ClearlyLicensed Metrics

This document is the proposal to define license clarity metrics for software components in the context of the ClearlyDefined project.

Metrics Usage

License clarity metrics are useful for several purposes such as:

- Evaluating the state of license clarity of a publicly available software project.
- Evaluating the overall state of license clarity in aggregations such as software repositories (e.g. MavenCentral or GitHub) and major software platforms (e.g. JBoss or Qt).
- Determining the level of effort required for the initial curation of license data for a ClearlyDefined component.
- Evaluating the reliability of the curated and or detected component metadata.
- Determining the level of effort required for curation of a new version of a component that has been previously curated.

Metrics Guiding Principles

Metrics must be closely aligned with what a ClearlyLicensed open source software component represents and offer a meaningful indication of license clarity, through a compound score. Metrics should also align with the concept of software project Facets, which determine the relative importance of the various sections of a software project:

- core - The files that go into making the release of the component (the deployed code).
- data - The files included in any data distribution of the component.
- dev - Files primarily used at development time (e.g. tools, build utilities) and not distributed with the component.
- docs - Documentation files may be distributed with the main component or separately.
- examples - Example files may be distributed with the main component or separately.
- tests - Test files (test code, data and other artifacts) may be distributed with the main component or separately.

The focus for now is only on the **core facet** when facets are available and in some cases only on source code in the core facet (as opposed to media files, scripts, etc.) The metrics could be broken down by facets at a later stage if needed or consider the relative importance of some facets, such as core that is more important than other facets such as dev, docs, examples, and tests but this is not the initial focus.

Metrics Score and Scaling factors

The score is:

1. bounded between 0 and 100.
2. composed of multiple scoring elements that are either binary or progressive.
3. constructed using the weight (total score points) of each element.
4. computed automatically from available data that are either collected automatically or updated from a curation.

There are two types of scoring elements:

- binary elements (e.g. we have a top level license or not, and licenses are standard) that grant a fixed number of points. A binary score element wins all the assigned weight point if present.
- progressive elements computed with a formula that grants a proportion of a score element weight.

Metrics Elements

License clarity metrics elements include the following:

- Clearly defined top-level, declared license
- Presence of file-level license and copyright
- License consistency between top level and file-level licenses
- Presence and coverage of full license texts for referenced licenses (vs. mentions).
- Use of standard license and copyright information texts (e.g. use of SPDX license identifiers/expressions, use of detectable and standard license notices and copyright statements, etc.).
- Ease of discovery of license and copyright information by compliance tools.

Clearly defined top-level, declared license

This is based on the presence and clarity of license information in key, top-level project files.

Open source packages often use text files in a codebase to document the applicable license(s) for a component. Some common names for these key files are LICENSE, COPYING, NOTICE, etc. In other cases a project will use a structured package manifest file(s) that contain coded or conventional licensing information (such as a Nuget .nuspec URL, a Maven POM license tags, a Python setup.py license classification, or an npm package.json license tags). Or there is a README file that provides semi-structured licensing information. These files are commonly stored in the root or top-level directory of a package archive or source code repository.

Collectively the files with either common licensing-related names, readme files and package manifests are referred to as "**key files**" when stored at the top-level or root of a package. These key files are easily discoverable and are typically the first place where one would look for license information.

The presence of license information in these key files in a form that is easy to detect mechanically and easy to read by humans contributes to a higher metrics score. For instance, a software project without a license notice present at the top level, but rather in a subdirectory (e.g. “docs”), is not as clearly licensed as one with a prominent notice found in a file with a common name and located at the top level of the package code tree.

File-level license and copyright

This is based on the presence, clarity and coverage of per-file license and copyright information to answer this question:

Do files that can contain licensing and copyright information reliably carry such information in the sense that these are reliable detected by tools?

Beyond top-level documentation of licensing in key files, individual files can (and should when this is possible) contain license and copyright information. The presence and proportion of files that have such per-file information is another metrics score element.

