib
PackageLicenseInfoFromFiles: LicenseRef-cmr-no
PackageLicenseInfoFromFiles: LicenseRef-gpl-2.0-plus-ada
PackageLicenseInfoFromFiles: LicenseRef-jboss-eula
PackageLicenseInfoFromFiles: LicenseRef-public-domain
PackageCopyrightText: <text>(c) 2004 by Henrik Ravn
(c) Copyright Henrik Ravn 2004
Copyright (c) 1991, 1999 Free Software Foundation, Inc.
Copyright (c) 1995-2005, 2010, 2011, 2012 Jean-loup Gailly.
Copyright (c) 1995-2008 Mark Adler
Copyright (c) 1995-2010 Jean-loup Gailly, Brian Raiter and Gilles Vollant.
Copyright (c) 1995-2011 Mark Adler
Copyright (c) 1995-2012 Jean-loup Gailly
```

```
# File

FileName: ./samples/JGroups/EULA
FileChecksum: SHA1: eb232aa0424eca9c4136904e6143b72aaa9cf4de
LicenseConcluded: NOASSERTION
LicenseInfoInFile: LicenseRef-jboss-eula
FileCopyrightText: <text>Copyright 2006 Red Hat, Inc.
</text>

# File

FileName: ./samples/JGroups/LICENSE
FileChecksum: SHA1: e60c2e780886f95df9c9ee36992b8edabec00bcc
LicenseConcluded: NOASSERTION
LicenseInfoInFile: LGPL-2.1+
FileCopyrightText: <text>Copyright (c) 1991, 1999 Free Software Foundation, Inc.
copyrighted by the Free Software Foundation
</text>
```

```
# Extracted Licenses

LicenseID: LicenseRef-cmr-no
LicenseComment: <text>See details at https://github.com/nexB/scancode-toolkit/blob/develop/src/licensedcode/data/licenses/cmr-no.yml
</text>
ExtractedText: <text>See details at https://github.com/nexB/scancode-toolkit/blob/develop/src/licensedcode/data/licenses/cmr-no.yml
</text>

LicenseID: LicenseRef-gpl-2.0-plus-ada
LicenseComment: <text>See details at https://github.com/nexB/scancode-toolkit/blob/develop/src/licensedcode/data/licenses/gpl-2.0-plus-ada.yml
</text>
ExtractedText: <text>See details at https://github.com/nexB/scancode-toolkit/blob/develop/src/licensedcode/data/licenses/gpl-2.0-plus-ada.yml
</text>
```

--html FILE

ScanCode supports formatting the Output result is a simple html format, to open with your favorite browser. This helps quick visualization of the detected license/copyright and other main information in the form of tables.

The following code performs a scan on the samples directory, and publishes the results in HTML format:

```
scancode -clpieu --html output.html samples
```

The HTML page generated has these following Tables:

- Copyright and Licenses Information
- File Information
- Package Information
- Licenses (Links to Dejacode/License Homepage)

Copyrights and Licenses Information				
path	start	end	what	value
samples/JGroups/EULA	3	108	license	jboss-eula
samples/JGroups/EULA	104	104	copyright	Copyright 2006 Red Hat, Inc.
samples/JGroups/LICENSE	1	502	license	lgpl-2.1-plus
samples/JGroups/LICENSE	4	7	copyright	Copyright (c) 1991, 1999 Free Software Foundation, Inc.
samples/JGroups/LICENSE	427	433	copyright	copyrighted by the Free Software Foundation
samples/JGroups/LICENSE	496	497	copyright	
samples/JGroups/licenses/gpl.txt	1	502	license	lgpl-2.1-plus
samples/JGroups/licenses/gpl.txt	4	7	copyright	Copyright (c) 1991, 1999 Free Software Foundation, Inc.
samples/JGroups/licenses/gpl.txt	427	433	copyright	copyrighted by the Free Software Foundation
samples/JGroups/licenses/gpl.txt	496	497	copyright	
samples/JGroups/licenses/bouncycastle.txt	5	5	copyright	Copyright (c) 2000 - 2006 The Legion Of The Bouncy Castle
samples/JGroups/licenses/bouncycastle.txt	7	18	license	mit
samples/JGroups/licenses/apache-1.1.txt	2	56	license	apache-1.1
samples/JGroups/licenses/apache-1.1.txt	4	5	copyright	Copyright (c) 2000 The Apache Software Foundation.
samples/JGroups/licenses/apache-1.1.txt	20	23	copyright	
samples/JGroups/licenses/apache-2.0.txt	2	202	license	apache-2.0

File Information							
path	type	name	extension	date	size	sha1	md5
samples/screenshot.png	file	screenshot.png	.png	2017-09-22	622754	01f4b1de0bc6c75c9cca6e46c80c1802d6976d4	b6ef5a90777147423c98b42a6a25e57a
samples/README	file	README		2019-09-18	238	598b3cadda718999199d9acbb7634f67dc31c9a6	f08e9167711dfa70aa0460f26cbd5cda
samples/JGroups	directory	JGroups		None	241280	None	None
samples/zlib	directory	zlib		None	268762	None	None
samples/arch	directory	arch		None	28103	None	None
samples/JGroups/EULA	file	EULA		2017-09-22	8156	eb232aa0424eca9c4136904e6143b72aaa9c4de	0be0aceb86296727eff0ac0bf8e6bdb3
samples/JGroups/LICENSE	file	LICENSE		2017-09-22	26430	e60c2e780886f95df9c9ee36992b8edabec00bcc	7fbc338309ac38fecd64b04bb903e34
samples/JGroups/licenses	directory	licenses		None	54604	None	None
samples/JGroups/src	directory	src		None	152090	None	None
samples/JGroups/licenses/lgpl.txt	file	lgpl.txt	.txt	2017-09-22	26934	8f1a637d2e2ed1bdb9eb01a7dccb5c12cc0557e1	f14599a2f0896f8c97e2baa4e3d575
samples/JGroups/licenses/bouncycastle.txt	file	bouncycastle.txt	.txt	2017-09-22	1186	74facb0e9a734479f9cd893b5be3fe1bf651b760	9fffd8de865a5705969f62b12838185
samples/JGroups/licenses/apache-1.1.txt	file	apache-1.1.txt	.txt	2019-09-18	2937	186d9195787fcbf2e5401b966159395640e06d11	8c909d7735f28f4db0128ee57fb430e
samples/JGroups/licenses/apache-2.0.txt	file	apache-2.0.txt	.txt	2017-09-22	11560	47b573e3824cd5e02a1a3ae99e2735b49e0256e4	d273d63619c9aef15cda7f6422c4f87
samples/JGroups/licenses/cpl-1.0.txt	file	cpl-1.0.txt	.txt	2017-09-22	11987	681cd776bcd79752543d42490ec7ed22a29fd888	9a6d2c9ae73d59eb3dd38e3909750d14
samples/JGroups/src/RouterStub.java	file	RouterStub.java	.java	2017-09-22	9913	c1f6818f8ee7bddcc9f444bc94c099729d716d52	eeefe23494acbcd8088c93bc1e83c7f2

Package Information		
path	type	packagin
samples/arch/zlib.tar.gz	plain tarball	archive

Licenses					
key	short_name	category	owner	reference_url	homepage_url
apache-1.1	Apache 1.1	Permissive	Apache Software Foundation	https://enterprise.dejacode.com/urn:urn:dje:license:apache-1.1	http://www.apache.org/licenses/
apache-2.0	Apache 2.0	Permissive	Apache Software Foundation	https://enterprise.dejacode.com/urn:urn:dje:license:apache-2.0	http://www.apache.org/licenses/
boost-1.0	Boost 1.0	Permissive	Boost	https://enterprise.dejacode.com/urn:urn:dje:license:boost-1.0	http://www.boost.org/users/license.html
cc-by-2.5	CC-BY-2.5	Permissive	Creative Commons	https://enterprise.dejacode.com/urn:urn:dje:license:cc-by-2.5	http://creativecommons.org/licenses/by/2.5/
cmr-no	CMR License	Permissive	CMR - Christian Michelsen Research AS	https://enterprise.dejacode.com/urn:urn:dje:license:cmr-no	
cpl-1.0	CPL 1.0	Copyleft Limited	IBM	https://enterprise.dejacode.com/urn:urn:dje:license:cpl-1.0	http://www.eclipse.org/legal/cpl-v1.0.html
gpl-2.0-plus-ada	GPL 2.0 or later with Ada exception	Copyleft Limited	Dmitriy Anisimkov	https://enterprise.dejacode.com/urn:urn:dje:license:gpl-2.0-plus-ada	
jboss-eula	JBoss EULA	Proprietary Free	JBoss Community	https://enterprise.dejacode.com/urn:urn:dje:license:iboss-eula	
lgpl-2.1-plus	LGPL 2.1 or later	Copyleft Limited	Free Software Foundation (FSF)	https://enterprise.dejacode.com/urn:urn:dje:license:lgpl-2.1-plus	http://www.gnu.org/licenses/old-licenses/lgpl-2.1-standalone.html
mit	MIT License	Permissive	MIT	https://enterprise.dejacode.com/urn:urn:dje:license:mit	http://opensource.org/licenses/mit-license.php
public-domain	Public Domain	Public Domain	Unspecified	https://enterprise.dejacode.com/urn:urn:dje:license:public-domain	http://www.linfo.org/publicdomain.html
zlib	ZLIB License	Permissive	zlib	https://enterprise.dejacode.com/urn:urn:dje:license:zlib	http://www.zlib.net/

Generated with ScanCode and provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. No content created from ScanCode should be considered scanning tool from nexB Inc. and others. Visit <http://www.nexb.com> and <https://github.com/nexB/scancode-toolkit/> for support and download.

--html-app FILE

ScanCode also supports formatting the output in a HTML visualization tool, which is more helpful than the standard HTML format.

Warning: The html-app feature has been deprecated and you should use Scancode Workbench instead to visualize scan results. The official Repository [link](#). Also refer *How to Visualize Scan results*.

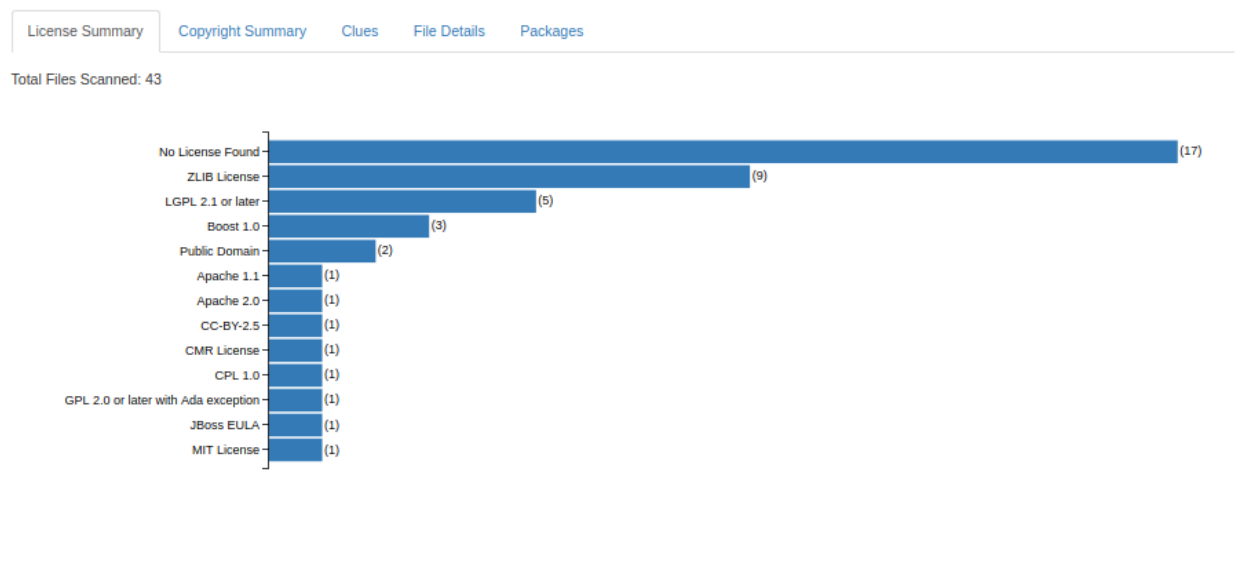
The following code performs a scan on the samples directory, and publishes the results in `html-app` format:

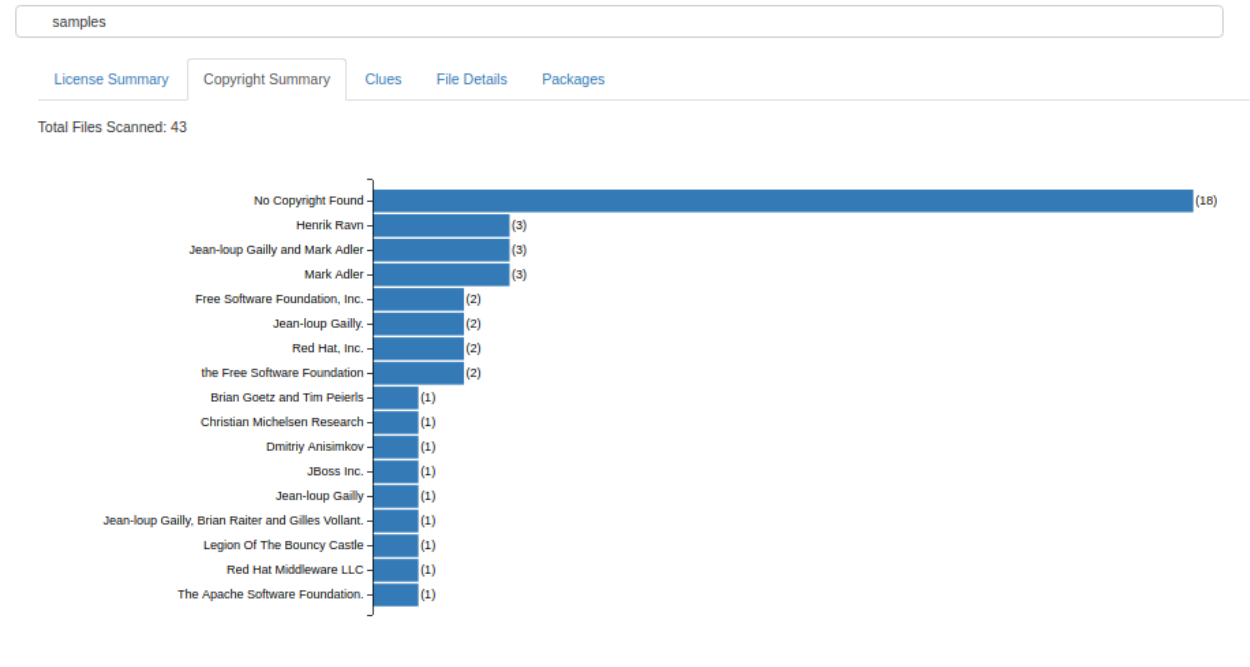
```
scancode -clpieu --html-app output.html samples
```

The Files scanned are shown in the left sidebar, and the section on the right contains separate tabs for the following:

- License Summary
- File Details
- Copyright Summary
- Packages
- Clues

Note: The HTML app also contains a Search option to easily find what you are looking for. But the HTML app output is deprecated and we recommend using `scancode-workbench` instead: <https://github.com/nexB/scancode-workbench>.





License Summary Copyright Summary Clues File Details Packages

Show 10 entries Search:

Path	Type	Name	Extension	Date	Size	SHA1	MD5	File count	MIME type
samples/arch	directory	arch			28103			1	
samples/arch/zlib.tar.gz	file	zlib.tar.gz	.tar.gz	2017-09-22	28103	576f0ccfe534d7f5ff5d6400078d3c6586de3abd	20b2370751abfc08bb3556c1d8114b5a		application/x-
samples/JGroups	directory	JGroups			241280			14	
samples/JGroups/EULA	file	EULA		2017-09-22	8156	eb232aa0424eca9c4136904e6143b72aaa9cf4de	0be0aceb8296727eff0ac0bf8e6bdb3		text/plain
samples/JGroups/LICENSE	file	LICENSE		2017-09-22	26430	e60c2e780886f95df9c9ee36992b8edabec00bcc	7fbc338309ac38fefd64b04bb903e34		text/plain
samples/JGroups/licenses	directory	licenses			54604			5	
samples/JGroups/licenses/apache-1.1.txt	file	apache-1.1.txt	.txt	2019-09-18	2937	186d9195787fcbf2e5401b966159395640e06d11	8c909d7735f28f4db0128ee57fb430e		text/plain
samples/JGroups/licenses/apache-2.0.txt	file	apache-2.0.txt	.txt	2017-09-22	11560	47b573e3824cd5e02a1a3ae99e2735b49e0256e4	d273d63619c9aeaf15cda176422c4f87		text/plain
samples/JGroups/licenses/bouncycastle.txt	file	bouncycastle.txt	.txt	2017-09-22	1186	74facb0e9a734479f9cd893b5be3fe1bf651b760	9fffd8de865a5705969f62b128381f85		text/plain
samples/JGroups/licenses/cpl-1.0.txt	file	cpl-1.0.txt	.txt	2017-09-22	11987	681cf776bcd79752543d42490ec7ed22a29fd888	9a6d2c9ae73d59eb3dd38e3909750d14		text/plain

Showing 1 to 10 of 43 entries

Previous 1 2 3 4 5 Next

--csv FILE

ScanCode can publish results in the useful `.csv` format.

Note: This option is deprecated and will be replaced by new CSV and tabular output formats in the next ScanCode release. Visit <https://github.com/nexB/scancode-toolkit/issues/3043> for details and to provide inputs and feedback.

The following code performs a scan on the samples directory, and publishes the results in `csv` format:

```
scancode -lpceiu --csv sample.csv samples
```

The first line of the `csv` file contains the headings, and they are:

- | | | |
|-------------------------|-------------------------------|----------------------------|
| • Resource, | • scan_errors, | • start_line, |
| • type, | • license__key, | • end_line, |
| • name, | • license__score, | • url, |
| • base_name, | • license__short_name, | • package__type, |
| • extension, | • license__category, | • package__name, |
| • date, | • license__owner, | • package__version, |
| • size, | • license__homepage_url, | • pack- |
| • sha1, | • license__text_url, | age__primary_language, |
| • md5, | • license__reference_url, | • package__summary, |
| • files_count, | • license__spdx_license_key, | • package__description, |
| • mime_type, | • license__spdx_url, | • package__size, |
| • file_type, | • matched_rule__identifier, | • package__release_date, |
| • programming_language, | • | • package__homepage_url, |
| • is_binary, | matched_rule__license_choice, | • package__notes, |
| • is_text, | • matched_rule__licenses, | • pack- |
| • is_archive, | • copyright, | age__bug_tracking_url, |
| • is_media, | • copyright_holder, | • package__vcs_repository, |
| • is_source, | • author, | • pack- |
| • is_script, | • email, | age__copyright_top_level |

Each subsequent line represents one element, i.e. can be any of the following:

- | | | | | |
|-----------|-------------|-----------|---------|-------|
| • license | • copyright | • package | • email | • url |
|-----------|-------------|-----------|---------|-------|

So if there's multiple elements in a file, they are each given an entry with the details mentioned earlier.

--cyclonedx FILE

ScanCode also supports the [CycloneDx](#) output format

Please note that this output format is only useful when scanning with the `--package` option

This output format is particularly useful if you want to process ScanCode results in downstream tools that can't process ScanCode's native JSON output, but do support CycloneDx BOMs.

To run an example scan on the test resources try: `./scancode --package --cyclonedx=bom.json tests/formattedcode/data/cyclonedx/simple`

If you prefer XML output over JSON, please have a look at the `--cyclonedx-xml` option instead

```

1 Resource,type,name,base name,extension,date,size,sha1,md5,files_count,mime_type,file_type,programming language,is binary,is text,is archive,is media,is source,i
s_script,scan_errors,license_key,license_score,license_short_name,license_category,license_owner,license_homepage_url,license_text_url,license_reference
url,license_spdx_license_key,license_spdx_url,matched_rule_identifier,matched_rule_license_choice,matched_rule_licenses,copyright,copyright_holder,author,
email,url,start_line,end_line,package_type,package_name,package_version,package_primary_language,package_summary,package_description,package_size,package
release_date,package_homepage_url,package_notes,package_bug_tracking_url,package_vcs_repository,package_copyright_top_level
2 /samples/screenshot.png,file,screenshot.png,screenshot.png,2017-09-22,622754,01ff4b1de0bc6c75c9cca6e46c80c1802d6976d4,b6ef5a90777147423c98b42a6a25e57a,,image/
png,"PNG image data, 2880 x 1666, 8-bit/color RGB, non-interlaced",,True,False,False,True,False,False,,text/plain,ASCII
3 /samples/README,file,README,README,,2019-09-18,238,598b3cadda718999199d9acbb7634f67dc31c9a6,f08e9167711dfa70aa0460f26cbd5cda,,text/plain,ASCII
text,,False,True,False,False,False,False,,
4 /samples/README,,http://zlib.net/zlib-1.2.8.tar.gz,3,3,,
5 /samples/README,,http://master.dl.sourceforge.net/project/javagroups/JGroups/2.10.0.GA/
JGroups-2.10.0.GA.src.zip,4,4,,
6 /samples/JGroups/,directory,JGroups,JGroups,,241280,,14,,False,False,False,False,False,False,,
7 /samples/zlib/,directory,zlib,zlib,,268762,,16,,False,False,False,False,False,False,,
8 /samples/arch/,directory,arch,arch,,28103,,1,,False,False,False,False,False,False,,
9 /samples/JGroups/EULA,file,EULA,EULA,,2017-09-22,8156,eb232aa0424eca9c4136904e6143b72aaa9cf4de,0be0aceb8296727efff0ac0bf8e6bdb3,,text/plain,ASCII
text,JavaScript+Lasso,False,True,False,False,True,False,,
10 /samples/JGroups/EULA,,jboss-eula,100.00,JBoss EULA,Proprietary Free,JBoss Community,http://repository.jboss.org/licenses/
jbossorg-eula.txt,https://enterprise.dejacode.com/urn:dje:license:jboss-eula,,jboss-eula.LICENSE,False,[u'jboss-eula'],,,3,108,,
11 /samples/JGroups/EULA,,Copyright 2006 Red Hat, Inc.,,,104,104,,
12 /samples/JGroups/EULA,,Red Hat, Inc.,,,104,104,,
13 /samples/JGroups/EULA,,http://www.opensource.org/licenses/index.php,24,24,,
14 /samples/JGroups/EULA,,http://www.jboss.org/,27,27,,
15 /samples/JGroups/EULA,,http://www.redhat.com/about/corporate/trademark,40,40,,
16 /samples/JGroups/EULA,,http://www.jboss.com/company/logos,43,43,,
17 /samples/JGroups/EULA,,http://www.redhat.com/licenses,94,94,,
18 /samples/JGroups/LICENSE,file,LICENSE,LICENSE,,2017-09-22,26430,e60c2e78886f95df9c9e36992b8edabec00bcc,7fbc338309ac38fefcd64b04bb903e34,,text/plain,ASCII
text,,False,True,False,False,False,,
19 /samples/JGroups/LICENSE,,lgpl-2.1-plus,100.00,LGPL 2.1 or later,Copyleft Limited,Free Software Foundation (FSF),http://www.gnu.org/licenses/
old-licenses/lgpl-2.1-standalone.html,http://www.gnu.org/licenses/old-licenses/lgpl-2.1-standalone.html,https://enterprise.dejacode.com/urn/
urn:dje:license:lgpl-2.1-plus,LGPL-2.1+,https://spdx.org/licenses/LGPL-2.1,lgpl-2.1-plus 2.RULE,False,[u'lgpl-2.1-plus'],,,1,502,,
20 /samples/JGroups/LICENSE,,Copyright (c) 1991, 1999 Free Software Foundation, Inc.,,,4,7,,
21 /samples/JGroups/LICENSE,,Free Software Foundation, Inc.,,,4,7,,
22 /samples/JGroups/LICENSE,,copyrighted by the Free Software Foundation,,427,433,,
23 /samples/JGroups/LICENSE,,the Free Software Foundation,,427,433,,
24 /samples/JGroups/LICENSE,,James Random Hacker,,496,497,,
25 /samples/JGroups/licenses/,directory,licenses,licenses,,54604,,5,,False,False,False,False,False,False,,
26 /samples/JGroups/src/,directory,src,src,,152090,,7,,False,False,False,False,False,False,,
27 /samples/JGroups/licenses/lgpl.txt,file,lgpl.txt,lgpl,,2017-09-22,26934,8f1a637d2e2ed1bdb9eb01a7dccb5c12cc0557e1,f14599a2f089f6ff8c97e2baa4e3d575,,text/
plain,"ASCII text, with CRLF line terminators",,False,True,False,False,False,,

```

--cyclonedx-xml FILE

This option allows outputting CycloneDx BOMs in XML format instead of JSON

To run an example scan on the test resources try: `./scancode --package --cyclonedx-xml=bom.xml tests/formattedcode/data/cyclonedx/simple`

Custom Output Format

While the three built-in output formats are convenient for a verity of use-cases, one may wish to create their own output template, using the following arguments:

```
``--custom-output FILE --custom-template TEMP_FILE``
```

ScanCode makes this very easy, as it uses the popular Jinja2 template engine. Simply pass the path to the custom template to the `--custom-template` argument, or drop it in a folder to `src/scancode/templates` directory.

For example, if I wanted a simple CLI output I would create a `template2.html` with the particular data I wish to see. In this case, I am only interested in the license and copyright data for this particular scan.

