04-08-2025

Software Licence Selection and Management in GÉANT – 2025 Update

Grant Agreement No.:	101194278
Work Package:	WP9
Task Item:	T2
Nature of Document:	Guide
Dissemination Level:	PU
Lead Partner:	UoB/AMRES
Document ID:	GN5-2-25-112DBD

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.

Contents

Figures

1 Introduction

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

This guide is divided into two main parts:

1. The first part, Essential Aspects and Elements of Software Licensing, provides developers with key insights into software licence selection and management, outlining the tasks they must perform, and the procedures required to engage with services that support licensing. This part provides a straightforward overview of the elements necessary for efficient preparation, information gathering, and software licensing.
2. The second part, Complying with a Selected Licence, explains how to implement the selected licence from a developer's perspective, providing instructions that ease the licensing process. It gives in-depth information, along with simple but vital guidance on creating essential licensing artefacts.

This document does not detail individual open source software (OSS) licences, software composition analysis (SCA) tool usage, licence compatibility, or the remediation of licence conflicts. This is intentional, as developers may not need to deal in depth with these complex tasks, because they can rely on support from the software licensing team. When such details are necessary, they are covered in separate guides provided by the software licensing team.

For comprehensive information on licences, compatibility, and licence selection, 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 commonly 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 Open Source Software and Licensing

Key factors related to open source software (OSS) include the following:

- Our work increasingly relies on OSS. Developers, National Research and Education Networks (NRENs), and GÉANT use, adapt, create, and endorse OSS.
- The ICT and R&E communities value OSS for its cost-effectiveness, transparency, collaboration, customisability, vendor independence, longevity, security, educational value, compatibility, ethical and societal values, accessibility, and more (detailed below).
- OSS licences ensure software remains free, prevent appropriation, and reduce the risk of abandonment.
- Declaring a software licence simplifies the selection of code and libraries for use in a project, and facilitates usage, adaptation, and contribution by other developers.
- Licensing is critical when distributing or sharing software, as OSS licences have specific conditions.
- Licence declaration and compliance are essential for legal and usability reasons, increasing transparency, and fostering collaboration.
- Adhering to applicable licences (including those of dependencies) enhances the visibility and credibility of a software project within the wider community.

OSS is extensively used in technology products, services, business, governments, research, and education. It provides and supports affordability, transparency, collaboration, and technical 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. Key benefits include:

1. **Cost-effectiveness** – OSS is often free to use, significantly reducing software acquisition and licensing costs for individuals, businesses, and organisations. This is especially critical for smaller businesses, educational institutions, and governments with budget constraints.
2. **Transparency** – OSS 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 is particularly important for software used in critical applications, such as cybersecurity and healthcare.
3. **Community collaboration** – OSS projects often have large and diverse communities of developers and users who collaborate to improve the software. This leads to rapid bug fixes, updates, and feature enhancements, fostering innovation and knowledge sharing.
4. **Customisation** – Users can modify and customise the code to meet specific needs. This flexibility allows businesses and individuals to adapt software to their unique requirements, offering them a competitive advantage.
5. **Vendor independence** – Unlike proprietary software, where users are often locked into a vendor's ecosystem, OSS offers greater freedom and flexibility, as users have access to the source code and can switch providers or modify software as needed, including integration with other products.
6. **Longevity** – While proprietary software may be discontinued by the vendor, leaving users without support, OSS tends to have a longer lifespan. Communities can continue maintaining and developing the software if the original team slows down or abandons it.
7. **Security** – Although OSS is not immune to vulnerabilities, its transparency allows a global community to audit the code continually. When vulnerabilities are discovered, they can be fixed quickly. Proprietary software security, on the other hand, depends on the vendor's resources and priorities.
8. **Education and learning** – OSS encourages learning and skill development. Students and aspiring developers can study, modify, and contribute to OSS projects, gaining practical experience in real-world software development.
9. **Compatibility** – Open standards and OSS promote compatibility and interoperability between different software, reducing barriers to data exchange and collaboration.
10. **Ethical values** – The OSS movement is rooted in values such as transparency, collaboration, and the belief that software should be a public good. Many individuals and organisations align with these values.
11. **Global accessibility** – OSS is accessible to users worldwide, regardless of location or economic status, promoting digital inclusion and levelling the playing field.

