Open Source
Tooling Group

# Workshop Capabilities and Functional Blocks

The OpenChain reference tooling work group 2021

## Agenda

| Top | Name | Actors |
|---|---|---|
| 1. | Recap of work done so far | All |
| 2. | Align Overview and description with capability model | All |

# Align Overview and description with capability model

| Capability model | Parts of the toolchain |
|---|---|
| Component<br>any software element that is treated as a uniquely identifiable entity within the compliance process (e.g. proprietary, commercial, or FOSS component) | 3rd party software elements<br>Any software element which is either licensed under a different license or from a different origin |
| Product<br>a piece of software that is analysed by the compliance process for purposes of distribution | "Product"<br>Any software that is analyzed by the compliance process |
| Business Context<br>the information defining the situation and circumstances of a foreseen distribution of a *product*, including the distribution license, distribution channel, any export related information, the foreseen audience (e.g. customers, partners, Open Source, regional constraints), etc. | Business Context<br>the information defining the situation and circumstances of a foreseen product business use |

## Agenda

| Top | Name | Actors |
|---|---|---|
| 1. | Recap of work done so far | All |
| 2. | Align Overview and description with capability model | All |

# Toolchain elements and capabilities

## Artifact Repository

A system or service providing (binary) software artifacts and metadata stored in a defined directory structure which is used to retrieve artifacts during a build process. This is used as a cache for the external Artifact Repository to ensure the availability of all components used within the company, it is also the storage for the build software artifacts of the company, used in the Continuous Integration Infrastructure to store the build results. Example: [Archiva]

## Package Crawler/Finder

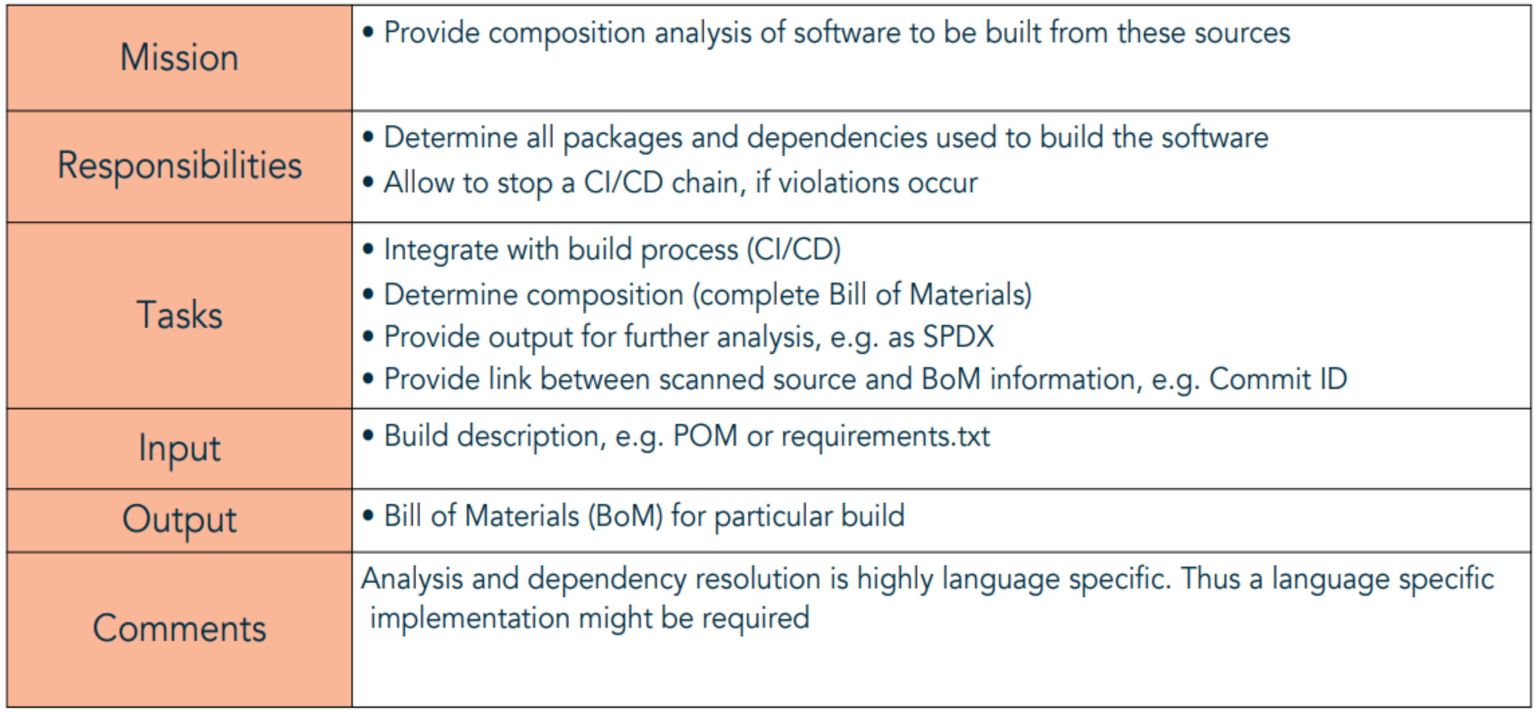
| | |
|---|---|
| Mission | • Research information on (new) components such as locate the repository, current and former versions, project homepage and viability information |
| Responsibilities | • Collect and provide accurate information about the component<br>• Alert, if component can't be matched/found |
| Tasks | • Scan package managers for new packages or versions of packages<br>• Collect package data<br>• Transfer data into package repository |
| Input | • Component descriptor or component name |
| Output | • Component Information, such as: source repository url, version history, branches, commit count, stars, last commit date, etc. |
| Comments | |

# Toolchain elements and capabilities

## Dependency Resolver

A tool that determines the dependencies of software projects. It is technology and package manager specific how dependencies are expressed - thus there need to be dedicated functionality of the different package managers in use. The dependency resolver ensures the all dependencies are resolved recursively. It ensures that the output is a technology and package manager neutral complete list of dependencies.

## Dependency Analyzer (Source)



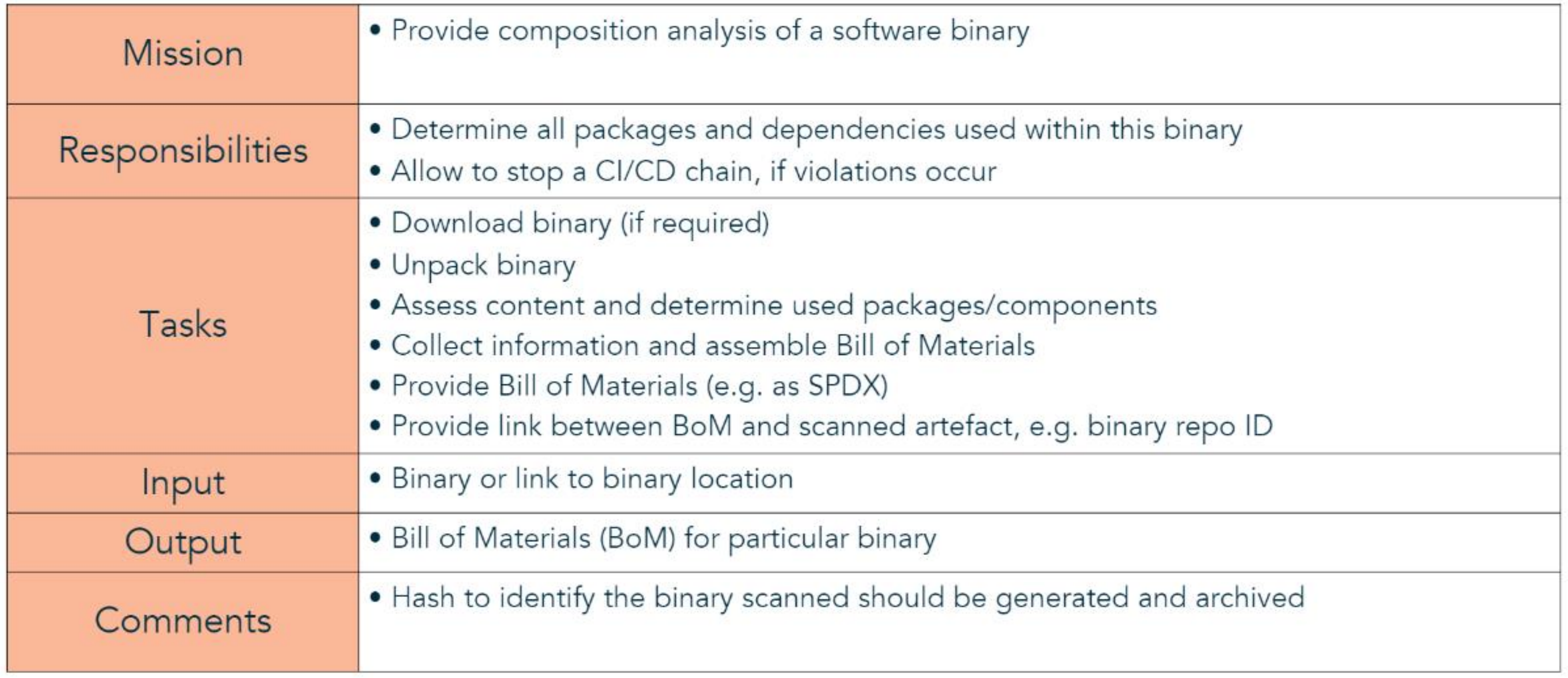
| | |
|---|---|
| Mission | • Provide composition analysis of software to be built from these sources |
| Responsibilities | • Determine all packages and dependencies used to build the software<br>• Allow to stop a CI/CD chain, if violations occur |
| Tasks | • Integrate with build process (CI/CD)<br>• Determine composition (complete Bill of Materials)<br>• Provide output for further analysis, e.g. as SPDX<br>• Provide link between scanned source and BoM information, e.g. Commit ID |
| Input | • Build description, e.g. POM or requirements.txt |
| Output | • Bill of Materials (BoM) for particular build |
| Comments | Analysis and dependency resolution is highly language specific. Thus a language specific implementation might be required |