```

## template.txt:
[
    {% if files.license_copyright %}
        {% for location, data in files.license_copyright.items() %}
            {% for row in data %}
                location:"{{ location }}",

```

(continues on next page)

(continued from previous page)

```
{% if row.what == 'copyright' %}copyright:"{{ row.value|escape }}",{% endif %}
    {% endfor %}
    {% endfor %}
{% endif %}
]

.. note::

    File name and extension does not matter for the template file.
```

Now I can run ScanCode using my newly created template:

```
$ scancode -clpeui --custom-output output.txt --custom-template template.txt samples
Scanning files...
[#####] 46
Scanning done.
```

Now the results are saved in `output.txt` and we can easily view them with `head output.txt`:

```
[
  location:"samples/JGroups/LICENSE",
  copyright:"Copyright (c) 1991, 1999 Free Software Foundation, Inc.",

  location:"samples/JGroups/LICENSE",
  copyright:"copyrighted by the Free Software Foundation",
]
```

For a more elaborate template, refer this [default template](#) given with ScanCode, to generate HTML output with the `--html` output format option.

Documentation on [Jinja templates](#).

3.1.9 Controlling ScanCode Output and Filters

All “Output Control” Scan Options

- | | |
|---------------------|--|
| --strip-root | Strip the root directory segment of all paths. |
| --full-root | Report full, absolute paths. |

Note: The options `--strip-root` and `--full-root` can’t be used together, i.e. Any one option may be used in a single scan.

Note: The default is to always include the last directory segment of the scanned path such that all paths have a common root directory.

- ignore-author <pattern>** Ignore a file (and all its findings) if an author contains a match to the `<pattern>` regular expression.
- ignore-copyright-holder <pattern>** Ignore a file (and all its findings) if a copyright holder contains a match to the `<pattern>` regular expression.

Note: Note that this both the options `--ignore-author` and `--ignore-copyright-holder` will ignore a file even if it has other scanned data such as a license or errors.

--only-findings	Only return files or directories with findings for the requested scans. Files and directories without findings are omitted (file information is not treated as findings).
------------------------	---

--strip-root Vs. --full-root

For a default scan of the “samples” folder, this a comparison between the default, `strip-root` and `full-root` options.

An example Scan

```
scancode -cplieu --json-pp output.json samples --full-root
```

These two changes only the “path” attribute of the file information. For this comparison we compare the “path” attributes of the file `LICENSE` inside `JGroups` directory.

The default path:

```
"path": "samples/JGroups/LICENSE",
```

For the `--full-root` option, the path relative to the Root of your local filesystem.

```
"path": "/home/aboutcode/scancode-toolkit/samples/JGroups/LICENSE"
```

For the `--strip-root` option, the root directory (here `/home/aboutcode/scancode-toolkit/samples/`) is removed from path :

```
"path": "JGroups/LICENSE"
```

Note: The options `--strip-root` and `--full-root` can’t be used together, i.e. Any one option may be used in a single scan.

Note: The default is to always include the last directory segment of the scanned path such that all paths have a common root directory.

--ignore-author <pattern> Option

In a normal scan, all files inside the directory specified as an input argument is scanned and subsequently included in the scan report. But if you want to run the scan on only some selective files, with some specific **common author** then `--ignore-author` option can be used to do the same.

This scan ignores all files with authors matching the string “Apache Software Foundation”:

```
scancode -cplieu --json-pp output.json samples --ignore-author "Apache↵
↵Software Foundation"
```

More information on *Glob Pattern Matching*.

Note: Note that this both the options `--ignore-author` and `--ignore-copyright-holder` will ignore a file even if it has other scanned data such as a license or errors.

`--ignore-copyright-holder <pattern>` Option

In a normal scan, all files inside the directory specified as an input argument is scanned and subsequently included in the scan report. But if you want to run the scan on only some selective files, with some specific **common copyright holder** then `--ignore-copyright-holder` option can be used to do the same.

This scan ignores all files with Copyright Holders matching the string “Free Software Foundation”:

```
scancode -cplieu --json-pp output.json samples --ignore-copyright-holder "Free↵
↵Software Foundation"
```

More information on *Glob Pattern Matching*.

`--only-findings` Plugin

This option removes from the scan results, the files where nothing significant has been detected, like files which doesn't contain any licenses, copyrights, emails or urls (if requested in the scan options), and isn't a package.

An example Scan:

```
scancode -cplieu --json-pp output.json samples --only-findings
```

Note: This also changes in the result displayed, the number of files scanned.

For example, scanning the `sample` files (distributed by default with `scancode-toolkit`) without this option, displays in it's report information of 43 files. But after enabling this option, the result shows information for only 31 files.

3.1.10 Pre-Scan Options

All “Pre-Scan” Options

- `--ignore <pattern>` Ignore files matching `<pattern>`.
 - `--include <pattern>` Include files matching `<pattern>`.
 - `--classify` Classify files with flags telling if the file is a legal, or readme or test file, etc.
- Sub-Options:

- `--license-clarity-score`
- `--tallies-key-files`

`--facet <facet_pattern>` Here `<facet_pattern>` represents `<facet>=<pattern>`. Add the `<facet>` to files with a path matching `<pattern>`.

Sub-Options:

- `--tallies-by-facet`
-

`--ignore` Option

In a scan, all files inside the directory specified as an input argument is scanned. But if there are some files which you don't want to scan, the `--ignore` option can be used to do the same.

A sample usage:

```
scancode --ignore "*.java" samples samples.json
```

Here, Scancode ignores files ending with `.java`, and continues with other files as usual.

More information on [Glob Pattern Matching](#).

`--include` Option

In a normal scan, all files inside the directory specified as an input argument is scanned. But if you want to run the scan on only some selective files, then `--include` option can be used to do the same.

A sample usage:

```
scancode --include "*.java" samples samples.json
```

Here, Scancode selectively scans files that has names ending with `.java`, and ignores all other files. This is basically complementary in behavior to the `--ignore` option.

More information on [Glob Pattern Matching](#).

`--classify`

Sub-Option

The options `--license-clarity-score` and `--tallies-key-files` are sub-options of `--classify`. `--license-clarity-score` and `--tallies-key-files` are Post-Scan Options.

The `--classify` option can be used like:

```
scancode -clpieu --json-pp sample_facet.json samples --classify
```

This option makes ScanCode further classify scanned files/directories, to determine whether they fall in these following categories

- legal
- readme
- top-level
- manifest

A manifest file in computing is a file containing metadata for a group of accompanying files that are part of a set or coherent unit.

- key-file

A KEY file is a generic file extension used by various programs when registering legal copies of the software. It may be saved in a plain text format, but generally contains some form of encrypted key string that authenticates the purchase and registers the software.

As in, to the JSON object of each file scanned, these extra attributes are added:

```
{
  "is_legal": false,
  "is_manifest": false,
  "is_readme": true,
  "is_top_level": true,
  "is_key_file": true
}
```

--facet Option

Sub-Option

The option `--summary-by-facet` is a sub-option of `--facet`. `--summary-by-facet` is a Post-Scan Option.

Valid `<facet>` values are:

- core,
- dev,
- tests,
- docs,
- data,
- examples.

You can use the `--facet` option in the following manner:

```
scancode -clpieu --json-pp sample_facet.json samples --facet dev="*.java" --
↪ facet dev="*.c"
```

This adds to the header object, the following attribute:

```
--facet": [
  "dev=*.java",
  "dev=*.c"
],
```

Here in this example, .java and .c files are marked as it belongs to facet dev.

As a result, .java file has the following attribute added:

```
"facets": [
  "dev"
],
```

Note: All other files which are not dev are marked to be included in the facet core.

For each facet, the --facet option precedes the <facet>=<pattern> argument. For specifying multiple facets, this whole part is repeated, including the --facet option.

For users who want to know *What is a Facet?*.

Glob Pattern Matching

All the Pre-Scan options use pattern matching, so the basics of Glob Pattern Matching is discussed briefly below.

Glob pattern matching is useful for matching a group of files, by using patterns in their names. Then using these patterns, files are grouped and treated differently as required.

Here are some rules from the [Linux Manual](#) on glob patterns. Refer the same for more detailed information.

A string is a wildcard pattern if it contains one of the characters '?', '*' or '['. Globbing is the operation that expands a wildcard pattern into the list of pathnames matching the pattern. Matching is defined by:

- A '?' (not between brackets) matches any single character.
- A '*' (not between brackets) matches any string, including the empty string.
- An expression "[...]" where the first character after the leading '[' is not an '!' matches a single character, namely any of the characters enclosed by the brackets.
- There is one special convention: two characters separated by '-' denote a range.
- An expression "[!...]" matches a single character, namely any character that is not matched by the expression obtained by removing the first '!' from it.
- A '/' in a pathname cannot be matched by a '?' or '*' wildcard, or by a range like "[.-0]".

Note that wildcard patterns are not regular expressions, although they are a bit similar.

For more information on Glob pattern matching refer these resources:

- [Linux Manual](#)
- [Wildcard Match Documentation](#).

You can also import these Python Libraries to practice UNIX style pattern matching:

- [fnmatch](#) for File Name matching

- [glob](#) for File Path matching

What is a Facet?

A facet is essentially a file purpose classification label. It is defined as follows (by ClearlyDefined):

A facet of a component is a subset of the files related to the component. It's really just a grouping that helps us understand the shape of the project. Each facet is described by a set of glob expressions, essentially wildcard patterns that are matched against file names.

Each facet definition can have zero or more glob expressions. A file can be captured by more than one facet. Any file found but not captured by a defined facet is automatically assigned to the core facet.

- **core** - The files that go into making the release of the component. Note that the core facet is not explicitly defined. Rather, it is made up of whatever is not in any other facet. So, by default, all files are in the core facet unless otherwise specified.
- **data** - The files included in any data distribution of the component.
- **dev** - Files primarily used at development time (e.g., build utilities) and not distributed with the component
- **docs** - Documentation files. Docs may be included with the executable component or separately or not at all.
- **examples** - Like docs, examples may be included in the main component release or separately.
- **tests** - Test files may include code, data and other artifacts.

Important Links:

- [Facets](#)
- [ClearlyDefined](#)

3.1.11 Post-Scan Options

Post-Scan options activate their respective post-scan plugins which execute the task.

All “Post-Scan” Options

--mark-source	Set the “is_source” flag to true for directories that contain over 90% of source files as direct children and descendants. Count the number of source files in a directory as a new “source_file_counts” attribute Sub-Option of: --url
--consolidate	Group resources by Packages or license and copyright holder and return those groupings as a list of consolidated packages and a list of consolidated components. The --consolidate option will be deprecated in a future version of scancode-toolkit as top level packages now provide improved consolidated data. Sub-Option of: --copyright , --license and --packages .
--filter-clues	Filter redundant duplicated clues already contained in detected licenses, copyright texts and notices.

- license-clarity-score** Compute a summary license clarity score at the codebase level.
Sub-Option of: **--classify**.
- license-policy FILE** Load a License Policy file and apply it to the scan at the Resource level.
- summary** Summarize scans by providing declared origin information and other detected info at the codebase attribute level.
- tallies** Summarize license, copyright and other scans at the codebase level with occurrence counts.
Sub-Options:
- **--tallies-by-facet**
 - **--tallies-key-files**
 - **--tallies-with-details**
- tallies-by-facet** Summarize license, copyright and other scans and group the results by facet.
Sub-Option of: **--tallies** and **--facet**.
- tallies-key-files** Summarize license, copyright and other scans for key, top-level files, with occurrence counts. Key files are top-level codebase files such as COPYING, README and package manifests as reported by the **--classify** option: “is_legal”, “is_readme”, “is_manifest” and “is_top_level” flags.
Sub-Option of: **--classify** and **--summary**.
- tallies-with-details** Summarize license, copyright and other scans at the codebase level with occurrence counts, while also keeping intermediate details at the file and directory level.

To see all plugins available via command line help, use **--plugins**.

Note: Plugins that are shown by using **--plugins** include the following:

1. Post-Scan Plugins (and, the following)
 2. Pre-Scan Plugins
 3. Output Options
 4. Output Control
 5. Basic Scan Options
-

--mark-source Option

Dependency

The option **--mark-source** is a sub-option of and **requires** the option **--info**.

The **mark-source** option marks the **is_source** attribute of a directory to be **True**, if more than 90% of the files under that directory is source files, and **False** otherwise.

When the following command is executed to scan the **samples** directory with this option enabled:

```
scancode -clpieu --json-pp output.json samples --mark-source
```

Then, the following directories are marked as “Source”, i.e. their `is_source` attribute is set to `True`, as they contain mostly source code.

- `samples/JGroups/src`
- `samples/zlib/iostream2`
- `samples/zlib/gcc_gvmat64`
- `samples/zlib/ada`
- `samples/zlib/infbck9`

--consolidate Option

Dependency

The option `--consolidate` is a sub-option of and **requires** the options `--license`, `--copyright` and `--package`.

Note: The `--consolidate` option will be deprecated in a future version of ScanCode Toolkit as top level packages, dependencies and licenses now provide improved consolidated data.

The JSON file containing scan results after using the `--consolidate` Plugin is structured as follows:

An example Scan:

```
scancode -clpieu --json-pp output.json samples --consolidate
```

The JSON output file is structured as follows:

```
{
  "headers": [...],
  "consolidated_components": [
    {
      "type": "license-holders",
      "identifier": "dmitriy_anisimkov_1",
      "consolidated_license_expression": "gpl-2.0-plus WITH ada-linking-
↪exception",
      "consolidated_holders": [
        "Dmitriy Anisimkov"
      ],
      "consolidated_copyright": "Copyright (c) Dmitriy Anisimkov",
      "core_license_expression": "gpl-2.0-plus WITH ada-linking-exception",
      "core_holders": [
        "Dmitriy Anisimkov"
      ],
      "other_license_expression": null,
      "other_holders": [],
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    "files_count": 1
  },
  { ...
  }
],
"consolidated_packages": [...],
"files": [...]
}

```

Each consolidated component has the following information:

```

"consolidated_components": [
{
  "type": "license-holders",
  "identifier": "dmitriy_anisimkov_1",
  "consolidated_license_expression": "gpl-2.0-plus WITH ada-linking-exception",
  "consolidated_holders": [
    "Dmitriy Anisimkov"
  ],
  "consolidated_copyright": "Copyright (c) Dmitriy Anisimkov",
  "core_license_expression": "gpl-2.0-plus WITH ada-linking-exception",
  "core_holders": [
    "Dmitriy Anisimkov"
  ],
  "other_license_expression": null,
  "other_holders": [],
  "files_count": 1
},

```

In addition to this, in every file/directory where the consolidated part (i.e. License information) was present, a “consolidated_to” attribute is added pointing to the “identifier” of “consolidated_components”:

```

"consolidated_to": [
  "dmitriy_anisimkov_1"
],

```

Note that multiple files may have the same “consolidated_to” attribute.

--filter-clues Option

The --filter-clues Plugin filters redundant duplicated clues already contained in detected licenses, copyright texts and notices, authors.

Consider the output of running the following scan (compared to running the scan without the --filter-clues option):

```
./scancode -clpieu --json-pp sample_filter_clues.json samples --filter-clues
```

When we run without the --filter-clues option, we have the following detections at "path": "samples/JGroups/src/FixedMembershipToken.java":

```
{
  "authors": [
    {
      "author": "Chris Mills (millsy@jboss.com)",
      "start_line": 51,
      "end_line": 51
    }
  ],
  "emails": [
    {
      "email": "millsy@jboss.com",
      "start_line": 51,
      "end_line": 51
    }
  ]
}
```

And when we run a scan with the `--filter-clues` option:

```
{
  "authors": [
    {
      "author": "Chris Mills (millsy@jboss.com)",
      "start_line": 51,
      "end_line": 51
    }
  ],
  "emails": []
}
```

Notice that when we run the scan with the `--filter-clues` option, we do not have the *millsy@jboss.com* in email detections as we already have it in author detections.

`--license-clarity-score` Option

Dependency

The option `--license-clarity-score` is a sub-option of and requires the option `--classify`.

Keep this doc section in sync with docstrings at: `src/summarycode/score.py::compute_license_score`

The `--license-clarity-score` plugin when used in a scan, computes a summary license clarity score at the codebase level. The license clarity score is a value from 0-100 calculated by combining the weighted values determined for each of the scoring elements:

Declared license:

- When true, indicates that the software package licensing is documented at top-level or well-known locations in the software project, typically in a package manifest, NOTICE, LICENSE, COPYING or README file.

- Scoring Weight = 40

Identification precision:

- Indicates how well the license statement(s) of the software identify known licenses that can be designated by precise keys (identifiers) as provided in a publicly available license list, such as the ScanCode LicenseDB, the SPDX license list, the OSI license list, or a URL pointing to a specific license text in a project or organization website.
- Scoring Weight = 40

License texts:

- License texts are provided to support the declared license expression in files such as a package manifest, NOTICE, LICENSE, COPYING or README.
- Scoring Weight = 10

Declared copyright:

- When true, indicates that the software package copyright is documented at top-level or well-known locations in the software project, typically in a package manifest, NOTICE, LICENSE, COPYING or README file.
- Scoring Weight = 10

Ambiguous compound licensing

- When true, indicates that the software has a license declaration that makes it difficult to construct a reliable license expression, such as in the case of multiple licenses where the conjunctive versus disjunctive relationship is not well defined.
- Scoring Weight = -10

Conflicting license categories

- When true, indicates the declared license expression of the software is in the permissive category, but that other potentially conflicting categories, such as copyleft and proprietary, have been detected in lower level code.
- Scoring Weight = -20

An example Scan:

```
scancode -clpieu --json-pp output.json samples --classify --license-clarity-
↪score
```

The “license_clarity_score” will have the following attributes:

- | | | |
|-----------------------------|-------------------------|-------------------------|
| • “score” | • “has_license_text” | ing_license_categories” |
| • “declared_license” | • “declared_copyrights” | • “ambigu- |
| • “identification_precision | • “conflict- | ous_compound_licensing” |

When the “license_clarity_score” is included, the entire JSON file is structured as follows:

```
{
  "headers": [...],
  "summary": {
    "declared_license_expression": "mit",
    "license_clarity_score": {
      "score": 100,
```

(continues on next page)

(continued from previous page)

```

    "declared_license": true,
    "identification_precision": true,
    "has_license_text": true,
    "declared_copyrights": true,
    "conflicting_license_categories": false,
    "ambiguous_compound_licensing": false
  },
  "files": [...]
}

```

Note: When the `--license-clarity-score` option is used, the output is added as the following attributes:

- `declared_license_expression`
- `license_clarity_score` (with the score and other flags as sub-attributes)

in the top-level `summary` attribute, but the `--summary` CLI option is not required for this. Using the `--summary` CLI option also populates the same top-level `summary` attribute with the license clarity score.

`--license-policy` FILE Option

Note: The `--license-policy` option does not have any required CLI options, but you would not have any usable information if you are using it without the `--license` option since this only gets license keys from the file license detections. We do not have licenses as a required option because this plugin would be upgraded to also include the license policy attribute for packages too.

The Policy file is a YAML (.yaml) document with the following structure:

```

license_policies:
- license_key: mit
  label: Approved License
  color_code: '#008000'
  icon: icon-ok-circle
- license_key: agpl-3.0
  label: Approved License
  color_code: '#008000'
  icon: icon-ok-circle

```

Note: In the policy file only the “`license_key`” is a required field.

Applying License Policies during a ScanCode scan, using the `--license-policy` Plugin:

```

scancode -clipeu --json-pp output.json samples --license-policy policy-file.yaml

```

This adds to every file/directory an object “`license_policy`”, having as further attributes under it the fields as specified in the .YAML file. Here according to our example .YAML file, the attributes will be:

- “license_key”
- “label”
- “color_code”
- “icon”

Here the samples directory is scanned, and the Scan Results for a sample file is as follows:

```
{
  "path": "samples/JGroups/licenses/apache-2.0.txt",
  "license_detections": [
    {
      "license_expression": "apache-2.0",
      "matches": {...}
      "identifier": "apache_2_0-9804422e-94ac-ad40-b53a-ee6f8ddb7a3b"
    }
  ],
  "detected_license_expression": "apache-2.0",
  "detected_license_expression_spdx": "Apache-2.0",
  "license_policy": {
    "license_key": "apache-2.0",
    "label": "Approved License",
    "color_code": "#008000",
    "icon": "icon-ok-circle"
  },
  ...
},
```

More information on the *License Policy Plugin* and usage.

--license-references Option

Dependency

The option `--license-references` is a sub-option of and requires the option `--license`.

Details about the matched license or license rule are not included with the license matches for license detections by default. These are instead reported optionally and separately as codebase-level reference data. There are two codebase-level attributes added with the `--license-references` option:

- `license_references` with details from scancode licenses (which are each a `.LICENSE` file)
- `license_rule_references` with details from scancode license rules (which are each a `.RULE` file)

Consider a file `mit.txt` with the following license declaration:

```
License: mit
```

We run the following scan on this file:

```
scancode -l --license-text --license-references mit.txt --json-pp mit.json
```

See the results for this license scan with `--license-references` enabled:

```
{
  "headers": [...],
  "license_detections": [
```

(continues on next page)

(continued from previous page)

```

{
  "identifier": "mit-3fce6ea2-8abd-6c6b-3ede-a37af7c6efee",
  "license_expression": "mit",
  "detection_count": 1
}
],
"license_references": [
{
  "key": "mit",
  "language": "en",
  "short_name": "MIT License",
  "name": "MIT License",
  "category": "Permissive",
  "owner": "MIT",
  "homepage_url": "http://opensource.org/licenses/mit-license.php",
  "notes": "Per SPDX.org, this license is OSI certified.",
  "is_builtin": true,
  "is_exception": false,
  "is_unknown": false,
  "is_generic": false,
  "spdx_license_key": "MIT",
  "other_spdx_license_keys": [],
  "osi_license_key": null,
  "text_urls": [
    "http://opensource.org/licenses/mit-license.php"
  ],
  "osi_url": "http://www.opensource.org/licenses/MIT",
  "faq_url": "https://ieeexplore.ieee.org/document/9263265",
  "other_urls": [
    "https://opensource.com/article/18/3/patent-grant-mit-license",
    "https://opensource.com/article/19/4/history-mit-license",
    "https://opensource.org/licenses/MIT"
  ],
  "key_aliases": [],
  "minimum_coverage": 0,
  "standard_notice": null,
  "ignorable_copyrights": [],
  "ignorable_holders": [],
  "ignorable_authors": [],
  "ignorable_urls": [],
  "ignorable_emails": [],
  "text": "Permission is hereby granted, free of charge, to any person_
↪ obtaining\
na 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,\
↪ \
ndistribute, sublicense, and/or sell copies of the Software, and to\
↪ \
npermit_
↪ persons to whom the Software is furnished to do so, subject to\
↪ \
nthe_
↪ following conditions:\
n\
nThe above copyright notice and this permission_
↪ notice shall be\
nincluded in all copies or substantial portions of the_
↪ Software.\
n\
nTHE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT WARRANTY OF ANY_
↪ KIND,\
nEXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF\
↪ \
nMERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.\
nIN_

```

(continues on next page)

(continued from previous page)