In the GÉANT context, specific requirements and recommendations are guided by its IPR Policy [1] (also see GÉANT Resources – Intellectual Property [2]):

1. **Establish clear naming and branding,** and define how the project should be **packaged into products and repositories**. Ideally, OSS should be hosted in a public versioned repository, with GÉANT GitLab as the preferred platform [3]. The relationship between packages and their components will likely influence the licensing decisions.
2. Every GÉANT software project should select and **apply an OSS licence** that meets the needs of both the development team and the user community.
3. **Start the licensing process early** to streamline licensing and compliance.
4. The **chosen licence must be compatible** with all used components' licences, to eliminate IPR and licensing risks for GÉANT.
5. It is preferable to place the OSS source code in a public and **versioned code repository,** with a **clear licence indication**.
6. **Copyright** information must acknowledge GÉANT's involvement and support, underscoring that the work was conducted within the GÉANT project or received support from it. The authors of the produced software should also be identified.

7. **Assess components and software** by using common software quality and trustworthiness checklists to ensure quality and reliability. Examples include *TinyMCE – Open Source Software Evaluation Checklist* [4], *Red Hat – Checklist for Measuring the Health of an Open Source Project* [5], and *EURISE Network Technical Reference – Software Quality Checklist* [6].
8. **Use GÉANT software composition and licence analysis (SCA and SLA) services** to conduct the related reviews and audits, designed to determine the appropriate OSS licence and ensure compliance. Identifying and addressing software vulnerabilities through SCA improves quality and benefits the broader community.
9. **Set up contribution, communication, and governance workflows** to ensure compliance with the software's licence.
10. **Adhere to the standards of the domain community** in software development, licensing, software metadata, documentation, registration in relevant community registries, citation, and promotion.
11. If applicable, **enable and advise on the citation and referencing** of software in scientific papers, presentations, and tutorials, ensuring clear and persistent references.

These and other recommendations for various software project stakeholders are detailed in the GN5-1 *Software Licence Selection and Management* guide [7] and *Deliverable D9.4 Open Source and Licence Support Report* [8].

2.2 OSS Conditions

Selecting a licence is not straightforward and is a task most developers prefer to avoid. While the GÉANT IPR Policy [9] favours permissive licences and names examples such as MIT, BSD, and EUPL, the available options are often constrained by the licences of core components or the frameworks the software relies on. It is therefore sensible for developers to explore specific licences, their implications, and compatibility only when the constraints and candidate licences are known within the licensing process, supported by 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 mandates it. Alternatively, if there are unavoidable GPL or AGPL dependencies, they may need to opt for a compatible licence, typically the most restrictive of the licences involved. Conversely, in cases where no pre-existing limitations exist, such as with a new project or when all essential components use permissive licences, the development team can select one or more candidate licences based on their key features regarding code sharing and modification (where copyleft licences are more restrictive), relicensing (also known as sublicensing, which the team may wish to prohibit or allow), and the handling of patents (waived, protected, or ignored). (For further information about software selection, see Section 2.9 Software Licence Analysis (SLA) Service.)

Various licence conditions are often classified as:

- Rights/permissions – what you are allowed to do.
- Requirements/obligations – mandatory compliance artefacts, such as copyright, disclaimers, licence text, notices, relevant source code, build and installation instructions, etc.
- Restrictions/limitations – what you are prohibited from doing.
- Other characteristics – typical uses, classification, compatibility, legal features, origin, community, endorsement, and more.

These features are further outlined in models such as the EC's Joinup Licensing Assistant tool (*JLA – Find and Compare Software Licenses* [10]), which, when options exist, greatly facilitates the selection of the most suitable licence aligned with the developers' strategy and user community.

