05-03-2024

Software Licence Selection and Management in GÉANT

Grant Agreement No.:	101100680
Work Package:	WP9
Task Item:	Task 2
Nature of Document:	Guide
Dissemination Level:	PU (Public)
Lead Partner:	UoB/AMRES
Document ID:	GN5-1-23-FB1864
Authors:	Branko Marović (UoB/AMRES); Magdalena Rząca (GÉANT Association)

Abstract

This document provides a detailed guide to software licence selection, declaration, compliance and management in GÉANT. Intended for software development teams, it contains both essential information and practical step-by-step instructions. It is divided into two main parts: Essential Aspects and Elements of Software Licensing, and Complying with a Selected Licence.

Co-funded by
the European Union

Table of Contents

Table of Figures

1 Introduction

The primary objective of this guide is to delve into the intricacies of licence selection, declaration, compliance and associated tasks, offering a step-by-step elaboration of licensing mechanics for software development teams.

This guide is divided into two main parts.

The first part, Essential Aspects and Elements of Software Licensing, provides software developers with crucial insights into software licence selection and related processes, highlighting the tasks they need to perform and the collaborative procedures required to engage with licensing support and related services. This part provides a straightforward overview of the elements necessary for efficient preparation, information gathering and conducting proper software licensing.

The second part, Complying with a Selected Licence, offers a detailed description of how to implement the selected licence from a developer's perspective, providing instructions facilitating the execution of the licensing process. It gives in-depth information alongside simple but critical guidance on creating essential licensing artefacts.

Note that this document does not delve into the specifics of individual open source software (OSS) licences, software composition analysis (SCA) tool usage, licence compatibility and the remediation of licence conflicts. This is intentional, recognising that some developers may not need to grapple with these complex and resource-intensive matters, while many others could be shielded from these issues through the support of the software licensing team and their services. In cases where such details are required, they are covered in separate guides provided by the software licensing team. For comprehensive information about individual licences, licence compatibility and the nuances of licence selection, please consult the training and reference materials listed in the Resources section of this guide.

2 Essential Aspects and Elements of Software Licensing

This part of the guide covers the following topics:

- Significance of open source software (OSS) and licensing.
- OSS conditions.
- Compatibility of licences frequently used in GÉANT.
- GÉANT Intellectual Property Rights (IPR) Policy.
- Licence governance in GÉANT.
- Software composition analysis (SCA) service.
- Mend SCA tool.
- Software licence analysis (SLA) tool.
- Licensing process, decisions and artefacts.
- Licences and tracking of documentation, data and other works.

2.1 Significance of Open Source Software and Licensing

These are the key factors related to the use of open source software (OSS):

- Our work increasingly relies on OSS. Developers, National Research and Education Networks (NRENs) and GÉANT widely and increasingly use, adapt, create and endorse OSS.
- In a broader context, the ICT and R&E communities value OSS for its cost-effectiveness, transparency, collaboration, customisability, vendor independence, longevity, security, educational value, compatibility, ethical and philosophical values, accessibility and more (detailed below).
- OSS licences ensure the licensed software remains free, prevent appropriation and help avoid abandonment.
- Declaring a software licence makes it easier to select what other code and libraries can be incorporated into the project and how others can use, adapt and contribute to the software.
- Licensing considerations are critical when there is a distribution or sharing of software, as OSS licences come with specific conditions.
- Declaring a licence and licence compliance are also essential for legal reasons and software usability, ensuring better transparency and collaboration.
- Adhering to the requirements of applied licences (including those of dependencies) enhances the transparency of the software project within the wider community.

Open source software is important in various domains, including technology, research and education, business and government. It plays a crucial role in promoting affordability, transparency, collaboration, and technology innovation. It empowers individuals and organisations to take control of their software solutions, adapt them to their needs, and contribute to a global community of developers and users.

1. **Cost-effective** – Open source software is often free to use, significantly reducing software acquisition and licensing costs for individuals, businesses and organisations. This cost-effectiveness is especially critical for smaller businesses, educational institutions and governments with budget constraints.
2. **Transparency** – Open source software is built on open and transparent development processes. Anyone can review the source code to understand how the software works, enhancing trust and security. This transparency is particularly important for software used in critical applications, such as cybersecurity and healthcare.
3. **Community collaboration** – Open source projects typically have large and diverse communities of developers and users who collaborate to improve the software. This collaborative approach results in rapid bug fixes, updates and feature enhancements. It also fosters innovation and the sharing of knowledge.
4. **Customisation** – Users of open source software have the freedom to modify and customise the code to suit their specific needs. This flexibility allows businesses and individuals to adapt software to their unique requirements, giving them a competitive edge.
5. **Vendor independence** – With proprietary software, users are often locked into a single vendor's ecosystem. Open source software reduces vendor lock-in, as users have access to the source code and can switch service providers or modify the software as needed.
6. **Longevity** – Proprietary software may be discontinued by the vendor, leaving users without support or updates. Open source software tends to have longer lifespans, as the community can take over maintenance and development if the original project loses momentum.
7. **Security** – While open source software is not immune to vulnerabilities, its transparency allows a global community to continually audit the code for security flaws. When vulnerabilities are discovered, they can be fixed quickly. In contrast, the security of proprietary software depends solely on the vendor's resources and priorities.
8. **Education and learning** – Open source software encourages learning and skill development. Students and aspiring developers can study, modify, and contribute to open source projects, gaining practical experience and exposure to real-world software development.
9. **Compatibility** – Open standards and open source software often go hand in hand, promoting compatibility and interoperability between different software and systems and reducing barriers to data exchange and collaboration.
10. **Ethical and philosophical values** – The open source movement is rooted in values such as transparency, collaboration, and the idea that software should be a public good. Many individuals and organisations choose open source software to align with these values and principles.
11. **Global accessibility** – Open source software is accessible to users worldwide, regardless of location or economic status. This accessibility promotes digital inclusion and levels the playing field for all users.

In addition to these general OSS-related factors, there are several requirements and recommendations applicable in the GÉANT context, some of which are implied by its IPR Policy [GN_IPRPolicy] (also see GÉANT Resources – Intellectual Property [GN_Resources_IP]):

- Every GÉANT software project should select and **apply a suitable OSS licence** that fits the needs of the software development team and those of the user community.
- **Start the licensing process early**, to make it easier to set up a licence and maintain compliance.
- The **chosen licence must be compatible** with licences of all used components so that the IPR and licensing risks on GÉANT are eliminated.
- It is preferable to place the OSS source code in a public and **versioned code repository** with a clear **indication of the used licence**.

- **Copyright** information must indicate GÉANT's involvement and support. This information underscores that work was conducted within the GÉANT project or received support from it and identifies who authored the produced software.
- **Assess the used components and software** by applying common software quality and trustworthiness checklists, to ensure the components used and software produced are reliable. Examples: TinyMCE – Open source software evaluation checklist [TinyMCE_OSSEC], Red Hat – Checklist for measuring the health of an open source project [RedHat_COSP], EURISE Network Technical Reference – Software quality checklist [EURISE_SQC].
- **Use software composition and license analysis (SCA and SLA) services** that conduct related reviews and audits designed to help determine the OSS licence appropriate for the software and ensure licence compliance. Identifying and addressing vulnerabilities in the software that may be detected by the SCA improves its quality and benefits the broader community your team contributes to.
- **Set up contribution, communication and governance workflows** that ensure compliance with the software's licence.
- **Adhere to the standards of the domain community** in software development, licensing, provision of metadata about software, documentation, registration in relevant community registries, citation and promotion of software.
- If applicable, **enable and advise on the citation and referencing** of software in scientific papers, presentations, tutorials, etc., ensuring that these references are unambiguous and permanent.

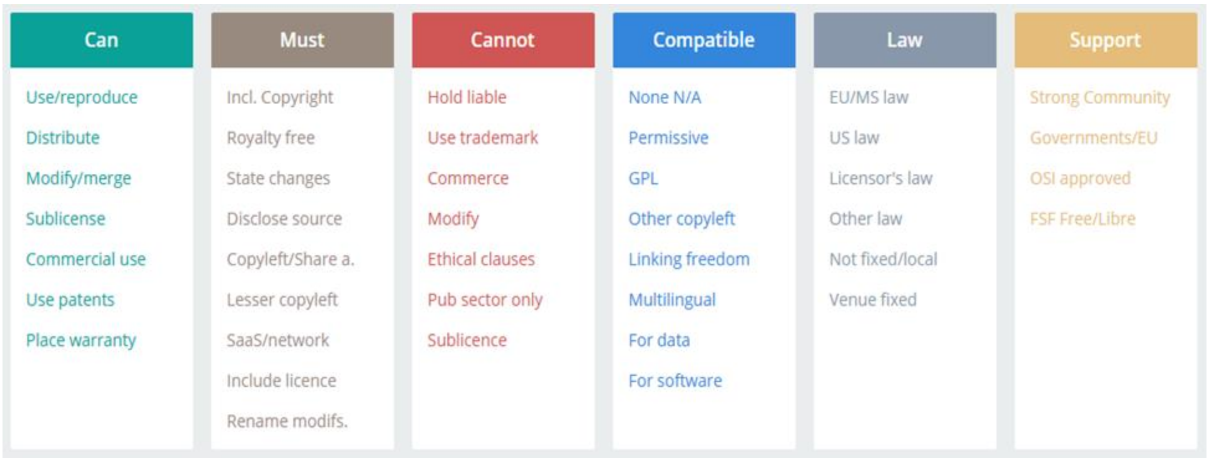
2.2 OSS Conditions

Selecting the licence is not straightforward, and it is a task most developers would prefer to avoid. While the GÉANT IPR Policy [GN_IPRPolicy] favours permissive licences and even names some, such as MIT, BSD and EUPL, the actual choices available are limited by the licences of the core components of the software or the frameworks it utilises. Therefore, it becomes meaningful for developers to delve into specific licences, their implications and compatibility only when the constraints are known and when it becomes necessary within the licensing process, with the assistance of the licensing team and the GÉANT IPR Coordinator.