## Toolchain elements and capabilities

Binary Analyzer

A system or service that analyses binary files, like executables or libraries. Its main purpose is to identify the contents of binary files and making this information available for further analysis, such as license compliance, security research or composition analysis..

### Dependency Analyzer (Binary)



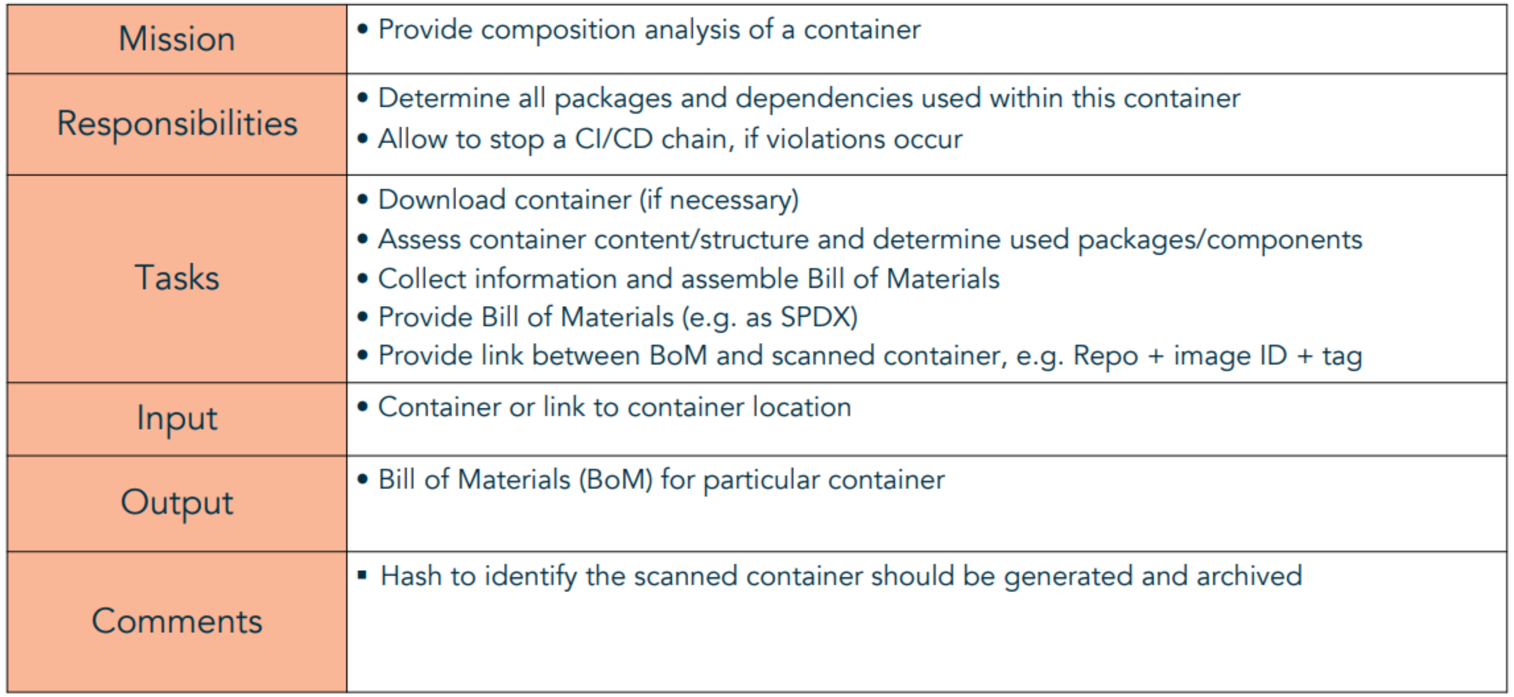
| | |
|---|---|
| Mission | • Provide composition analysis of a software binary |
| Responsibilities | • Determine all packages and dependencies used within this binary<br>• Allow to stop a CI/CD chain, if violations occur |
| Tasks | • Download binary (if required)<br>• Unpack binary<br>• Assess content and determine used packages/components<br>• Collect information and assemble Bill of Materials<br>• Provide Bill of Materials (e.g. as SPDX)<br>• Provide link between BoM and scanned artefact, e.g. binary repo ID |
| Input | • Binary or link to binary location |
| Output | • Bill of Materials (BoM) for particular binary |
| Comments | • Hash to identify the binary scanned should be generated and archived |

# Toolchain elements and capabilities

Container Content Resolver

A system or service that analyses a given software container to determine the packages and their metadata installed in the container image. Thus the container content resolver provides the bill of material of a container image.

## Dependency Analyzer (Container)



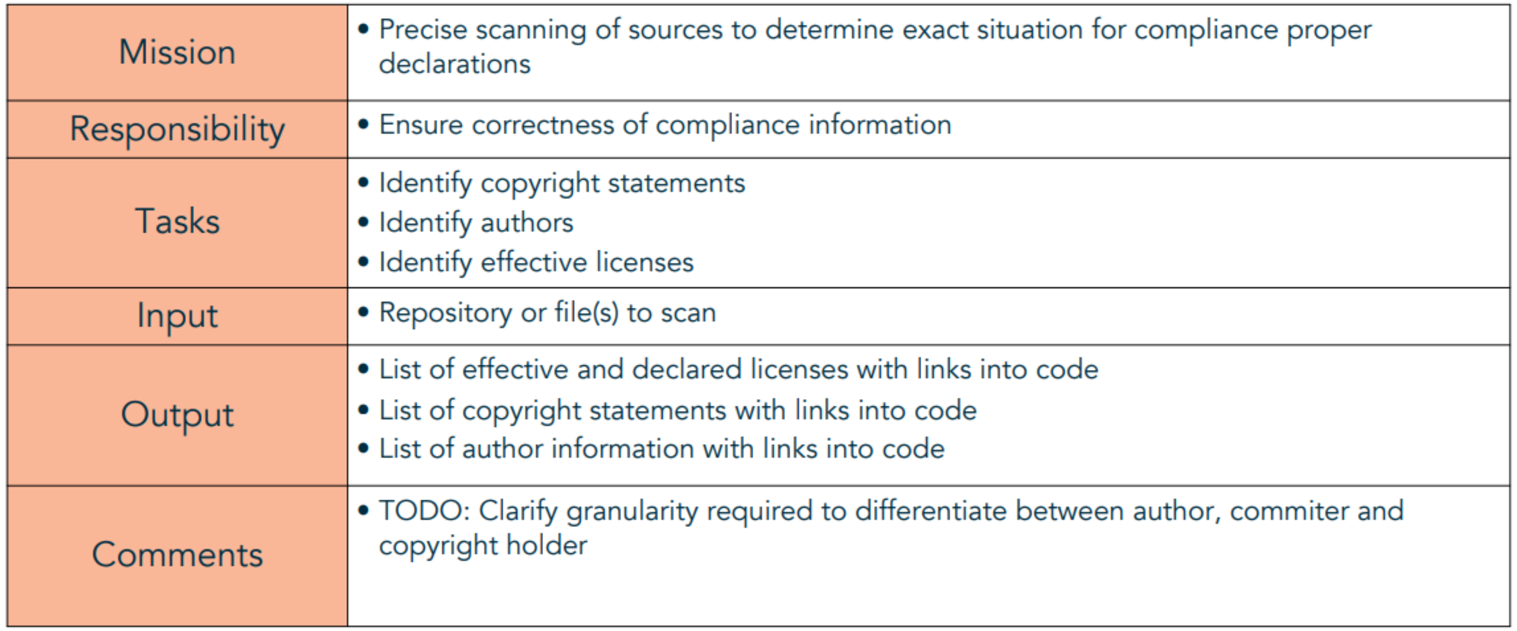
| | |
|---|---|
| Mission | • Provide composition analysis of a container |
| Responsibilities | • Determine all packages and dependencies used within this container<br>• Allow to stop a CI/CD chain, if violations occur |
| Tasks | • Download container (if necessary)<br>• Assess container content/structure and determine used packages/components<br>• Collect information and assemble Bill of Materials<br>• Provide Bill of Materials (e.g. as SPDX)<br>• Provide link between BoM and scanned container, e.g. Repo + image ID + tag |
| Input | • Container or link to container location |
| Output | • Bill of Materials (BoM) for particular container |
| Comments | ▪ Hash to identify the scanned container should be generated and archived |