```

→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.",
    "scancode_url": "https://github.com/nexB/scancode-toolkit/tree/develop/
→src/licensedcode/data/licenses/mit.LICENSE",
    "licensedb_url": "https://scancode-licensedb.aboutcode.org/mit",
    "spdx_url": "https://spdx.org/licenses/MIT"
  }
],
"license_rule_references": [
  {
    "license_expression": "mit",
    "identifier": "mit_30.RULE",
    "language": "en",
    "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
→licensedcode/data/rules/mit_30.RULE",
    "is_license_text": false,
    "is_license_notice": false,
    "is_license_reference": false,
    "is_license_tag": true,
    "is_license_intro": false,
    "is_continuous": false,
    "is_builtin": true,
    "is_from_license": false,
    "is_synthetic": false,
    "length": 2,
    "relevance": 100,
    "minimum_coverage": 100,
    "referenced_filenames": [],
    "notes": null,
    "ignorable_copyrights": [],
    "ignorable_holders": [],
    "ignorable_authors": [],
    "ignorable_urls": [],
    "ignorable_emails": [],
    "text": "License: MIT"
  }
],
"files": [
  {
    "path": "mit.txt",
    "type": "file",
    "detected_license_expression": "mit",
    "detected_license_expression_spdx": "MIT",
    "license_detections": [
      {
        "license_expression": "mit",
        "matches": [
          {
            "score": 100.0,
            "start_line": 1,

```

(continues on next page)

(continued from previous page)

```

        "end_line": 1,
        "matched_length": 2,
        "match_coverage": 100.0,
        "matcher": "1-hash",
        "license_expression": "mit",
        "rule_identifier": "mit_30.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/
→develop/src/licensedcode/data/rules/mit_30.RULE",
        "matched_text": "License: mit"
    }
],
    "identifier": "mit-3fce6ea2-8abd-6c6b-3ede-a37af7c6efee"
}
],
"license_clues": [],
"percentage_of_license_text": 100.0,
"scan_errors": []
}
]
}

```

See *Only reference License related data* for more details on license references and a comparison with previous scancode output formats.

--summary Option

Sub-Option

The option `--summary-by-facet`, `--summary-key-files` and `--summary-with-details` are sub-options of `--summary`. These Sub-Options are all Post-Scan Options.

An example Scan:

```
scancode -clpieu --json-pp output.json samples --summary
```

The whole JSON file is structured as follows, when the `--summary` plugin is applied:

```

{
  "headers": [...],
  "summary": {
    "declared_license_expression": null,
    "license_clarity_score": {...},
    "declared_holder": "",
    "primary_language": "C",
    "other_license_expressions": [...],
    "other_holders": [...],
    "other_languages": [...]
  },

```

(continues on next page)

(continued from previous page)

```

"files": [...]
}

```

Each attribute in `other_license_expressions`, `other_holders`, `other_languages` has multiple entries each containing “value” and “count”, with their values having the summary information inside them.

See below a sample fully populated summary object:

```

{
  "summary": {
    "declared_license_expression": "commercial-license AND other-permissive AND ↵
↵mit",
    "license_clarity_score": {
      "score": 100,
      "declared_license": true,
      "identification_precision": true,
      "has_license_text": true,
      "declared_copyrights": true,
      "conflicting_license_categories": false,
      "ambiguous_compound_licensing": false
    },
    "declared_holder": "Strapi Solutions SAS",
    "primary_language": "JavaScript",
    "other_license_expressions": [
      {
        "value": "commercial-license AND other-permissive AND mit",
        "count": 65
      },
      {
        "value": "mit",
        "count": 7
      },
      {
        "value": null,
        "count": 1
      },
      {
        "value": "apache-2.0",
        "count": 1
      },
      {
        "value": "generic-cla",
        "count": 1
      }
    ],
    "other_holders": [
      {
        "value": null,
        "count": 3572
      },
      {
        "value": "Jon Schlinkert",
        "count": 2
      }
    ]
  }
}

```

(continues on next page)

(continued from previous page)

```

    }
  ],
  "other_languages": [
    {
      "value": "TypeScript",
      "count": 91
    },
    {
      "value": "GAS",
      "count": 28
    },
    {
      "value": "HTML",
      "count": 6
    },
    {
      "value": "Bash",
      "count": 5
    },
    {
      "value": "verilog",
      "count": 1
    }
  ]
}

```

--tallies Option

Optional Dependency

The `--tallies` option does not have any required CLI option dependencies, but as it contains license, copyright, holder, author, packages and programming language information, it is recommended to use this option with `--license`, `--package`, `--copyright` and `--info` options enabled, or there will not be any corresponding data for these.

An example scan using the `--tallies` Plugin:

```
scancode -clipecu --json-pp strapi.json strapi-main/ --tallies
```

The JSON file containing the `--tallies` scan results are as follows:

```

{
  "headers": [...],
  "packages": [...],
  "dependencies": [...],
  "license_detections": [...],
  "tallies": {
    "detected_license_expression": [
      {

```

(continues on next page)

(continued from previous page)

```

    "value": "commercial-license AND other-permissive AND mit",
    "count": 65
  },
  {
    "value": "mit",
    "count": 7
  },
  {
    "value": null,
    "count": 1
  },
  {
    "value": "apache-2.0",
    "count": 1
  },
  {
    "value": "generic-cla",
    "count": 1
  }
],
"copyrights": [
  {
    "value": null,
    "count": 3572
  },
  {
    "value": "Copyright (c) Strapi Solutions SAS",
    "count": 31
  },
  {
    "value": "Copyright (c) Jon Schlinkert",
    "count": 2
  }
],
"holders": [
  {
    "value": null,
    "count": 3572
  },
  {
    "value": "Strapi Solutions SAS",
    "count": 31
  },
  {
    "value": "Jon Schlinkert",
    "count": 2
  }
],
"authors": [
  {
    "value": null,
    "count": 3567
  }
]

```

(continues on next page)

(continued from previous page)

```
    },
    {
      "value": "name' Strapi Solutions",
      "count": 30
    },
    {
      "value": "the community",
      "count": 4
    },
    {
      "value": "name' A Strapi developer",
      "count": 3
    },
    {
      "value": "name A Strapi",
      "count": 1
    },
    {
      "value": "name' Yurii Tykhomyrov",
      "count": 1
    }
  ],
  "programming_language": [
    {
      "value": "JavaScript",
      "count": 2854
    },
    {
      "value": "TypeScript",
      "count": 91
    },
    {
      "value": "GAS",
      "count": 28
    },
    {
      "value": "HTML",
      "count": 6
    },
    {
      "value": "Bash",
      "count": 5
    },
    {
      "value": "verilog",
      "count": 1
    }
  ],
  "packages": [...],
  "files": [...]
}
```

This adds a top-level “tallius” attribute and the sub-attributes will be:

- “detected_license_expression”
- “copyright”
- “holders”
- “programming_language”
- “ages”
- “authors”
- “packaging”

These are all lists with the corresponding “value” and their respective “count”, basically tallies of all different values.

--tallies-by-facet Option

Dependency

The option `--tallies-by-facet` is a sub-option of and requires the options `--facet` and `--tallies`.

For users who want to know *What is a Facet?*.

Running the scan with `--tallies --tallies-by-facet` Plugins creates individual summaries for all the facets with the same license, copyright and other scan information, at a codebase level (in addition to the codebase level general summary generated by `--tallies` Plugin). Once all files have been assigned a facet, files without a facet are assigned to the core facet.

An example scan using the `--tallies-by-facet` Plugin:

```
scancode -clipecu --json-pp strapi.json strapi-main/ --tallies --facet dev="*.js"
→ " --facet dev="*.ts" --tallies-by-facet
```

We have used the [github:strapi/strapi](https://github.com:strapi/strapi) project to generate example results for this CLI option.

Note: All other files which are not dev are marked to be included in the facet `core`.

A sample “summary_by_facet” object generated by the previous scan (shortened):

```
{
  "headers": [...],
  "packages": [...],
  "dependencies": [...],
  "license_detections": [...],
  "tallies": {...}
  "tallies_by_facet": [
    {
      "facet": "core",
      "tallies": {
        "detected_license_expression": [
          {
            "value": "commercial-license AND other-permissive AND mit",
            "count": 65
          },
          {
            "value": "mit",
            "count": 5
          }
        ]
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
    },
    {
      "value": "generic-cla",
      "count": 1
    }
  ],
  "copyrights": [
    {
      "value": "Copyright (c) Strapi Solutions SAS",
      "count": 31
    }
  ],
  "holders": [
    {
      "value": "Strapi Solutions SAS",
      "count": 31
    }
  ],
  "authors": [
    {
      "value": "name' Strapi Solutions",
      "count": 30
    },
    {
      "value": "name' A Strapi developer",
      "count": 3
    },
    {
      "value": "name' Yurii Tykhomyrov",
      "count": 1
    },
    {
      "value": "the community",
      "count": 1
    }
  ],
  "programming_language": [
    {
      "value": "GAS",
      "count": 28
    },
    {
      "value": "TypeScript",
      "count": 7
    },
    {
      "value": "HTML",
      "count": 6
    },
    {
      "value": "Bash",
      "count": 5
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
    },
    {
      "value": "verilog",
      "count": 1
    }
  ]
}
},
{
  "facet": "dev",
  "tallies": {
    "detected_license_expression": [
      {
        "value": "mit",
        "count": 2
      },
      {
        "value": "apache-2.0",
        "count": 1
      }
    ],
    "copyrights": [
      {
        "value": "Copyright (c) Jon Schlinkert",
        "count": 2
      }
    ],
    "holders": [
      {
        "value": "Jon Schlinkert",
        "count": 2
      }
    ],
    "authors": [
      {
        "value": "the community",
        "count": 3
      },
      {
        "value": "name A Strapi",
        "count": 1
      }
    ],
    "programming_language": [
      {
        "value": "JavaScript",
        "count": 2854
      },
      {
        "value": "TypeScript",
        "count": 84
      }
    ]
  }
}
```

(continues on next page)

(continued from previous page)

```
    ]
  }
},
{
  "facet": "tests",
  "tallies": {
    "detected_license_expression": [],
    "copyrights": [],
    "holders": [],
    "authors": [],
    "programming_language": []
  }
},
{
  "facet": "docs",
  "tallies": {
    "detected_license_expression": [],
    "copyrights": [],
    "holders": [],
    "authors": [],
    "programming_language": []
  }
},
{
  "facet": "data",
  "tallies": {
    "detected_license_expression": [],
    "copyrights": [],
    "holders": [],
    "authors": [],
    "programming_language": []
  }
},
{
  "facet": "examples",
  "tallies": {
    "detected_license_expression": [],
    "copyrights": [],
    "holders": [],
    "authors": [],
    "programming_language": []
  }
}
],
"files": [...]
}
```

Note: Summaries for all the facets are generated by default, regardless of facets not having any files under them.

--tallies-key-files Option

Dependency

The option `--tallies-key-files` is a sub-option of and requires the options `--classify` and `--tallies`.

An example Scan:

```
scancode -clipecu --json-pp strapi.json strapi-main/ --classify --tallies --  
↪tallies-key-files
```

Running the scan with `--tallies --tallies-key-files` plugins creates summaries for key files with the same license, copyright and other scan information, at a codebase level (in addition to the codebase level general summary generated by `--tallies` Plugin).

The resulting JSON file containing the scan results is structured as follows:

```
{  
  "headers": [...],  
  "packages": [...],  
  "dependencies": [...],  
  "license_detections": [...],  
  "tallies": {...},  
  "tallies_of_key_files": {  
    "license_expressions": [  
      {  
        "value": null,  
        "count": 1  
      }  
    ],  
    "copyrights": [  
      {  
        "value": null,  
        "count": 1  
      }  
    ],  
    "holders": [  
      {  
        "value": null,  
        "count": 1  
      }  
    ],  
    "authors": [  
      {  
        "value": null,  
        "count": 1  
      }  
    ],  
    "programming_language": [  
      {  
        "value": null,  
        "count": 1  
      }  
    ]  
  }  
}
```

(continues on next page)

(continued from previous page)

```

    }
  ]
},
"files": [...]
}

```

These following flags for each file/directory is also present (generated by `--classify`)

- “is_legal”
- “is_readme”
- “is_key_file”
- “is_manifest”
- “is_top_level”

A key-file is a top-level file, that is either a legal (LICENSE/COPYING etc), manifest or a readme file.

--tallies-with-details Option

The `--tallies` plugin summarizes license, copyright and other scan information at the codebase level. Now running the scan with the `--tallies-with-details` plugin instead creates summaries at individual file/directories with the same license, copyright and other scan information, but at a file/directory level (in addition to the the codebase level summary).

An example Scan:

```
scancode -clipecu --json-pp strapi.json strapi-main/ --tallies-with-details
```

Note: The option `--tallies-with-details` is not a dependency of `--tallies` and can be used individually. `--tallies` is redundant in a scan when `--tallies-with-details` is already selected, because both of them add codebase-level tallies.

A sample scan result is structured as follows:

```

{
  "headers": [...],
  "packages": [...],
  "dependencies": [...],
  "license_detections": [...],
  "tallies": {...},
  "files": [
    {
      "path": "strapi-main",
      "type": "directory",
      "name": "strapi-main",
      "base_name": "strapi-main",
      "extension": "",
      "size": 0,
      "date": null,
      "sha1": null,
      "md5": null,
      "sha256": null,

```

(continues on next page)

(continued from previous page)

```

"mime_type": null,
"file_type": null,
"programming_language": null,
"is_binary": false,
"is_text": false,
"is_archive": false,
"is_media": false,
"is_source": false,
"is_script": false,
"package_data": [],
"for_packages": [],
"detected_license_expression": null,
"detected_license_expression_spdx": null,
"license_detections": [],
"license_clues": [],
"percentage_of_license_text": 0,
"copyrights": [],
"holders": [],
"authors": [],
"emails": [],
"urls": [],
"facets": [],
"is_legal": false,
"is_manifest": false,
"is_readme": false,
"is_top_level": true,
"is_key_file": false,
"tallies": {
  "detected_license_expression": [
    {
      "value": "commercial-license AND other-permissive AND mit",
      "count": 65
    },
    {
      "value": "mit",
      "count": 7
    },
    {
      "value": null,
      "count": 1
    },
    {
      "value": "apache-2.0",
      "count": 1
    },
    {
      "value": "generic-cla",
      "count": 1
    }
  ],
  "copyrights": [
    {

```

(continues on next page)

(continued from previous page)

```

        "value": null,
        "count": 3572
    },
    {
        "value": "Copyright (c) Strapi Solutions SAS",
        "count": 31
    },
    {
        "value": "Copyright (c) Jon Schlinkert",
        "count": 2
    }
],
"holders": [
    {
        "value": null,
        "count": 3572
    },
    {
        "value": "Strapi Solutions SAS",
        "count": 31
    },
    {
        "value": "Jon Schlinkert",
        "count": 2
    }
],
"authors": [
    {
        "value": null,
        "count": 3567
    },
    {
        "value": "name' Strapi Solutions",
        "count": 30
    },
    {
        "value": "the community",
        "count": 4
    },
    {
        "value": "name' A Strapi developer",
        "count": 3
    },
    {
        "value": "name A Strapi",
        "count": 1
    },
    {
        "value": "name' Yurii Tykhomyrov",
        "count": 1
    }
],

```

(continues on next page)

(continued from previous page)

```
"programming_language": [  
  {  
    "value": "JavaScript",  
    "count": 2854  
  },  
  {  
    "value": "TypeScript",  
    "count": 91  
  },  
  {  
    "value": "GAS",  
    "count": 28  
  },  
  {  
    "value": "HTML",  
    "count": 6  
  },  
  {  
    "value": "Bash",  
    "count": 5  
  },  
  {  
    "value": "verilog",  
    "count": 1  
  }  
],  
"files_count": 3604,  
"dirs_count": 1603,  
"size_count": 15175739,  
"scan_errors": []  
},  
{...}  
]  
}
```

TUTORIALS

Tutorial documents provide specific instructions to help you get started.

4.1 Basic Tutorials

4.1.1 How to Run a Scan

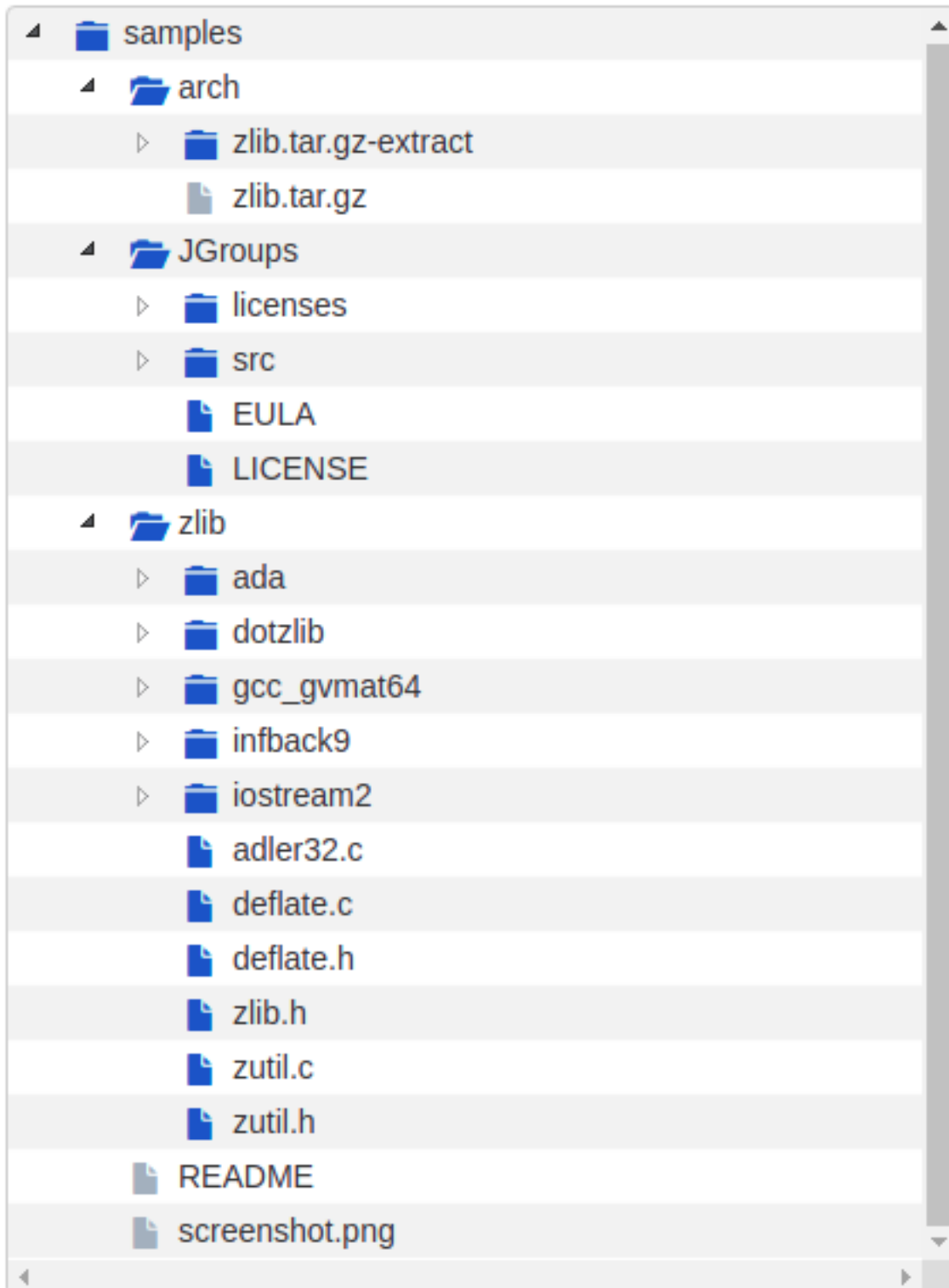
In this simple tutorial example, we perform a basic scan on the `samples` directory distributed by default with Scancode.

Prerequisites

Refer to the *Comprehensive Installation* installation guide.

Looking into Files

As mentioned previously, we are going to perform the scan on the `samples` directory distributed by default with Scancode Toolkit. Here's the directory structure and respective files:



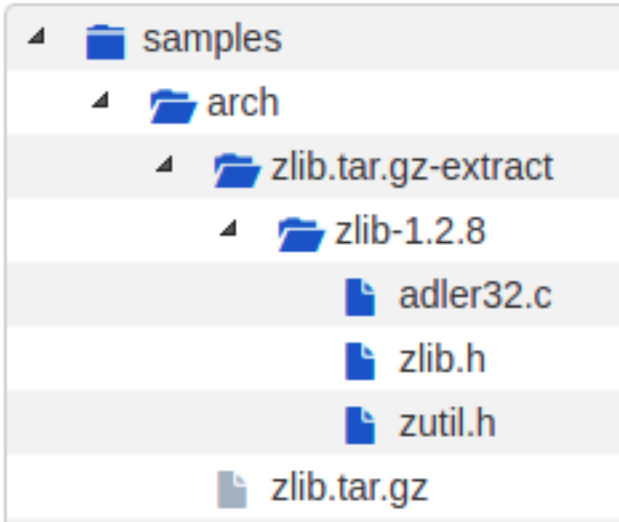
We notice here that the sample files contain a package `zlib.tar.gz`. So we have to extract the archive before running the scan, to also scan the files inside this package.

Performing Extraction

To extract the packages inside `samples` directory:

```
extractcode samples
```

This extracts the `zlib.tar.gz` package:



Note: Use the `--shallow` option to prevent recursive extraction of nested archives.

Deciding Scan Options

These are some common scan options you should consider using before you start the actual scan, according to your requirements.

1. The basic scan options, i.e. `-c` or `--copyright`, `-l` or `--license`, `-p` or `--package`, `-e` or `--email`, `-u` or `--url`, and `-i` or `--info` can be selected according to your requirements. If you do not need one specific type of information (say, licenses), consider removing it because the more options you scan for, the longer it will take for the scan to complete.
2. `--license-score` `INTEGER` is to be set if license matching accuracy is desired (Default is 0, and increasing this means a more accurate match). Also, using `--license-text` includes the matched text to the result.
3. `-n` `INTEGER` option can be used to speed up the scan using multiple parallel processes.
4. `--timeout` `FLOAT` option can be used to skip files taking a long time to scan.
5. `--ignore` `<pattern>` can be used to skip certain group of files.
6. `<OUTPUT FORMAT OPTION(s)>` is also a very important decision when you want to use the output for specific tasks/have requirements. Here we are using `json` as ScanCode Workbench imports `json` files only.

For the complete list of options, refer *All Available Options*.

Running The Scan

Now, run the scan with the options decided:

```
scancode -clpeui -n 2 --ignore "*.java" --json-pp sample.json samples
```

A Progress report is shown:

```
Setup plugins...
Collect file inventory...
Scan files for: info, licenses, copyrights, packages, emails, urls with 2 process(es)...
[#####] 29
Scanning done.
Summary:      info, licenses, copyrights, packages, emails, urls with 2 process(es)
Errors count: 0
Scan Speed:   1.09 files/sec. 40.67 KB/sec.
Initial counts: 49 resource(s): 36 file(s) and 13 directorie(s)
Final counts:  42 resource(s): 29 file(s) and 13 directorie(s) for 1.06 MB
Timings:
  scan_start: 2019-09-24T203514.573671
  scan_end:   2019-09-24T203545.649805
  setup_scan:licenses: 4.30s
  setup: 4.30s
  scan: 26.62s
  total: 31.14s
Removing temporary files...done.
```

Other Important Documentation

1. *Type of Options*
2. *How to Run a Scan*
3. *Basic Tutorials*
4. *How-To Guides*
5. *Reference Docs*
6. *Contributing to Code Development*
7. *Contributing to the Documentation*
8. *Plugin Architecture*
9. *FAQ*
10. *Support*

4.1.2 How to Visualize Scan results



To help visualize the scans, we have a dedicated tool [Scancode workbench](#) which is a desktop application that allows you to visualize and explore the results of one or more scans. It is a cross-platform application that runs on Windows, Mac OS X and Linux. It is built using the Electron framework and is built using Electron, Typescript & React

Detailed Installation and Usage guide can be found here - [Getting Started](#)

Warning: This tutorial uses the 32.x version of Scancode Toolkit, and Scancode Workbench 4.0.x (This version of ScanCode Workbench is compatible with scans from any ScanCode Toolkit develop version/branch at or after v32.x). If you are using an older version of Scancode Toolkit, check respective versions of this documentation. Also refer the Scancode Workbench [release highlights](#).

4.1.3 How To Extract Archives

ScanCode Toolkit provides archive extraction. This command can be used before running a scan over a codebase in order to ensure all archives are extracted. Archives found inside an extracted archive are extracted recursively. Extraction is done in-place in a directory and named after the archive with '-extract' appended.

Name	Date Modified	Size	Kind
 zlib.tar.gz	Feb 8, 2016, 4:04 PM	28 KB	gzip compressed archive
 zlib.tar.gz-extract	Today, 4:18 PM	--	Folder

Usage:

```
extractcode [OPTIONS] <input>
```

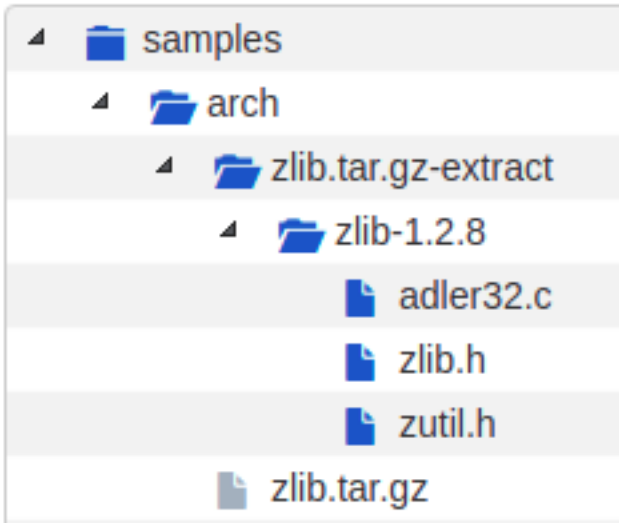
All Extractcode Options

This is intended to be used as an input preparation step, before running the scan. Archives found in an extracted archive are extracted **recursively** by default. Extraction is done in-place in a directory named '-extract' side-by-side with an archive.

To extract the packages in the `samples` directory

```
extractcode samples
```

This extracts the `zlib.tar.gz` package:



--shallow	Do not extract recursively nested archives (e.g. Not archives in archives).
--verbose	Print verbose file-by-file progress messages.
--quiet	Do not print any summary or progress message.
-h, --help	Show the extractcode help message and exit.
--about	Show information about ScanCode and licensing and exit.
--version	Show the version and exit.

4.1.4 How to specify Scancode Output Format

A basic overview of formatting Scancode Output is presented here.

More information on *Scancode Output Formats*.

JSON

If you want JSON output of ScanCode results, you can pass the `--json` argument to ScanCode. The following commands will output scan results in a formatted json file:

- `scancode --json /path/to/output.json /path/to/target/dir`
- `scancode --json-pp /path/to/output.json /path/to/target/dir`
- `scancode --json-lines /path/to/output.json /path/to/target/dir`

To compare the JSON output in different formats refer *Comparing Different json Output Formats*.

Print to stdout (Terminal)

If you want to format the output in JSON and print it at stdout, you can replace the JSON filename with a “-”, like `--json-pp -` instead of `--json-pp output.json`.

The following command will output the scan results in JSON format to stdout (In the Terminal):

```
./scancode -clpieu --json-pp - samples/
```

HTML

If you want HTML output of ScanCode results, you can pass the `--html` argument to ScanCode. The following commands will output scan results in a formatted HTML page or simple web application:

- `scancode --html /path/to/output.html /path/to/target/dir`
- `scancode --html-app /path/to/output.html /path/to/target/dir`

For more details on the HTML output format refer [--html FILE](#).

Warning: The `--html-app` option has been deprecated, use ScanCode Workbench instead.

Custom Output Format

While the three built-in output formats are convenient for a verity of use-cases, one may wish to create their own output template, using the following arguments:

```
`--custom-output FILE --custom-template TEMP_FILE`
```

ScanCode makes this very easy, as it uses the popular Jinja2 template engine. Simply pass the path to the custom template to the `--custom-template` argument, or drop it in a folder to `src/scancode/templates` directory.

For example, if I wanted a simple CLI output I would create a `template2.html` with the particular data I wish to see. In this case, I am only interested in the license and copyright data for this particular scan.

```
## template.txt:
[
    {% if files.license_copyright %}
        {% for location, data in files.license_copyright.items() %}
            {% for row in data %}
location:"{{ location }}",
{% if row.what == 'copyright' %}copyright:"{{ row.value|escape }}"},{% endif %}
            {% endfor %}
        {% endfor %}
    {% endif %}
]

.. note::

    File name and extension does not matter for the template file.
```

Now I can run ScanCode using my newly created template:

```
$ scancode -clpeui --custom-output output.txt --custom-template template.txt samples
Scanning files...
[#####] 46
Scanning done.
```

Now the results are saved in `output.txt` and we can easily view them with `head output.txt`:

```
[
  location:"samples/JGroups/LICENSE",
  copyright:"Copyright (c) 1991, 1999 Free Software Foundation, Inc.",

  location:"samples/JGroups/LICENSE",
  copyright:"copyrighted by the Free Software Foundation",
]
```

For a more elaborate template, refer this [default template](#) given with ScanCode, to generate HTML output with the `--html` output format option.

Documentation on [Jinja templates](#).

4.1.5 How to set what will be detected in Scan

ScanCode allows you to scan a codebase for license, copyright and other interesting information that can be discovered in files. The following options are available for detection when using ScanCode Toolkit:

All “Basic” Scan Options

Option lists are two-column lists of command-line options and descriptions, documenting a program’s options. For example:

-c, --copyright	Scan <input> for copyrights. Sub-Options: <ul style="list-style-type: none"> • <code>--consolidate</code>
-l, --license	Scan <input> for licenses. Sub-Options: <ul style="list-style-type: none"> • <code>--license-references</code> • <code>--license-text</code> • <code>--license-text-diagnostics</code> • <code>--license-diagnostics</code> • <code>--license-url-template TEXT</code> • <code>--license-score INT</code> • <code>--license-clarity-score</code> • <code>--consolidate</code> • <code>--unknown-licenses</code>
-p, --package	Scan <input> for packages. Sub-Options:

	<ul style="list-style-type: none"> • <code>--consolidate</code>
<code>--system-package</code>	Scan <code><input></code> for installed system package databases.
<code>--package-only</code>	Scan <code><input></code> for system and application only for package metadata, without license/ copyright detection and package assembly.
<code>-e, --email</code>	Scan <code><input></code> for emails. Sub-Options: <ul style="list-style-type: none"> • <code>--max-email INT</code>
<code>-u, --url</code>	Scan <code><input></code> for urls. Sub-Options: <ul style="list-style-type: none"> • <code>--max-url INT</code>
<code>-i, --info</code>	Scan for and include information such as: <ul style="list-style-type: none"> • Size, • Type, • Date, • Programming language, • sha1 and md5 hashes, • binary/text/archive/media/source/script flags • Additional options through more CLI options Sub-Options: <ul style="list-style-type: none"> • <code>--mark-source</code>

Note: Unlike previous 2.x versions, `-c`, `-l`, and `-p` are not default. If any combination of these options are used, ScanCode performs only that specific task, and not the others. `scancode -l` scans only for licenses, and doesn't scan for copyright/packages/general information/emails/urls. The only notable exception: a `--package` scan also has license information for package manifests and top-level packages, which are derived regardless of `--license` option being used.

Note: These options, i.e. `-c`, `-l`, `-p`, `-e`, `-u`, and `-i` can be used together. As in, instead of `scancode -c -i -p`, you can write `scancode -cip` and it will be the same.

<code>--generated</code>	Classify automatically generated code files with a flag.
<code>--max-email INT</code>	Report only up to INT emails found in a file. Use 0 for no limit. [Default: 50] Sub-Option of: <code>--email</code>
<code>--max-url INT</code>	Report only up to INT urls found in a file. Use 0 for no limit. [Default: 50] Sub-Option of: <code>--url</code>
<code>--license-score INTEGER</code>	Do not return license matches with scores lower than this score. A number between 0 and 100. [Default: 0] Here, a bigger number means a better match, i.e. Setting a higher license score translates to a higher threshold (with equal or smaller number of matches).

- Sub-Option of: `--license`
- license-text** Include the matched text for the detected licenses in the output report.
- Sub-Option of: `--license`
- Sub-Options:
- `--license-text-diagnostics`
- license-url-template TEXT** Set the template URL used for the license reference URLs.
- In a template URL, curly braces ({}) are replaced by the license key. [Default: default: <https://scancode-licensedb.aboutcode.org/{}>]
- Sub-Option of: `--license`
- license-text-diagnostics** In the matched license text, include diagnostic highlights surrounding with square brackets [] words that are not matched.
- Sub-Option of: `--license` and `--license-text`
- license-diagnostics** In license detections, include diagnostic details to figure out the license detection post processing steps applied.
- Sub-Option of: `--license`
- unknown-licenses** [EXPERIMENTAL] Detect unknown licenses.
- Sub-Option of: `--license`

Different Scans

The following examples will use the `samples` directory that is provided with the [ScanCode Toolkit code](#). All examples will be saved in the JSON format, which can be loaded into Scancode Workbench for visualization. See [How to Visualize Scan results](#) for more information. Another output format option is a static html file. See [Scancode Output Formats](#) for more information.

Scan for all clues:

To scan for licenses, copyrights, urls, emails, package information, and file information

```
scancode -clipecu --json output.json samples
```

Scan for license and copyright clues:

```
scancode -cl --json-pp output.json samples
```

Scan for emails and URLs:

```
scancode -eu --json-pp output.json samples
```

Scan for package information:

```
scancode -p --json-pp output.json samples
```

Scan for file information:

```
scancode -i --json-pp output.json samples
```

To see more example scans:

```
scancode --examples
```

For more information, refer [All Available Options](#).

4.1.6 Add A Post-Scan Plugin

Scan plugins in scancode-toolkit

A lot of scancode features are built-in plugins which are present with scancode-toolkit source code and are usually enabled via the different scancode-toolkit CLI options and are grouped by the types of plugins.

Here are the major types of plugins:

1. Pre-scan plugins (*scancode_pre_scan* in entry points)

These plugins are run before the main scanning steps and are usually filtering of input files, or file classification steps, on whose results the main scan plugins depend on. The base plugin class to be extended is `PreScanPlugin` at `/src/plugincode/pre_scan.py`.

2. Scan plugins (*scancode_scan* in entry points)

These are the scancode plugins which do the file scanning for useful information like license, copyrights, packages and others. These are run on multiprocessing for speed as they are done on a per-file basis, but there can also be post-processing steps on these which are run afterwards and have access to all the per-file scan results. The base plugin class to be extended is `ScanPlugin` at `/src/plugincode/scan.py`.

3. Post-scan plugins (*scancode_post_scan* in entry points)

These are mainly data processing, summarizing and reporting plugins which depend on all the results for the scan plugins. These add new codebase level or file-level attributes, and even removes/modifies data as required for consolidation or summarization. The base plugin class to be extended is `PostScanPlugin` at `/src/plugincode/post_scan.py`.

4. Output plugins (*scancode_output* in entry points)

Supported output options in scancode-toolkit are all plugins and these can also be multiple output options selected. These convert, process and write the data in the specific file format as the output of the scanning procedures. The base plugin class to be extended is `OutputPlugin` at `/src/plugincode/output.py`.

5. Output Filter Plugins (*scancode_output_filter* in entry points)

There are also output filter plugins which apply filters to the outputs and is modified. These filters can be based on whether resources had any detections, ignorables present in licenses and others. The base plugin class to be extended is `OutputFilterPlugin` at [/src/plugincode/output_filter.py](#).

6. Location Provider Plugins

These plugins provide pre-built binary libraries and utilities and their locations which are packaged to be used in scancode-toolkit. The base plugin class to be extended is `LocationProviderPlugin` at [/src/plugincode/location_provider.py](#).

Built-In vs. Optional Installation

Built-In

Some post-scan plugins are installed when ScanCode itself is installed, and they are specified at `[options.entry_points]` in the `setup.cfg` file. For example, the *License Policy Plugin* is a built-in plugin, whose code is located here:

```
https://github.com/nexB/scancode-toolkit/blob/develop/src/licensedcode/plugin_license_
↪policy.py
```

These plugins do not require any additional installation steps and can be used as soon as ScanCode is up and running.

Optional

ScanCode is also designed to use post-scan plugins that must be installed separately from the installation of ScanCode. The code for this sort of plugin is located here:

```
https://github.com/nexB/scancode-plugins
```

This wiki page will focus on optional post-scan plugins.

Example Post-Scan Plugin: Hello ScanCode

To illustrate the creation of a simple post-scan plugin, we'll create a hypothetical plugin named `Hello ScanCode`, which will print `Hello ScanCode!` in your terminal after you've run a scan. Your command will look like something like this:

```
scancode -i -n 2 <path to target codebase> --hello --json <path to JSON output file>
```

We'll start by creating three folders:

1. Top-level folder – `/scancode-hello/`
2. 2nd-level folder – `/src/`
3. 3rd-level folder – `/hello_scancode/`

1. Top-level folder – /scancode-hello/

- In the scancode-plugins repository, in the misc directory, add a folder with a relevant name, e.g., scancode-hello. This folder will hold all of your plugin code.
- Inside the /scancode-hello/ folder you'll need to add a folder named `src` and 7 files. `/src/` – This folder will contain your primary Python code and is discussed in more detail in the following section.

The 7 Files are:

1. `.gitignore` – See, e.g., [/scancode-ignore-binaries/.gitignore](#)

```
/build/
/dist/
```

2. `apache-2.0.LICENSE` – See, e.g., [/scancode-ignore-binaries/apache-2.0.LICENSE](#)

3. `MANIFEST.in`

```
graft src

include setup.py
include setup.cfg
include .gitignore
include README.md
include MANIFEST.in
include NOTICE
include apache-2.0.LICENSE

global-exclude *.py[co] __pycache__ *.~
```

4. `NOTICE` – See, e.g., [/scancode-ignore-binaries/NOTICE](#)

5. `README.md`

6. `setup.cfg`

```
[metadata]
license_file = NOTICE

[bdist_wheel]
universal = 1

[aliases]
release = clean --all bdist_wheel
```

7. `setup.py` – This is an example of what our `setup.py` file would look like:

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-

from __future__ import absolute_import
from __future__ import print_function

from glob import glob
from os.path import basename
from os.path import join
```

(continues on next page)

(continued from previous page)

```

from os.path import splitext

from setuptools import find_packages
from setuptools import setup

desc = '''A ScanCode post-scan plugin to to illustrate the creation of a simple post-
↳scan plugin.'''

setup(
    name='scancode-hello',
    version='1.0.0',
    license='Apache-2.0 with ScanCode acknowledgment',
    description=desc,
    long_description=desc,
    author='nexB',
    author_email='info@aboutcode.org',
    url='https://github.com/nexB/scancode-plugins/blob/main/misc/scancode-hello/',
    packages=find_packages('src'),
    package_dir={'': 'src'},
    py_modules=[splitext(basename(path))[0] for path in glob('src/*.py')],
    include_package_data=True,
    zip_safe=False,
    classifiers=[
        # complete classifier list: http://pypi.python.org/pypi?%3Aaction=list_
        ↳classifiers
        'Development Status :: 4 - Beta',
        'Intended Audience :: Developers',
        'License :: OSI Approved :: Apache Software License',
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
        'Topic :: Utilities',
    ],
    keywords=[
        'scancode', 'plugin', 'post-scan'
    ],
    install_requires=[
        'scancode-toolkit',
    ],
    entry_points={
        'scancode_post_scan': [
            'hello = hello_scancode.hello_scancode:SayHello',
        ],
    }
)

```

2. 2nd-level folder – /src/

1. Add an `__init__.py` file inside the `src` folder. This file can be empty, and is used to indicate that the folder should be treated as a Python package directory.
2. Add a folder that will contain our primary code – we'll name the folder `hello_scancode`. If you look at the example of the `setup.py` file above, you'll see this line in the `entry_points` section:

```
'hello = hello_scancode.hello_scancode:SayHello',
```

- `hello` refers to the name of the command flag.
- The first `hello_scancode` is the name of the folder we just created.
- The second `hello_scancode` is the name of the `.py` file containing our code (discussed in the next section).
- `SayHello` is the name of the `PostScanPlugin` class we create in that file (see sample code below).

3. 3rd-level folder – /hello_scancode/

1. Add an `__init__.py` file inside the `hello_scancode` folder. As noted above, this file can be empty.
2. Add a `hello_scancode.py` file.

Imports

```
from plugincode.post_scan import PostScanPlugin
from plugincode.post_scan import post_scan_impl
from scancode import CommandLineOption
from scancode import POST_SCAN_GROUP
```

Create a PostScanPlugin class

The `PostScanPlugin` class ([PostScanPlugin code](#)) inherits from the `CodebasePlugin` class (see [CodebasePlugin code](#)), which inherits from the `BasePlugin` class (see [BasePlugin code](#)).

```
@post_scan_impl
class SayHello(PostScanPlugin):
    """
    Illustrate a simple "Hello World" post-scan plugin.
    """

    options = [
        CommandLineOption('--hello',),
        is_flag=True, default=False,
        help='Generate a simple "Hello ScanCode" greeting in the terminal.',
        help_group=POST_SCAN_GROUP)
    ]

    def is_enabled(self, hello, **kwargs):
        return hello
```

(continues on next page)

(continued from previous page)

```
def process_codebase(self, codebase, hello, **kwargs):
    """
    Say hello.
    """
    if not self.is_enabled(hello):
        return

    print('Hello ScanCode!!')
```

Load the plugin

- To load and use the plugin in the normal course, navigate to the plugin's root folder (in this example: `/plugins/scancode-hello/`) and run `pip install .` (don't forget the final `.`).
- If you're developing and want to test your work, save your edits and run `pip install -e .` from the same folder.

More-complex examples

This Hello ScanCode example is quite simple. For examples of more-complex structures and functionalities you can take a look at the other post-scan plugins for guidance and ideas.

One good example is the License Policy post-scan plugin. This plugin is installed when ScanCode is installed and consequently is not located in the `/plugins/` directory used for manually-installed post-scan plugins. The code for the License Policy plugin can be found at `/scancode-toolkit/src/licensedcode/plugin_license_policy.py` and illustrates how a plugin can be used to analyze the results of a ScanCode scan using external data files and add the results of that analysis as a new field in the ScanCode JSON output file.

HOW-TO DOCUMENTS

How-To documents explain how to accomplish specific tasks.

5.1 How-To Guides

5.1.1 How To Add a New License for Detection

How to add a new license for detection?

To add a new license, you first need to select a new and unique license *key* (`mit` and `gpl-2.0` are some of the existing license keys).

The key name can contain only these symbols:

- lowercase letters from a to z,
- numbers from 0 to 9,
- dash - and . period signs. No spaces or underscore.

The license key also has to be fewer than 50 characters (same for *short_name*).

We also have to add a *spdx_license_key* which is either a valid SPDX license key at `The SPDX license list <<https://spdx.org/licenses/>>`_, or a *Licenseref-scancode*-<key>.

All licenses are stored as a plain text file in the *src/licensedcode/data/licenses* directory using their key as base for the file name. For example the filename for a license with *key*: *mit* would be *mit.LICENSE*.

You need to create a file with:

- the text of the license saved in plain text. We usually get rid of HTML tags or other special characters. We also remove copyrights and only keep the original text as is, with the original formatting intact.
- the data attributes for the license in YAML format as [YAML frontmatter](#).

See an example license: [apache-2.0.LICENSE](#)

There are a couple of mandatory attributes:

- *key*
- *spdx_license_key*
- *short_name*
- *name*
- *category* (Use “Unstated License” if not known)

- *owner* (Use “Unspecified” if not known)

And more attributes which are not mandatory but always nice to have (if applicable):

- *other_spdx_license_keys*
- *osi_license_key*
- *minimum_coverage*
- *standard_notice*
- *notes*

We want to use *minimum_coverage* when there are other licenses that are very similar and we want to make sure we match these licenses correctly, and *notes* for interesting cases of licenses with descriptions to help identify origin, similarities to other licenses, notes about the SPDX keys and others.

Some URLs:

- *homepage_url*
- *text_urls*
- *osi_url*
- *faq_url*
- *other_urls*

Also attributes having ignorables in the license text:

- *ignorable_urls*
- *ignorable_copyrights*
- *ignorable_authors*
- *ignorable_holders*
- *ignorable_emails*

See the `src/licensedcode/data/licenses/` directory for many more examples.

Note: Add licenses in a local development installation and run *scancode-reindex-licenses* to make sure we reindex the licenses and this validates the new licenses.

5.1.2 How to Add New License Rules for Enhanced Detection

ScanCode relies on license rules to detect licenses. A rule is a simple text file containing a license text or notice or mention with YAML frontmatter with data attributes that tells ScanCode which license expression to report when the text is detected, and other properties.

See the [FAQ](#) for a high level description of adding license detection rules.

How to add a new license detection rule?

A license detection rule is a file with:

- a plain text that is typically a variant of a license text, notice or license mention.
- data as YAML frontmatter documenting license expression and other rule attributes.

To add a new rule, you need to pick a unique base file name. As a convention, we like to include the license expression that should be detected in that name to make it more descriptive. For example: `mit_and_gpl-2.0` is a good base name for a rule that would detect an MIT and GPL-2.0 license combination at once. Add a suffix (usually numeric) to make it unique if there is already a rule with this base name. Do not use spaces or special characters in that name.

Then create the rule file in the `src/licensedcode/data/rules/` directory using this name; for example a rule with *license_expression* as `mit AND apache-2.0` might have a filename: `mit_and_apache-2.0_10.RULE`.

Save your rule text in this file; if there are specific words like company names, projects or other, it is better to have rules with and without these so we have better detection.

For a simple *mit AND apache-2.0* license expression detection, here is an example rule file:

```
---
license_expression: mit AND apache-2.0
is_license_notice: yes
relevance: 100
referenced_filenames:
  - LICENSE
---

## License
The MIT License (MIT) + Apache 2.0. Read [LICENSE](LICENSE).
```

See the `src/licensedcode/data/rules/` directory for many examples.

More (advanced) rules options:

- you can use a *notes* text field to document this rule and explain where you found it first.
- if no license should be detected for your `.RULE` text, do not add a license expression, just add a *notes* field.
- Each rule needs to have one flag to describe the type of license rule. The options are:
 - *is_license_notice*
 - *is_license_text*
 - *is_license_tag*
 - *is_license_reference*
 - *is_license_intro*
- There can also be false positive rules, which if detected in the file scanned, will not be present in the result license detections. These just have the license text and a *is_false_positive* flag set to True.
- you can specify key phrases by surrounding one or more words between the `{{` and `}}` tags. Key phrases are words that **must** be matched/present in order for a RULE to be considered a match.

See the `src/licensedcode/models.py` directory for a list of all possible values and other options.

Note: Add rules in a local development installation and run `scancode-reindex-licenses` to make sure we reindex the rules and this validates the new licenses.