For example, a developer might have to use the Apache licence if the framework they are using or relying on supports it. Alternatively, if there are unavoidable GPL or AGPL dependencies, they may opt for a compatible licence, which is often the most restrictive of the licences involved. Conversely, in cases where there are no preexisting limitations, as with an unconstrained new project or when all essential components use permissive licences, the development team can choose one or several candidate licences based on their key features with regard to code sharing and modification (where copyleft licences are more stringent), relicensing (also referred to as sublicensing, which the team may wish to prohibit or allow), and the way patents are handled (waived, protected or ignored). (For further information about software selection, see Section 2.8 Software Licence Analysis (SLA) Service.) Various licence conditions are often classified as:

- Rights/permissions – what you can do.
- Requirements/obligations – related to behaviour and the provision of compliance artefacts such as copyright, disclaimer, licence text, notices, relevant source code, build and installation instructions, etc.
- Restrictions/limitations – limits of use or what you must not do.
- Other characteristics – typical uses, classification, compatibility, legal features, origin, community, endorsement and more.

These features are further differentiated in various models, one of which is provided through the EC's Joinup Licensing Assistant tool (JLA – Find and compare software licenses [JLA]), which, when choices exist, greatly facilitates the selection of the most appropriate licence aligned with the developers' strategy and user community.



Can	Must	Cannot	Compatible	Law	Support
Use/reproduce	Incl. Copyright	Hold liable	None N/A	EU/MS law	Strong Community
Distribute	Royalty free	Use trademark	Permissive	US law	Governments/EU
Modify/merge	State changes	Commerce	GPL	Licensor's law	OSI approved
Sublicense	Disclose source	Modify	Other copyleft	Other law	FSF Free/Libre
Commercial use	Copyleft/Share a.	Ethical clauses	Linking freedom	Not fixed/local	
Use patents	Lesser copyleft	Pub sector only	Multilingual	Venue fixed	
Place warranty	SaaS/network	Sublicence	For data		
	Include licence		For software		
	Rename modifs.				

Figure 2.1: OSS conditions from JLA – Find and compare software licenses [JLA]

2.3 Compatibility of Licences Frequently Used in GÉANT

We have compiled comprehensive information about OSS licences, some of which is also available via Mend, the tool used for SCA (described in Section 2.7). However, there is a limited set of licences that are frequently used in GÉANT or are common sources of compatibility issues. Figure 2.2 presents an orientational diagram describing the relationships and compatibility of these licences. Please note that there are two distinct interpretations of licence compatibility. A less restrictive, more commonly used and symmetrical type of compatibility indicates that components with two distinct licences can be used in the same project, which may be achieved by relicensing one or both of them or by selecting a third licence for the encompassing product. A more restrictive and direct, yet asymmetrical, interpretation determines whether a component under one licence may be used in software under another licence. Although the first interpretation is dependent on the second, various types of "use" by the produced software exist and sometimes can be altered by modifying the system architecture to allow the integration of a problematic component without needing to change the licence of the created software.

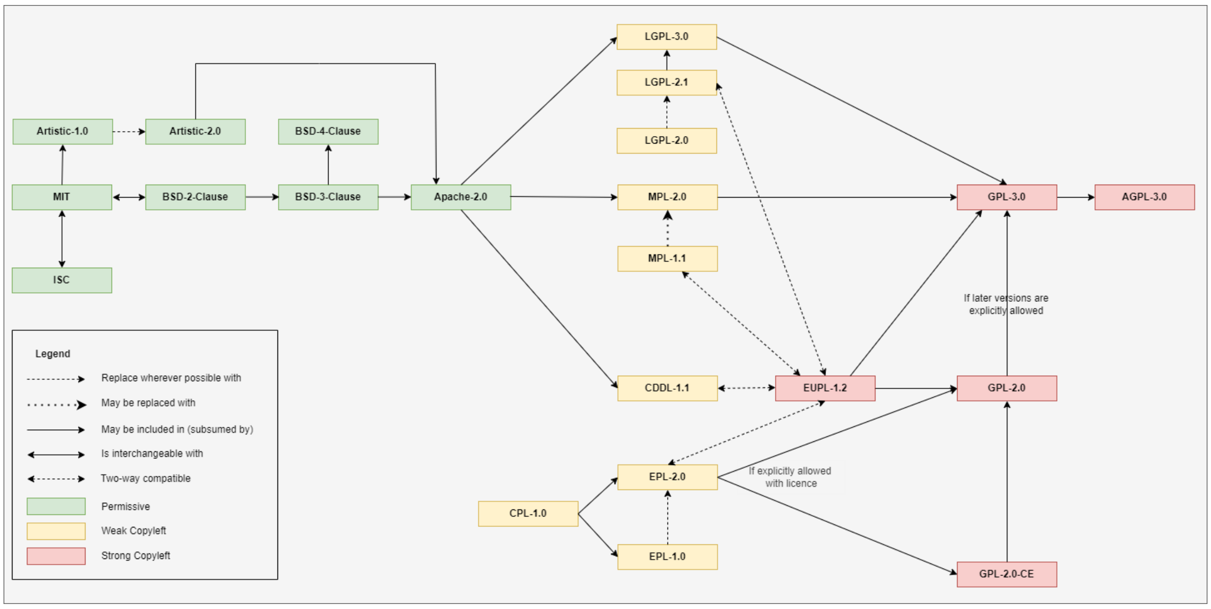


Figure 2.2: Relationships between OSS licences frequently used in GÉANT projects

OSS licences are described in several documents produced by the licensing team. For additional details, please refer to the following guides: *Reference information about OSS licences and tools* [Wiki_OSSL_RefInfo], *OSS licences and licence selection* [Wiki_OSSL&LS] and *Important licences for licence selection* [Wiki_ImportantLicences].

2.4 GÉANT IPR Policy

The GÉANT IPR Policy applies to IP generated within the GÉANT project, including open source software. It also provides related recommendations and rules which are made concrete in the present document through practical and actionable instructions. The policy is available at [GN_IPRPolicy]. The IPR Policy seeks to establish a framework for the intellectual property (IP) generated by the GÉANT project. It applies to all project participants of the GN5-*n* project and any other EC-funded GÉANT projects and provides practical and useful guidance in the area of IPR. Most importantly, the IPR Policy aims to establish a cooperation modus operandi and proper protection as well as fair use with regard to any IP created by GÉANT projects. The IPR Policy also aims to apply the principles of findability, accessibility, interoperability and reusability (FAIR) in the use of the project IP. The IPR Policy has been binding from the moment of its approval by the General Assembly and as of the start of the GN5-1 project (January 2023).

What is important from a software development perspective is that there is a software composition analysis (SCA) tool that allows a software scan to check and verify the licences of used components and determine which licence shall be used.

Failure to comply with OSS licence terms can have significant legal and monetary consequences, hence due diligence is required. The IPR Policy emphasises the importance of IP protection, introduces the GÉANT IP Register where project results will be documented, and highlights the significance and necessity of having the code scanned with an SCA tool to ensure licence compliance and compatibility. It also highlights the role of the IPR Coordinator in supporting project participants with the licence selection process.

To summarise the IPR Policy:

- **All OSS licences** [OSI_Licences] **are allowed**.

- The IPR Policy strongly **recommends the selection of permissive licences**.
- Copyleft licences (weak, strong or network protective) can be applied as necessary, in consultation with the IPR Coordinator.
- The IPR Coordinator provides the final recommendations and maintains the GÉANT IP Register.

The WP9 Task 2 software licensing team provides related services, support and guidance. Related guides developed by the team are listed in Section 4.3 Further Reading, while training and infoshare presentations are listed in Section 4.2 Training Materials.

2.5 Licence Governance in GÉANT

The goal of licence governance in the GÉANT project is to ensure compliance with GÉANT's IPR Policy while respecting dependencies' licences and domain community standards. It is **led by the IPR Coordinator** and **supported by the WP9 Task 2 activity Open Source and Licence Support (OSLS)**, or simply software licensing. The OSLS team:

- Helps those who prefer to invest in understanding and managing OSS licences, master the provided tools, and apply them.
- Offers knowledge and support to solution designers, developers and skilled promoters on licences and intellectual property rights (IPR).

The licensing team provides technical and implementation support on open source software and licence management through two services:

- **Software composition analysis (SCA)** – Technical and practical assistance for software development teams with managing software components and their software licences.
- **Software licence analysis (SLA)** – Assistance to software teams in aligning their project and licensing decisions with the GÉANT IPR Policy and the guidance provided by the IPR Coordinator.

These services complement other software review services provided by WP9 Task 2 Software Governance and Support, namely SonarQube Setup Assistance and Extended Source Code Review [Wiki_SWReviews]. Through SCA and SLA services, the licensing team ensures the handling of key licensing concerns by:

- Assessing the situation with licences of components and prior IP.
- Selecting an open source licence for the software project's needs.
- Making sure the selected licence is compatible with the components' licences.
- Ensuring compliance with the chosen licence.

The licensing team helps GÉANT software teams address IPR and licensing issues by implementing robust processes for managing dependencies and licences and achieving compliance with the GÉANT IPR Policy. It provides mediated access to expert tools for analysing and managing open source licences. The OSLS has worked with many GÉANT software development teams to assess their licensing situations and decisions, with resulting recommendations to support reliable and effective IPR management, in line with the GÉANT IPR Policy.

GÉANT development and maintenance teams can contact the OSLS through the GÉANT Slack channel or email. SCA and SLA services are requested by submitting a software review request to the GÉANT Jira Software Tools Help Desk [Jira_RSWR], which also serves to track the progress of the work on them. Several iterations of analysis and licence and dependency adjustments may be required to reach satisfactory IPR status. The IPR Coordinator can be reached when assistance with licensing decisions is needed.

Our guides help with the licensing and services. For more information about OSS licences, read the *OSS licences and licence selection* guide first [Wiki_OSSL&LS]. Since the SLA service requires the software development team's involvement in licence selection, it is recommended that you read this guide before requesting the service. It also helps with interpreting the software composition analysis results. However, if you are already generally familiar with OSS licences or just want to get summaries of licences that are frequently present in GÉANT software projects or are typical of licence compatibility problems, you may go directly to the *Important licences for licence selection* guide instead [Wiki_ImportantLicences].