# Toolchain elements and capabilities

## License & Copyright Scanner

A tool that analyses source code to identify license and copyright information. usually these tools need to provide a UI in order to be able to review the tool findings and if necessary to correct the tool findings.

## License, Copyright & Authors Scanner



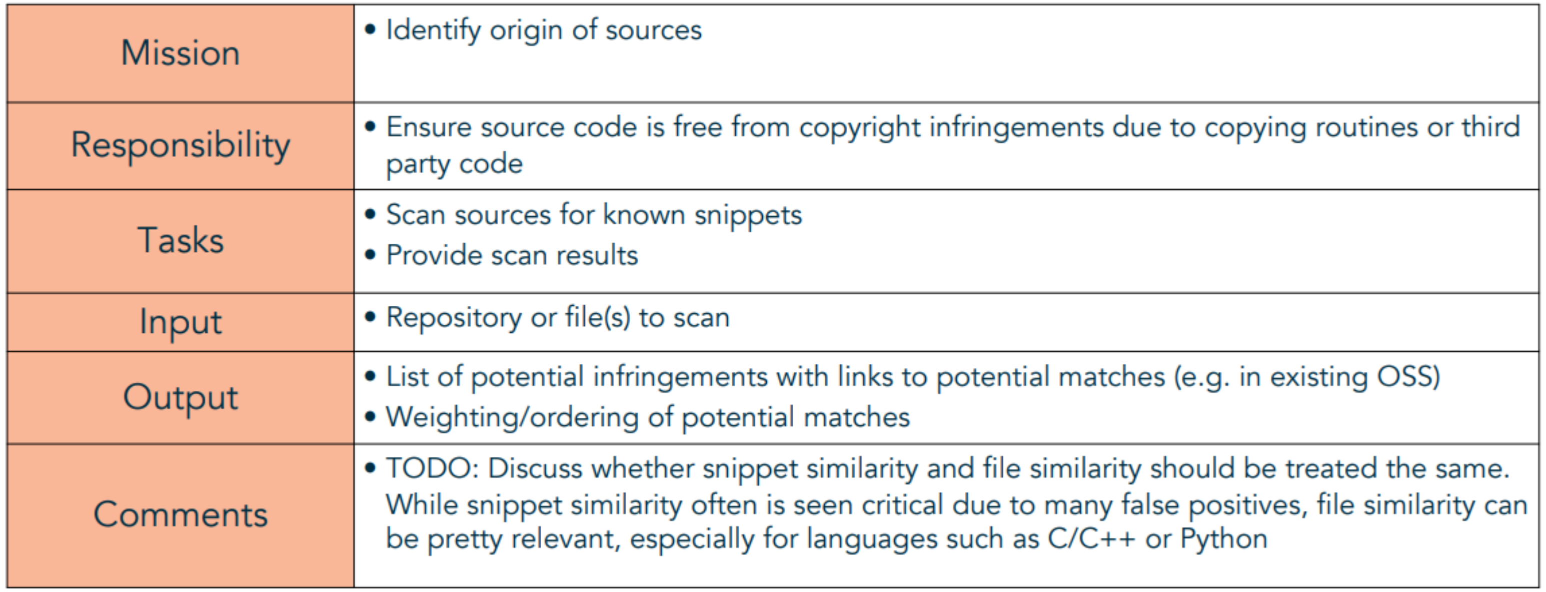
| | |
|---|---|
| Mission | • Precise scanning of sources to determine exact situation for compliance proper declarations |
| Responsibility | • Ensure correctness of compliance information |
| Tasks | • Identify copyright statements<br>• Identify authors<br>• Identify effective licenses |
| Input | • Repository or file(s) to scan |
| Output | • List of effective and declared licenses with links into code<br>• List of copyright statements with links into code<br>• List of author information with links into code |
| Comments | • TODO: Clarify granularity required to differentiate between author, commiter and copyright holder |

# Toolchain elements and capabilities

Forensic Code Analysis Service

Dedicated tools and services which scan the source code concerning integrated 3rd party code snippets and aim to provide license information about the origin and applicable license of the source code snippets. These kind of tools or services are often called **scanner for plagiarism**.

## Snippet Scanner



| | |
|---|---|
| Mission | • Identify origin of sources |
| Responsibility | • Ensure source code is free from copyright infringements due to copying routines or third party code |
| Tasks | • Scan sources for known snippets<br>• Provide scan results |
| Input | • Repository or file(s) to scan |
| Output | • List of potential infringements with links to potential matches (e.g. in existing OSS)<br>• Weighting/ordering of potential matches |
| Comments | • TODO: Discuss whether snippet similarity and file similarity should be treated the same. While snippet similarity often is seen critical due to many false positives, file similarity can be pretty relevant, especially for languages such as C/C++ or Python |

## Toolchain elements and capabilities

Open Source
Tooling Group

### Component Inventory (Metadata Repository)

A system or service that stores metadata about used software components. This includes meta data like ids of the components in other systems, licenses, copyrights, known vulnerabilities and information, that is needed to do export classifications (ECCN), such as information about the contained cryptographic functionality. The Component Metadata Repository can be linked to an external FOSS Metadata Database to retrieve commonly known information and make it usable within the organization. Also Security Vulnerability Database and other sources for e.g. export classification-relevant data, can be linked to retrieve the necessary information and to make it available within the company.