5.1.3 How to Install External Licenses to Use in License Detection

Users can install external licenses and rules in the form of:

1. reusable plugins
2. license directories

These licenses and rules are then used in license detection.

How to install a plugin containing external licenses and/or rules

To create a plugin with external licenses or rules, we must create a Python package containing the license and/or rule files. Python packages can have many different file structures. You can find an example package in:

tests/licensedcode/data/additional_licenses/additional_plugin_1.

This is the basic structure of the example plugin:

```
licenses_to_install1/
├── src/
│   └── licenses_to_install1/
│       ├── licenses/
│       │   └── example-installed-1.LICENSE
│       ├── rules/
│       │   └── example-installed-1.RULE
│       └── __init__.py
├── apache-2.0.LICENSE
├── MANIFEST.in
├── setup.cfg
└── setup.py
```

Entry points definition in setup.py

First, in `setup.py`, you must provide an entry point called `scancode_location_provider`. This allows ScanCode Toolkit to discover the plugin and use it in license detection. Here is the definition of `entry_points` in `setup.py`:

```
entry_points={
    'scancode_location_provider': [
        'licenses_to_install1 = licenses_to_install1:LicensesToInstall1Paths',
    ],
}
```

The `scancode_location_provider` entry point maps to a list with information about the plugin. The variable `licenses_to_install1` is the name of the entry point. All entry point names **must** start with the prefix `licenses`, or else ScanCode Toolkit will not use them in license detection.

Directory structure

`licenses_to_install1` is set to `licenses_to_install1:LicensesToInstall1Paths`. Note that in `src`, we have another directory called `licenses_to_install1` and in `licenses_to_install1/__init__.py`, we define the class `LicensesToInstall1Paths`. These two values make up the entry point definition.

`LicensesToInstall1Paths` is a subclass of `LocationProviderPlugin` and implements the method `get_locations()`. The class you define in `__init__.py` must also subclass `LocationProviderPlugin` and implement this method.

Finally, the same directory containing the class definition must also contain the licenses and/or rules. Licenses must be contained in a directory called `licenses` and rules must be contained in a directory called `rules`.

See [How To Add a New License for Detection](#) and [How to Add New License Rules for Enhanced Detection](#) to understand the structure of license and rule files, respectively.

After creating this plugin, you can upload it to PyPI so that others can use it, or you can leave it as a local directory.

Installing and using the plugin

To use the plugin in license detection, all you need to do is:

1. Configure the `scancode-toolkit` virtualenv and activate.
2. Install the package with `pip` like the following: `pip install tests/licensedcode/data/additional_licenses/additional_plugin_2/`
3. Reindex licenses using `scancode-reindex-licenses`.

Note: Installing the plugin will not add the licenses/rules to the index automatically, they will be indexed only after running `scancode-reindex-licenses`.

Once it is installed, the contained licenses and rules will automatically be used in license detection assuming the plugin follows the correct directory structure conventions.

Writing tests for new installed licenses

Look at `tests/licensedcode/data/example_external_licenses/licenses_to_install1` to see an example of a plugin with tests. The tests are contained in the `tests` directory:

```
licenses_to_install1/
├── src/
│   └── licenses_to_install1/
│       ├── licenses/
│       │   └── example-installed-1.LICENSE
│       ├── rules/
│       │   └── example-installed-1.RULE
│       └── __init__.py/
└── tests/
    ├── data/
    │   ├── example-installed-1.txt
    │   └── example-installed-1.txt.yml
    └── test_detection_datadriven.py
```

(continues on next page)

(continued from previous page)

```

├─ apache-2.0.LICENSE
├─ MANIFEST.in
├─ setup.cfg
└─ setup.py

```

To write your own tests, first make sure `setup.py` includes `scancode-toolkit` as a dependency:

```

...
install_requires=[
    'scancode-toolkit',
],
...

```

Then you can define a test class and call the `build_tests` method defined in `licensedcode_test_utils`, passing in the test directory and the test class as parameters:

```

TEST_DIR = abspath(join(dirname(__file__), 'data'))

class TestLicenseDataDriven1(unittest.TestCase):
    pass

licensedcode_test_utils.build_tests(
    TEST_DIR,
    clazz=TestLicenseDataDriven1, regen=scancode_config.REGEN_TEST_FIXTURES)

```

The `tests/data` directory contains one file for each license: a license text file with a YAML frontmatter specifying the expected license expression from the test.

Finally, install the plugin and run the test:

```
pytest -vvs tests/test_detection_datadriven.py.
```

Note: Once you install an external license plugin, you have to reconfigure `scancode-toolkit` (or use `pip uninstall`) to uninstall the plugin to completely remove it. Otherwise using the `—only-builtin` option only regenerates the index without the installed plugins, but another Reindex would have the licenses/rules from the installed plugins.

How to add external licenses and/or rules from a directory

This is the basic structure of the example license directory:

```

additional_license_directory/
├─ licenses/
│   └─ example-installed-1.LICENSE
├─ rules/
│   └─ example-installed-1.RULE

```

Adding the licenses to the index

To add the licenses in the directory to the index, all you need to do is:

1. Configure the scancode-toolkit virtualenv and activate.
2. Run `scancode-reindex-licenses` with:


```
--additional-directory tests/licensedcode/data/additional_licenses/additional_dir/
```

Note: Adding licenses/rules from an additional directory is not permanent. Another reindexing without the additional directory option would just use the builtin scancode licenses and rules, and will not have these additional licenses/rules anymore.

Once the licenses/rules are in the index, they will automatically be used in license detection.

scancode-reindex-licenses Usage

Usage: `scancode-reindex-licenses` [OPTIONS]

Reindex scancode licenses and exit

Options

--all-languages	[EXPERIMENTAL] Rebuild the license index including texts all languages (and not only English) and exit.
--only-builtin	Rebuild the license index excluding any additional license directory or additional license plugins which were added previously, i.e. with only builtin scancode license and rules.
--additional-directory DIR	Include this directory with additional custom licenses and license rules in the license detection index.
--load-dump	Load all license and rules from their respective files and then dump them back to those same files.
-h, --help	Shows the options and explanations.

5.1.4 How To Generate Attribution from a ScanCode Scan

How To generate attribution from a ScanCode scan?

Users can use an Open Source Project “AboutCode Toolkit” to generate attribution document from a ScanCode scan.

Read more about AboutCode Toolkit here: <https://aboutcode-toolkit.readthedocs.io/>.

Check out the code at <https://github.com/nexB/aboutcode-toolkit>

Command in AboutCode Toolkit to generate attribution: <https://aboutcode-toolkit.readthedocs.io/en/latest/reference.html#attrib>.

Attention: The attribution requires the ScanCode scan to have at least the *-info* and *-license* option flagged

CONTRIBUTE TO SCANCODE

6.1 Contribute

6.1.1 Contributing to Code Development

TL;DR:

- Contributions comes as bugs/questions/issues and as pull requests.
- Source code and runtime data are in the `/src/` directory.
- Test code and test data are in the `/tests/` directory.
- Datasets (including licenses) and test data are in `/data/` sub-directories.
- We use DCO signoff in commit messages, like Linux does.
- Porting ScanCode to other OS (FreeBSD is supported, etc.) is possible. Enter an issue for help.

See [CONTRIBUTING.rst](#) for details.

Code layout and conventions

Source code is in the `src/` directory, tests are in the `tests/` directory. Miscellaneous scripts and configuration files are in the `etc/` directory.

There is one Python package for each major feature under `src/` and a corresponding directory with the same name under `tests` (but this is not a package by design as it would not make sense to have a top level “tests” package which is a name that’s too common).

Each test script is named `test_XXXX`; we prefer organizing tests in subclasses of the standard library `unittest` module. But we also use plain functions that are discovered nicely by `pytest`.

When source or tests need data files, we store these in a `data` subdirectory. This is used extensively in tests and also in source code for the reference license texts and data and license detection rules files.

We use PEP8 conventions with a relaxed line length that can be up to 90’ish characters long when needed to keep the code clear and readable.

We write tests, a lot of tests, thousands of tests. When finding bugs or adding new features, we add tests. See existing test code for examples which form also a good specification for the supported features.

The tests should pass on Linux 64 bits, Windows 64 bits and on macOS 10.14 and up. We maintain multiple CI loops with Azure (all OSes) at https://dev.azure.com/nexB/scancode-toolkit/_build and Appveyor (Windows) at <https://ci.appveyor.com/project/nexB/scancode-toolkit>.

Several tests are data-driven and use data files as test input and sometimes data files as test expectation (in this case using either JSON or YAML files); a large number of copyright, license and package manifest parsing tests are such data-driven tests.

Running tests

ScanCode comes with over 29,000 unit tests to ensure detection accuracy and stability across Linux, Windows and macOS OSes: we kinda love tests, do we?

We use pytest to run the tests: call the `pytest` script to run the whole test suite. This is installed with the `pytest` package which is installed when you run `./configure --dev`.

If you are running from a fresh git clone and you run `./configure` and then `source venv/bin/activate` the `pytest` command will be available in your path.

Alternatively, if you have already configured but are not in an activated “virtualenv” the `pytest` command is available under `<root of your checkout>/venv/bin/pytest`

(Note: paths here are for POSIX, but mostly the same applies to Windows)

If you have a multiprocessor machine you might want to run the tests in parallel (and faster). For instance: `pytest -n4` runs the tests on 4 CPUs. We typically run the tests in verbose mode with `pytest -vvs -n4`.

You can also run a subset of the test suite as shown in the CI configs <https://github.com/nexB/scancode-toolkit/blob/develop/azure-pipelines.yml> e.g. `pytest -n 2 -vvs tests/scancode` runs only the test scripts present in the `tests/scancode` directory. (You can give the path to a specific test script file there too).

See also <https://docs.pytest.org> for details or use the `pytest -h` command to show the many other options available.

One useful option is to run a select subset of the test functions matching a pattern with the `-k` option, for instance: `pytest -vvs -k tcpdump` would only run test functions that contain the string “tcpdump” in their name or their class name or module name.

Another useful option after a test run with some failures is to re-run only the failed tests with the `--lf` option, for instance: `pytest -vvs --lf` would only run only test functions that failed in the previous run.

Because we have a lot of tests (over 29,000), we organized theses in test suites using pytest markers that are defined in the `conf/test.py` pytest plugin. These are enabled by adding a `--test-suite` option to the `pytest` command.

- `--test-suite=standard` is the default and runs a decent but basic test suite
- `--test-suite=all` runs the standard test and adds a comprehensive test suite
- `--test-suite=validate` runs the `standra` and `all` test and adds extensive data-driven and data validations (for package, copyright and license detection)

In some cases we need to regenerate test data when expected behaviour/result data structures change, and we have an environment variable to regenerate test data. `SCANCODE_REGEN_TEST_FIXTURES` is present in `scancode_config` and this can be set to regenerate test data for specific tests like this:

```
SCANCODE_REGEN_TEST_FIXTURES=yes pytest -vvs tests/packagedcode/test_package_models.py
```

This command will only regenerate test data for only the tests in `test_package_models.py`, and we can further specify the tests to regen by using more pytest options like `-lf` and `-k test_instances`.

If test data is regenerated, it is important to review the diff for test files and carefully go through all of it to make sure there are no unintended changes there, and then commit all the regenerated test data.

To help debug in scancode, we use logging. There are different environment variables you need to set to turn on logging. In `packagedcode`:

```
``SCANCODE_DEBUG_PACKAGE=yes pytest -vvs tests/packagedcode/ --lf``
```

Or set the `TRACE` variable to `True`. This enables `logger_debug` functions logging variables and shows code execution paths by logging and printing the logs in the terminal. If debugging full scans run by click, you have to raise exceptions in addition to setting the `TRACE` to enable logging.

Thirdparty libraries and dependencies management

ScanCode uses the `configure` and `configure.bat` scripts to install a `virtualenv`, install required packaged dependencies using `setuptools` and such that ScanCode can be installed in a repeatable and consistent manner on all OSes and Python versions.

For this we maintain a `setup.cfg` with our direct dependencies with loose minimum version constraints; and we keep pinned exact versions of these dependencies in the `requirements.txt` and `requirements-dev.txt` (for testing and development).

Note: we also have a `setup-mini.cfg` used to create a ScanCode PyPI package with minimal dependencies (and limited features). This is mostly duplicated from `setup.cfg`.

And to ensure that we also all use well known version of the core `virtualenv`, `pip`, `setuptools` and `wheel` libraries, we use the `virtualenv.pyz` Python zipp app from <https://github.com/pypa/get-virtualenv/tree/main/public> and store it in the Git repo in the `etc/thirdparty` directory.

We bundle pre-built bundled native binaries as plugins which are installed as wheels. These binaries are organized by OS and architecture; they ensure that ScanCode works out of the box either using a checkout or a download, without needing a compiler and toolchain to be installed.

The corresponding source code and build scripts for all for the pre-built binaries are stored in a separate repository at <https://github.com/nexB/scancode-plugins>

ScanCode app archives should not require network access for installation or configuration of its third-party libraries and dependencies. To enable this, we store bundled thirdparty components and libraries in the `thirdparty` directory of released app archives; this is done at build time. These dependencies are stored as pre-built wheels. These wheels are sometimes built by us when there is no wheel available upstream on PyPI. We store all these prebuilt wheels with corresponding `.ABOUT` and `.LICENSE` files in <https://github.com/nexB/thirdparty-packages/tree/main/pypi> which is published for download at <https://thirdparty.aboutcode.org/pypi/>

Because this is used by the `configure` script, all the thirdparty dependencies used in ScanCode MUST be available there first. Therefore adding a new dependency means requesting a merge/PR in <https://github.com/nexB/thirdparty-packages/> first that contains all the recursive dependencies.

There are utility scripts in `etc/release` that can help with the dependencies management process in particular to build or update wheels with native code for multiple OSes (Linux, macOS and Windows) and multiple Python versions (3.7+), which is not a completely simple operation (and requires eventually 12 wheels and one source distribution to be published as we support 3 OSes and 4 Python versions).

Using ScanCode as a Python library

ScanCode can be used also as a Python library and is available as a Python wheel in PyPI and installed with `pip install scancode-toolkit` or `pip install scancode-toolkit-mini`.

Since we do not pin dependencies to avoid dependency resolution conflicts for downstream users, there are possibilities of issues arising from dependencies silently changing API/functions which scanCode uses.

6.1.2 How to cut a new release

Update version

- Bump version to update major, minor or patch version in `setup.cfg` `setup-mini.cfg` and `src/scancode_config.py`. Note that this is SemVer, though we used CalVer previously, we have switched back to SemVer.
- If scancode output data format is changed, increment manually the major, minor or patch version to bump the version in `src/scancode_config.py`. Note that this is SemVer.

See our [:ref:versioning](#) for more details.

Tag and publish

- Changes for a release should also be pushed to a branch and a Pull Request should be created for it, for review.
- Update the `CHANGELOG.rst` with detailed documentation of updates and API/CLI option changes, or any significant changes.
- Commit these changes and push changes to develop (here we use an example tag `v1.6.1`):
 - `git commit -s`
 - `git push --set-upstream origin release-prep-v1.6.1`
- Merge this `release-prep-v1.6.1` branch in develop after review approval and tag the release:
 - `git tag -a v1.6.1 -m "Release v1.6.1"`
 - `git push --set-upstream origin release-prep-v1.6.1`
 - `git push --set-upstream origin v1.6.1`

Automated Release Process

- We have an [automated release script](#) triggered by a pushed tag, where jobs run to:
 - Build pypi wheels and sdist archives
 - Build app release archives for linux/mac/windows
 - This happens for all supported python versions
 - Test these wheels and app archives in linux/mac/windows for all supported versions of python
 - Create a GitHub release (draft by default) with all wheels, sdists and app archives (for all os/python combinations)
 - Upload sdists and wheels (all python versions) and publish a release (This won't be a stable release for beta/release-candidate tags)
- Populate the draft GitHub release by clicking the **Generate Release Notes** button and this pre-populates the release notes with PRs and contributors.
- Add more details to the release notes talking about the key features and changes in the release.
- Publish the release on GitHub (Note the **Set as a pre-release** vs **Set as the latest release** checkboxes)
- Announce in public channels and chats about the release
- Do test the release archives yourself.

6.1.3 Contributing to the Documentation

Setup Local Build

To get started, create or identify a working directory on your local machine.

Open that directory and execute the following command in a terminal session:

```
git clone https://github.com/nexB/scancode-toolkit.git
```

That will create an `/scancode-toolkit` directory in your working directory. Now you can install the dependencies in a virtualenv:

```
cd scancode-toolkit
./configure --docs
```

Note: In case of windows, run `configure --docs` instead of this.

Now, this will install the following prerequisites:

- Sphinx
- sphinx_rtd_theme (the format theme used by ReadTheDocs)
- docs8 (style linter)

These requirements are already present in `setup.cfg` and `./configure --docs` installs them.

Now you can build the HTML documents locally:

```
source venv/bin/activate
cd docs
make html
```

Assuming that your Sphinx installation was successful, Sphinx should build a local instance of the documentation .html files:

```
open build/html/index.html
```

Note: In case this command did not work, for example on Ubuntu 18.04 you may get a message like “Couldn’t get a file descriptor referring to the console”, try:

```
see build/html/index.html
```

You now have a local build of the AboutCode documents.

Share Document Improvements

Ensure that you have the latest files:

```
git pull
git status
```

Before committing changes run Continuous Integration Scripts locally to run tests. Refer *Continuous Integration* for instructions on the same.

Follow standard git procedures to upload your new and modified files. The following commands are examples:

```
git status
git add source/index.rst
git add source/how-to-scan.rst
git status
git commit -m "New how-to document that explains how to scan"
git status
git push
git status
```

The Scancode-Toolkit webhook with ReadTheDocs should rebuild the documentation after your Pull Request is Merged.

Refer the [Pro Git Book](#) available online for Git tutorials covering more complex topics on Branching, Merging, Rebasing etc.

Continuous Integration

The documentations are checked on every new commit through Travis-CI, so that common errors are avoided and documentation standards are enforced. Travis-CI presently checks for these 3 aspects of the documentation :

1. Successful Builds (By using sphinx-build)
2. No Broken Links (By Using link-check)
3. Linting Errors (By Using Doc8)

So run these scripts at your local system before creating a Pull Request:

```
cd docs
./scripts/sphinx_build_link_check.sh
./scripts/doc8_style_check.sh
```

If you don't have permission to run the scripts, run:

```
chmod u+x ./scripts/doc8_style_check.sh
```

Style Checks Using Doc8

How To Run Style Tests

In the project root, run the following commands:

```
$ cd docs
$ ./scripts/doc8_style_check.sh
```

A sample output is:

```
Scanning...
Validating...
docs/source/misc/licence_policy_plugin.rst:37: D002 Trailing whitespace
docs/source/misc/faq.rst:45: D003 Tabulation used for indentation
docs/source/misc/faq.rst:9: D001 Line too long
docs/source/misc/support.rst:6: D005 No newline at end of file
=====
Total files scanned = 34
Total files ignored = 0
Total accumulated errors = 326
Detailed error counts:
  - CheckCarriageReturn = 0
  - CheckIndentationNoTab = 75
  - CheckMaxLineLength = 190
  - CheckNewlineEndOfFile = 13
  - CheckTrailingWhitespace = 47
  - CheckValidity = 1
```

Now fix the errors and run again till there isn't any style error in the documentation.

What is Checked?

PyCQA is an Organization for code quality tools (and plugins) for the Python programming language. Doc8 is a sub-project of the same Organization. Refer this [README](#) for more details.

What is checked:

- invalid rst format - D000
- lines should not be longer than 100 characters - D001
 - RST exception: line with no whitespace except in the beginning
 - RST exception: lines with http or https URLs
 - RST exception: literal blocks
 - RST exception: rst target directives
- no trailing whitespace - D002
- no tabulation for indentation - D003
- no carriage returns (use UNIX newlines) - D004
- no newline at end of file - D005

Intersphinx

ScanCode toolkit documentation uses [Intersphinx](#) to link to other Sphinx Documentations, to maintain links to other Aboutcode Projects.

To link sections in the same documentation, standart reST labels are used. Refer [Cross-Referencing](#) for more information.

For example:

```
.. _my-reference-label:

Section to cross-reference
-----

This is the text of the section.

It refers to the section itself, see :ref:`my-reference-label`.
```

Now, using Intersphinx, you can create these labels in one Sphinx Documentation and then reference these labels from another Sphinx Documentation, hosted in different locations.

You just have to add the following in the `conf.py` file for your Sphinx Documentation, where you want to add the links:

```
extensions = [
    'sphinx.ext.intersphinx'
]

intersphinx_mapping = {'aboutcode': ('https://aboutcode.readthedocs.io/en/latest/',
    ↪None)}
```

To show all Intersphinx links and their targets of an Intersphinx mapping file, run:

```
python -msphinx.ext.intersphinx https://aboutcode.readthedocs.io/en/latest/objects.inv
```

Warning: `python -msphinx.ext.intersphinx https://aboutcode.readthedocs.io/objects.inv` will give error.

This enables you to create links to the aboutcode Documentation in your own Documentation, where you modified the configuration file. Links can be added like this:

```
For more details refer :ref:`aboutcode:doc_style_guide`.
```

You can also not use the aboutcode label assigned to all links from aboutcode.readthedocs.io, if you don't have a label having the same name in your Sphinx Documentation. Example:

```
For more details refer :ref:`doc_style_guide`.
```

If you have a label in your documentation which is also present in the documentation linked by Intersphinx, and you link to that label, it will create a link to the local label.

For more information, refer this tutorial named [Using Intersphinx](#).

Style Conventions for the Documentaion

1. Headings

(Refer) Normally, there are no heading levels assigned to certain characters as the structure is determined from the succession of headings. However, this convention is used in Python's Style Guide for documenting which you may follow:

with overline, for parts

- with overline, for chapters

=, for sections

-, for subsections

^, for sub-subsections

“, for paragraphs

2. Heading Underlines

Do not use underlines that are longer/shorter than the title headline itself. As in:

Correct :

Extra Style Checks

Incorrect :

Extra Style Checks

Note: Underlines shorter than the Title text generates Errors on sphinx-build.

3. Internal Links

Using `:ref:` is advised over standard reStructuredText links to sections (like ``Section title`_`) because it works across files, when section headings are changed, will raise warnings if incorrect, and works for all builders that support cross-references. However, external links are created by using the standard ``Section title`_` method.

4. Eliminate Redundancy

If a section/file has to be repeated somewhere else, do not write the exact same section/file twice. Use `.. include: ../README.rst` instead. Here, `../` refers to the documentation root, so file location can be used accordingly. This enables us to link documents from other upstream folders.

5. Using `:ref:` only when necessary

Use `:ref:` to create internal links only when needed, i.e. it is referenced somewhere. Do not create references for all the sections and then only reference some of them, because this created unnecessary references. This also generates ERROR in `restructuredtext-lint`.

6. Spelling

You should check for spelling errors before you push changes. [Aspell](#) is a GNU project Command Line tool you can use for this purpose. Download and install Aspell, then execute `aspell check <file-name>` for all the files changed. Be careful about not changing commands or other stuff as

Aspell gives prompts for a lot of them. Also delete the temporary .bak files generated. Refer the [manual](#) for more information on how to use.

7. Notes and Warning Snippets

Every Note and Warning sections are to be kept in `rst_snippets/note_snippets/` and `rst_snippets/warning_snippets/` and then included to eliminate redundancy, as these are frequently used in multiple files.

8. Redirects

If layouts of doc pages are being changed and these could be referenced elsewhere, these should be added in the *redirects* mapping in *conf.py*. For examples on using these see <https://documatt.gitlab.io/sphinx-rerredirects/usage.html>

Converting from Markdown

If you want to convert a .md file to a .rst file, this [tool](#) does it pretty well. You'd still have to clean up and check for errors as this contains a lot of bugs. But this is definitely better than converting everything by yourself.

This will be helpful in converting GitHub wiki's (Markdown Files) to reStructuredtext files for Sphinx/ReadTheDocs hosting.