Note that earlier versions were focusing only on source code files rather than any file type.

License consistency

This is to capture the coherence and alignment of top-level, key file information with file-level licensing information.

License data is most clear when the top-level or structured license information is aligned and in agreement with the aggregated per-file details. Metrics scoring is higher for a software package with accurate license alignment, and the scoring should consider the relative importance of some facets, such as core code and data.

SPDX standard licenses

This is to capture the use of standard licenses: a license is considered as standard if it is present in the SPDX license list. The use of fully detectable, standard license notices and texts and copyright statements, as well as the use of SPDX license identifiers and expressions, contribute to a higher metrics score. The use of non-detectable, non-standard licenses makes the licensing less clear.

A custom, non-SPDX license might be a well-crafted piece of legal writing, but it would have to be reviewed by the user (which, in principle, has may already has reviewed and set policy for the SPDX Standard Licenses), then it means that the legal terms and conditions are not so clearly defined.

A standard copyright statement is one that can be detected by tools. A typical copyright statement contains the following elements: Copyright [(c)] [year or range of years] [name or names]. The “(c)” or © after the word “Copyright” are optional. The year range is optional.

License texts

This is to capture the presence of full license texts for any referenced licenses (vs. notice or mere mentions) found in a package.

Most open source licenses require to make available their full license text. Some packages may contain only license names, a license tag, a license identifier (such as an SPDX license identifier) or a license notice text but they may not contain the complete corresponding license text. Metrics scoring is higher for a software package that contains the full license texts because the absence of such text would require users to fetch these texts separately. Also there could be some ambiguity if the full text is not provided.

ClearlyLicensed Scoring Formula

This table describes the formula to compute a value for each of the scoring elements and the weight of each element.
The total score is the sum of the weight contributed to the score by each element.

Name	Weight	Formula	Notes
Clearly defined top-level, declared license	30	binary	A project has specific key file(s) at the top level of its code hierarchy such as LICENSE, NOTICE or similar (and/or a package manifest) containing structured license information such as an SPDX license expression or SPDX license identifier, and the file(s) contain “clearly defined” declared license information (a license declaration such as a license expression and/or a series of license statements or notices). This is a **binary** score element.
File-level license and copyright	25	LICCOP / TOT	Do files that **can contain** licensing and copyright information **reliably** carry such information? This is based on a percentage of files in the core facet of the project that have both: ● A license statement such as a text, notice or an SPDX-License-Identifier and, ● A copyright statement in standard format that can be detected by tools. Here "reliably" means that these are reliably detected by tool(s) with a high level of confidence This is a **progressive** element that is computed based on: ● LICCOP: the number of files with a license notice and copyright statement ● TOT: the total number of files
License Consistency	15	binary	Are all file-level licenses consistent and documented in top-level key files? This is based on files in the core facet. This is a **binary** score element awarded if all the licenses found anywhere in the core facet are also found in the top-level key files.
SPDX Standard Licenses	15	binary	Are all the licenses standard SPDX-listed licenses? This is based on files in the core facet. This is a **binary** score element awarded if all the licenses found anywhere in the core facet use an SPDX-listed license, identified by a standard license notice or the SPDX identifier.
License Texts	15	binary	Is a copy of the complete license text available in the project for every referenced license? This is based on files in the core facet. This is a **binary** score element awarded if the package contains the full license text for all the licenses found anywhere in the core facet.

Base Score vs. Curated Score

1. **Base score** is computed on the available data collected from a package as it exists "upstream", as redistributed by the project.
2. **Curated score** is computed from the combination of the initial base score and the curated data overlaid.

A curation can have several impacts on the score:

- In the most typical case, licensing and copyright may have been reviewed and this review has been contributed as part of the curation.
- Updated licensing metadata, notices and copyright may have been created for upstream contribution as part of the curation.
- Facets may have been contributed and the computed score is now different for the core facet.