### Package Data Repository

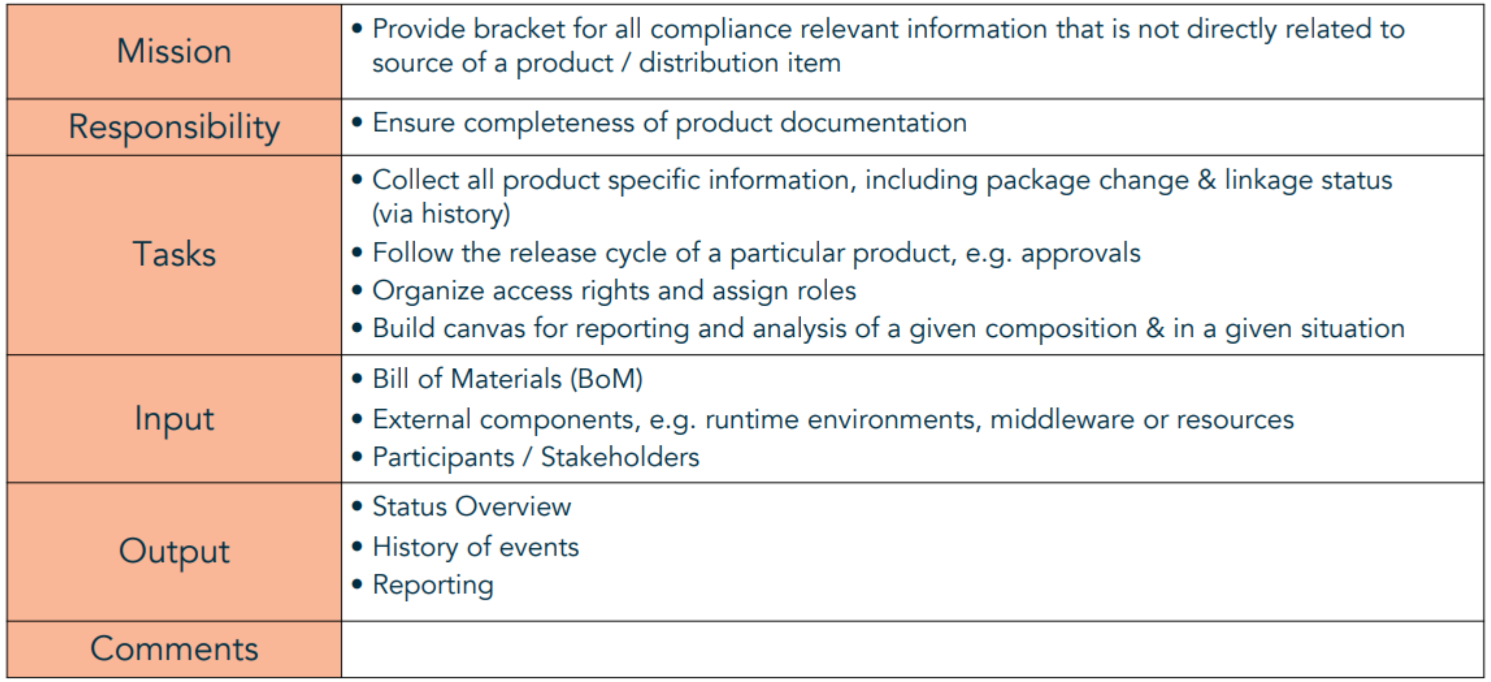
| | |
|---|---|
| Mission | • Collect package information and clearing data on packages |
| Responsibility | • Single point of truth for package information |
| Tasks | • Store packages data<br>• Support composition analysis (verification of dependency analysis)<br>• Provide search capabilities to identify existing packages<br>• Support authentication/authorization to ensure responsible data handling/editing |
| Input | • Package data and metadata (if known) |
| Output | • Package data and metadata, including package type (e.g. OSS, COTS, internal) and completion/verification status of associated metadata<br>• Containment structures (consists of)<br>• Dependency structures (depends on)<br>• Optional: relate known vulnerability information (not OSC specific, but a good place) |
| Comments | • TODO: Clarify role or repo in relation to the archive function. SW360 comes as archive, which actually could also be served by Git or binary repositories. Thus adding an archive function here might just duplicate the code<br>• TODO: Clarify unique identification of OSS pkgs,  e.g. package URL,  to be mandatory |

## Toolchain elements and capabilities

Open Source
Tooling Group

Not existing

### Situation Data(Structure of Solution, Circumstances, etc.)



| | |
|---|---|
| Mission | • Provide bracket for all compliance relevant information that is not directly related to source of a product / distribution item |
| Responsibility | • Ensure completeness of product documentation |
| Tasks | • Collect all product specific information, including package change & linkage status (via history)<br>• Follow the release cycle of a particular product, e.g. approvals<br>• Organize access rights and assign roles<br>• Build canvas for reporting and analysis of a given composition & in a given situation |
| Input | • Bill of Materials (BoM)<br>• External components, e.g. runtime environments, middleware or resources<br>• Participants / Stakeholders |
| Output | • Status Overview<br>• History of events<br>• Reporting |
| Comments | |

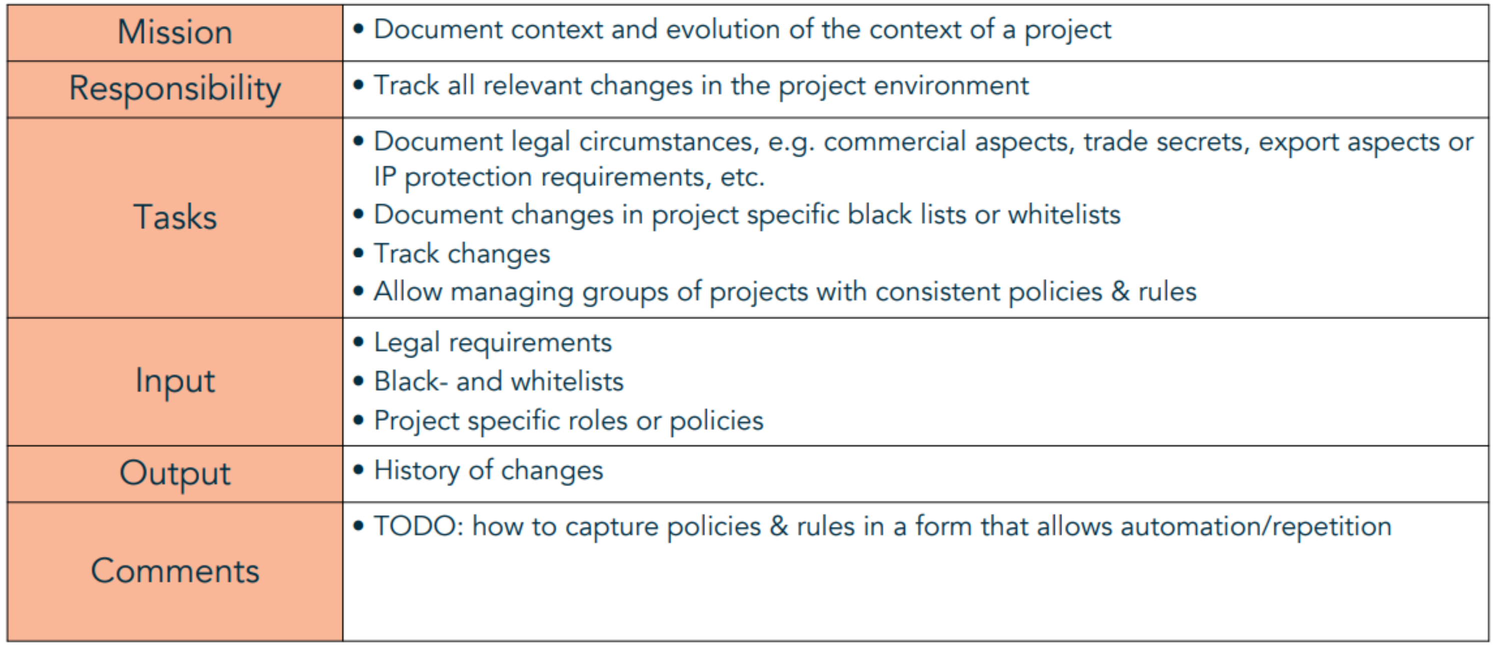
# Toolchain elements and capabilities

## Policy Checker (Compliance Checker)

System or service which takes at least the concrete deliverable of a software product or service, the licenses of the the integrated OSS components and checks whether the concrete deliverable is in-line with the company policy for the specific product category and the deliverable.

E.g. Not all licenses may be allowed for a certain product or service category or certain licenses may not be allowed for a certain deliverable, etc.