SCA and SLA present a valuable opportunity to elevate and align a software project with both GÉANT's and external expectations, encompassing licensing and policies. The analysis, selection and validation of software licences can greatly enhance the projects and bolster their credibility within GÉANT and beyond.

Engaging in software licensing offers a chance to meticulously evaluate the project's components, review their licences and consider aspects related to authorship, ownership, external relations and expectations, and associated documentation artefacts. This concerted effort contributes to standardising various projects in these respects. Licensing reviews engage individuals who were not the original developers to assess and validate the software project, injecting a significant impetus for its developers to critically evaluate and consolidate their work.

Furthermore, this activity entails registering and publishing software using GÉANT's internal software development tools and aligning them with established practices and expectations within GÉANT. The successful completion of licensing and the formalisation of the licence stand as positive indicators for the project within GÉANT. This holds particular significance for smaller and relatively autonomous developments, such as those undertaken within the GÉANT Trust and Identity Incubator, as it can enhance visibility and overall improvement in the practices and visibility of the originating activity. Moreover, addressing issues through licensing analysis and the resultant reports and decisions yield valuable insights for assessing software solutions and the services based on them during their evaluation at GÉANT Product Lifecycle Management (PLM) gates [PLM].

The performed analyses, when conducted for a module that is an add-on for an externally developed open source platform, may also benefit the broader community of the software platform the development team used or contributed to, by assessing the overall status of its licensing and the security of its components. This is a side effect of the analysis of software produced by GÉANT's software development teams, as it transitively includes an analysis of involved components and licences.

2.6 Software Composition Analysis (SCA) Service

This service assists software development teams by establishing a project within an SCA tool and providing valuable insights into external components. It is suitable for **one-time software analysis** but also **continuous monitoring**, identifying third-party components used and their licences, and offering information about potential IPR infringements and security vulnerabilities. The service can be used in combination with other software review services or performed exclusively. Repeated analyses can determine how changes in software and dependencies impact licence compliance and identify new or pending vulnerabilities. For ongoing monitoring, the produced analysis setup can be integrated into the project's continuous integration platform.

The SCA is currently based on Mend (described in more detail in Section 2.7), which is used to identify third-party components in projects and gather information about their licences and security vulnerabilities. Mend employs a comprehensive database to assist in two critical aspects:

- The analysis of components and their licences helps reduce risks associated with IPR infringements, which could have significant financial consequences, by achieving licence compatibility and compliance.

- The security of OSS is another important issue. Mend reports vulnerabilities through a report that complements SonarQube and extended code reviews.

The licensing team sets up the project in the Mend SCA tool, which generates reports on the software composition and potential deviations from established policies.

The visibility of produced reports and the created Mend project can be established while the software project is being set up, but can also be adjusted after the results of the analysis have been obtained or even at the end of the review.

The primary report ("Risk Report") is on the software composition with its components, their licences and related risks and vulnerabilities.

The designated leader or expert from the software development team receives this report and provides support in interpreting it, although the software development team should be able to interpret this report themselves.

The licensing team helps with this report if needed, and the developers can ask for additional feedback on the reported and other risks related to licences and IPR infringements.

A summary of the SCA service is available in *Software Reviews* [Wiki_SWReviews], with additional details in the *Client Guide for Software Composition Analysis (SCA)* [Wiki_CGSCA].

2.7 Mend SCA Tool

Mend is an online tool provided through a service purchased by GÉANT for open source licence and security compliance [Mend_SCA]. It is designed for in-house use by the customer and does not offer direct consultancy by legal experts. Mend can **detect software components**, **identify their open source licences** and **uncover vulnerabilities**. It can seamlessly integrate with the development environment, building a pipeline to detect open source libraries with security or compliance issues. Mend reports severe software bugs, problematic licences, new versions and available fixes. It simplifies the management of open source libraries and the detection and remediation of compliance and security issues. It builds an **inventory of software components** by detecting declared dependencies, matching them with a rich database providing **licence information**, warnings about outdated or risky open source libraries, and details of associated security vulnerabilities and issues. The provided licence information includes licence type, risk level, handling of patents, summary descriptions, and excerpts from original licence texts, etc.

The Mend tool, offered by WP9 Task 2, streamlines the process of verifying software IPR compliance and partially automates it. Mend provides visibility and control over the risks associated with open source. The licensing team sets up and maintains the Mend configuration, including the list of approved and rejected libraries provided by the software development team.

A short overview of Mend usage is available in the *Mend short guide for end users* [Wiki_MendGuide].

Mend can analyse projects in several ways. The provided code may be locally stored and a Mend scan can be manually triggered whenever the development team needs to assess the effects of a recent code change (details in *Adding project to Mend (Scan Flow)* [Wiki_MendAP]). Scanning of GÉANT software can be conducted by performing one integral Unified Agent (UA) project scan or multiple per-product scans. Currently, there is no versioning in Mend, so each software version is scanned as a separate Mend project.

Mend scans directories to find software components and identify vulnerable libraries, licensing conflicts or risks. After scanning the source code it displays the results in the Mend web application. By default, it checks the digital signatures of used components in the Mend database to detect and describe open source or commercial components in the product. Mend is a platform that enables users to connect to a GÉANT product (without having to review the code) and assess its compliance with a predefined IPR policy. Verification is accomplished by scanning the project, which populates the Mend web application dashboard with data about the project and enables the creation of reports on compliance with the help of Mend's backend database.

The web-based GUI provides numerous options and panels for reviewing and analysing scans of open source software in an organisation's products and projects. Each scanned product or project is displayed on the corresponding page displaying summary information about a specific product or project and offering various dashboard options, providing a comprehensive view of the organisation's open source status. The product/project page provides access to all contained projects and libraries used by the product/project.

Each Mend dashboard segment leads to more detailed pages and reports with charts and tables. The dashboard displays the following information:

- **Product Alerts** – displays valuable information about library (component) alerts generated for a product. The **New Versions** category shows the number of alerts triggered for scanned libraries that are out of date (i.e., not the latest version). Whenever an out-of-date library is found, a new alert is generated and displayed in the **Alerts** report. The alert indicates the out-of-date library and its new version.
- **Security and Quality** – displays the number of libraries containing vulnerabilities, sorted by severity, the score of the most vulnerable library, the count of libraries with newer versions and vulnerabilities, and the count of "buggy" libraries.
- **Libraries** – presents detailed information about the product libraries (components), including library name, library licence, and per-product or per-project occurrences of libraries.
- **Licence Analysis** – provides data on the distribution of licences used by product or project components. It displays the number of different licence types.

Administrators can customise system settings, manage user permissions, and configure integration with third-party components.

Additional and detailed information on licences is available in reports, which are available from the Report menu. The Risk Report contains useful information for analysis and is the most detailed in terms of content. It is a tool that provides a view of all aspects of libraries, their licences, security and quality. The report contains several panels and tables displaying risk-related information. Security and licence analysis data is also presented in other parts of Mend, such as the Product Dashboard.

The displayed information is based on an internal database of libraries, their obsolete versions and vulnerabilities, licences and licence conflicts. Since this database is continually updated, the produced reports can change over time even if a scan has not been performed in the meantime.

Mend information on OSS licences includes licence type, copyright, handling of patents and royalties, linking requirements, and compliance with free and open source software norms. Mend's experts have conducted an analysis of many licence types and defined risk scores to help developers easily assess risks associated with a particular licence. The primary score is the Copyright risk score calculated based on several factors (Risk Score Attribution [Mend_RSA]). Its purpose is to quantify, on a linear scale, the degree of loss of exclusive control over the code using a library or source code governed by that licence. The Copyright risk score is, therefore, more suitable for commercial organisations wanting to quantify or audit the level of exclusivity over their software assets and associated risks than it is for a software project willing to share, or interested in sharing, the code they developed. Since low values of this score, associated with the colour green, generally correspond to permissive licences, while high values, associated with red, correspond to strong copyleft licences, it can be used to quickly identify and assess the present licences. Licences are also quantified in terms of copyleft (no, partial, full) and linking (non-viral, dynamic, viral). There is also a Patent and Royalty risk score and a related attribute that indicates whether the software under the described licence is royalty-free (yes, conditional, no).

Mend can integrate with development environments and with build tools. It can be incorporated into a continuous integration (CI) pipeline, triggering scans with each commit in host repositories such as GitHub [GitHub], GÉANT GitLab [GN_GitLab] and Bitbucket [GN_Bitbucket]. GÉANT already uses Bamboo [GN_Bamboo] as the CI/CD software between the host repository and Mend (details in *Automated Mend scans with Bamboo* [Wiki_MendASB]).

Mend's functionality, originally tailored for commercial organisations and projects, is gradually moving towards licence compatibility checks more suitable for use within OSS projects. However, developers and project leaders still need to familiarise themselves with it and with licence peculiarities and limitations.

Mend does not provide full and entirely accurate licence detection. It is possible to manually correct licences of individual components at the organisational level. This helps with some libraries that do not have licence information or do not have a licence version specified. Sometimes, only suspect licences are indicated and they must be verified. Multi-licences and allowed relicensing are not always reliably handled and not all allowed licences are always listed. Also, there is no support for software versioning or differential reports.

All this prevents fully automated licence control and alerts. However, although an effort to interpret and improve the Mend analysis is sometimes needed, its use is much more efficient than manual analysis. Besides, even after it completes its composition analysis work, some decision-making and remediation are needed.

This is why the service described in the next section, software licence analysis (SLA), is needed. Mend provides a licence compatibility analysis which indicates the (likely) compatibility of other components with the selected one. However, this is far from an automatic licence selection, which will probably always require human decision-making and trade-offs.

It should be noted that other tools can be used in software composition and licence analysis; some are listed in *Other software composition analysis (SCA, software inventory) tools* [Wiki_OtherSCATools].