We will compute a new Curated score based on curated data and in particular:

- Take facets in consideration when facets were contributed as part of the curation.
- Treat a contributed curated top-level license as the new "clearly defined top-level, declared license" scoring element.

Previous versions ClearlyLicensed Scoring Method proposals

Metrics Calculation Proposal #4.

This is a proposal evolved from #3 and integrates the community feedback. There are two types of scoring elements: **binary elements** (e.g. we have a top level license or not, and licenses are standard one) that grant a fixed number of points and **proportional elements** computed with a formula that grants of a proportion of a score element weight. A binary score element wins all the assigned weight if present.

NOTE: The focus for now is only on the core facet when facets are available and in some cases only on source code in the core facet (as opposed to media, etc.) This could be broken down by facets at a later stage if needed but this is not the initial focus.

Name	**Weight**	**Formula**	**Notes**
Clearly defined top-level, declared license	25	binary	A project has specific key file(s) at the top level of its code hierarchy such as LICENSE, NOTICE or similar (and/or a package manifest) that contains structured license information such as an SPDX license expression or SPDx license identifier, and the file(s) contain “clearly defined” declared license information: ● A license declaration such as a license expression and/or a series of license statements or notices, This is a **binary** score element
~~Project Key File Copyright Notices~~	~~15~~	~~binary~~	~~A project has specific key file(s) at the top level of its code hierarchy such as LICENSE, NOTICE or similar (and/or a package manifest that contains structured license information), and the file(s) contain:~~ ~~● A copyright statement usable for attribution~~ ~~This is a **binary** score element.~~
File-level license and copyright	25	SFLICCOP / SFTOT	Do files that can contain licensing and copyright information reliably carry such information? Based on a percentage of files in the core facet of the project that have both: ● A license statement such as a text, notice or an SPDX-License-Identifier and, ● A copyright statement. Here "reliably" would mean that the tool(s) has a high confidence in the detection This is a **progressive** element that is computed based on: ● LICCOP: the number of files with a license notice and copyright statement ● TOT: the total number of files
License Consistency	15	binary	Are file-level licenses consistent and documented with top-level, key files licenses? Based on files in the core facet. This is a **binary** score element awarded if all the licenses found anywhere in the core facet are also found in the top-level key files

Standard licenses	10	binary	Are the licenses common and standard (e.g. SPDX-listed licenses)? Based on files in the core facet. This is a **binary** score element awarded if all the licenses found anywhere in the core facet are use an SPDX-listed license
~~Non ambiguous licenses~~	~~5~~	~~HIGHDET / DETLIC~~	~~Are the licenses detected unambiguously? Based on files in the core facet.~~ ~~This is a **progressive** element that is computed based on:~~ ~~● HIGHDET: the number of detected licenses that are detected with high confidence (e.g. over 90% score)~~ ~~● DETLIC: the total number of license detections~~
License Texts	10	LICTEXT/ LICTOT	Is a copy of the complete license texts available in the project for every referenced licenses? This is a **progressive** element that is computed based on: ● LICTEXT: the number of unique detected licenses with a full license text present ● LICTOT: the total number of unique detected licenses

Discussion on score proposal #4

We need eventually two scores:

3. a score computed on the available data collected from a package as it exists "upstream", as redistributed by the project
4. a score computed from the combination of the initial inherent score in 1. and the curated data overlaid

The reason for this is that curation can have several impacts on the score:

- licensing and copyright may have been reviewed and this review has been contributed as part of the curation
- updated licensing metadata, notices and copyright may have been created for upstream contribution as part of the curation
- facets may have been contributed and the computed score is now different for the core facet

In these cases what should be done? Should we compute a new "curated" score? This can make sense when facets are defined as part of the curation. But this makes less sense when the curation may provide an effective top-level license and copyright and therefore would not contribute details that could be reused to compute an updated score.