## Policies & Rules



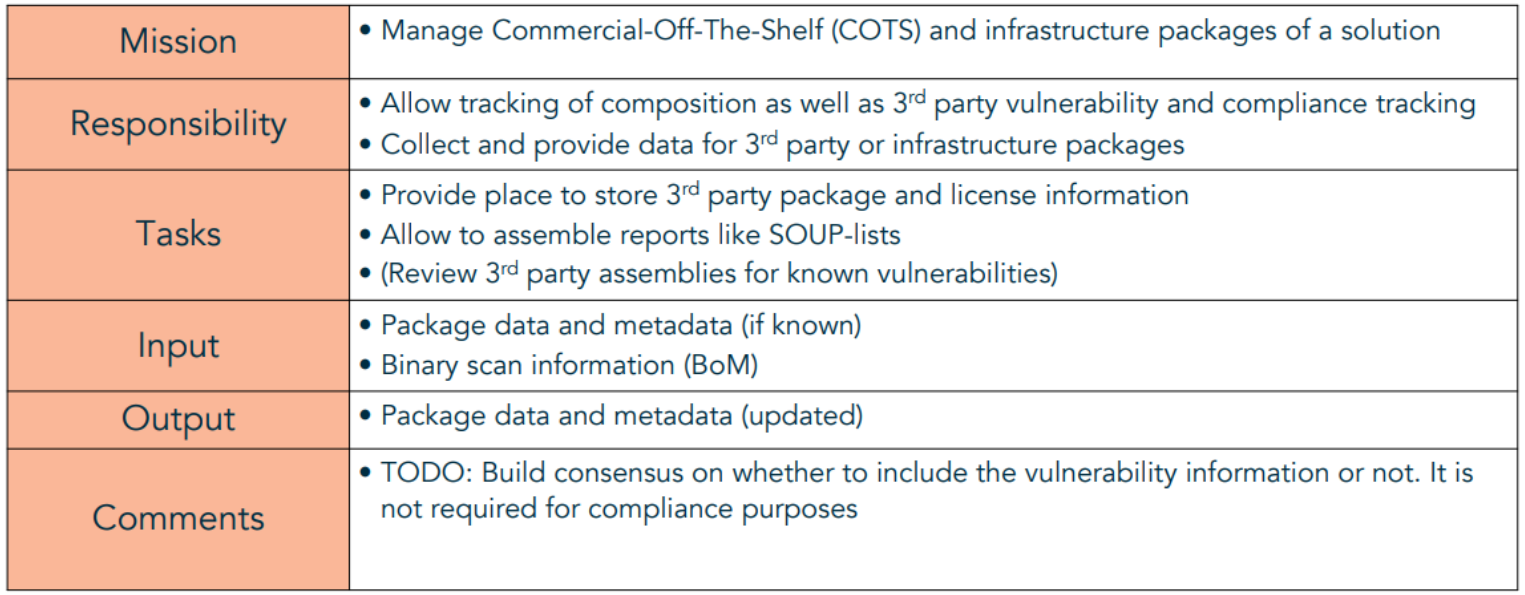
| | |
|---|---|
| Mission | • Document context and evolution of the context of a project |
| Responsibility | • Track all relevant changes in the project environment |
| Tasks | • Document legal circumstances, e.g. commercial aspects, trade secrets, export aspects or IP protection requirements, etc.<br>• Document changes in project specific black lists or whitelists<br>• Track changes<br>• Allow managing groups of projects with consistent policies & rules |
| Input | • Legal requirements<br>• Black- and whitelists<br>• Project specific roles or policies |
| Output | • History of changes |
| Comments | • TODO: how to capture policies & rules in a form that allows automation/repetition |

## Toolchain elements and capabilities

### Component Inventory (Metadata Repository)

A system or service that stores metadata about used software components. This includes meta data like ids of the components in other systems, licenses, copyrights, known vulnerabilities and information, that is needed to do export classifications (ECCN), such as information about the contained cryptographic functionality. The Component Metadata Repository can be linked to an external FOSS Metadata Database to retrieve commonly known information and make it usable within the organization. Also Security Vulnerability Database and other sources for e.g. export classification-relevant data, can be linked to retrieve the necessary information and to make it available within the company.

### COTS Management



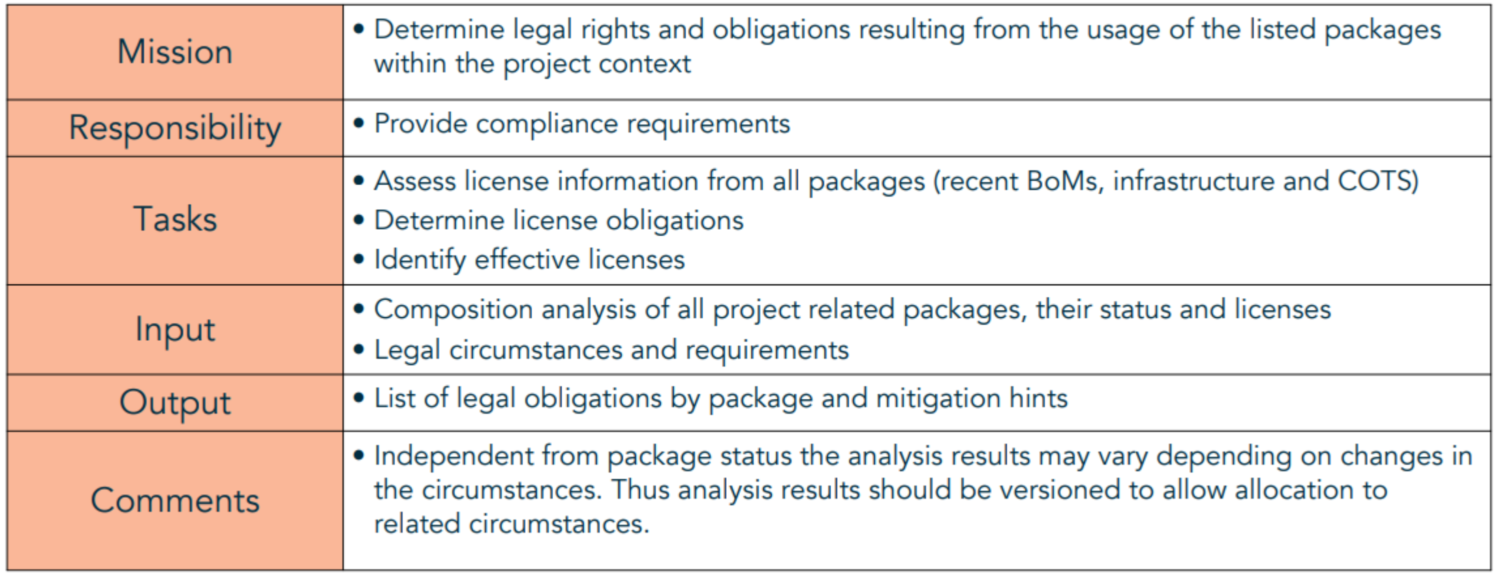
| | |
|---|---|
| Mission | • Manage Commercial-Off-The-Shelf (COTS) and infrastructure packages of a solution |
| Responsibility | • Allow tracking of composition as well as $3^{rd}$ party vulnerability and compliance tracking<br>• Collect and provide data for $3^{rd}$ party or infrastructure packages |
| Tasks | • Provide place to store $3^{rd}$ party package and license information<br>• Allow to assemble reports like SOUP-lists<br>• (Review $3^{rd}$ party assemblies for known vulnerabilities) |
| Input | • Package data and metadata (if known)<br>• Binary scan information (BoM) |
| Output | • Package data and metadata (updated) |
| Comments | • TODO: Build consensus on whether to include the vulnerability information or not. It is not required for compliance purposes |

## Toolchain elements and capabilities

### Obligation Fulfilment

A system or service where all the obligations resulting from the integrated OSS components are aggregated. How these obligations are fulfilled for a certain deliverable is documented and made available to provide an audit trail that a certain deliverable is made available in a license compliant way.