2.8 Software Licence Analysis (SLA) Service

Software licence selection involves choosing the appropriate licence for distributing and using the software. This decision is crucial as it defines the terms under which the software can be shared, modified and distributed. The selection process considers the project's goals, the developers' preferences, the desired level of collaboration, and the constraints imposed by licences of dependencies and other sideground IP. As touched on in Section 2.2, the most restrictive licence is often the only one compatible with all those present. However, this is not necessarily the case, as when permissive licences have been used. Furthermore, at times, a licence that is compatible with the most, or with all, may not even be among those that are present. Selection of a subsuming licence should also consider the effort needed to remediate detected licence compatibility problems, involve assessing the impact on the software's ecosystem, and ensure compliance with legal, regulatory and funding requirements. The chosen licence plays a pivotal role in fostering a collaborative and transparent development environment while providing clarity on how others can use and contribute to the open source project. All of the above illustrates the complexity of licence selection, and explains why the SLA service was designed and is needed.

The SLA service is a technical consultancy service providing a comprehensive understanding of third-party libraries within a software project and their licences. This understanding is crucial for selecting appropriate software and ensuring compatibility among all licences involved. The service is highly recommended for software development teams seeking to validate third-party licences, establish or review their software's licence, verify compliance with it and the GÉANT IPR Policy, or assess the implications of potential changes in the project licence, the effects of major changes in their software or changes in licences of used libraries or frameworks.

It provides a deeper insight into third-party library licences and their relationship with the project. A prior software composition analysis (SCA) is a prerequisite. The service relies on the results obtained from SCA but also conducts manual checks of detected libraries as needed.

The service includes customising the licence settings of the SCA tool, project licence selection with analysis of the relationship between the project licence and those of its dependencies, and checking alignment with licence requirements and the rules and recommendations of the GÉANT IPR Policy, both requiring verification of related documentation artefacts. If the SCA tool is used with continuous integration, the service team works with the customer to customise related settings.

A summary of the SLA service is available in *Software Reviews* [Wiki_SWReviews].

2.9 Licensing Process, Decisions and Artefacts

The typical steps for licence management include:

1. Gather information (can be done with Mend).
2. Document (can be partially done with Mend).
3. Remediate.
4. Create licence-related artefacts (to ensure compliance).

These are preceded by a number of preparatory activities and decisions, and should be followed by measures that ensure long-term, continuous licence management. Details of the preparation required for the process, the above steps, and ongoing licence management activities in GÉANT are provided in the following sections. (For further information about the four steps, see GÉANT's Open Source Licensing and Compliance training [OSLC_Training].)

Preparation

- Decide on the software name, **grouping of subprojects** and use of available contributions.
- New projects might require a proof of concept or **prototype** to identify and validate key components.
- Gather **preexisting information** and documentation.
- Consolidate the project's components in repositories into a **single project** or clarify their relationships if it is more advantageous for them to remain separate.
- Make sure your software is on GÉANT GitLab [GN_GitLab] or GitHub [GitHub].
- Register the software project in the GÉANT Software Catalogue [GN_SC].
- Internally address authorship and copyright matters.

All components of closely interconnected software development should reside in one project repository, preferably GÉANT GitLab. Although some developers may choose GitHub or opt to mirror their GitLab project on GitHub to obtain permanent URLs for accessing the latest release and its assets, such functionality has been available in GitLab since version 14.9 (GitLab_ReleaseFields). Furthermore, permanent links to assets from specific releases, including those from private releases, have been accessible since GitLab version 15.9. Users can access private release assets using a Personal Access Token. Nevertheless, for certain users, the dual use of both GitLab and GitHub projects remains a viable option. In this approach, the actual development work and intricate pipelines are concealed within the more private space of GÉANT GitLab, with only the final outcome visible on GitHub for public viewing.

The software may include non-original artefacts and assets or those with different licences. These assets, which may not be easily detected with SCA tools, should be documented with their origin, copyright and licence as soon as they are added to the project. The methods for accomplishing this are detailed in Section 2.10 Licences and Tracking of Documentation, Data and Other Works. Failing to document them promptly can complicate their identification and tracking in the future.

One or Several Projects?

When handling multiple projects, it is crucial to determine and specify which dependencies should be incorporated into the SCA analysis. This decision may also depend on the relationship between components and their respective responsibilities. For example, whether one project serves as a subproject managed by the same team or may be intended to function as a module within a larger project overseen by different developers. If so, there may be a need to comprehensively analyse both projects, including their dependencies and, potentially, their source code, even if it is kept in separate repositories.

The rationale for this extended analysis is twofold. In the first scenario, by undertaking an integrated analysis of the larger project and its subprojects, a more comprehensive and holistic understanding of the whole project and its constituent parts can be achieved. In the second scenario, where the main responsibilities and governance lie with the larger project to which the developers contribute with their part, an integrated view of the entire broader undertaking is essential for the contributing team, as they should know whether the licence governance of the larger project is solid. This is particularly significant since although the wider audience and possible suitor will perceive the contributed component or extension as part of someone else's project, that does not fully protect contributors from a potential licensing dispute. To reassure the contributor, it is prudent to analyse the larger work in the same systematic manner as the contributed part.

Developers should indicate to the licensing team that one of the above situations is the case and which other components they think should be included in the analysis. This may be further facilitated through the use of adequate settings within the project's build tool configuration. For example, dependencies with Maven's default scope declaration "compile" are available in all build tasks and propagated to all dependent projects, when their dependencies are transitively included in the Mend analysis. Dependencies with "provided", "runtime" and "test" scopes should be provided by the execution environment, at runtime, or during testing. They are not considered an integral part of the project but as external libraries that need to be used and are, therefore, not transitively analysed.

The licensing team can also provide ideas about grouping features or products within the project and branding for the software. This will be done strictly from a practical technical and licensing perspective. If you need a more authoritative consultation on the product or service that you are developing, please contact the GÉANT Marcomms Team at marcomms@geant.org.

Information Gathering and Documenting

Document copyrights, findings about licences, and decisions:

- **Note the current licence** of your product (entire bundle of created components) or project (one program or standalone component), if set.
- **Request** software composition analysis (**SCA**) **and** software licence analysis (**SLA**) **through GÉANT Jira** [Jira_RSWR].
- Scan the project codebase, producing an **inventory of used components** and their licences **with the SCA service**.
- **Interpret SCA outputs**.
- Additionally, analyse and **document dependencies** and licences **with the SLA service**.
- As a part of (or based on the outputs of) SCA, SLA and, potentially, dedicated security and vulnerability reviews of your software, **review and update the inventory of used components**. Identify:

    - Vulnerable open source components that should be removed or replaced.
    - Outdated open source libraries that should be updated.
    - Confirmed licences of used components (in-licences).
- The above review may also include **clarifying these ambiguities** or doubts:
    - A tool may not be able to properly identify a licence – some licences are reported as suspect or ambiguous.
    - Information about the applied licence may be false, unclear or contradictory.
    - Some licences may be recognised under several names.
    - Some (permissive) licences (BSD, Artistic, etc.) have unnumbered variants or are sometimes edited by authors.
    - Applicability of "or later" may be unclear or even wrongly declared by editing the original licence text.
- **Document gathered analysis and review information** through reports, UI and data exports.
- **Consult on the findings** with the licensing team IPR Coordinator, who will assist with determining which licence shall be applied.
- The **response will be sent by the IPR Coordinator** with guidance regarding the applicable licences.
- **Select an appropriate licence** – There is a dedicated overview document, *Important licences for licence selection* [Wiki_ImportantLicences], about OSS licences and their relationships. Reading it helps improve awareness of OSS licences and selection, but the final decision must be made with the IPR Coordinator after the SCA scan.
- **Document decisions that were made** – Some licences may be additionally refined during remediation and decision-making, but you should also record licensing selection arguments, detected issues and how they were (or will be) addressed. The IPR Coordinator's arguments and clearance should also be recorded.

For additional details about the licence selection process and related recommendations, refer to the *OSS licences and licence selection* guide [Wiki_OSSL&LS] and *OSS Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* white paper [Wiki_OSSLWP].

Remediation

During this step, the goal is to identify and resolve licence conflicts and obligations. Resolving these conflicts may involve trivial or highly complex actions, such as removing or replacing dependencies, developing substitutes, or refactoring existing code.

In some cases, additional remediation after the repeated SCA scan might be necessary. The process may include one or several of the following:

- **Choose a product/project licence** (subsuming licence) compatible with key dependencies.
- **Perform initial easy-to-achieve improvements**:
    - Remedy vulnerable open source components.
    - Update outdated open source libraries (where possible).
    - Ask component authors to clarify their licence or to relicense.
    - Pay for the required proprietarily licensed software.
    - Choose among dual licences of components.
- **Identify remaining incompatible licences**.
- **Decide what to do** with components that use these licences:

  - ○ Remove (component and corresponding functionality) if not necessary.
  - ○ Replace with an existing equivalent.
  - ○ Move to server-side (central service).
  - ○ Write your replacement.
- **Accept some risks**.
- **Internally document dependencies**, their licences, responsibilities, **decisions made and performed remediations**.
- If needed, repeat the SCA and the above-listed steps.

Creating Licence-Related Artefacts

A series of steps needs to be performed to ensure OSS licence compliance. Some details related to the key artefacts listed below are included in Section 3 Complying with a Selected Licence.

- **Provide a LICENSE file**.
  - ○ Clearly state the chosen OSS licence for the project by including a suitable LICENSE file in the root directory of the project repository.
  - ○ Ensure the licence text precisely aligns with its official text. Some licences require the inclusion of copyright information.
  - ○ When a LICENSE file is present, header comments in source code files (with licence and copyright information and disclaimers) are unnecessary, except in special circumstances.
- **Declare the licence in project documentation, website and repository UI**.
  - ○ Review and update project documentation to reflect the chosen OSS licence.
  - ○ Utilise any available repository UI features to declare the project's licence.
- **Declare the project's licence in the GÉANT Software Catalogue and IP Register**.
  - ○ The Software Catalogue has introduced a feature to declare the licence on the project home page.
  - ○ The IPR Coordinator maintains the GÉANT IP Register.
- **Provide a copyright notice**.
  - ○ Add a copyright notice for the project work in a COPYRIGHT file.
  - ○ Include copyright information for components developed by others in compliance with their licence requirements by including each component's name, year and copyright holder.
- **Produce a README file** containing licence and copyright information.
  - ○ Include project licensing information in the README file, detailing the licences for used components where necessary.
  - ○ Provide instructions for access to the full licence text if not provided in the LICENSE file.
  - ○ Briefly explain the implications of the licence for both users and contributors.
  - ○ If licensing badges are available, add a licensing badge to the README file to visually communicate the project's licensing information and make it easily recognisable.
- **Document code modifications** as required for the licence.
  - ○ Providing a history of changes may be required by the applied licence.
  - ○ If the modified software has a CHANGELOG file or similar, extend it with a description of changes using the same format.
  - ○ To learn whether documenting modifications to code in a CHANGELOG file or elsewhere is required, check the licence text or licence summaries such as those in *Important licences for licence selection* [Wiki_ImportantLicences].