Automatic Docs Generation

It's possible to generate docs automatically from data by using a combination of:

- [shell scripts: example](#)
- [python scripts: example](#)
- [jinja templates: example](#)

And we do this currently to keep a documentation page for all the supported package formats. See *Supported package manifests and package datafiles* for details.

6.1.4 Roadmap

This is a high level list of what we are working on and what is completed.

This is not updated regularly, see the [milestones](#) instead for updated shorter and longer term roadmaps.

Legend

✓ completed 🔄 In progress ☐ Planned, not started


Work in Progress

(see Completed features below)

Package manifest and dependency parsers

-  Docker image base (as part of: <https://github.com/pombredanne/conan>) #651
-  RubyGems base and dependencies #650 (code in <https://github.com/nexB/scancode-toolkit-contrib/>)
-  Perl, CPAN (basic in <https://github.com/nexB/scancode-toolkit-contrib/>)
-  Go : parsing for Godep in <https://github.com/nexB/scancode-toolkit-contrib/>
-  Windows PE #652
- ☐ RPM dependencies #649
- ☐ Windows Nuget dependencies #648
- ☒ Bower packages #654
- ☒ Python dependencies #653
- ☒ CRAN
- ☒ Plain packages
- ☐ other Java-related meta files (SBT, Ivy, Gradle, etc.)
- ☐ Debian debs
- ☐ other JavaScript (jspm, etc.)
- ☐ other Linux distro packages

License Detection

- ☒ support and detect license expressions (code in <https://github.com/nexB/license-expression>)
-  support and detect composite licenses
- ☒ support custom licenses
- ☐ move licenses data set to external separate repository
- ☒ Improved unknown license detection
- ☒ sync with external sources (DejaCode, SPDX, etc.)

Copyrights

- ☒ speed up copyright detection
- ☒ improved detected lines range
- ☒ streamline grammar of copyright parser
- ☒ normalize holders and authors for summarizing
- ☒ normalize and streamline results data format

Core features

- ✓ pre scan filtering (ignore binaries, etc)
- ✓ pre/post/output plugins! (worked as part of the GSoC by @yadsharaf)
- ✓ scan plugins (e.g. plugins that run a scan to collect data)
- ✓ support Python 3 #295
- 🕒 transparent archive extraction (as opposed to on-demand with extractcode)
- ☐ scancode.yml configuration file for exclusions, defaults, scan failure conditions, etc.
- ☐ support scan pipelines and rules to organize more complex scans
- ✓ scan baselining, delta scan and failure conditions (such as license change, etc) (spawned as its the [DeltaCode](#) project)
- ☐ dedupe and similarities to avoid re-scanning. For now only identical files are scanned only once.
- 🕒 Improved logging, tracing and error diagnostics
- ✓ native support for ABC Data (See [AboutCode Data Structure \(ABCD\)](#))

Classification, summarization and deduction

- 🕒 File classification #426
- ✓ summarize and aggregate data #377 at the top level

Source code support (some will be spawned as their own tool)

- 🕒 symbols : parsing complete in <https://github.com/nexB/scancode-toolkit-contrib/>
- 🕒 metrics : some elements in <https://github.com/nexB/scancode-toolkit-contrib/>

Compiled code support (will be spawned as their own tool)

- 🕒 ELF's : parsing complete in <https://github.com/nexB/scancode-toolkit-contrib/>
- 🕒 Java bytecode : parsing complete in <https://github.com/nexB/scancode-toolkit-contrib/>
- 🕒 Windows PE : parsing complete in <https://github.com/nexB/scancode-toolkit-contrib/>
- 🕒 Mach-O : parsing complete in <https://github.com/nexB/scancode-toolkit-contrib/>
- ☐ Dalvik/dex

Data exchange

- ✓ SPDX data conversion #338

Packaging

- ☐ simpler installation, automated installer
- ✓ distro-friendly packaging
- 🕒 unbundle and package as multiple libraries (commoncode, extractcode, etc)

Documentation

- ☐ integration in a build/CI loop
- ☐ end to end guide to analyze a codebase
- ☐ hacking guides
- ☐ API doc when using ScanCode as a library

CI integration

- ☐ Plugins for CI (Jenkins, etc)
- ☐ Integration for CI (Travis, Appveyor, Drone, etc)

Other work in progress

- 🕒 ScanCode server: Separate project: <https://github.com/nexB/scancode-server>. Will include Integration / webhooks for Github, Bitbucket.
- 🕒 VulnerableCode: NVD and CVE lookups: Separate project: <https://github.com/nexB/vulnerablecode>
- ✓ ScanCode Workbench: desktop app for scan review: Separate project: <https://github.com/nexB/scancode-workbench>
- ☐ DependentCode: dynamic dependencies resolutions: Separate project: <https://github.com/nexB/dependentcode>

Package mining and matching

(Note that this will be a separate project) Some code is in <https://github.com/nexB/scancode-toolkit-contrib/>

- 🕒 exact matching
- 🕒 attribute-based matching
- 🕒 fuzzy matching
- ☐ peer-reviewed meta packages repo
- ☐ basic mining of package repositories

Other

- ☐ Crypto code detection

Completed features

Core scans

- ✓ exact license detection
- ✓ approximate license detection
- ✓ copyright detection
- ✓ file information (size, type, etc.)
- ✓ URLs, emails, authors

Outputs and UI

- ✓ JSON compact and pretty
- ✓ plain HTML tables, also usable in a spreadsheet
- ✓ fancy HTML ‘app’ with a file tree navigation, and scan results filtering, search and sorting
- ✓ simple scan summary
- ✓ SPDX output

Package and dependencies

- ✓ common model for package data
- ✓ basic support for common package format
- ✓ RPM package base
- ✓ NuGet package base
- ✓ Python package base
- ✓ PHP Composer package support with dependencies
- ✓ Java Maven POM package support with dependencies
- ✓ npm package support with dependencies

Speed!

- ✓ accelerate license detection indexing and scanning; include caching
- ✓ scan using multiple processes to speed up overall scan
- ✓ cache per-file scan to disk and stream final results

Other

- ✓ archive extraction with extractcode
- ✓ conversion of scan results to CSV
- ✓ improved error handling, verbose and diagnostic output

6.1.5 Google Summer of Code 2017 - Final report

Project: Plugin architecture for ScanCode

Yash D. Saraf yashdsaraf@gmail.com

This project's purpose was to create a decoupled plugin architecture for ScanCode such that it can handle plugins at different stages of a scan and can be coupled at runtime. These stages were,

1. Format :

In this stage, the plugins are supposed to run **after** the scanning is done and **post-scan** plugins are called. These plugins could be used for:

- **converting the scanned output to the given format (say csv, json, etc.)**

HOWTO

Here, a plugin needs to add an entry in the `scancode_output_writers` entry point in the following format : `'<format> = <module>:<function>'`.

- `<format>` is the format name which will be used as the command line option name (e.g csv or json).
- `<module>` is a python module which implements the output hook specification.
- `<function>` is the function to which the scan output will be passed if this plugin is called.

The `<format>` name will be automatically added to the `--format` command line option and (if called) the scanned data will be passed to the plugin.

2. Post-scan :

In this stage, the plugins are supposed to run **after** the scanning is done. Some uses for these plugins were:

- **summarization of scan outputs**

e.g A post-scan plugin for marking `is_source` to true for directories with ~90% of source files.

- **simplification of scan outputs**

e.g The `--only-findings` option to return files or directories with findings for the requested scans. Files and directories without findings are omitted (not considering basic file information as findings)).

This option already existed, I just ported it to a post-scan plugin.

HOWTO

Here, a plugin needs to add an entry in the `scancode_post_scan` entry point in the following format '`<name> = <module>:<function>`'

- `<name>` is the command line option name (e.g **only-findings**).
- `<module>` is a python module which implements the `post_scan` hook specification.
- `<function>` is the function to which the scanned files will be passed if this plugin is called

The command line option for this plugin will be automatically created using the `<function>` 's doctring as its help text and (if called) the scanned files will be passed to the plugin.

3. Pre-scan :

In this stage, the plugins are supposed to run **before** the scan starts. So the potential uses for these types of plugins were to:

- **ignore files based on a given pattern (glob)**
- **ignore files based on their info i.e size, type etc.**
- **extract archives before scanning**

HOWTO

Here, a plugin needs to add an entry in the `scancode_pre_scan` entry point in the following format : '`<name> = <module>:<class>`'

- `<name>` is the command line option name (e.g **ignore**).
- `<module>` is a python module which implements the `pre_scan` hook specification.
- `<class>` is the class which is instantiated and its appropriate method is invoked if this plugin is called. This needs to extend the `plugincode.pre_scan.PreScanPlugin` class.

The command line option for this plugin will be automatically created using the `<class>` 's doctring as its help text. Since there isn't a single spot where pre-scan plugins can be plugged in, more methods to `PreScanPlugin` class can be added which can represent different hooks, say to add or delete a scan there might be a method called `process_scan`.

If a plugin's option is passed by the user, then the `<class>` is instantiated with the user input and its appropriate aforementioned methods are called.

4. Scan (proper):

In this stage, the plugins are supposed to run **before** the scan starts and **after** the pre-scan plugins are called. These plugins would have been used for

- adding or deleting scans
- adding dependency scans (whose data could be used in other scans)

No development has been done for this stage, but it will be quite similar to pre-scan.

5. Other work:

Group cli options in cli help

Here, the goal was to add command line options to pre-defined groups such that they are displayed in their respective groups when `scancode -h` or `scancode --help` is called. This helped to better visually represent the command line options and determine more easily what context they belong to.

Add a Resource class to hold all scanned info * Ongoing *

Here, the goal was to create a `Resource` class, such that it holds all the scanned data for a resource (i.e a file or a directory). This class would go on to eventually encapsulate the caching logic entirely. For now, it just holds the `info` and `path` of a resource.

6. What's left?

- Pre-scan plugin for archive extractions
- Scan (proper) plugins
- More complex post-scan plugins
- Support plugins written in languages other than python

Additionally, all my commits can be found [here](#).

6.1.6 Google Summer of Code 2019 - Final report

Project: `scancode-toolkit` to Python 3

Owner: [Abhishek Kumar](#)

Mentor: [Philippe Ombredanne](#)

Overview

Problem: Since Python 2.7 will retire in few months and will not be maintained any longer.

Solution: `Scancode` needs to be ported to python 3 and all test suites must pass on both version of Python. The main difference that makes Python 3 better than Python 2.x is that the support for unicode is greatly improved in Python 3. This will also be useful for `scancode` as `scancode` has users in more than 100 languages and it's easy to translate strings from unicode to other languages.

Objective: To make `scancode-toolkit` installable on on Python 3.6 and higher, as presently it installs with Python 2.7 only.

Implementation

- It was started in development mode(editable mode) and then it was moved to work in virtual environments.
- I have worked module by module according to the order of hierarchy of modules. For example :All module is dependent on commoncode, so it must be ported first. In this way we have created the Porting order:
 1. commoncode
 2. plugincode
 3. typecode
 4. extractcode
 5. textcode
 6. scancode basics (some tests are integration tests and will have to wait to be ported)
 7. formattedcode, starting with JSON (some tests are integration tests and will have to wait to be ported)
 8. cluecode
 9. licensedcode
 10. packagedcode (depends on licensecode)
 11. summarycode
 12. fixup the remaining bits and tests

After porting each module, I have marked these modules as ported `scanpy3` with help of **conffest** plugin (created by @pombredanne). **Conffest** plugin is heart of this project. Without this, it was very difficult to do. Dependencies was fixed at the time of porting the module where it was used.

Challenging part of Project

It is very difficult to deal with paths on different operating systems.The issue is around macOS/Windows/Linux. The first two OS handle unicode paths comfortably on Python 2 and 3 but not completely on macOS Mojave because its filesystem encoding is APFS. Linux paths are bytes and `os.listdir` is broken on Python 2. As a result you can only sanely handle Linux paths as bytes on Python 2. But on Python 3 path seems to be corrected as `unicode` on Linux.

For more details visit here :

- <https://vstinner.github.io/painful-history-python-file-system-encoding.html>
- [jaraco/path.py#130](#)

We came with various Solution:

- To use `pathlib` which generally handle paths correctly across platforms. And for backports we use `pathlib 2`. But this solution also fails because `pathlib 2` does not work as expected wrt unicode vs bytes. And `os.listdir` also doesn't work properly.
- To use `path.py` which handles the paths across all the platforms even on macOS Mojave .
- Use bytes on linux and python 3 and `unicode` everywhere.

We choose the third solution because it is most fundamental and simple and easy to use.

Project was tracked in this ticket [nexB/scancode-toolkit#295](#)

Project link : [Port Scancode to Python 3](#)

My contribution : [List of Commits](#)

Note : Please give your feedback [here](#)

Outcome

Now we have liftoff on Python 3 . We are able to run basic scans without errors on develop branch. You check it by running `scancode -clipeu samples/ --json-pp - -n4` .

At last I would like to thanks my Mentor @**pombredanne** aka [Philippe Ombredanne](#) . He has helped lot in completing this project. He is very supportive and responsive. I have learned a lot from him. By his encouragement and motivation, I am very improving day by day, building and developing my skills. I have completed all the tasks that were in the scope of this GSoC project.

6.1.7 Google Summer of Code 2021 Final report

Organisation - AboutCode

Akanksha Garg <akanksha.garg2k@gmail.com>

[GITHUB](#)

Project: Detect Unknown Licenses and Indirect License References in ScanCode

[ScanCode-toolkit](#)

[Project Link](#)

[Proposal](#)

Description

The main motive of this project was to improve license detection of unknown licenses and follow references to indirect license references in ScanCode-TK

Improvement in the License Data Model Definition

Unknown Licenses are the ones which are matched to a license rule tagged with ‘unknown’ license key. Since these are some of the ‘special’ licenses , reporting them with special attributes will provide more clarification. Now unknown licenses are tagged with a new flag **“is_unknown”** to identify them beyond just the naming convention of having “unknown” as part of their name.

Rules that match at least one unknown license have a flag **“has_unknown”** set in the returned match results.

[nexB/scancode-toolkit#2548](#)

Reporting known and Unknown licenses separately

We considered having a separate section for of scan results to report ‘unknown licenses’ separately and not mixed with main license detection results. But after implementing a separate section for unknown ones ,it doesn’t seem to be good idea to have currently.

[nexB/scancode-toolkit#2578](#)

Follow License References to another file

Some license references such as “see license in file LICENSE.txt” e.g. mentions to look for license details in another file are reported as unknown license references and we could instead follow the referenced file to find what was detected there. The approach was to use already contained attribute `referenced_filenames` in license RULE data files. Since this was a `process_codebase` step in scan plugin , it was needed that our API function should return `referenced_filenames` to keep track of these files corresponding to licenses detected. This was tracked in -

[nexB/scancode-toolkit#2632](#)

The ``process_codebase`` step is tracked in -

[nexB/scancode-toolkit#2616](#)

Improve license detection of Unknown Licenses

The approach was to use index of n-grams for detecting unknowns besides having our actual detection of “unknown” license rules. Firstly matches were filtered after running our normal procedure of license detection and the remaining spans are run through a automaton index containing n-grams from all regular license texts and rules. This is tracked in -

[nexB/scancode-toolkit#2592](#)

Addition of some new Licenses

There were some licenses that were not present in Scancode-toolkit as for now. They have been added now.

[nexB/scancode-toolkit#2625](#)

Pre-GSoC

Contributions

- [nexB/scancode-toolkit#2423](#)
- [nexB/scancode-toolkit#2473](#)
- [nexB/scancode-toolkit#2464](#)
- [nexB/scancode-toolkit#2381](#)

I’ve had a wonderful summer during these 10 weeks journey and have learned plenty of things. I am thankful to Google and Aboutcode for giving me this opportunity to work with such an amazing community. I am fortunate to have mentors [Philippe Ombredanne](#) and [Ayan Sinha Mahapatra](#) who helped me a lot throughout my GSoC project and provided constant support.

PLUGINS DOCUMENTATION

7.1 Plugins

7.1.1 Plugin Architecture

Abstract:

The purpose of plugins is to create a decoupled architecture such that ScanCode can support extensibility at different stages of a scan. These stages are:

- Pre-scan: Before starting the scan proper, such as plugins to handle extraction of different archive types or instructions on how to handle certain types of files, or to collect filetypes. These plugins process a whole codebase at once.
- Scan proper: plugins to scan a file e.g. collect data and evidence from the files. These plugins process one file at a time and can do a whole codebase pass once all files are scanned.
- Post-scan: After the scan, e.g plugins for summarization and other aggregated operation once all scans are completed. These plugins process a whole codebase at once.
- Output and output filter: plugins for output creation and filtering such as formatting or converting output to other formats (such as json, spdx, csv, yaml). These plugins process a whole codebase at once.

Description:

This project aims at making scancode a “pluggable” system, where new functionalities can be added to scancode at runtime as “plugins”. These plugins can be hooked into scancode using some predefined hooks. I would consider pluggy as the way to go for a plugin management system.

Why pluggy?

Pluggy is well documented and maintained regularly, and has proved its worth in projects such as pytest. Pluggy relies on hook specifications and hook implementations (callbacks) instead of the conventional subclassing approach which may encourage tight-coupling in the overlying framework. Basically a hook specification contains method signatures (no code), these are defined by the application. A hook implementation contains definitions for methods declared in the corresponding hook specification implemented by a plugin.

As mentioned in the abstract, the plugin architecture will have 3 hook specifications (can be increased if required)

1. Pre - scan hook

- Structure -

```
prescan_hookspec = HookspecMarker('prescan')

@prescan_hookspec
def extract_archive(args):
```

Here the path of the archive to be extracted will be passed as an argument to the `extract_archive` function which will be called before scan, at the time of extraction. This will process the archive type and extract the contents accordingly. This functionality can be further extended by calling this function if any archive is found inside the scanning tree.

2. Scan proper hook

- Structure

```
scanproper_hookspec = HookspecMarker('scanproper')

@scanproper_hookspec
def add_cmdline_option(args):
```

This function will be called before starting the scan, without any arguments, it will return a dict containing the click extension details and possibly some help text. If this option is called by the user then the call will be rerouted to the callback defined by the click extension. For instance say a plugin implements functionality to add regex as a valid ignore pattern, then this function will return a dict as:

```
{
    'name': '--ignore-regex',
    'options': {
        'default': None,
        'multiple': True,
        'metavar': '<pattern>'
    },
    'help': 'Ignore files matching regex <pattern>'
    'call_after': 'is_ignored'
}
```

According to the above dict, if the option `--ignore-regex` is supplied, this function will be called after the `is_ignored` function and the data returned by the `is_ignored` function will be supplied to this function as its argument(s). So if the program flow was:

```
scancode()  scan()  resource_paths()  is_ignored()
```

It will now be edited to

```
scancode()  scan()  resource_paths()  is_ignored()  add_cmdline_option()
```

Options such as `call_after`, `call_before`, `call_first`, `call_last` can be defined to determine when the function is to be executed.

```
@scanproper_hookspec
def dependency_scan(args):
```

This function will be called before starting the scan without any arguments, it will return a list of file types or attributes which if encountered in the scanned tree, will call this function with the path to the file as an argument. This function can do some extra processing on those files and return the data to be processed as a dependency for the normal scanning process. E.g. It can return a list such as:

```
[ 'debian/copyright' ]
```

Whenever a file matches this pattern, this function will be called and the data returned will be supplied to the main scancode function.

3. Post - scan hook

- **Structure -**

```
postscan_hookspec = HookspecMarker('postscan')
```

```
@postscan_hookspec
def format_output(args):
```

This function will be called after a scan is finished. It will be supplied with path to the ABC data generated from the scan, path to the root of the scanned code and a path where the output is expected to be stored. The function will store the processed data in the output path supplied. This can be used to convert output to other formats such as CSV, SPDX, JSON, etc.

```
@postscan_hookspec
def summarize_output(args):
```

This function will be called after a scan is finished. It will be supplied the data to be reported to the user as well as a path to the root of the scanned node. The data returned can then be reported to the user. This can be used to summarize output, maybe encapsulate the data to be reported or omit similar file metadata or even classify files such as tests, code proper, licenses, readme, configs, build scripts etc.

- **Identifying or configuring plugins**

For python plugins, pluggy supports loading modules from setuptools entrypoints, E.g.

```
entry_points = {
    'scancode_plugins': [
        'name_of_plugin = ignore_regex',
    ]
}
```

This plugin can be loaded using the PluginManager class's load_setuptools_entrypoints('scancode_plugins') method which will return a list of loaded plugins.

For non python plugins, all such plugins will be stored in a common directory and each of these plugins will have a manifest configuration in YAML format. This directory will be scanned at startup for plugins. After parsing the config file of a plugin, the data will be supplied to the plugin manager as if it were supplied using setuptools entrypoints.

In case of non python plugins, the plugin executables will be spawned in their own processes and according to their config data, they will be passed arguments and would return data as necessary. In addition to this, the desired hook function can be called from a non python plugin using certain arguments, which again can be mapped in the config file.

Sample config file for a ignore_regex plugin calling scanproper hook would be:

```

name: ignore_regex
hook: scanproper
hookfunctions:
  add_cmdline_option: '-aco'
  dependency_scan: '-dc'
data:
  add_cmdline_option':
    - name: '--ignore-regex'
    - options:
      - default: None
      - multiple: True
      - metavar: <pattern>
    - help: 'Ignore files matching regex <pattern>'
    - call_after: 'is_ignored'

```

Existing solutions:

An alternate solution to a “pluggable” system would be the more conventional approach of adding functionalities directly to the core codebase, which removes the abstraction layer provided by a plugin management and hook calling system.

7.1.2 License Policy Plugin

This plugin allows the user to apply policy details to a scancode scan, depending on which licenses are detected in a particular file. If a license specified in the Policy file is detected by scancode, this plugin will apply that policy information to the Resource as a new attribute: `license_policy`.

Policy File Specification

The Policy file is a YAML (.yaml) document with the following structure:

```

license_policies:
- license_key: mit
  label: Approved License
  color_code: '#008000'
  icon: icon-ok-circle
- license_key: agpl-3.0
  label: Approved License
  color_code: '#008000'
  icon: icon-ok-circle
- license_key: broadcom-commercial
  label: Restricted License
  color_code: '#FFcc33'
  icon: icon-warning-sign

```

The only required key is `license_key`, which represents the ScanCode license key to match against the detected licenses in the scan results.

In the above example, a descriptive label is added along with a color code and CSS id name for potential visual display.

Using the Plugin

To apply License Policies during a ScanCode scan, specify the `--license-policy` option.

For example, use the following command to run a File Info and License scan on `/path/to/codebase/`, using a License Policy file found at `~/path/to/policy-file.yml`:

```
$ scancode -clipecu /path/to/codebase/ --license-policy ~/path/to/policy-file.yml --json-
  ↳pp
  ~/path/to/scan-output.json
```

Example Output

Here is an example of the ScanCode output after running `--license-policy`:

```
{
  "path": "samples/zlib/deflate.c",
  "type": "file",
  "detected_license_expression": "zlib",
  "detected_license_expression_spdx": "Zlib",
  "license_detections": [
    {
      "license-expression": "zlib",
      ...
      ...
      ...
    }
  ],
  "license_policy": {
    "license_key": "zlib",
    "label": "Approved License",
    "color_code": "#008000",
    "icon": "icon-ok-circle"
  },
  "scan_errors": []
}
```

7.1.3 Plugin Tutorials

- *Add A Post-Scan Plugin*

7.1.4 CPP Includes Plugin

This plugin allows users to collect the `#includes` statements in C/C++ files.

Using the Plugin

User needs to use the `--cpp-includes` option.

The following command will collect the `#includes` statements from C/C++ files.:

```
$ scancode --cpp-includes /path/to/codebase/ --json-pp ~/path/to/scan-output.json
```

Example Output

Here is an sample output:

```
{
  "path": "zlib_deflate/deflate.c",
  "type": "file",
  "cpp_includes": [
    "<linux/module.h",
    "<linux/zutil.h",
    "\"defutil.h"
  ],
  "scan_errors": []
},
{
  "path": "zlib_deflate/deflate_syms.c",
  "type": "file",
  "cpp_includes": [
    "<linux/module.h",
    "<linux/init.h",
    "<linux/zlib.h"
  ],
  "scan_errors": []
}
```

7.1.5 LKMClue Plugin

This plugin allows users to collect LKM module clues and type indicating a possible Linux Kernel Module.