### Legal Solver



| | |
|---|---|
| Mission | • Determine legal rights and obligations resulting from the usage of the listed packages within the project context |
| Responsibility | • Provide compliance requirements |
| Tasks | • Assess license information from all packages (recent BoMs, infrastructure and COTS)<br>• Determine license obligations<br>• Identify effective licenses |
| Input | • Composition analysis of all project related packages, their status and licenses<br>• Legal circumstances and requirements |
| Output | • List of legal obligations by package and mitigation hints |
| Comments | • Independent from package status the analysis results may vary depending on changes in the circumstances. Thus analysis results should be versioned to allow allocation to related circumstances. |

# Toolchain elements and capabilities

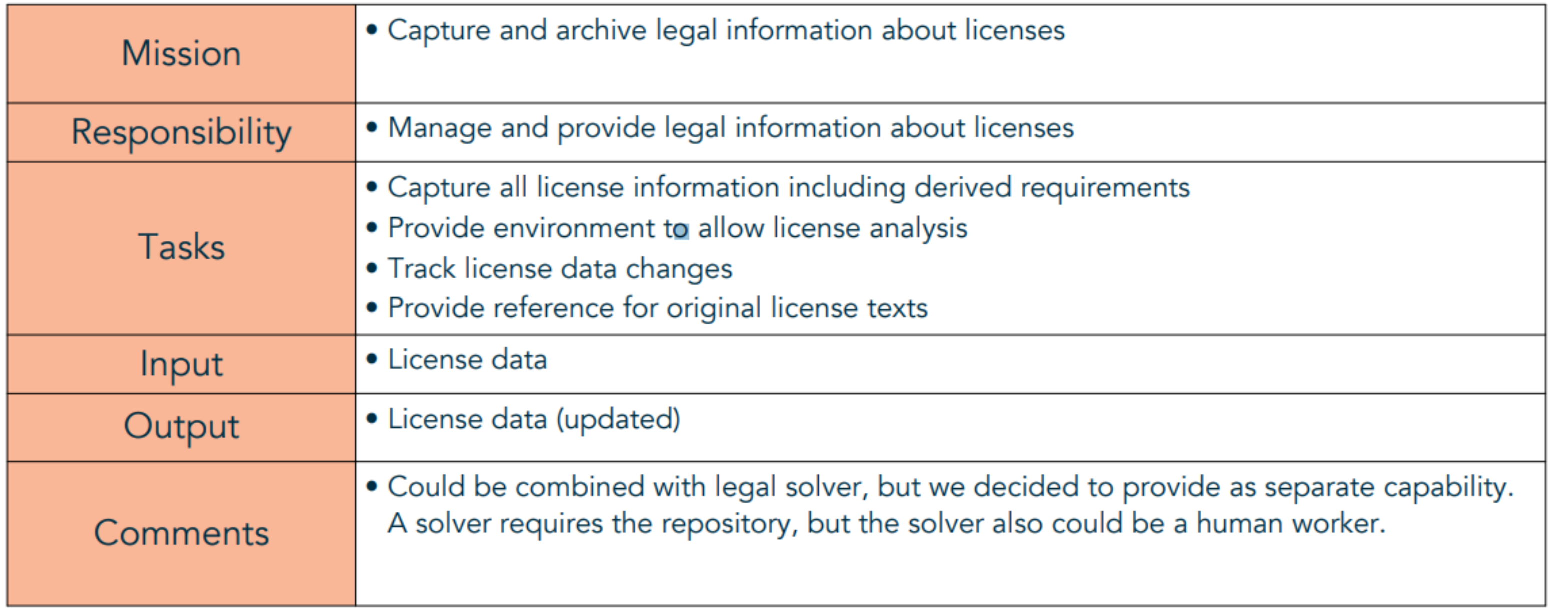
## License Obligations Database

System or service which provides for licenses its legal analysis. Usually the licenses analysis results in the more easy to understand:

- obligations
- restrictions
- permissions
- risks

An obligation is for example that the text of the license has to be provided together with the software. A restriction is for example that one is not allowed to impose further restrictions on the recipients exercise of the rights granted in the license.

## License Repository



| | |
|---|---|
| Mission | • Capture and archive legal information about licenses |
| Responsibility | • Manage and provide legal information about licenses |
| Tasks | • Capture all license information including derived requirements<br>• Provide environment to allow license analysis<br>• Track license data changes<br>• Provide reference for original license texts |
| Input | • License data |
| Output | • License data (updated) |
| Comments | • Could be combined with legal solver, but we decided to provide as separate capability. A solver requires the repository, but the solver also could be a human worker. |

# Toolchain elements and capabilities

## FOSS Compliance Bundle Generator

A tool that creates all the necessary documentation needed for the distribution of a software e.g. a so called FOSS disclosure document consisting of:

•required legal notices

•optional: a written offer to provide the complete corresponding source code with data whom to contact

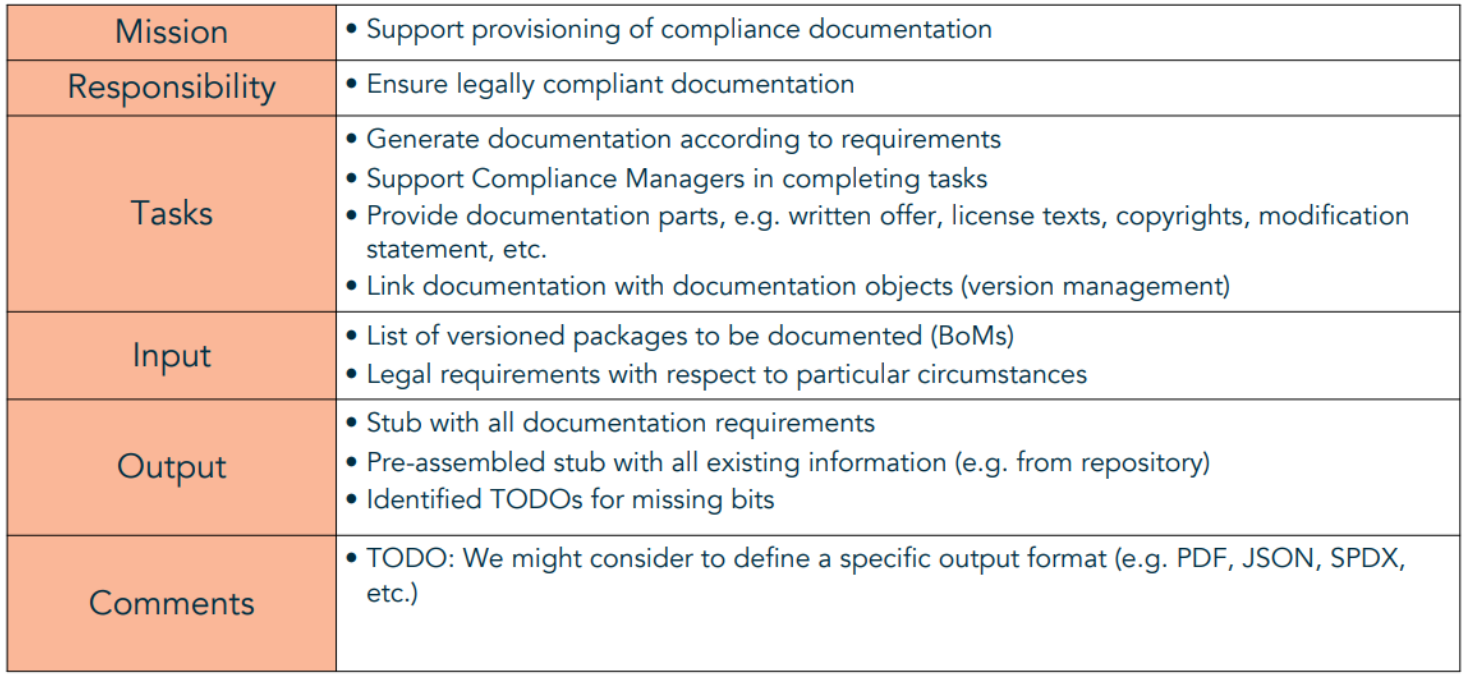
•list of all integrated components with all:

- copyright notices
- acknowledgments
- applicable licenses

Another output the FOSS compliance bundle generator produces is the source code bundle consisting of the source code of the integrated FOSS packages. As well as the description of the build environment.

Note that different delivery models of a product may require different FOSS compliance bundles.