Because of this, I propose the following:

- if facets have been contributed as part of the curation, we can re-compute a score that includes this curation.
- If a curation has been done, and it effectively confirm or update or amend the licensing and origin, a separate score should be computed that could consist of

1. the actual curated score (e.g. using the new or updated facets)
2. the consistency of the actual detected licenses with the curated/reviewed/confirmed licenses and copyright information?

Should this be broken down between top-level and file level?

Metrics Calculation Proposal #3.

This is a proposal evolved from #2 with refined definitions and point assignments. There are two types of scoring elements: **binary elements** (e.g. we have a top level license or not, and licenses are standard one) that grant a fixed number of points and **progressive elements** typically based on a number of files or percentage that grant a number of points on a sliding scale.

Metric	**Element Name**	**Element Weight**	**Notes**
1	Project Key files Notice	40	A project has specific key file(s) at the top level of its code hierarchy such as LICENSE, NOTICE or similar (and/or a package manifest that contains structured license information), and the file(s) contain: ● A license declaration (25 points) - a license expression and/or a series of license statements or notices, ● A copyright statement (15 points)
2	File-level, Source File Headers Notice	25	Do source code files contain a license and copyright notice? Based on a percentage of the source files in the core facet of the project that have both: ● A license statement as a license text, a license notice or a license “tag” (such as as SPDX-License-Identifier or similar convention) and, ● A copyright statement. ● 100% = 10 points …. 20% = 2 points …. 10% = 1 points
3	Common and Detectable	15	Are the licenses and copyright statements common and detectable? Focus is on the core facet. ● Standard licenses (e.g. SPDX-listed licenses) = 5 points. ● Licenses are detected unambiguously = 5 points. ● Well formed/detectable copyright statements = 5 points.
4	License Consistency	10	Based on a percentage of the source files in the core facet of the project that have a license that is found/documented in the Project Key files ● 100% = 10 points ….20% = 2 points 10% = 1 points
5	License Texts	10	A copy of the complete license text is available in the project for every referenced licenses. Based on the percentage of licenses that have such a license text (as opposed to only a simple notice or reference) ● 100% = 10 points ….20% = 2 points 10% = 1 points

Metrics Calculation Proposal #2.

This is a proposal evolved from #1 with refined definitions and point assignments.

Metric	Element Name	Element Weight	Notes
1	Project Key files Notice	40	A project has specific key file(s) at the top level of its code hierarchy such as LICENSE, NOTICE or similar (and/or a package manifest that contains structured license information), and the file(s) contain: ● A license declaration (25 points) - a license expression and/or a series of license statements or notices, ● A copyright statement (15 points)
2	File-level, Source File Headers Notice	25	Source code files contain a license and copyright notice. Based on a percentage of the source files in the core facet of the project that have both: ● A license statement as a license text, a license notice or a license “tag” (such as as SPDX-License-Identifier or similar convention) and, ● A copyright statement. ● 80-100% = 25 points ● 50-79% = 20 points ● 30-49% = 15 points ● 20-39% = 10 points ● 0-10% = 0 points
3	Common and Detectable	15	Are the licenses and copyright statements common and detectable? Focus is on the core facet. ● Standard licenses (e.g. SPDX-listed licenses) = 5 points. ● Licenses are detected unambiguously = 5 points. ● Well formed/detectable copyright statements = 5 points.
4	Consistency	10	Over 2/3 of the source files in the core facet have a copyright and license that are mentioned in the Project Key files = 10 points.
5	License Texts	10	A copy of the complete license text is available in the project for every referenced licenses. ● 100% = 10 points ● 80-99% = 8 points ● 50-79% = 4 points ● 0-49% = 0 points

Metrics Calculation Proposal #1.

This is a “straw-man” proposal intended to get the discussion going in a concrete and pragmatic manner. Each Metric Element contributes a numeric value and a “metric weight”. The combined total for a perfect score would be 100. Since the calculation is inherently an automated process, the use of one or more license compliance discovery tools is assumed. The logic to evaluate each element is subject to further detailed clarification.