Using the Plugin

User needs to use the `--lkmclue` option.

The following command will collect the LKM module clues from the input location:

```
$ scancode --lkmclue /path/to/codebase/ --json-pp ~/path/to/scan-output.json
```

Example Output

Here is an sample output:

```
{
  "path": "zlib_deflate/deflate.c",
  "type": "file",
  "lkm_clue": {
    "lkm-header-include": [
      "include <linux/module.h>"
    ]
  },
  "scan_errors": []
},
{
  "path": "zlib_deflate/deflate_syms.c",
  "type": "file",
  "lkm_clue": {
    "lkm-header-include": [
      "include <linux/module.h>"
    ],
    "lkm-license": [
      "GPL"
    ]
  },
  "scan_errors": []
}
```

7.1.6 Dwarf Plugin

This plugin allows users to collect source code path/name from compilation units found in ELF DWARFs.

Specification

This plugin will only work with non-stripped ELF's with debug symbols.

Using the Plugin

User needs to use the `--dwarf` option.

The following command will collect all the dwarf references found in non-stripped ELF's:

```
$ scancode --dwarf /path/to/codebase/ --json-pp ~/path/to/scan-output.json
```


Example Output

Here is an sample output:

```
{
  "path": "project/stripped.ELF",
  "type": "file",
  "dwarf_source_path": [],
  "scan_errors": []
},
{
  "path": "project/non-stripped.ELF",
  "type": "file",
  "dwarf_source_path": ['/tmp/test.c'],
  "scan_errors": []
}
```

MISCELLANEOUS DOCUMENTS

8.1 Miscellaneous

8.1.1 FAQ

Why ScanCode?

We could not find an existing tool (open source or commercial) meeting our needs:

- usable from the command line or as library
- running on Linux, Mac and Windows
- written in a higher level language such as Python
- easy to extend and evolve
- accurately detecting most licenses and copyrights

How is ScanCode different from Debian licensecheck?

At a high level, ScanCode detects more licenses and copyrights than licensecheck does, reporting more details about the matches. It is likely slower.

In more details: ScanCode is a Python app using a data-driven approach (as opposed to carefully crafted regex like licensecheck uses):

- for license scan, the detection is based on a (large) number of license full texts (~2100) and license notices, mentions and variants (~32,000) and is data-driven as opposed to regex-driven. It detects and reports exactly where license text is found in a file. Just throw in more license texts to improve the detection.
- for copyright scan, the approach is natural language parsing grammar; it has a few thousand tests.
- licenses and copyrights are detected in texts and binaries
- licenses and copyrights are also detected in structured package manifests

Licensecheck (available here for reference: <https://metacpan.org/pod/App::Licensecheck>) is a Perl script using hand-crafted regex patterns to find typical copyright statements and about 50 common licenses. There are about 50 license detection tests.

A quick test (in July 2015, before a major refactoring, but for this may still be still valid) shows several things that are not detected by licensecheck that are detected by ScanCode.

How can I integrate ScanCode in my application?

More specifically, does this tool provide an API which can be used by us for the integration with my system to trigger the license check and to use the result?

In terms of API, there are two stable entry points:

- #. The JSON output when you use it as a command line tool from any language or when you call the `scan-code.cli.scancode` function from a Python script.
- #. Otherwise the `scancode.cli.api` module provides a simple function if you are only interested in calling a certain service on a given file (such as license detection or copyright detection)

Can I install ScanCode in a Unicode path?

Yes and this is fully supported and tested. See <https://github.com/nexB/scancode-toolkit/issues/867> for a previous bug that was preventing this.

There was a bug in virtualenv <https://github.com/pypa/virtualenv/issues/457> that is now fixed and has been extensively tested for ScanCode.

The line numbers for a copyright found in a binary are weird. What do they mean?

When scanning binaries, the line numbers are just a relative indication of where a detection was found: there is no such thing as lines in a binary. The numbers reported are based on the strings extracted from the binaries, typically broken as new lines with each NULL character.

8.1.2 Support

Documentation

The ScanCode toolkit documentation lives at <https://scancode-toolkit.readthedocs.io/>.

Issue Tracker

Post issues you are having and bugs as [GitHub tickets](#)

Discussions

If you want to ask questions or anything else that you think are not bugs/new features, open a [discussion](#)

Join the conversation

Join our [general chatroom](#) to chat with aboutcode community members, and if you want to talk to users and developers of ScanCode Toolkit, use [scancode room](#)

8.1.3 Runtime Performance Reports

These are reports of runtimes for real life scans:

2015-09-03 by @rrjohnston

- On Ubuntu 12.04 x86_64 Python 2.7.3 and ScanCode Version 1.3.1
- Specs: 40 threads (2 processors, 10 cores each, with hyperthreading) 3.1 GHz 128GB RAM 8TB controller RAID5
- scanned 195676 files in about 16.7 hours or about 3.25 file per second (using defaults licenses and copyrights)
- notes: this version of ScanCode runs on a single thread so it does not make good use of extra processing power.

8.1.4 Versioning approach

ScanCode is composed of code and data (mostly license data used for license detection). In the past, we have tried using calver for code versioning to also convey that the data contained in ScanCode was updated but it proved to be not as clear and as effective as planned so we are switching back to semver which is simpler and overall more useful for users. We also want to provide hints about JSON output data format changes.

Therefore, this is our versioning approach starting with version 30.0.0:

- ScanCode releases are versioned using semver as documented at <https://semver.org> using major.minor.patch versioning.
- Significant changes to the data (license or copyright detection) is considered a major version change even if there are no code changes. The rationale is that in our case the data has the same impact as the code. Using outdated data is like using old code and means that several licenses may not be detected correctly. Any data change triggers at least a minor version change.
- We will signal separately to users with warnings messages when ScanCode needs to be upgraded because its data and/or code are out of date.

In addition to the main code version, we also maintain a secondary output data format version using also semver with two segments. The versioning approach is adapted for data this way:

- The first segment –the major version– is incremented when data attributes that are removed, renamed, changed or moved (but not reordered) in the JSON output. Reordering the attributes of a JSON object is not considered as a change and does not trigger a version change.
- The second segment –the minor version– of the output format is incremented for an addition of attributes to the JSON output.
- We store the output format version string in the JSON output object as the first attribute and display that also in the help.
- This output format versioning applies only to the JSON, pretty-printed JSON, YAML and JSON lines formats. It does not apply to CSV and any other formats. For these other formats there is no versioning and guaranteed format stability (or there may be some other rationale and convention for versioning like for SPDX).
- The output format version is incremented by when a new ScanCode tagged release is published
- We document in the CHANGELOG the output format changes in any new format version.
- For any format version changes, we will provide a documentation on the format and its updates using JSON examples and a comprehensive and updated data dictionary. See <https://github.com/nexB/scancode-toolkit/issues/2008> for details.

REFERENCE DOCUMENTS

Reference documents provide reference pages for technical reference information about ScanCode Toolkit, including how it works and supported features.

9.1 Reference Docs

9.1.1 Overview

How does ScanCode detect licenses?

For license detection, ScanCode uses a (large) number of license texts and license detection ‘rules’ that are compiled in a search index. When scanning, the text of the target file is extracted and used to query the license search index and find license matches.

For copyright detection, ScanCode uses a grammar that defines the most common and less common forms of copyright statements. When scanning, the target file text is extracted and ‘parsed’ with this grammar to extract copyright statements.

ScanCode-Toolkit performs the scan on a codebase in the following steps :

1. Collect an inventory of the code files and classify the code using file types,
2. Extract files from any archive using a general purpose extractor
3. Extract texts from binary files if needed
4. Use an extensible rules engine to detect open source license text and notices
5. Use a specialized parser to capture copyright statements
6. Identify packaged code and collect metadata from packages
7. Report the results in the formats of your choice (JSON, CSV, etc.) for integration with other tools

Scan results are provided in various formats:

- a JSON file simple or pretty-printed,
- SPDX tag value or XML, RDF formats,
- CSV,
- a simple unformatted HTML file that can be opened in browser or as a spreadsheet.

For each scanned file, the result contains:

- its location in the codebase,

- the detected licenses and copyright statements,
- the start and end line numbers identifying where the license or copyright was found in the scanned file, and
- reference information for the detected license.

For archive extraction, ScanCode uses a combination of Python modules, 7zip and libarchive/bsdtar to detect archive types and extract these recursively.

Several other utility modules are used such as libmagic for file and mime type detection.

9.1.2 License Detection Updates

References:

- [Issue](#)
- [Pull Request](#)
- [A presentation on this](#)

The Problem:

The goal was to reduce false-positives in scancode license detection results, especially *unknown-license-reference* detections and approximate detections reporting best-guess license_expressions. To tackle this the following solution elements were discussed and implemented:

1. Reporting the primary, declared license in a scan summary record
2. tagging mandatory portions in rules [#2773](#)
3. Adding license detections by combine multiple license matches [#2961](#)
4. Integrating the existing scancode-analyzer tool into SCTK to combine multiple matches based on statistics and heuristics [#2961](#)
5. Reporting license clues when the matched license rule data is not sufficient to create a LicenseDetection [#2961](#)
6. web app for efficient scan and review of a single license to ease reporting license detection issues [nexB/scancode.io#450](#)
7. also apply LicenseDetection to package license detections [#2961](#)
8. rename resource and package license fields [#2961](#)

Some other elements are still WIP, see [issue #3300](#) for more details on this.

What is a LicenseDetection?

A detection which can have one or multiple LicenseMatch in them, and creates a License Expression that we finally report.

Properties:

- A file can have multiple LicenseDetections (separated by non-legalese lines)
- This can be from a file directly or a package.
- We should be mostly certain of a proper license detection to report a LicenseDetection, i.e. we should have ideally gotten rid of false positives and wrong license matches, or improved them.
- One LicenseDetection can have matches from different files, in case of local license references.

- We don't remove any detection matches, but we add more matches only to rectify and correct the license_expression.

Also there are two levels of reporting license detections:

- File/package level License Detections
- Codebase level unique License Detections (summarized from the file/package level detections)

Examples

A License Intro example:

Consider the following text:

```

/*****
* Copyright (c) 2019 Red Hat, Inc.
*
* This program and the accompanying materials are made
* available under the terms of the Eclipse Public License 2.0
* which is available at https://www.eclipse.org/legal/epl-2.0/
*
* SPDX-License-Identifier: EPL-2.0
*****/

```

The text:

```

"This program and the accompanying materials are made\n* available under the terms of the
↪"

```

is detected as `unknown-license-reference` with `is_license_intro` as `True`, and has several `ep1-2.0` detections after that.

This can be considered as a single License Detection with its detected license-expression as `ep1-2.0`. The matches of this license detection would also have the matches with the `unknown-license-reference`, but they will not be present in the final `license_expression`.

A License Reference example:

Consider the two following files:

file.py:

```

This is free software. See COPYING for details.

```

COPYING:

```

license: apache 2.0

```

Here there will be a `unknown-license-reference` detected in `file.py` and this actually references the license detected in `COPYING` which is `apache-2.0`.

This can be considered a single LicenseDetection with both the license matches from both files, and a concluded `license_expression` `apache-2.0` instead of the `unknown-license-reference`.

Chnagelog Summary

- There is a new `license_detections` codebase level attribute with all the unique license detections in the whole scan, both in resources and packages.
- The data structure of the JSON output has changed for licenses at resource level, also with new attribute names, `licenses` -> `license_detections` and `license_expressions` -> `detected_license_expression` also with a SPDX version of the same. As license detection attributes we have: `license_expression`, `identifier` and `matches`. We also have a `detection_log` (present optionally if the `--license-diagnostics` option is enabled).
- There are `license_detections` now reported at packages, and the data structure of license attributes in `package_data` and the codebase level `packages` has been also updated: `license_expression` -> `declared_license_expression`, also with its SPDX version, `declared_license` -> `extracted_license_statement`, and also secondary license detections data in: `other_license_expression` and `other_license_detections`.
- Instead of reporting one match for each license key of a matched license expression, we now report one single match for each matched license expression, avoiding data duplication. Inside each match, we also list each match and matched rule attributes directly to avoiding nesting.
- License and Rule reference data is not reported at match level in license detections and instead is reported at codebase-level with a new CLI option `--license-references` as new attributes: `license_references` and `license_rule_references` that list unique detected license and license rules with their details.

Change in License Data format: Resource

The data structure of the JSON output has changed for licenses at file level:

- The `licenses` attribute is deleted.
- A new `license_detections` attribute contains license detections in that file. This object has three attributes: `license_expression`, `detection_log` and `matches`. `matches` is a list of license matches and is roughly the same as `licenses` in the previous version with additional structure changes detailed below.
- A new attribute `license_clues` contains license matches with the same data structure as the `matches` attribute in `license_detections`. This contains license matches that are mere clues and were not considered to be a proper conclusive license detection.
- The `license_expressions` list of license expressions is deleted and replaced by a `detected_license_expression` single expression. Similarly `spdx_license_expressions` was removed and replaced by `detected_license_expression_spdx`.

See the before/after results for a file to compare the changes.

Before:

```
{
  "licenses": [
    {
      "key": "apache-2.0",
      "score": 100.0,
      "name": "Apache License 2.0",
      "short_name": "Apache 2.0",
      "category": "Permissive",
      "is_exception": false,
      "is_unknown": false,
      "owner": "Apache Software Foundation",
```

(continues on next page)

(continued from previous page)

```

    "homepage_url": "http://www.apache.org/licenses/",
    "text_url": "http://www.apache.org/licenses/LICENSE-2.0",
    "reference_url": "https://scancode-licensedb.aboutcode.org/apache-2.0",
    "scancode_text_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/licenses/apache-2.0.LICENSE",
    "scancode_data_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/licenses/apache-2.0.yml",
    "spdx_license_key": "Apache-2.0",
    "spdx_url": "https://spdx.org/licenses/Apache-2.0",
    "start_line": 1,
    "end_line": 1,
    "matched_rule": {
        "identifier": "apache-2.0_65.RULE",
        "license_expression": "apache-2.0",
        "licenses": [
            "apache-2.0"
        ],
        "referenced_filenames": [],
        "is_license_text": false,
        "is_license_notice": false,
        "is_license_reference": false,
        "is_license_tag": true,
        "is_license_intro": false,
        "has_unknown": false,
        "matcher": "1-hash",
        "rule_length": 4,
        "matched_length": 4,
        "match_coverage": 100.0,
        "rule_relevance": 100,
        "is_builtin": true
    },
    "matched_text": "License: Apache-2.0"
}
],
"license_expressions": [
    "apache-2.0"
]
}

```

After:

```

"detected_license_expression": "apache-2.0",
"detected_license_expression_spdx": "Apache-2.0",
"license_detections": [
    {
        "license_expression": "apache-2.0",
        "matches": [
            {
                "score": 100.0,
                "start_line": 1,
                "end_line": 1,
                "matched_length": 4,

```

(continues on next page)

(continued from previous page)

```

        "match_coverage": 100.0,
        "matcher": "1-hash",
        "license_expression": "apache-2.0",
        "rule_identifier": "apache-2.0_65.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
→licensedcode/data/rules/apache-2.0_65.RULE",
        "matched_text": "license: apache 2.0"
    }
],
    "detection_log": [],
    "identifier": "apache_2_0-ec759ae0-ea5a-f138-793e-388520e080c0"
}
],
"license_clues": [],

```

Change in License Data format: Package

License data attributes has also changed in packages:

Before:

```

{
  "type": "cocoapods",
  "namespace": null,
  "name": "LoadingShimmer",
  "version": "1.0.3",
  "license_expression": "mit AND unknown",
  "declared_license": ":type = MIT, :file = LICENSE",
  "datasource_id": "cocoapods_podspec",
  "purl": "pkg:cocoapods/LoadingShimmer@1.0.3"
}

```

After:

```

"declared_license_expression": "mit",
"declared_license_expression_spdx": "MIT",
"license_detections": [
  {
    "license_expression": "mit",
    "matches": [
      {
        "score": 100.0,
        "start_line": 1,
        "end_line": 1,
        "matched_length": 4,
        "match_coverage": 100.0,
        "matcher": "1-hash",
        "license_expression": "mit",
        "rule_identifier": "mit_in_manifest.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/

```

(continues on next page)

(continued from previous page)

```

↪licensedcode/data/rules/mit_in_manifest.RULE",
    "matched_text": ":type = MIT, :file = LICENSE"
  }
],
  "identifier": "mit-74f1df5b-f94d-2423-6bb8-3e4d809c26a5"
}
],
"other_license_expression": null,
"other_license_expression_spdx": null,
"other_license_detections": [],
"extracted_license_statement": ":type = MIT, :file = LICENSE",

```

Previously in package data only the `license_expression` was present and it was very hard to debug license detections. Now there's a `license_detections` field with the detections, same as the resource `license_detections`, with additional `declared_license_expression` and `other_license_expression` with their SPDX counterparts. The `declared_license` field also has been renamed to `extracted_license_statement`.

Codebase level Unique License Detection

We now have a new codebase level attribute `license_detections` which has Unique License Detection across the codebase, in both packages and resources. They are linked by a common attribute `identifier` containing the `license_expression` and a UUID generated from the match content. The match level data is only present at the resource level if needed, to look at details.

New codebase level attribute:

```

{
  "license_detections": [
    {
      "identifier": "epl-1.0-583490fb-0b3a-f445-a1b9-1b96423b9ec3",
      "license_expression": "epl-1.0",
      "detection_count": 2,
      "detection_log": []
    }
  ]
}

```

For the corresponding resource level license detection:

```

"license_detections": [
  {
    "license_expression": "epl-1.0",
    "matches": [
      {
        "score": 99.34,
        "start_line": 12,
        "end_line": 25,
        "matched_length": 150,
        "match_coverage": 99.34,
        "matcher": "3-seq",
        "license_expression": "epl-1.0",
        "rule_identifier": "epl-1.0_3.RULE",

```

(continues on next page)

(continued from previous page)

```

        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/rules/epl-1.0_3.RULE",
    },
    {
        "score": 100.0,
        "start_line": 17,
        "end_line": 17,
        "matched_length": 8,
        "match_coverage": 100.0,
        "matcher": "2-aho",
        "license_expression": "epl-1.0",
        "rule_identifier": "epl-1.0_7.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/rules/epl-1.0_7.RULE",
    }
],
"detection_log": [],
"identifier": "epl_1_0-583490fb-0b3a-f445-a1b9-1b96423b9ec3"
}
]

```

LicenseMatch Result Data

LicenseMatch data was based on a `license key` instead of being based on a `license-expression`.

So if there is a `gpl-2.0 AND patent-disclaimer` license expression detected from a single LicenseMatch, there were two entries in the `licenses` list for that resource, one for each license key, (here `gpl-2.0` and `patent-disclaimer` respectively). This repeats the match details as these two entries have the same details except the license key.

We should only add one entry per match (and therefore per rule) and here the primary attribute should be the `license-expression`, rather than the `license-key`.

We also used to create a mapping inside a mapping in these license details to refer to the license rule (and there are other inconsistencies in how we report here). We are now just reporting a flat mapping here, and all the rule details are also not present in the license match, and only available as an optional reference.

See this before/after comparison to see how the license data in results has evolved.

Before:

```

"licenses": [
  {
    "key": "gpl-2.0",
    "score": 100.0,
    "name": "GNU General Public License 2.0",
    "short_name": "GPL 2.0",
    "category": "Copyleft",
    "is_exception": false,
    "is_unknown": false,
    "owner": "Free Software Foundation (FSF)",

```

(continues on next page)

(continued from previous page)

```

    "homepage_url": "http://www.gnu.org/licenses/gpl-2.0.html",
    "text_url": "http://www.gnu.org/licenses/gpl-2.0.txt",
    "reference_url": "https://scancode-licensedb.aboutcode.org/gpl-2.0",
    "scancode_text_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/licenses/gpl-2.0.LICENSE",
    "scancode_data_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/licenses/gpl-2.0.yml",
    "spdx_license_key": "GPL-2.0-only",
    "spdx_url": "https://spdx.org/licenses/GPL-2.0-only",
    "start_line": 4,
    "end_line": 30,
    "matched_rule": {
      "identifier": "gpl-2.0_and_patent-disclaimer_3.RULE",
      "license_expression": "gpl-2.0 AND patent-disclaimer",
      "licenses": [
        "gpl-2.0",
        "patent-disclaimer"
      ],
      "referenced_filenames": [],
      "is_license_text": false,
      "is_license_notice": true,
      "is_license_reference": false,
      "is_license_tag": false,
      "is_license_intro": false,
      "has_unknown": false,
      "matcher": "2-aho",
      "rule_length": 185,
      "matched_length": 185,
      "match_coverage": 100.0,
      "rule_relevance": 100
    },
  },
  {
    "key": "patent-disclaimer",
    "score": 100.0,
    "name": "Generic patent disclaimer",
    "short_name": "Generic patent disclaimer",
    "category": "Permissive",
    "is_exception": false,
    "is_unknown": false,
    "owner": "Unspecified",
    "homepage_url": null,
    "text_url": "",
    "reference_url": "https://scancode-licensedb.aboutcode.org/patent-disclaimer",
    "scancode_text_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/licenses/patent-disclaimer.LICENSE",
    "scancode_data_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/licenses/patent-disclaimer.yml",
    "spdx_license_key": "LicenseRef-scancode-patent-disclaimer",
    "spdx_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/licensedcode/
↪data/licenses/patent-disclaimer.LICENSE",
    "start_line": 4,
  }

```

(continues on next page)

(continued from previous page)

```

    "end_line": 30,
    "matched_rule": {
      "identifier": "gpl-2.0_and_patent-disclaimer_3.RULE",
      "license_expression": "gpl-2.0 AND patent-disclaimer",
      "licenses": [
        "gpl-2.0",
        "patent-disclaimer"
      ],
      "referenced_filenames": [],
      "is_license_text": false,
      "is_license_notice": true,
      "is_license_reference": false,
      "is_license_tag": false,
      "is_license_intro": false,
      "has_unknown": false,
      "matcher": "2-aho",
      "rule_length": 185,
      "matched_length": 185,
      "match_coverage": 100.0,
      "rule_relevance": 100
    }
  },
  "license_expressions": [
    "gpl-2.0 AND patent-disclaimer"
  ],

```

After:

```

"license_detections": [
  {
    "license_expression": "gpl-2.0 AND patent-disclaimer",
    "matches": [
      {
        "score": 100.0,
        "start_line": 4,
        "end_line": 30,
        "matched_length": 185,
        "match_coverage": 100.0,
        "matcher": "2-aho",
        "license_expression": "gpl-2.0 AND patent-disclaimer",
        "rule_identifier": "gpl-2.0_and_patent-disclaimer_3.RULE",
        "rule_relevance": 100,
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪ licensedcode/data/rules/gpl-2.0_and_patent-disclaimer_3.RULE"
      }
    ],
    "identifier": "gpl_2_0_and_patent_disclaimer-3bb2602f-86f5-b9da-9bf5-b52e6920c8d1"
  }
],

```

Only reference License related data

Before 32.x all license related data was inlined in each match, and this repeats a lot of information. This repetition exists in three levels:

- License-level Data (a license-key)
- Rule-level Data (a license rule)
- LicenseDetection Data (a license detection)

License Data

This is referencing data related to whole licenses, references by their license key.

Example: `apache-2.0`

Other attributes are it's full test, links to origin, licenseDB, spdx, osi etc.

Rule Data

This is referencing data related to a LicenseDB entry. I.e. the identifier is a *RULE* or a *LICENSE* file.

Example: `apache-2.0_2.RULE`

Other attributes are it's license-expression, the boolean fields, length, relevance etc.

CLI option

This is now default with the CLI option `--license`, which references from the match License-level Data and LicenseDB-level Data, and removes the actual data from the matches, and adds them to two top-level lists.