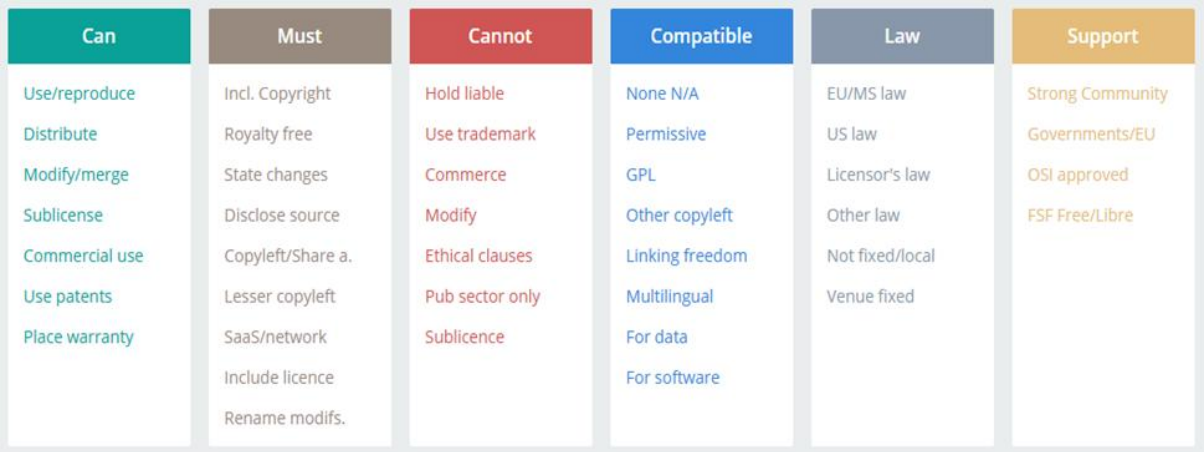

Figure 2.1: OSS conditions from *JLA – Find and Compare Software Licenses* [11]

2.3 Compatibility of Licences Frequently Used in GÉANT

We have compiled comprehensive information about OSS licences, some of which is also available through Mend, the tool used for SCA (described in Section 2.8). However, only a limited set of licences is frequently used in GÉANT, and only a few licences are common sources of compatibility issues. Figure 2.2 provides an orientational diagram describing the relationships and compatibility of these licences.

There are two interpretations of licence compatibility. A less restrictive, more commonly used, and symmetrical type of compatibility indicates that components with distinct licences can be used in the same project. It may be achieved by relicensing one or both, or 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 governed by another licence. Although the first interpretation is an extension of the second, various types of "use" by the software can sometimes be achieved by altering the system architecture, allowing the integration of a problematic component without the need to change the software licence.