## Compliance Artefact Generator

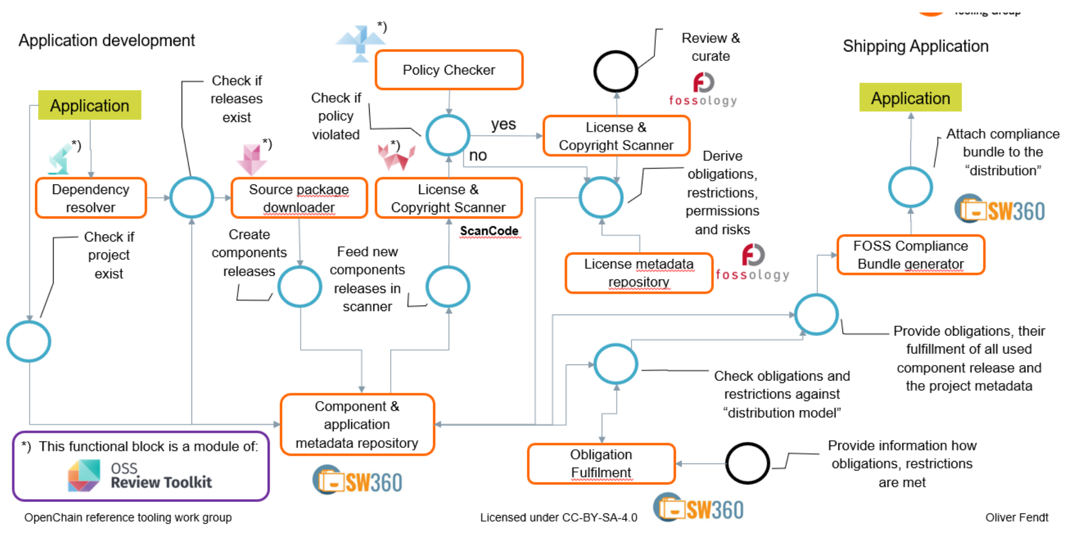


| | |
|---|---|
| Mission | • Support provisioning of compliance documentation |
| Responsibility | • Ensure legally compliant documentation |
| Tasks | • Generate documentation according to requirements<br>• Support Compliance Managers in completing tasks<br>• Provide documentation parts, e.g. written offer, license texts, copyrights, modification statement, etc.<br>• Link documentation with documentation objects (version management) |
| Input | • List of versioned packages to be documented (BoMs)<br>• Legal requirements with respect to particular circumstances |
| Output | • Stub with all documentation requirements<br>• Pre-assembled stub with all existing information (e.g. from repository)<br>• Identified TODOs for missing bits |
| Comments | • TODO: We might consider to define a specific output format (e.g. PDF, JSON, SPDX, etc.) |

# Toolchain elements and capabilities

Open Source Tooling Group

## Reference workflow



Application development
Application
Dependency resolver
Check if project exist
Check if releases exist
Source package downloader
Create components releases
Check if policy violated
Policy Checker
yes
no
License & Copyright Scanner
ScanCode
Feed new components releases in scanner
Component & application metadata repository
SW360
*) This functional block is a module of: OSS Review Toolkit
Review & curate
fossology
License & Copyright Scanner
Derive obligations, restrictions, permissions and risks
License metadata repository
fossology
Check obligations and restrictions against "distribution model"
Obligation Fulfilment
SW360
Provide information how obligations, restrictions are met
Shipping Application
Application
Attach compliance bundle to the "distribution"
SW360
FOSS Compliance Bundle generator
Provide obligations, their fulfillment of all used component release and the project metadata
OpenChain reference tooling work group
Licensed under CC-BY-SA-4.0
Oliver Fendt

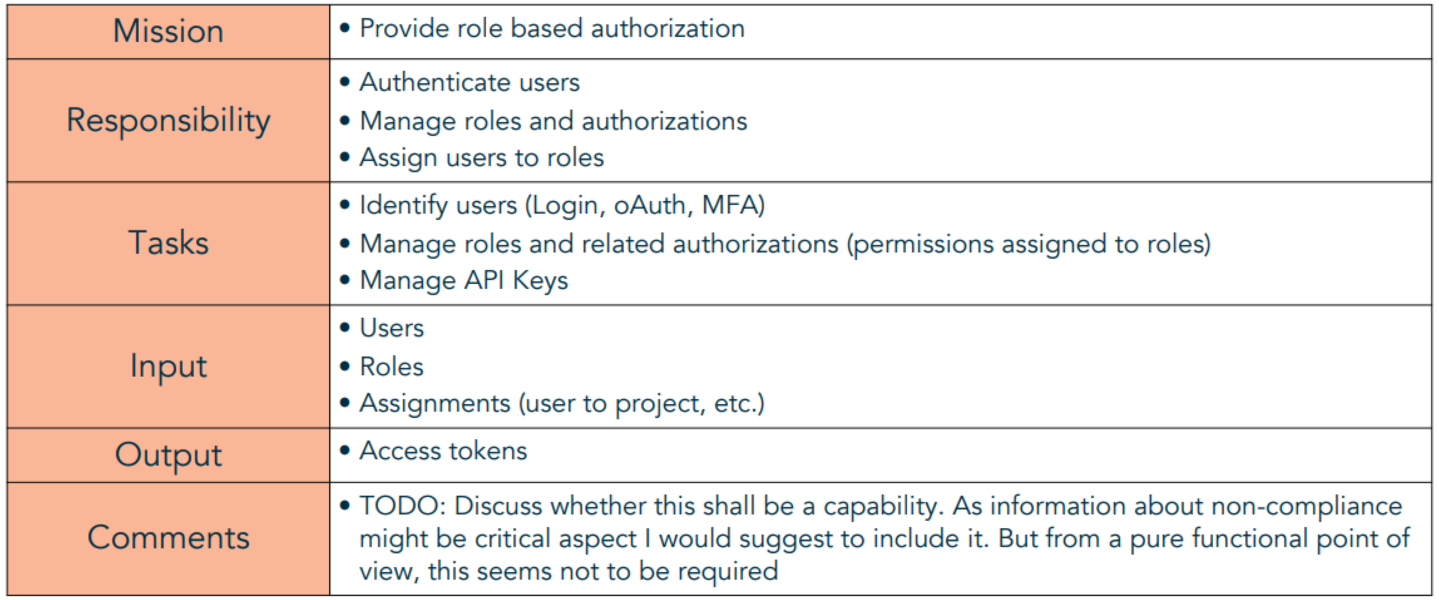
## Approval Flow

| | |
|---|---|
| Mission | • Help decentralising compliance work through approval |
| Responsibility | • Provide approval flow appropriate for audit |
| Tasks | • Track all legally relevant changes to products and packages<br>• Identify authors of change<br>• Provide compliance status and overview<br>• Allow to approve or reject an approval request<br>• Document/archive all decisions (auditing) |
| Input | • Approval request for (list of packages, legal situation, compliance documentation) |
| Output | • State of compliance analysis for approval request<br>• Approval / Rejection documentation |
| Comments | • The approval by a dedicated, skilled resource (Compliance Manager) combined with the automation support for all prior steps reduces the need for Compliance Managers |