- **Declare the use of licence options** if available for the chosen licence.
  - Some licences provide options that should be explicitly stated if they are applied.
  - Options may include accepting later or future versions of the licence or relicensing to specific licences endorsed by the original licence.
- **Document dependencies**, their licences, notices and copyright information.
  - Document the licences of directly included dependencies in a dedicated dependencies file or within the project documentation.
  - Include copyright information for these dependencies and links to official licence texts.
  - For source code components in subfolders, store their licences, copyright and notices files there.
- **Document licence adherence, contribution and updating**.
  - Provide clear guidance for users and contributors on adhering to licensing requirements in a README, CONTRIBUTING or CODE_OF_CONDUCT file. Examples of this are the guidelines provided by FileSender [FileSender_Contrib] and Atom [Atom_Contrib].
  - Provide contribution and copyright instructions and rules, even if external contributors are not expected.
- **Prepare and apply a Contributor License Agreement** (CLA) if necessary.
  - If the chosen licence necessitates a CLA, establish it to define terms for contributions and ensure understanding and adherence.
  - For some licences, suitable CLA forms are available, and the appropriate one needs to be selected.
  - Clearly communicate (in a README or CONTRIBUTING file) the process for contributors to sign CLAs, ensuring legal clarity for all involved parties.
  - A CLA is placed in a file named CONTRIBUTOR_LICENSE_AGREEMENT, CLA or as a part of the broader contribution guidelines in the CONTRIBUTING file.
- **Establish a licence notification mechanism** – Implement a notification mechanism to alert contributors and users about the project's licensing terms and their updates. This can include prompts during the build process, clear notifications in the project repository, use of the project's general notification channels, and providing licence and copyright information through the application UI.

Continuous Licence Management

The goal of long-term, continuous licence management is to **integrate licence management into the regular software development lifecycle**.

- **Integrate licence-related checks into the build process**.
  - Incorporate SCA and other licence-related tools into the build process. This can be at least partially automated by seamlessly integrating dependencies and licence checks into CI/CD toolchains. Use the Mend service provided by GÉANT or other available tools [Wiki_OtherSCATools] that identify dependencies and their licences or verify licence compliance. This ensures that accidental licence violations are caught early.
  - Some tools such as License Maven Plugin [LMP] can generate a file with lists of dependencies and their licences, download dependencies' licence files, check, update or remove licence headers in source files and update (or create) the main project licence file.
  - Verify and scrutinise the outputs of employed tools, bearing in mind that they are not foolproof.
- **Establish continuous monitoring and compliance checks**.
  - Stay updated on changes to the chosen OSS licence.
  - Review the potential impact of any licence updates on your project.

  - Establish processes for continuous compliance checks to ensure that licensing obligations are consistently met by **repeating SCA and SLA as needed** for new dependencies or licences.
- **Perform regular audits of licence-related artefacts**.
  - Conduct regular audits of the project's licence-related artefacts to ensure they remain accurate and up to date.
  - Promptly address any discrepancies to uphold legal clarity and compliance. Any delay is likely to complicate their resolution.
- **Seek legal advice when necessary** – For complex licensing situations, seek advice from the IPR Coordinator, who can help ensure proper interpretation and compliance.

GÉANT software best practice *BP-B.6: Manage sideground IPR* [GN_BP_B6] recommends dealing early with the preexisting and external IP and repeating the process periodically.

2.10 Licences and Tracking of Documentation, Data and Other Works

Software-related artefacts and assets distributed with the software or stored in its source code repository typically adhere to the same open source licence. These include data, technical documentation, configurations and user manuals. For separate tutorials, presentations, training and promotional materials, it is advisable to use the Creative Commons Attribution (CC BY) or Attribution-NonCommercial (CC BY-NC) licence. Another noteworthy licence is the GNU Free Documentation License (GFDL). While data is occasionally licensed under OSS licences, datasets more commonly use licences formulated by Creative Commons and Open Data Commons.

Software should acknowledge its use of external data by clearly documenting and attributing data sources in its documentation or within the software. This acknowledgement should ensure transparency and adherence to licensing or usage terms linked to the external data. Consequently, the software's licence may be impacted if the data comes with specific licensing requirements or restrictions. In such cases, the software must adhere to the terms of the external data licence. This is particularly relevant if data obtained from another source is hardcoded in software, integrated into software data structures, part of its knowledge base, or incorporated into software configuration or database bootstrap scripts. This applies to all data provided with the software, used to configure settings or define parameters, or essential for the software's operation. Such data should be listed among the software's references and included in the software's licensing analysis. If the data used in this way is proprietary or licensed under an open data or OSS licence, compliance with licence requirements is imperative, and the used data should be mentioned, at least, in the NOTICE file. However, if the data is reference or lookup information in public and widespread use, such as a list of country codes from the International Organisation for Standardisation, it should be acknowledged in software documentation and project artefacts but typically does not need inclusion in the analysis of licences. Even if the use of such data is not explicitly credited, its presence and source should be mentioned in the documentation to explain how this information can be updated.

The above also applies to data contained in external code libraries or modules.

If the data is dynamically fetched from external services and APIs during software initialisation or used regularly as contextual or supporting information, it must be prominently referenced in the README or NOTICE file and project documentation. Examples include maps, environmental and sensory information and the presentation of data from external sources. Furthermore, software may persist, aggregate, or otherwise process data obtained from its users and other services. This includes user-created or shared collaborative content, usage information, logs or data items harvested from other services, personal data from authentication services, information about network resources, topologies or traffic, and datasets for training machine learning models. Software documentation should state the use of such data by the application and provide instructions for administrators or users on how to subscribe to external sources or access them, as registration with third-party services is often necessary. Ideally, the software should allow integration with multiple alternative services according to customer preferences, thereby decoupling the software from specific data and services. Users must be informed that they can opt for various data sources.

Processing of external or user-created data may require explicit user consent, be allowed by the terms of use for the service from which data is coming, or be subject to arrangements between the provider or controller of the system based on the software and those who manage the external source. These arrangements are not the primary concern for software developers and they do not affect software licensing. Still, they should be reasonably achievable with the software. Developers should ensure that software and data are secure, design software for personal data protection, and provide features supporting data-related arrangements, such as obtaining user consent, cookies management, and the display of the privacy notice, terms of use, service policy or data retention policy. On the other hand, virtually all OSS licences include disclaimers of warranties and liability, so software authors cannot be legally liable for malfunctions, damages or misuse suffered or caused by users of OSS software. GÉANT offers security-focused code reviews using automated code analysis and expert assessments [Wiki_SWReviews], coupled with related training [Wiki_SCT]. This is complemented by infrastructure-level support from GÉANT Security [GN_Security].

The use or modification of some other types of externally developed work can significantly impact software licensing, especially if this work includes elements such as database models, architectural designs, development and execution frameworks, and code generated by tools. If the work is under a specific licence, the software incorporating it must adhere to the licence terms, considering requirements such as attribution, restrictions on commercial use, or obligations for derivative works. Furthermore, if the software integrates an external database model, framework, or generated content in such a way that the new work is extensively influenced or permeated by the used prior work, the licensing terms of that work may extend to the entire software project. This is especially true if copyleft OSS licences, and non-OSS licences such as Creative Commons licences with NC (NonCommercial), ND (NoDerivatives) and SA (ShareAlike) clauses, are applied. For instance, if development is based on someone else's database model, the used model will likely need modifications, extensions and optimisations, making the allowance for modifications (derivatives) crucial. Therefore, it is essential to review the licences and terms of any such work before significant development begins. These works typically come with copyrights, which should be documented (e.g., in a NOTICE file) and directly included in the code repository, particularly if original or modified artefacts are present, preferably in a dedicated folder.

Given that many of the above-described works are unlikely to be detected with SCA tools, clear documentation and communication of their use and licences in software documentation are imperative as soon as they are adopted. This also applies to non-original software-related artefacts (assets, configuration files, scripts, technical documentation and user guides). If they are distributed with software, they should be kept in its source code repository and under the same licence as the piece of software they originally came with. If many such works are included, they should be annotated with easy-to-aggregate and searchable provenance and licensing details in the software repository, stated in the project's Software Bill of Materials (SBOM) [Mend_SBOM], or marked with easily extractable metadata and comments. The included details should encompass the place of use, origin, copyright and licence. Omitting to do so upon including these assets can complicate their identification and tracing at a later stage. This particularly applies to non-original graphical or UI assets, such as images, vector graphics, JavaScript code or GUI layouts that do not directly belong to external components and are not placed within them.

The original assets distributed with software are best kept under the same licence when they do not have to be individually tracked.

3 Complying with a Selected Licence

This is the primary section for developers after selecting a licence. It facilitates preparatory work and internal compliance checks before reviewing licence adherence with the licensing team, and provides essential information for developers seeking to address licensing issues independently. Additional templates and links to example files with GÉANT-approved content will be provided as they become available from software projects.

Developers are obligated to adhere to the requirements of the chosen OSS licence and ensure licence compatibility. Failure to comply constitutes a breach, potentially leading to legal challenges and significant financial loss.

Even if the software and its dependencies are aligned with the chosen licence, this **licence must be clearly stated** in the documentation, including the README file. Most licences require that a copy of the licence be included, which is typically placed in the LICENSE file in the root folder. Some licences may require only their name or URL in documentation, but having the licence text in a dedicated file is standard. A clear and explicit statement of the specific licensing is necessary, beyond just including the licence text. For instance, the GPL family of licences articulates this requirement in the "How to Apply These Terms to Your New Programs" section at the end of their text. Simply including the licence text in the LICENSE file is insufficient; a distinct statement affirming the applied licence is required.