Metric	**Element Name**	**Element Weight**	**Notes**
1	Project License Notice	30	A project has a specific file at the top level of the code hierarchy called NOTICE or similar, and that file contains: ● A copyright statement (15 points), and ● A license declaration (15 points) - a license expression and/or a series of license statements.
2	Source File Headers	40	Based on a percentage of the source files in the core and data facets of the project that have both: ● A copyright statement, AND ● A license statement in SPDX format OR a standard license notice. 80-100% = 40 points 50-79% = 30 points 30-49% = 20 points 20-39% = 10 points 0-10% = 0 points
3	Consistency	10	Fewer than half of the source files in the project have a copyright and license that are not mentioned in the Project License Notice = 10 points.
4	License Texts	10	A copy of the complete license text is available in the project for each license mentioned in a notice. 100% = 10 points 20-99% = 5 points 0-19% = 0 points
5	Standards Compliance	10	Focus is on the core and data facets. Well formed copyright statements = 5 points.

			Standard license notices = 5 points.

Additional Background Material

ClearlyDefined Project Context

The following section from https://docs.clearlydefined.io/clearly#licensed defines “ClearlyLIcensed”

> Being ClearlyLicensed is a main focus of ClearlyDefined. Many projects have a license. Not all are clearly identified or identified uniformly. Often times there are more licenses at play in the code than are stated in the project. Even when the license is known, the information required to comply with the license (e.g., source location and the parties to the attribution requirement) are not always clear.
>
> This ambiguity results in component users who either end up not following the project requirements or who spend enormous effort trying to “get it right”. ClearlyDefined addresses both scenarios by clarifying the license information around a component.
>
> License information is broken down by facet (*see below*). So if you are consuming just the core of a component (the common case), then you need only be concerned with the licensing data in the core facet.
>
> Each facet includes the following licensing-related information:
>
> - declared - The SPDX license expression that was declared to cover the component in general. For example, the value of the license property in an NPM or the license declared in the LICENSE file of a Git repo.
> - files - Total number of files related to the facet.
> - discovered - License information found in facet files. In particular, the following properties:
>   - expressions - The set of unique SPDX license expressions found across all files in the facet.
>   - unknown - The number of files in the facet that have no discernable license information.
> - attribution - Information related to how attribution should be done for the files in the facet. Typically this equates to a list of copyright holders, but projects vary as to how they want attribution to be done.
>   - parties - The set of unique entities that are to receive attribution (e.g., copyright holders)
>   - unknown - The number of files in the facet that have no discernable attribution information.

Software Project Facets

> Many projects contain an assortment of code, only some of which is actually part of the shipped component. For example, tests and samples generally are not part of the release of a project. As such, consumers generally need not be concerned with the licenses of that content. For example, the packaged TypeScript component includes about 200 files but the full source of the component has >36,000 files – lots of tests.

To accommodate this level of detail and variation, ClearlyDefined's *described* domain includes the notion of *facets*. 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 filenames.

While there are many different ways to slice up one's source, for simplicity and generality, ClearlyDefined identifies a fixed set of canonical facets as discussed below. The facet declarations are not meant to be hard and fast rules nor are they intended for anything other than the uses in ClearlyDefined – differentiating files that go in the delivery/consumption forms of the component. In short, think of this as specifying what goes into the consumed outputs. You need not fill in each facet. Key element is that all files that go into the release are part of the relevant facet.

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.

Scoring Discussion

Consider two extreme cases:

1. a component provides no licensing or copyright information. Nothing at all. This would likely be the lowest score
2. a component provides explicit licensing or copyright information both in a human and machine form, both at the top level and in every file. The licensing and copyright is detected without ambiguities and all top-level and per-file documentation is consistent. This would likely be the highest score of licensing clarity.

Between these two extreme and likely rare cases we have a continuum of increasingly clear licensing information available with some important thresholds where some usable and consistent set of information is provided.

The selection of which documentation elements contribute to this clarity increase is the essence of the score design.