## Toolchain elements and capabilities

Not existing

### User & Role Management

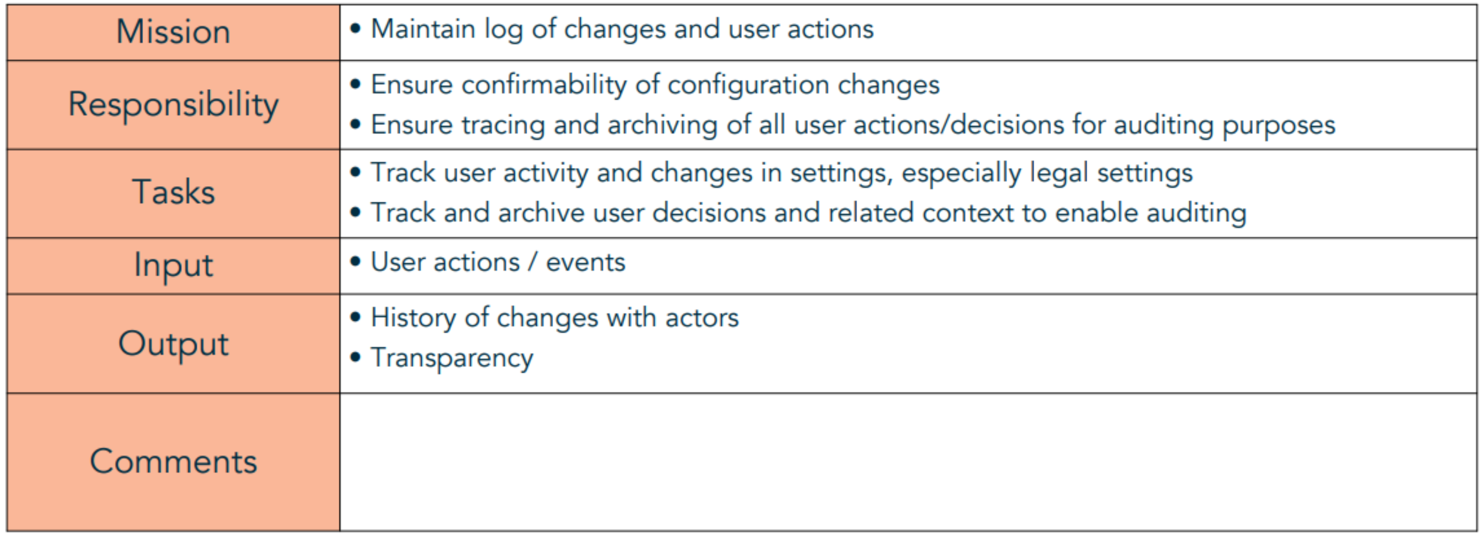


| | |
|---|---|
| Mission | • Provide role based authorization |
| Responsibility | • Authenticate users<br>• Manage roles and authorizations<br>• Assign users to roles |
| Tasks | • Identify users (Login, oAuth, MFA)<br>• Manage roles and related authorizations (permissions assigned to roles)<br>• Manage API Keys |
| Input | • Users<br>• Roles<br>• Assignments (user to project, etc.) |
| Output | • Access tokens |
| Comments | • TODO: Discuss whether this shall be a capability. As information about non-compliance might be critical aspect I would suggest to include it. But from a pure functional point of view, this seems not to be required |

## Toolchain elements and capabilities

Not existing

### Audit Log

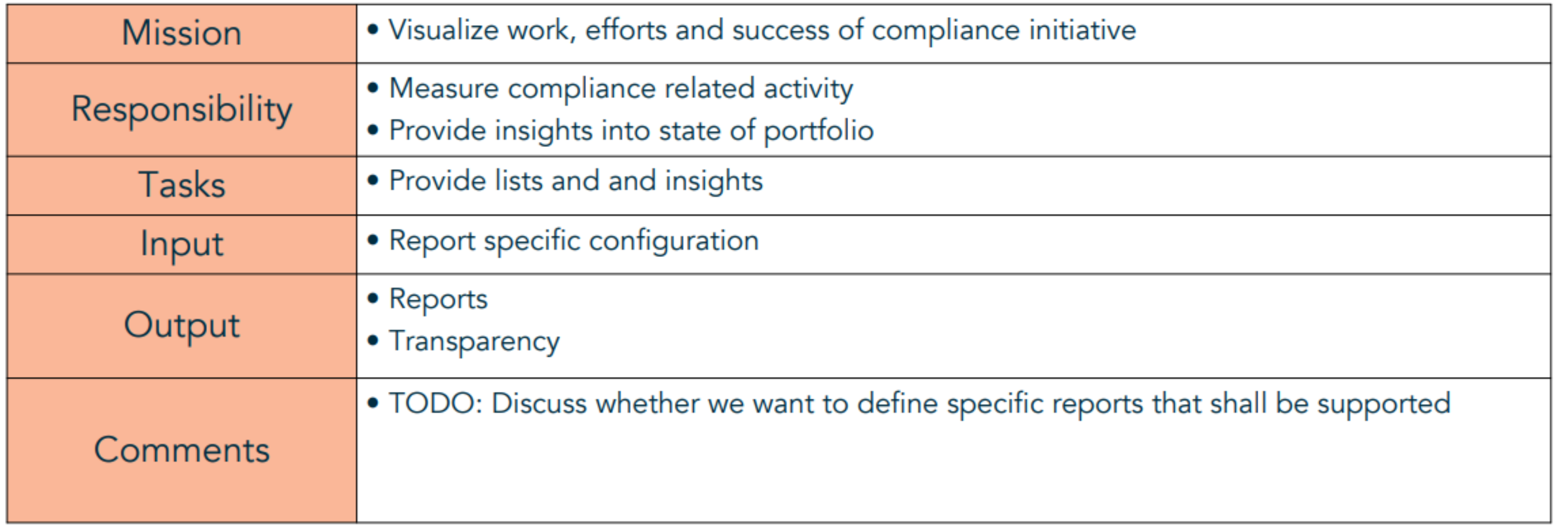


| | |
|---|---|
| Mission | • Maintain log of changes and user actions |
| Responsibility | • Ensure confirmability of configuration changes<br>• Ensure tracing and archiving of all user actions/decisions for auditing purposes |
| Tasks | • Track user activity and changes in settings, especially legal settings<br>• Track and archive user decisions and related context to enable auditing |
| Input | • User actions / events |
| Output | • History of changes with actors<br>• Transparency |
| Comments | |

# Toolchain elements and capabilities

Not existing

## Reporting & Analytics

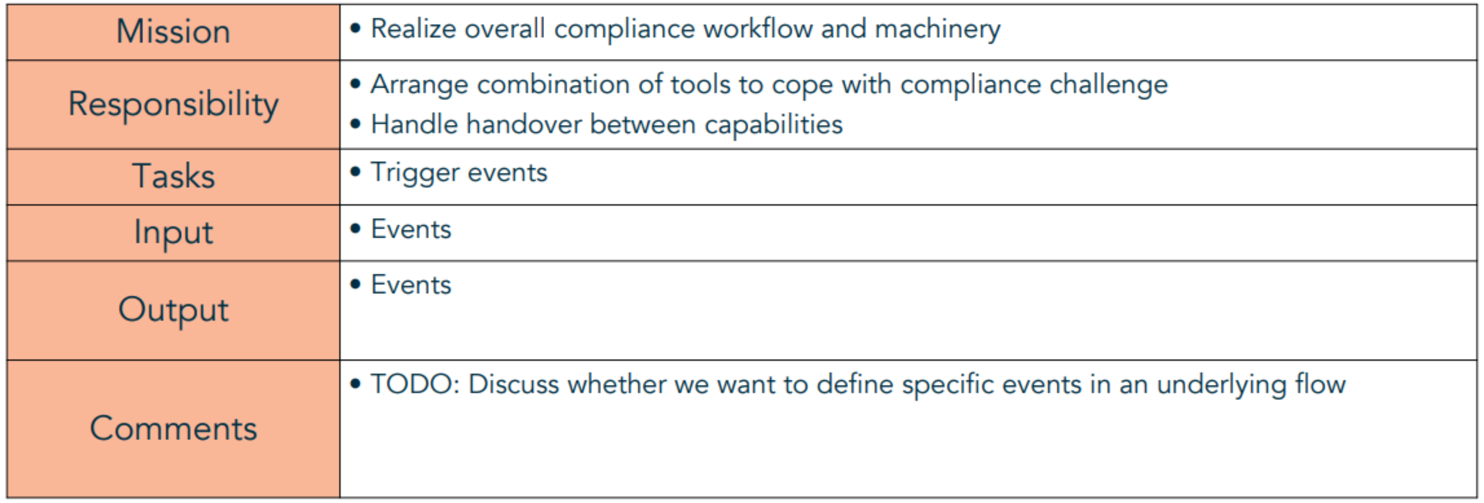


| | |
|---|---|
| Mission | • Visualize work, efforts and success of compliance initiative |
| Responsibility | • Measure compliance related activity<br>• Provide insights into state of portfolio |
| Tasks | • Provide lists and and insights |
| Input | • Report specific configuration |
| Output | • Reports<br>• Transparency |
| Comments | • TODO: Discuss whether we want to define specific reports that shall be supported |

## Toolchain elements and capabilities

Not existing

### Reporting & Analytics



| | |
|---|---|
| Mission | • Realize overall compliance workflow and machinery |
| Responsibility | • Arrange combination of tools to cope with compliance challenge<br>• Handle handover between capabilities |
| Tasks | • Trigger events |
| Input | • Events |
| Output | • Events |
| Comments | • TODO: Discuss whether we want to define specific events in an underlying flow |

## Links / Communication

Where to communicate what?

Github:
https://github.com/Open-Source-Compliance/Sharing-creates-value

Slack:
https://join.slack.com/t/ossbasedcompl-bhx9742/shared_invite/enQtODE2MTMxNzUyNDY1LWQyNWVlNzkyMjhhOWUyNDdjNDJlMzk0YzU0NDUwNzQ2YzY0Mzc1N2Y2NjhhZGEyN2JmNDE0ZTg2MTBjYmM3MWI

Mailing List:

Subscription page: https://groups.io/g/oss-based-compliance-tooling

Email address: oss-based-compliance-tooling@groups.io