While licence and copyright information usually do not need to be in every source file header, the applied licence and funding should be clearly declared and strategically placed so that anyone interested could easily find and see them. If the software has a website or webpage, the licences should also be stated there. The source code repository used may also have a mechanism for specifying the used licence. GitHub and GitLab provide features that allow declaration of the licence by using the repository's user interface. GitHub is able to automatically recognise the used licence from the LICENSE file in the root folder.

Software authors may specify alternative licences for a project, offering it under a dual licence or multiple licences. However, when such software is used in another project, only one of its available licences is applied when considering compatibility with the licence of the main project. Also, the component's licence must be compatible with the licence (or all offered licences) of the main project.

The minimal set of typical documenting files in a software project usually includes README for project information, LICENSE for licensing details and CONTRIBUTING for contribution guidelines. The applied licence may require the inclusion of CHANGELOG or CHANGES for tracking project changes. These files may optionally use markdown when their names end with the .md extension. If so, you can edit them using an online markdown editor or checker such as Dillinger [Dillinger] or StackEdit [StackEdit].

The following sections address these compliance requirements in more detail, covering:

- Copyright and licence notices in source code.
- Placement of information.
- Project licence options.
- Complying with licences of used code.
- README file.
- COPYRIGHT file.

- Acknowledgements in AUTHORS, NOTICE and README files.
- CHANGELOG file.

3.1 Copyright and Licence Notices in Source Code

A simple copyright line and a licence indicator may be placed in the header of every source file, as shown in the example below; replace the text in angle brackets with the appropriate information. While not mandatory, these notices can be useful if you anticipate that individual project files may be accessed, reused or modified independently, and you are concerned that users or modifiers might overlook or not receive the project-level copyright and licence information.

```
/* Copyright (c) <Year> <Copyright Holder>
 * This code is released under <Licence Name>. See file LICENSE or
visit <Licence URL>
 * for full licence details.
 */
```

3.2 Placement of Information

The README file is the primary starting point and is, therefore, emphasised by GitHub. It includes a description of the software along with licensing information. Hyperlinks in README files can connect users to external resources such as the full licence text, project website or funding details. In online platforms such as GitLab, GitHub or the GÉANT Software Catalogue, relevant information may be linked in the project's profile or description.

It is essential to create or review and update the copyright statement. Typically, a COPYRIGHT file should indicate that the software is copyrighted by GÉANT, NRENs or other organisations. The copyright holder should also be specified within the README file, and sometimes within the LICENSE file if there is a placeholder for this in the licence text or it is otherwise required by the licence, or in copyright notices within headers of source files.

Contributors are acknowledged in the AUTHORS file, providing a list of individuals or entities who have contributed to the software and who may be grouped by type (e.g., code, documentation, testing). The source code repository, if regularly used, tracks contributions and may provide information for the AUTHORS file. COPYRIGHT, NOTICE or CONTRIBUTORS files sometimes acknowledge the project team and individual contributors and authors. It is better to adhere to the AUTHORS file, unless this information is already provided elsewhere in the existing software.

Funding information may be mentioned in the README, COPYRIGHT and AUTHORS files. It may include logos or links to websites of sponsors or grant providers. It should be transparently disclosed if funding influences project direction or goals, which is certainly the case for the developments in GÉANT. The information about EC funding is obligatory if the code was developed with the money provided by the EC. Acknowledgement of funders might also be included in the detailed documentation.

Contribution guidelines are documented in the README or CONTRIBUTING file. These guidelines specify how new contributors can engage with the project, submit changes (including committing, branching, testing and releasing), and adhere to the project's coding standards and licensing requirements. They may link to templates, coding style profiles or guidelines and specific instructions for different kinds of contributions.

Software, its licence, associated background IP and sideground IP should be recorded in the GÉANT IP Register. This is done by the licensing team and the IPR Coordinator.

Modules located in subfolders may have their own licences. They should include a separate LICENSE file in each subfolder containing a module with a different licence from that of the main project and provide any necessary attribution or copyright notices for that module. It is crucial to provide this information for subprojects or folders if it differs from what is stated in the main project or the root folder. If their other important information is also different, it should be provided in their respective folders in the same manner as for the main project.

3.3 Project Licence Options

The applied licence may offer the option to apply additional conditions or permissions. These additional statements help clarify how others can use, modify and distribute the software. If the licence used offers some licensing options, these options and their declaration are explained in the licence text. Some common software licence options that code owners may explicitly state include:

- **Permitting users to choose between the original licence version or any later version** – One option is to permit users to choose between the original licence version or any later version approved by the licensor. Allowing such multiversioning relicensing can be interpreted as licence-endorsed open-ended multilicensing. For example, a clear statement indicating the code's specific GPL licence is required and is typically found in the README file. The GPL phrase "This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version" allows the choice of the referenced version or any later version. Simplifying it to "This software is licensed under GNU General Public License version 3 or any later version" is acceptable. If the licence notice statement ends with "version 3" or "version 3 only", then only GPL 3.0 can be applied. If the version number is not mentioned, the recipient can choose any published version of the licence. The notice can even specify that a proxy can decide which future versions of the GNU General Public License can be used, but this option is rarely used. For LGPL, this notice should mention "GNU Lesser General Public License" instead of "GNU General Public License".

- **Relicensing under a different licence** – Some licences allow relicensing of the software under a different licence endorsed by the original licence or even a licence chosen by the licensor. For EPL 2.0 licensed software, its notice or file headers should state "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/." With this, software is to be used with EPL 2.0 only. But a Secondary License can be introduced, where recipients can choose to comply with either the EPL or the Secondary License. Adding the following text to README introduces GPL 2.0 with Classpath-exception as the Secondary License: "This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: The GNU General Public License (GPL), version 2 with Classpath-exception." Any other licence that grants the recipients rights that are at least as broad as those granted under the EPL can be declared as a Secondary License. Adding the latter clause is effectively relicensing, and all copyright holders must agree to the licence change. When MPL 2.0 is used, the notice must state "This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at https://mozilla.org/MPL/2.0/." By default, MPL 2.0 offers most GPL licences as Secondary Licenses. However, the licensor may prohibit this by stating "This Source Code Form is 'Incompatible With Secondary Licenses,' as defined by the Mozilla Public License, v. 2.0."

- **Extending certain rights beyond standard terms** – In some cases, it is possible to extend certain rights beyond the standard terms of the original licence, such as allowing commercial use without open sourcing modifications, providing a patent grant and permitting the distribution of proprietary versions. Any extension of rights should be clearly articulated and explicitly state that these extensions work in conjunction with the original licence without replacing or modifying its conditions. For example, the notice suggested for Apache 2.0 states that software is on an “AS IS” basis. However, additional warranties can be agreed in writing.

- **Placing limitations on certain uses or modifications** – Some licences may allow or not prohibit the imposition of additional conditions such as restricting non-commercial use, requiring the availability of modified source code and ensuring compatibility with the original version. It is crucial to note that adding limitations should be done carefully and explicitly and ensuring that these limitations work alongside the original licence. The original Apache 2.0 notice opens up a space for providing additional restricting clauses, but licensors should avoid introducing restrictions that contradict or undermine the fundamental principles of open source licensing, such as the ability for users to freely use, modify and distribute the software.

- **Choosing the jurisdiction under which the licence is governed** – With EUPL, the licensor can choose the jurisdiction under which the licence is governed. This allows them to specify the legal framework to be applied, which can be helpful when dealing with legal matters.

It should be noted that SCA tools cannot interpret most options, nor additional rights or conditions introduced in free text.

3.4 Complying with Licences of Used Code

In addition to meeting your project’s licence requirements, it is imperative to address the obligations imposed by licences governing dependencies or reused code, encompassing associated copyright and patent rules. This necessitates continuous attention throughout the development process. The essential actions are as follows:

- Extend README, COPYRIGHT and NOTICE files to explicitly declare and credit the use of dependencies or other utilised code, clearly stating the application of their licence options.
- Retain all preexisting licence- and copyright-related files and notices to ensure comprehensive documentation and compliance with their respective licences.
- Properly attribute and document any modifications made to reused code, updating the history of modifications and the list of contributors as necessary.
- Stay informed about licence updates and changes for used code, as they can impact compliance requirements and necessitate adjustments.

Eclipse- or Mozilla-licensed dependencies (direct or transitive) require explicit reference within the project’s documentation.

Using code under Apache 2.0 with a different licence for your project involves specific obligations:

- Include the original copyright notice.
- Provide a copy of the Apache licence.
- Describe any significant changes made to the original code.
- Maintain a NOTICE file with attribution notes (either the original or a new one with your additions).

3.5 README File

The README file should include basic information about the software. It should clearly and concisely state the licence and copyright, typically in one short line each. It should also state the origin of the development, give credit to GÉANT and refer to COPYRIGHT and LICENSE files for further details.

The README file should provide guidance and instructions related to the software, covering:

- Purpose or intent, which authors may sometimes omit as it may appear self-evident to them.
- Scope, supported settings, requirements or constraints of the application which may not be apparent to a reader encountering the project on the internet.
- Installation and configuration.
- Usage.
- Roadmap and known issues.
- Community contributions.
- Acknowledgements, dependencies and used tools.
- Software licence and licences of differently licensed components.

A useful README template is available at Make a README [Make a README]. After providing a sample markdown, it offers a more detailed section titled “Suggestions for a good README”.

3.6 COPYRIGHT File

Software is protected by copyright law. Various OSS licences have different requirements under which software developers grant other users specific rights while retaining copyright. Developers must clearly indicate copyright in addition to the licence.

Copyright management within the project is expressed through the copyright statement. Copyright is sometimes embedded in the LICENSE file. For some short licences such as MIT, BSD and their variants, the copyright notice is an integral part of the LICENSE file and is provided at the beginning of the file, although a place for this may be reserved elsewhere in the licence text. In some cases, the copyright information is placed in a separate COPYRIGHT file to maintain the LICENSE file’s integrity.

Not all software developed within the GÉANT project correctly indicates copyright. This issue needs addressing during work on software licensing. While core copyright information can be provided on the project website, in the project’s LICENSE file (if this is a convention for the applied licence) and in the README file, creating a dedicated COPYRIGHT file with more comprehensive details is advisable for software created or modified within GÉANT.