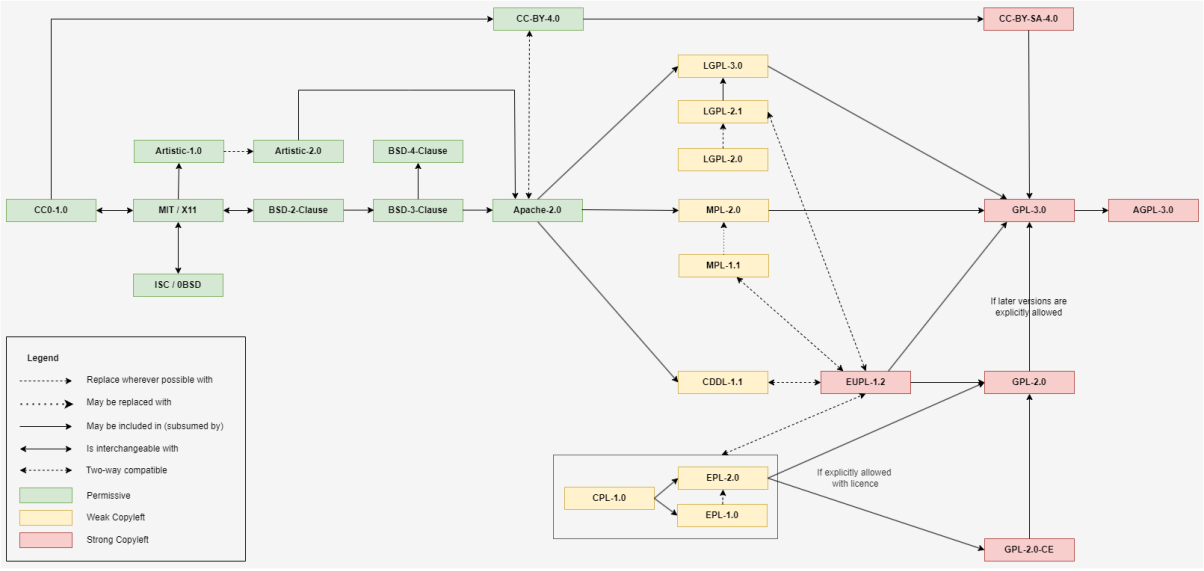

Figure 2.2: Relationships between OSS licences commonly used in GÉANT projects

OSS licences are described in several documents produced by the licensing team. For more details, refer to *Reference Information about OSS Licences and Tools* [12], *OSS Licences and Licence Selection* [13] and *Open Source Licences Used in GÉANT* [14].

2.4 GÉANT IPR Policy

The GÉANT IPR Policy applies to intellectual property (IP) generated within the GÉANT project, including open source software. It provides recommendations and rules, which this document translates into practical and actionable instructions. The policy is available at [15]. It aims to establish a framework for IP generated by the GÉANT project, applying to all GÉANT participants. It offers practical guidance on IPR and, most importantly, seeks to establish a cooperative approach to IP protection and fair use in GÉANT projects. The IPR Policy also upholds the principles of findability, accessibility, interoperability, and reusability (FAIR) for the use of GÉANT project IP. Upon approval by the General Assembly, it has been binding since January 2023.

From a software development perspective, having software composition analysis (SCA) with a tool that scans components to identify their licences is crucial for determining which licence to apply. Non-compliance with OSS licence terms can have serious legal and financial consequences, so due diligence is essential. The IPR Policy stresses the importance of IP protection, introduces the GÉANT IP Register to document project results, and highlights the need for scanning code with an SCA tool to ensure licence compliance. It also underlines the role of the IPR Coordinator in supporting development teams through the licence selection process.

To summarise the IPR Policy:

- **All OSS licences** [16] [17] **are permitted**.
- The IPR Policy strongly recommends the use of permissive licences.
- Copyleft licences (weak, strong, or network-protective) can be applied as needed, in consultation with the IPR Coordinator.
- The IPR Coordinator provides final recommendations and maintains the GÉANT IP Register.

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

2.5 Licence Governance in GÉANT

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

- Assists those wishing to understand and manage OSS licences, master the tools provided, and apply them.
- Provides knowledge and support to solution designers, developers, and skilled promoters on licences and IPR.

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

- **Software composition analysis (SCA)** – Technical and practical assistance for development teams in managing software components and their licences.
- **Software licence analysis (SLA)** – Assistance for software teams in aligning 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, such as SonarQube Setup Assistance and Extended Source Code Review [18]. Through SCA and SLA services, the licensing team ensures key licensing concerns are addressed by:

- Assessing the licences of components and prior IP.
- Selecting an open source licence that meets the project's needs.
- Ensuring the selected licence is compatible with the components' licences.
- Ensuring compliance with the chosen licence.

The licensing team helps GÉANT software teams with IPR and licensing issues by implementing robust processes for managing dependencies and licences and achieving compliance with the GÉANT IPR Policy. It provides access to expert tools for analysing and managing open source licences. The OSLS has collaborated with many GÉANT software teams to assess their licensing situations and decisions, offering 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 by email. SCA and SLA services are requested via a software review request submitted to the GÉANT Jira Software Tools Help Desk [19], which also tracks their progress. Several iterations of analysis and adjustments to licences or dependencies may be required to achieve satisfactory IPR status. The IPR Coordinator is available to assist with licensing decisions.

Our guides assist with licensing and services. For more information on OSS licences, start with the *OSS Licences and Licence Selection* guide [20]. Since the SLA service requires the involvement of the software team in licence selection, it is recommended that this guide is read before requesting the service. It also helps with interpreting software composition analysis results. However, if you are already familiar with OSS licences or are seeking summaries of licences commonly found in GÉANT projects or those typically involved in licence compatibility issues, you may proceed directly to the *Open Source Licences Used in GÉANT* guide [21].

SCA and SLA offer valuable opportunities to align software projects with GÉANT's and external licensing and policy expectations. The analysis, selection, and validation of software licences can significantly enhance a software project's standing within GÉANT and beyond.