Comparison: Before/After license references

To compare how the license output data changes between when license references are not collected vs when they are collected (which is default from version 32.x), check out the before/after comparison below.

Before:

```
{
  "files": [
    {
      "detected_license_expression": "apache-2.0",
      "detected_license_expression_spdx": "Apache-2.0",
      "license_detections": [
        {
          "license_expression": "apache-2.0",
          "detection_log": [
            "not-combined"
          ],
          "matches": [
            {
              "score": 100.0,
```

(continues on next page)

(continued from previous page)

```

        "start_line": 1,
        "end_line": 1,
        "matched_length": 4,
        "match_coverage": 100.0,
        "matcher": "1-hash",
        "license_expression": "apache-2.0",
        "rule_identifier": "apache-2.0_65.RULE",
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/rules/apache-2.0_65.RULE",
        "referenced_filenames": [],
        "is_license_text": false,
        "is_license_notice": false,
        "is_license_reference": false,
        "is_license_tag": true,
        "is_license_intro": false,
        "rule_length": 4,
        "rule_relevance": 100,
        "matched_text": "License: Apache-2.0",
        "licenses": [
            {
                "key": "apache-2.0",
                "name": "Apache License 2.0",
                "short_name": "Apache 2.0",
                "category": "Permissive",
                "is_exception": false,
                "is_unknown": false,
                "owner": "Apache Software Foundation",
                "homepage_url": "http://www.apache.org/licenses/",
                "text_url": "http://www.apache.org/licenses/LICENSE-2.0",
                "reference_url": "https://scancode-licensedb.aboutcode.org/apache-2.0",
                "scancode_url": "https://github.com/nexB/scancode-toolkit/tree/develop/
↪src/licensedcode/data/licenses/apache-2.0.LICENSE",
                "spdx_license_key": "Apache-2.0",
                "spdx_url": "https://spdx.org/licenses/Apache-2.0"
            }
        ]
    },
    "license_clues": [],
}

```

After:

```

{
  "license_references": [
    {
      "key": "apache-2.0",
      "short_name": "Apache 2.0",

```

(continues on next page)

(continued from previous page)

```

    "name": "Apache License 2.0",
    "category": "Permissive",
    "owner": "Apache Software Foundation",
    "homepage_url": "http://www.apache.org/licenses/",
    "notes": "Per SPDX.org, this version was released January 2004 This license is OSI\
↪ncertified\n",
    "is_builtin": true,
    "spdx_license_key": "Apache-2.0",
    "other_spdx_license_keys": [
        "LicenseRef-Apache",
        "LicenseRef-Apache-2.0"
    ],
    "osi_license_key": "Apache-2.0",
    "text_urls": [
        "http://www.apache.org/licenses/LICENSE-2.0"
    ],
    "osi_url": "http://opensource.org/licenses/apache2.0.php",
    "faq_url": "http://www.apache.org/foundation/licence-FAQ.html",
    "other_urls": [
        "http://www.opensource.org/licenses/Apache-2.0",
        "https://opensource.org/licenses/Apache-2.0",
        "https://www.apache.org/licenses/LICENSE-2.0"
    ],
    "text": "Apache License\nVersion 2.0, {Truncated text}"
  }
],
"license_rule_references": [
  {
    "license_expression": "apache-2.0",
    "rule_identifier": "apache-2.0_65.RULE",
    "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/rules/apache-2.0_65.RULE",
    "referenced_filenames": [],
    "is_license_text": false,
    "is_license_notice": false,
    "is_license_reference": false,
    "is_license_tag": true,
    "is_license_intro": false,
    "rule_length": 4,
    "rule_relevance": 100,
    "rule_text": "license: Apache-2.0"
  }
],
"files": [
  {
    "detected_license_expression": "apache-2.0",
    "detected_license_expression_spdx": "Apache-2.0",
    "license_detections": [
      {
        "license_expression": "apache-2.0",
        "detection_log": [
          "not-combined"
        ]
      }
    ]
  }
]

```

(continues on next page)

(continued from previous page)

```

    ],
    "matches": [
      {
        "score": 100.0,
        "start_line": 1,
        "end_line": 1,
        "matched_length": 4,
        "match_coverage": 100.0,
        "matcher": "1-hash",
        "license_expression": "apache-2.0",
        "rule_identifier": "apache-2.0_65.RULE",
        "matched_text": "License: Apache-2.0",
        "rule_url": "https://github.com/nexB/scancode-toolkit/tree/develop/src/
↪licensedcode/data/rules/apache-2.0_65.RULE"
      }
    ]
  },
  "license_clues": [],
}
]
}

```

LicenseDetection Data

This is referencing by LicenseDetections objects, and has one or multiple license matches. This is linked to the resource level detections through an `identifier` attribute present in both resource and codebase level detections. See the *Codebase level Unique License Detection* above for more details on this.

There could be a list of ambiguous detections as a summary to review. This is WIP, see [scancode-toolkit#3122](#).

9.1.3 Supported package manifests and package datafiles

Scancode supports a wide variety of package manifests, lockfiles and other package datafiles containing package and dependency information.

This documentation page is generated automatically from available package parsers in scancode-toolkit during documentation builds.

Table 1: Supported Package Parsers

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
AboutCode ABOUT file	*.ABOUT	about	about_file	None	https://aboutcode-toolkit.readthedocs.io/en/latest/specification.html
Alpine Linux .apk package archive	*.apk	alpine	alpine_apk_archive	None	https://wiki.alpinelinux.org/wiki/Alpine_package_format
Alpine Linux APKBUILD package script	*APKBUILD	alpine	alpine_apkbuild	None	https://wiki.alpinelinux.org/wiki/APKBUILD_Reference
Alpine Linux installed package database	*lib/apk/db/installed	alpine	alpine_installed_db	None	None
Android application package	*.apk	android	android_apk	Java	https://en.wikipedia.org/wiki/Apk_(file_format)
Android library archive	*.aar	android_lib	android_aar_library	Java	https://developer.android.com/studio/projects/android-library

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Autotools configure script	*/configure */configure.ac	autotools	autotools_configure	None	https://www.gnu.org/software/automake/
Apache Axis2 module archive	*.mar	axis2	axis2_mar	Java	https://axis.apache.org/axis2/java/core/docs/modules.html
Apache Axis2 module.xml	*/meta-inf/module.xml	axis2	axis2_module_xml	Java	https://axis.apache.org/axis2/java/core/docs/modules.html
Bazel BUILD	*/BUILD	bazel	bazel_build	None	https://bazel.build/
Bower package	*/bower.json */.bower.json	bower	bower_json	JavaScript	https://bower.io/
Buck file	*/BUCK	buck	buck_file	None	https://buck.build/
Buck metadata file	*/METADATA.bz1	buck	buck_metadata	None	https://buck.build/

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Microsoft cabinet archive	*.cab	cab	microsoft_cabinet	C#	https://docs.microsoft.com/en-us/windows/win32/msi/cabinet-files
Rust Cargo.lock dependencies lockfile	*/Cargo.lock */cargo.lock	cargo	cargo_lock	Rust	https://doc.rust-lang.org/cargo/guide/cargo-toml-vs-cargo-lock.html
Rust Cargo.toml package manifest	*/Cargo.toml */cargo.toml	cargo	cargo_toml	Rust	https://doc.rust-lang.org/cargo/reference/manifest.html
Chef cookbook metadata.json	*/metadata.json	chef	chef_cookbook_metadata	Python	https://docs.chef.io/config_rb_metadata/
Chef cookbook metadata.rb	*/metadata.rb	chef	chef_cookbook_metadata	Python	https://docs.chef.io/config_rb_metadata/

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Chrome extension	*.crx	chrome	chrome_crx	JavaScript	https://chrome.google.com/extensions
Cocoapods Podfile	*Podfile	cocoapods	cocoapods_podfile	Objective-C	https://guides.cocoapods.org/using/the-podfile.html
Cocoapods Podfile.lock	*Podfile.lock	cocoapods	cocoapods_podfile.lock	Objective-C	https://guides.cocoapods.org/using/the-podfile.html
Cocoapods .podspec	*.podspec	cocoapods	cocoapods_podspec	Objective-C	https://guides.cocoapods.org/syntax/podspec.html
Cocoapods .podspec.json	*.podspec.json	cocoapods	cocoapods_podspec.json	Objective-C	https://guides.cocoapods.org/syntax/podspec.html
PHP composer manifest	*composer.json	composer	php_composer_json	PHP	https://getcomposer.org/doc/04-schema.md

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
PHP composer lockfile	*composer.lock	composer	php_composer_lock	PHP	https://getcomposer.org/doc/01-basic-usage.md#commit-your-compo
conan external source	*/conandata.yml	conan	conan_conandata.yml	C++	https://docs.conan.io/en/2/tutorial/creating_packages/handle_sources_in_packages.html#using-the-conandata
conan recipe	*/conanfile.py	conan	conan_conanfile.py	C++	https://docs.conan.io/en/2.0/reference/conanfile.html
Conda meta.yml manifest	*/meta.yml	conda	conda_meta_yaml	None	https://docs.conda.io/
CPAN Perl dist.ini	*/dist.ini	cpan	cpan_dist_ini	Perl	https://metacpan.org/pod/Dist::Zilla::Tutorial

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
CPAN Perl Makefile.PL	*/Makefile.PL	cpan	cpan_makefile	Perl	https://www.perlmonks.org/?node_id=128077
CPAN Perl module MANIFEST	*/MANIFEST	cpan	cpan_manifest	Perl	https://metacpan.org/pod/Module::Manifest
CPAN Perl META.json	*/META.json	cpan	cpan_meta_json	Perl	https://metacpan.org/pod/Parse::CPAN::Meta
CPAN Perl META.yml	*/META.yml	cpan	cpan_meta_yaml	Perl	https://metacpan.org/pod/CPAN::Meta::YAML
CRAN package DESCRIPTION	*/DESCRIPTION	cran	cran_description	R	https://r-pkgs.org/description.html
Debian control file - extracted layout	*/control.tar.gz-extract/control	deb	debian_control	Not extracted	https://www.debian.org/doc/debian-policy/ch-controlfields.html

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Debian control file - source layout	<code>*/debian/control</code>	deb	debian_control	Non-source	https://www.debian.org/doc/debian-policy/ch-controlfields.html
Debian machine readable file in source	<code>*usr/share/doc/*/copyright</code>	deb	debian_copyright	Non-package	https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Debian machine readable file in source	<code>*/debian/copyright</code>	deb	debian_copyright	Non-source	https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Debian machine readable file standalone	<code>*/copyright</code> <code>*_copyright</code>	deb	debian_copyright	Non-standalone	https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Debian binary package archive	*.deb	deb	debian_deb	None	https://manpages.debian.org/unstable/dpkg-dev/deb.5.en.html
Debian distro-less installed database	*var/lib/dpkg/status.d/*	deb	debian_distroless_installed_db	None	https://www.debian.org/doc/debian-policy/ch-controlfields.html
Debian installed file paths list	*var/lib/dpkg/info/*.list	deb	debian_installed_files_list	None	None
Debian installed file MD5 and paths list	*var/lib/dpkg/info/*.md5sums	deb	debian_installed_md5sums	None	https://www.debian.org/doc/manuals/debian-handbook/sect.package-meta-information.html#sect.configuration-scripts
Debian installed packages database	*var/lib/dpkg/status	deb	debian_installed_status_db	None	https://www.debian.org/doc/debian-policy/ch-controlfields.html

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Debian file MD5 and paths list in .deb archive	*/control.tar.gz-extract/md5sums */control.tar.xz-extract/md5sums	deb	debian_md5sums	None	https://www.debian.org/doc/manuals/debian-handbook/sect.package-meta-information.html#sect.configuration-scripts
Debian package original source archive	*.orig.tar.xz *.orig.tar.gz	deb	debian_original_source_tarballs	None	https://manpages.debian.org/unstable/dpkg-dev/deb.5.en.html
Debian source control file	*.dsc	deb	debian_source_control	Control_dsc	https://wiki.debian.org/dsc
Debian source package meta-data archive	*.debian.tar.xz *.debian.tar.gz	deb	debian_source_metadata_tarballs	None	https://manpages.debian.org/unstable/dpkg-dev/deb.5.en.html

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
macOS disk image file	*.dmg *.sparseimage	dmg	apple_dmg	None	https://en.wikipedia.org/wiki/Apple_Disk_Image
Java EAR application.xml	*/META-INF/application.xml	ear	java_ear_application.xml	Java	https://en.wikipedia.org/wiki/EAR_(file_format)
Java EAR Enterprise application archive	*.ear	ear	java_ear_archive	Java	https://en.wikipedia.org/wiki/EAR_(file_format)
FreeBSD compact package manifest	*/+COMPACT_MANIFEST	freebsd	freebsd_compact_manifest	None	https://www.freebsd.org/cgi/man.cgi?pkg-create(8)#MANIFEST

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
RubyGems gem package archive	*.gem	gem	gem_archive	Ruby	https://web.archive.org/web/20220326093616/https://piotrmurach.com/articles/looking-inside-a-rub
RubyGems gem package extracted archive	*/metadata.gz-extract	gem	gem_archive_extracted	Ruby	https://web.archive.org/web/20220326093616/https://piotrmurach.com/articles/looking-inside-a-rub
RubyGems gemspec manifest - installed vendor/bundle/specifications layout	*/specifications/*.gemspec	gem	gem_gemspec_installed_specifications	Ruby	https://guides.rubygems.org/specification-reference
RubyGems Bundler Gemfile	*/Gemfile */*.gemfile */Gemfile-*	gem	gemfile	Ruby	https://bundler.io/man/gemfile.5.html

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
RubyGems Bundler Gemfile - extracted layout	*/data.gz-extract/Gemfile	gem	gemfile_extracted	Ruby	https://bundler.io/man/gemfile.5.html
RubyGems Bundler Gemfile.lock	*/Gemfile.lock	gem	gemfile_lock	Ruby	https://bundler.io/man/gemfile.5.html
RubyGems Bundler Gemfile.lock - extracted layout	*/data.gz-extract/Gemfile.lock	gem	gemfile_lock_extracted	Ruby	https://bundler.io/man/gemfile.5.html
RubyGems gemspec manifest	*.gemspec	gem	gemspec	Ruby	https://guides.rubygems.org/specification-reference
RubyGems gemspec manifest - extracted data layout	*/data.gz-extract/*.gemspec	gem	gemspec_extracted	Ruby	https://guides.rubygems.org/specification-reference
Go modules file	*/go.mod	golang	go_mod	Go	https://go.dev/ref/mod

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Go module checksums file	*/go.sum	golang	go_sum	Go	https://go.dev/ref/mod#go-sum-files
Go Godeps	*/Godeps.json	golang	godeps	Go	https://github.com/tools/godep
Haxe haxelib.json metadata file	*/haxelib.json	haxe	haxelib_json	Haxe	https://lib.haxe.org/documentation/creating-a-haxelib-p
InstallShield installer	*.exe	installshield	installshield_installer	Installer	https://www.revenera.com/install/products/installshield
iOS package archive	*.ipa	ios	ios_ipa	Objective-C	https://en.wikipedia.org/wiki/.ipa
ISO disk image	*.iso *.udf *.img	iso	iso_disk_image	None	https://en.wikipedia.org/wiki/ISO_9660

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Ant IVY dependency file	*/ivy.xml	ivy	ant_ivy_xml	Java	https://ant.apache.org/ivy/history/latest-milestone/ivyfile.html
JAR Archive	*.jar	jar	java_jar	None	https://en.wikipedia.org/wiki/JAR_(file_format)
Java JAR MANIFEST.MF	*/META-INF/MANIFEST.MF	jar	java_jar_manifest	Java	https://docs.oracle.com/javase/tutorial/deployment/jar/manifestindex.html
JBOSS service archive	*.sar	jboss-service	jboss_sar	Java	https://docs.jboss.org/jbossas/docs/Server_Configuration_Guide/4/html/ch02s01.html

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
JBOSS service.xml	*/meta-inf/jboss-service.xml	jboss-service	jboss_service.xml	Java	https://docs.jboss.org/jbossas/docs/Server_Configuration_Guide/4/html/ch02s01.html
Linux OS release metadata file	*etc/os-release *usr/lib/os-release	linux-distro	etc_os_release	None	https://www.freedesktop.org/software/systemd/man/os-release.html
Gradle build script	*/build.gradle */build.gradle.kts	maven	build_gradle	None	None
Apache Maven pom	*.pom *pom.xml	maven	maven_pom	Java	https://maven.apache.org/pom.html
Apache Maven pom properties file	*/pom.properties	maven	maven_pom_properties	Java	https://maven.apache.org/pom.html

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Meteor package.js	*/package.js	meteor	meteor_package	JavaScript	https://docs.meteor.com/api/packagejs.html
Mozilla XPI extension	*.xpi	mozilla	mozilla_xpi	JavaScript	https://en.wikipedia.org/wiki/XPIInstall
Microsoft MSI installer	*.msi	msi	msi_installer	None	https://docs.microsoft.com/en-us/windows/win32/msi/windows-installer-p
npm package.json	*/package.json	npm	npm_package_json	JavaScript	https://docs.npmjs.com/cli/v8/configuring-npm/package-json
npm package-lock.json lockfile	*/package-lock.json */package-lock.json	npm	npm_package_lock_json	JavaScript	https://docs.npmjs.com/cli/v8/configuring-npm/package-lock-json

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
npm shrinkwrap.json lockfile	*/npm-shrinkwrap.json	npm	npm_shrinkwrap_json	JavaScript	https://docs.npmjs.com/cli/v8/configuring-npm/npm-shrinkwrap-json
yarn.lock lockfile v1 format	*/yarn.lock	npm	yarn_lock_v1	JavaScript	https://classic.yarnpkg.com/lang/en/docs/yarn-lock/
yarn.lock lockfile v2 format	*/yarn.lock	npm	yarn_lock_v2	JavaScript	https://classic.yarnpkg.com/lang/en/docs/yarn-lock/
NSIS installer	*.exe	nsis	nsis_installer	None	https://nsis.sourceforge.io/Main_Page
NuGet nupkg package archive	*.nupkg	nuget	nuget_nupkg	None	https://en.wikipedia.org/wiki/Open_Packaging_Conventions

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
NuGet nuspec package manifest	*.nuspec	nuget	nuget_nuspec	None	https://docs.microsoft.com/en-us/nuget/reference/nuspec
Ocaml Opam file	*opam	opam	opam_file	Ocaml	https://opam.ocaml.org/doc/Manual.html#Common-file-formats
Java OSGi MANIFEST.MF	None	osgi	java_osgi_manifest	Java	https://docs.oracle.com/javase/tutorial/deployment/jar/manifestindex.html
Dart pubspec lockfile	*pubspec.lock	pubspec	pubspec_lock	dart	https://web.archive.org/web/20220330081004/https://gpalma.pt/blog/what-is-the-pubspec-lockfile

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Dart pubspec manifest	*pubspec.yaml	pubspec	pubspec_yaml	dart	https://dart.dev/tools/pub/pubspec
Conda yaml manifest	*conda.yaml *conda.yml	pypi	conda_yaml	Python	https://docs.conda.io/
pip requirements file	*requirements.txt *requirements*.txt pip *requirements*.in *requires.txt *requirements/*.txt *requirements/*.pip *requirements/*.in *reqs.txt	pypi	pip_requirements	Python	https://pip.pypa.io/en/latest/reference/requirements-file-format/
Pipfile	*Pipfile	pypi	pipfile	Python	https://github.com/pypa/pipfile
Pipfile.lock	*Pipfile.lock	pypi	pipfile_lock	Python	https://github.com/pypa/pipfile
PyPI editable local installation PKG-INFO	*.egg-info/PKG-INFO	pypi	pypi_editable_egg_pkginfo	Python	https://peps.python.org/pep-0376/

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
PyPI egg	*.egg	pypi	pypi_egg	Python	https://web.archive.org/web/20210604075235/http://peak.telecommunity.com/DevCenter/PythonEggs
PyPI extracted egg PKG-INFO	*/EGG-INFO/PKG-INFO	pypi	pypi_egg_pkginfo	Python	https://peps.python.org/pep-0376/
Python pyproject.toml	*pyproject.toml	pypi	pypi_pyproject_toml	Python	https://peps.python.org/pep-0621/
PyPI extracted sdist PKG-INFO	*/PKG-INFO	pypi	pypi_sdist_pkginfo	Python	https://peps.python.org/pep-0314/
Python setup.cfg	*setup.cfg	pypi	pypi_setup_cfg	Python	https://peps.python.org/pep-0390/

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Python setup.py	*setup.py	pypi	pypi_setup_py	Python	https://docs.python.org/3.11/distutils/setupscript.html
PyPI wheel	*.whl	pypi	pypi_wheel	Python	https://peps.python.org/pep-0427/
PyPI installed wheel META-DATA	*.dist-info/METADATA	pypi	pypi_wheel_metadata	Python	https://packaging.python.org/en/latest/specifications/core-metadata/
None	*/README. android */README. chromium */README. facebook */README. google */README. thirdparty	readme	readme	None	None
RPM package archive	*.rpm *.src. rpm *.srpm *. mvl *.vip	rpm	rpm_archive	None	https://en.wikipedia.org/wiki/RPM_Package_Manager

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
RPM installed package BDB database	*var/lib/rpm/Packages	rpm	rpm_installed_database_bdb	None	https://man7.org/linux/man-pages/man8/rpmdb.8.html
RPM installed package NDB database	*usr/lib/sysimage/rpm/Packages.db	rpm	rpm_installed_database_ndb	None	https://fedoraproject.org/wiki/Changes/NewRpmDBFormat
RPM installed package SQLite database	*rpm/rpmdb.sqlite	rpm	rpm_installed_database_sqlite	None	https://fedoraproject.org/wiki/Changes/Sqlite_Rpmdb
RPM specfile	*.spec	rpm	rpm_specfile	None	https://en.wikipedia.org/wiki/RPM_Package_Manager
shell archive	*.shar	shar	shar_shell_archive	None	https://en.wikipedia.org/wiki/Shar

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Squashfs disk image	None	squashfs	squashfs_disk_image	None	https://en.wikipedia.org/wiki/SquashFS
Java Web Application Archive	*.war	war	java_war_archive	Java	https://en.wikipedia.org/wiki/WAR_(file_format)
Java WAR web/xml	*/WEB-INF/web.xml	war	java_war_web_xml	Java	https://en.wikipedia.org/wiki/WAR_(file_format)
Windows Registry Installed Program - Docker SOFTWARE	*/Files/Windows/System32/config/SOFTWARE	windows-program	win_reg_installed_programs_docker_file_software	None	https://en.wikipedia.org/wiki/Windows_Registry
Windows Registry Installed Program - Docker Software Delta	*/Hives/Software_Delta	windows-program	win_reg_installed_programs_docker_software_delta	None	https://en.wikipedia.org/wiki/Windows_Registry

continues on next page

Table 1 – continued from previous page

Description	Path Patterns	Package type	Datasource ID	Primary Language	Documentation URL
Windows Registry Installed Program - Docker UtilityVM SOFTWARE	*/UtilityVM/Files/Windows/System32/config/SOFTWARE	windows-program	win_reg_installed_programs_docker_utility_so	None	https://en.wikipedia.org/wiki/Windows_Registry
Microsoft Update Manifest .mum file	*.mum	windows-update	microsoft_update_manifest_mum	None	None
Windows Portable Executable metadata	*.exe *.dll *.mui *.mun *.com *.winmd *.sys *.tlb *.exe_* *.dll_* *.mui_* *.mun_* *.com_* *.winmd_* *.sys_* *.tlb_* *.ocx	winexe	windows_executable	None	https://en.wikipedia.org/wiki/Portable_Executable

INDICES AND TABLES

- [genindex](#)
- [modindex](#)
- [search](#)

10.1 Something Missing?

If something is missing in the documentation or if you found some part confusing, please file an [issue](#) with your suggestions for improvement. Use the “Documentation Improvement” template. Your help makes ScanCode docs better, we love hearing from you!