The COPYRIGHT file should be placed in the project root folder or in the component’s root folder if it differs from the rest of the project. It should include a reference to the GÉANT project and specify the years in which the work was carried out. It should state that the software is copyrighted by GÉANT and other contributing organisations. In addition to copyrighting new code to GÉANT, indicate whether any code within was reused, adapted or relied upon from elsewhere. If the work includes contributions from NRENs that were created independently of GÉANT or direct insertions or adaptations of code from other projects, copyrights of the original copyright holders should be indicated or preserved if they were already present.

If you are creating a COPYRIGHT file in markdown format, which is not necessary if the copyright information is short and simple, you may use Dillinger, StackEdit or another online markdown editor or checker.

An example of a COPYRIGHT file with a disclaimer that can be tailored to your needs is provided here:

> Copyright (c) 2023-2024 GÉANT Association on behalf of the GN5-1 project
>
> THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE

> LIABLE FOR ANY CLAIM, DAMAGES, OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF, OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
>
> Project IP was generated and/or developed during the GÉANT GN5-1 Project, a project that has received funding from the Horizon Europe research and innovation programme under Grant Agreement No. 101100680 (GN5-1).
>
> The Partner that developed the Project IP remains the sole owner of the Project IP developed during the Project. However, GÉANT Vereniging (Association), registered with the Chamber of Commerce in Amsterdam with registration number 40535155, and operating in the UK as a branch of GÉANT Vereniging (Association), registered office: Hoekenrode 3, 1102BR Amsterdam, The Netherlands, UK branch address: City House, 126-130 Hills Road, Cambridge CB2 1PQ, UK, has the free-of-charge, non-exclusive, perpetual, irrevocable, worldwide right to exploit the Project IP, including any Background and Sideground IP, and any IPR attached to the respective IP, including the right to sublicense the Project IP through multiple levels of sublicences and/or other licensing arrangements and to release to third parties the Project IP, including Public Disclosure in accordance with this IPR Policy.

The COPYRIGHT file commences with a copyright statement. The first line contains “Copyright (c) <Year> <Copyright Holder>”, where <Year> indicates the year or range of years when the copyright for the software was established or updated, and <Copyright Holder> is the name or organisation. In the case of GÉANT Association, include a reference to the project phase during which the licensed software was developed. If the software was developed during several iterations of the project, all of them shall be mentioned, for example: “GÉANT Association on behalf of GN4-2, GN4-3 and GN5-1)”.

If there are additional agreed copyright statements confirmed by the IPR Coordinator, such as those independently developed by other partners such as NRENs, they should be added in lines following the GÉANT Association copyright.

The above template includes a disclaimer of warranty and a limitation of liability clause. If the software is not open source and you retain all rights provided by copyright law, you may use “All rights reserved” instead. Even if you skip this statement, the work is automatically protected by copyright; another person cannot reproduce, distribute or adapt any part of the work without your permission.

Whenever your code was developed with EC funding, the EU emblem shall be included. The EU emblem should be added to information about funding whenever feasible (in README.md, project documentation, the application’s “About” page, screen or window, etc.), following the guidelines from the GÉANT IPR Policy [GN_IPRPolicy], which may be summarised as follows:

- The minimum emblem height is 1 cm.
- “European Union” shall be used in conjunction with the name of the programme or fund and spelt out in full.
- Use one of the listed non-serif fonts without any font effects.
- The text should not interfere with the emblem in any way.
- The text should be proportionate to the size of the emblem.
- The text should be in the same blue colour as the EU flag, black or white, depending on the background.
- Along with the EU emblem, information about the funding shall be provided.

Figure 3.1 presents an example of the use of the EU emblem with the appropriate text about GÉANT and its funding:

GN5-1 project is funded from the Horizon Europe research and innovation programme under Grant Agreement No. 101100680 (GN5-1)

Figure 3.1: The EU emblem with text about GÉANT and its funding

(Use the above image or download and adapt and resize a hi-res image from [EC_Downloads].)

For the prior projects, the text is:

- GN4-3 project is funded from the Horizon 2020 research and innovation programme under Grant Agreement No. 856726
- GN4-2 project is funded from the Horizon 2020 research and innovation programme under Grant Agreement No. 731122
- GN4-1 project is funded from the Horizon 2020 research and innovation programme under Grant Agreement No. 691567

Contact the IPR Coordinator if you have any questions about specific copyright. Licence text must be prepared for every software you are developing, and the selected licence will depend on the licences used for components, while the copyright statement might vary depending on the institutions involved.

3.7 Acknowledgements in AUTHORS, NOTICE and README Files

An open source project typically results from the collective efforts of various developers or teams who have made valuable contributions. It is essential to acknowledge their contributions and efforts.

An AUTHORS (or CONTRIBUTORS) file is a straightforward way to acknowledge and credit individual contributors and reference the GÉANT Work Package or Task of the project team. This file is placed in the project's source code root.

This file can also acknowledge funding as required by the contract or programme. If detailed information about individual contributors is not provided, funding credits can be moved to the COPYRIGHT file. Some of the information from COPYRIGHT and AUTHORS may also be summarised in the README file and on the project's standalone website, as instructed by the GÉANT Marcomms Team. However, there is no need to provide funding credits for software descriptions within GÉANT-branded websites or within its internal collaboration platforms.

Below is a markdown template for an AUTHORS file:

```
<Project Name> has been created by the <GNx-y> WP<x> Task <y> <Recognisable Task Name>.

## Developers

- [<Author 1 Name>] (<Author 1 Contact>): <Author 1 Role>

- [<Author 2 Name>] (<Author 2 Contact>): <Author 2 Role>

## Relevant Contributors
```

- [<Contributor 1 Name>] (<Contributor 1 Contact>): <Contributor 1 Role>

- [<Contributor 2 Name>] (<Contributor 2 Contact>): <Contributor 2 Role>

## Funding

- [GN5-1] (https://geant.org/gn5-1/) is funded from the European Union's GN5 FPA partnership framework agreement, Horizon Europe research and innovation programme under Grant Agreement No. 101100680 (GN5-1).

- [GN4-3] (https://geant.org/projects/) is funded from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726.

- [GN4-2] (https://geant.org/projects/) is funded from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122.

- [GN4-1] (https://geant.org/projects/) is funded from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 691567.

<Project Name> should be replaced with the name of the software project. If you want to indicate the project's GÉANT origin, the file may optionally start with the line provided at the beginning of the template.

When individuals are listed, their names may be accompanied by optional email addresses or other relevant contact information (e.g., a reference to a catalogue or repository, or ORCID iD) and descriptions of their notable contributions or roles in the project, if desired. These descriptions should be brief, such as "Implemented the core functionality of the project" and "Designed the project's user interface". This not only gives credit to those who have worked on the project but also helps other contributors find each other, which can foster collaboration. The contributors section may also include individuals involved in related activities who did not directly author the code, if their contributions were significant. Developers and contributors should be clearly distinguished from the holder(s) of copyright stated in the COPYRIGHT file. Their lines may also start with time period references, such as "2021:" and "2020 – today:".

Make sure to keep this section up to date as new contributors join the project or make significant contributions.

If your project is part of a larger ecosystem or relies on third-party libraries, it is good practice to acknowledge these dependencies in your documentation as well, either in the README file or in a separate NOTICE file. Also, the licence that you use may require that you create a NOTICE file, or you may be modifying a project with an existing one, into which you should add your entries. Mention all the libraries and frameworks that your project uses and provide links to their respective websites or repositories. README, COPYRIGHT and NOTICE files of unmodified components should remain undisturbed.

Below is a markdown template for a NOTICE file listing relevant third-party components, their licences, URLs and copyright information:

# <Project Name> – NOTICE File

This project includes third-party software components subject to open source licences. Please read and comply with the licensing terms and attribution requirements of these components, as listed:

1. <Dependency 1 Name>

   - Version: <Dependency 1 Version Number>

   - URL: <Dependency 1 URL>

   - Licence: <Dependency 1 Licence>

   - Copyright (c) <Dependency 1 Year Range> <Dependency 1 Copyright Holder>

2. <Dependency 2 Name>
   - Version: <Dependency 2 Version Number>
   - URL: <Dependency 2 URL>
   - Licence: <Dependency 2 Licence>
   - Copyright (c) <Dependency 2 Year Range> <Dependency 2 Copyright Holder>

3. <Dependency 3 Name>
   - Version: <Dependency 3 Version Number>
   - URL: <Dependency 3 URL>
   - Licence: <Dependency 3 Licence>
   - Copyright (c) < Dependency 3 Year Range> < Dependency 3 Copyright Holder>

We thank all open source developers who contributed to these dependencies.

The actual content and format of a NOTICE file may vary depending on the specific project and its dependencies' licensing requirements. Some licences require preserving the content from existing NOTICE files of dependencies. If a NOTICE file is unnecessary, you may opt to provide this information in a dedicated section of the README file in a more concise form, also including information about the tools used, as given in this markdown example:

```
## Dependencies

This project relies on the following third-party libraries and tools:

- [Library A] (https://librarya.com) – Used for data processing.
- [Framework B] (https://frameworkb.io) – Provides essential functionalities for the project.
- [Tool C] (https://toolc.org) – Assisted in automating tasks.
```

Scanning a project with Mend is useful for checking dependencies to avoid forgetting significant ones. However, transitive dependencies listed in Mend reports should not be included in the above lists.

It is your responsibility to review and comply with the respective licences and attribution requirements for these components, as stated in their respective files or documentation, as the project's dependencies evolve. Acknowledging contributions, dependencies and tools is not only a gesture of appreciation but also a way to foster a collaborative and supportive open source community.

3.8 CHANGELOG File

Some licences require a history of software changes to be maintained and provided. It is commonly in a file named CHANGELOG, CHANGES or HISTORY, placed in the root directory of the project. Some licences have explicit naming requirements or even requirements on the content of entries. The history should provide a chronological summary of new features, significant changes, additions, bug fixes and other modifications, serving as a record of changes made to a software project over the project's releases and time. This helps users and contributors understand what is new and different in each release and understand the evolution of a software project. It also assists with troubleshooting and identifying potential upgrade issues. One common approach is to generate the history of changes by concatenating release notes, the creation of which may also be facilitated with a tool using commit messages.

Entries should be organised chronologically, with the newest changes at the top. Each entry typically corresponds to a single release. Include release or version numbers or tags (e.g., “1.2.3”) and indicate release (or change) dates. For unreleased changes, consider using “Unreleased” as the version until a release occurs. Summarise changes in bullet points or brief paragraphs, optionally grouping them by type (e.g., features, bug fixes, improvements). Alternatively, bullets may have indicative prefixes, such as “Added:”, “Changed:”, “Fixed:”, “Deprecated:”, “Removed:” and “Security:” for security-related changes. Descriptions of changes may reference the relevant commit IDs and code files for detailed information. Highlight important changes by visually emphasising significant updates.

The information from commits into the source code repository should be used to track changes systematically. Having a practice of frequent commits, development in branches, and meaningful, clear and concise commit messages helps in this, but the information from the repository is often too specific. It is also important to tag releases with version numbers for easy identification. Other valuable sources of information about changes, the content of which can be readily transferred to the history, are release notes and lists of features and modifications planned for a release, developer tasks and fixed issues. The history should remain meaningful throughout the software lifecycle, so it is useful to mention why a change is made or include the context or background for the change. Features such as pull requests and code reviews can offer additional context.

Use a consistent and clear format for each entry, such as that shown in the following markdown example:

```
## [1.2.3] – 2024-01-01

### Added
- Newly added feature 1
- Newly added feature 2

### Changed
- Updated existing feature or its implementation

### Fixed
- Bug fix 1
- Bug fix 2
```

Markdown for linking to specific commits or issues related to each change facilitates access to additional information:

```
- New feature 1 (#123)
- Bug fix 1 ([commit hash])
```

The project’s documentation should explain how users and contributors can check the file for release notes.

4 Resources

4.1 Contact

- IPR Coordinator email: iprcoordinator@geant.org
- WP9 Task 2 Open Source and Licence Support (OSLS, software licensing)
  - Slack: **#sw-licences** at https://geant-project.slack.com workspace
  - Email: sw-licences@software.geant.org
- GÉANT Marcomms Team email: marcomms@geant.org

4.2 Training Materials

- Training course: Open Source Licensing and Compliance [OSLC_Training]
- Webinar: License Dependencies Analysis with WhiteSource [LDAwithWS_Webinar]
- Training course: Introduction to Open Source Licensing and Compliance 2023 [IntroOSLC_Training]
- Infoshare: Software Licences Management in GÉANT [SWLMinGN_Infoshare]

4.3 Further Reading

- GÉANT IPR Policy [GN_IPRPolicy]
- *OSS licences and licence selection* [Wiki_OSSL&LS] – an introductory guide
- *OSS Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* [Wiki_OSSLWP] – project-oriented white paper for GÉANT participants. A guide on licence selection with an appendix describing the OSS licences most frequently used by analysed projects
- *Software licence selection and management in GÉANT* [Wiki_SWLS&M] – a maintained online version of this guide.
- *Important licences for licence selection* [Wiki_ImportantLicences] – descriptions and compatibility of most frequently used licences in GÉANT
- *Reference information about OSS licences and tools* [Wiki_OSSL_RefInfo] – a comprehensive catalogue of thematically organised resources and pointers
- GÉANT software best practice *BP-B.6: Manage sideground IPR* [GN_BP_B6]
- OSI Approved Licenses [OSI_Licences]

4.4 Services

- GÉANT Software Licence Management (homepage) [Wiki_SWLM]
- Jira requests for SCA and SLA [Jira_RSWR]
- *Software Reviews* [Wiki_SWReviews] – GÉANT Software Development Support
- GÉANT Mend, with login via GÉANT SSO, special permissions may apply [GN_Mend]

- *Accessing Mend and visibility levels* [Wiki_MendAccess] – visibility of Mend scan results
- *Mend short guide for end users* [Wiki_MendGuide] – also explains the interpretation of Mend reports
- The Risk Report [Mend_TRR] – short Mend report guide for end users
- GÉANT GitLab [GN_GitLab]
- GÉANT Software Catalogue [GN_SC]

References

[Atom_Contrib]	https://github.com/atom/atom/blob/master/CONTRIBUTING.md
[Dillinger]	https://dillinger.io/
[EC_Downloads]	https://ec.europa.eu/regional_policy/information-sources/logo-download-center_en
[EURISE_SQC]	https://technical-reference.readthedocs.io/en/latest/quality/software-checklist.html
[FileSender_Contrib]	https://github.com/filesender/filesender/blob/development/CONTRIBUTE.md
[GitHub]	https://github.com/
[GitLab_ReleaseFields]	https://docs.gitlab.com/ee/user/project/releases/release_fields.html
[GN_Bamboo]	https://bamboo.software.geant.org/
[GN_Bitbucket]	https://bitbucket.software.geant.org/repos?visibility=public
[GN_BP_B6]	https://wiki.geant.org/display/GSD/BP-B.6%3A+Manage+sideground+IPR
[GN_GitLab]	Community Edition instance, hosting most projects: https://gitlab.software.geant.org/public Ultimate Edition, hosting a few selected projects: https://gitlab.geant.org/
[GN_IPRPolicy]	https://resources.geant.org/wp-content/uploads/2022/09/GEANT-_IPR_Policy_2022.pdf
[GN_Mend]	https://app-eu.whitesourcesoftware.com
[GN_Resources_IP]	https://resources.geant.org/publications/intellectual-property/
[GN_SC]	https://sc.geant.org/
[GN_Security]	https://security.geant.org/
[IntroOSLC_Training]	https://e-academy.geant.org/moodle/course/view.php?id=478
[Jira_RSWR]	https://jira.software.geant.org/servicedesk/customer/portal/2/create/55
[JLA]	https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant/jla-find-and-compare-software-licenses
[LDAwithWS_Webinar]	https://e-academy.geant.org/moodle/course/view.php?id=220
[LMP]	https://github.com/mojohaus/license-maven-plugin
[Make_a_README]	https://www.makeareadme.com/
[Mend_SBOM]	https://www.mend.io/blog/guide-to-standard-sbom-formats/
[Mend_SCA]	https://www.mend.io/sca/
[Mend_RSA]	https://docs.mend.io/bundle/sca_user_guide/page/understanding_risk_score_attribution_and_license_analysis.html#Risk-Score-Attribution
[Mend_TRR]	https://docs.mend.io/bundle/sca_user_guide/page/the_risk_report.html
[OSI_Licences]	https://opensource.org/license
[OSLC_Training]	https://e-academy.geant.org/moodle/course/view.php?id=214
[PLM]	https://geantprojects.sharepoint.com/sites/plm
[RedHat_COSP]	https://www.redhat.com/en/resources/open-source-project-health-checklist
[SWLMinGN_Infoshare]	https://wiki.geant.org/pages/viewpage.action?pageId=633276866
[StackEdit]	https://stackedit.io/
[TinyMCE_OSSEC]	https://www.tiny.cloud/software-evaluation-criteria-checklist/
[Wiki_CGSCA]	https://wiki.geant.org/pages/viewpage.action?pageId=599785535
[Wiki_ImportantLicences]	https://wiki.geant.org/display/GSD/Important+licences+for+licence+selection
[Wiki_MendAccess]	https://wiki.geant.org/display/gn51wp9t2/Accessing+Mend+and+visibility+levels
[Wiki_MendAP]	https://wiki.geant.org/pages/viewpage.action?pageId=240844905

[Wiki_MendASB]	https://wiki.geant.org/pages/viewpage.action?pageId=219938818
[Wiki_MendGuide]	https://wiki.geant.org/display/GSD/Mend+short+guide+for+end+users
[Wiki_OSSL_RefInfo]	https://wiki.geant.org/display/GSD/Reference+information+about +OSS+licences+and+tools
[Wiki_OSSL&LS]	https://wiki.geant.org/display/GSD/OSS+licences+and+licence+selection
[Wiki_OSSLWP]	https://wiki.geant.org/pages/viewpage.action?pageId=633275197
[Wiki_OtherSCATools]	https://wiki.geant.org/display/GSD/Reference+information+about +OSS+licences+and+tools#ReferenceinformationaboutOSSlicencesandtools-Othersoftwarecompositionanalysis(SCA,softwareinventory)tools
[Wiki_SCT]	https://wiki.geant.org/display/GSD/Secure+Code+Training
[Wiki_SWLM]	https://wiki.geant.org/display/GSD/Software+Licence+Management
[Wiki_SWLS&M]	https://wiki.geant.org/pages/viewpage.action?pageId=725614690
[Wiki_SWReviews]	https://wiki.geant.org/display/GSD/Software+Reviews

Glossary

AGPL	GNU Affero General Public Licence
API	Application Programming Interface
BSD	Berkeley Source Distribution
CC	Creative Commons
CC BY	Creative Commons Attribution licence
CC BY-NC	Creative Commons Attribution-NonCommercial licence
CI	Continuous Integration
CI/CD	Continuous Integration / Continuous Delivery
CLA	Contributor License Agreement
EC	European Commission
EPL	Eclipse Public License
EU	European Union
EUPL	European Union Public Licence
EURISE	European Research Infrastructure Software Engineers
FAIR	Findability, Accessibility, Interoperability and Reusability
GFDL	GNU Free Documentation License
GPL	GNU General Public License
GUI	Graphical User Interface
ICT	Information and Communication Technology
IP	Intellectual Property
IPR	Intellectual Property Rights
JLA	Joinup Licensing Assistant
MIT	Massachusetts Institute of Technology
MPL	Mozilla Public License
NC	NonCommercial
ND	NoDerivatives
NREN	National Research and Education Network
OSI	Open Source Initiative
OSLS	Open Source and Licence Support
OSS	Open Source Software
PLM	Product Lifecycle Management
R&E	Research and Education
SA	ShareAlike
SBOM	Software Bill of Materials
SCA	Software Composition Analysis
SLA	Software Licence Analysis
UA	Unified Agent
UI	User Interface
WP	Work Package
WP9	Work Package 9 Operations Support
WP9 Task 2	WP9 Task 2 Software Governance and Support