Engaging with software licensing is an opportunity to evaluate project components and licences while considering aspects related to authorship, ownership, external relations, and documentation artefacts. This structured effort helps standardise projects in these aspects. Licensing reviews may also engage individuals such as new team members to assess and validate the software and motivate the original developers to critically evaluate and consolidate their work.

Furthermore, this activity involves registering and publishing software using GÉANT's internal software tools, aligning with established practices and expectations within GÉANT. Successfully completing licensing and formalising the licence are positive indicators for the project within GÉANT. This is particularly significant for smaller and relatively autonomous developments, such as those within the GÉANT Trust and Identity Incubator, as it enhances visibility and improves the practices and visibility of the originating activity. Addressing issues through licensing analysis, along with the resultant reports and decisions, provides valuable insights for evaluating software solutions and services at GÉANT Product Lifecycle Management (PLM) gates [22].

The analyses, especially when conducted for a module that is an add-on for an externally developed open source platform, can also benefit the broader software platform community by assessing its overall licensing status and component security. This is a side effect of analysing software produced by GÉANT development teams, as it includes assessing involved components and licences.

2.6 Importance of Declaring a Licence in Code Repositories and FAIR Compliance

Declaring an open source licence when publishing code in a public repository is essential for legal clarity, responsible reuse, and alignment with community and funder expectations. Legally, **code shared without an explicit licence** defaults to "all rights reserved", prohibiting others from using, modifying, or redistributing it – even when publicly accessible. Applying a well-defined licence ensures compliance with legal requirements while empowering others to reuse the work. Specifying a licence communicates the terms of use, protecting both the **developer's rights** and **users' obligations** (including attribution requirements, disclaimers, and redistribution conditions). This becomes particularly crucial in collaborative environments and for publicly funded projects, where compliance, reuse potential, and transparency are essential values.

Licensing also plays a pivotal role in upholding the **FAIR principles (Findable, Accessible, Interoperable, and Reusable)**, widely endorsed in research and open science. The **Reusability** of digital assets, including software, hinges on a clear licence. Without one, code cannot be integrated into other projects, workflows, or datasets, undermining reproducibility and collaboration. A declared licence further supports **Accessibility** by resolving legal ambiguity and enhances **Interoperability**, particularly when integrating projects with differing licensing terms.

Special Considerations for the GPL

Among open source licences, the GNU General Public License (**GPL**) family imposes distinct obligations. As **copyleft** licences, they require derivative works or redistributed versions to adopt the same GPL terms. Distributing software that incorporates GPL-licensed code legally obligates you to **release the source code publicly** under identical conditions. While this aligns with open science ideals of transparency and shared knowledge, compliance demands deliberate planning – especially in collaborative or commercial settings.

2.7 Software Composition Analysis (SCA) Service

This service assists development teams by setting up a project within an SCA tool and providing insights into external components. It is suitable for both **one-time software analysis** and **continuous monitoring**, identifying third-party components, their licences, and potential IPR infringements or security vulnerabilities. The service can be combined with other software review services or performed independently. Repeated analyses can determine how changes in software and dependencies affect licence compliance and identify new or pending vulnerabilities. For ongoing monitoring, the 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 2.8), which identifies third-party components in projects and gathers information about their licences and security vulnerabilities. Mend uses a comprehensive database to address two critical aspects:

- Analysing components and their licences to reduce the risk of IPR infringement, which could have significant financial consequences, by supporting licence compatibility and compliance.
- Reporting security vulnerabilities, which complements SonarQube and extended code reviews.

The licensing team sets up the project in the Mend SCA tool, which generates reports on software composition and potential deviations from established policies. The visibility of reports and the created Mend project can be configured during the project setup or adjusted after results are obtained. The primary report, the "Risk Report", details the software composition, including components, their licences, and related risks and vulnerabilities.

The designated leader or expert from the development team receives this report and supports its interpretation, though the development team should be able to interpret it independently. The licensing team assists where needed, and developers can request additional feedback on the risks related to licences and IPR infringements.

A summary of the SCA service is available in *Software Reviews* [23], with more details in the *Client Guide for Software Composition Analysis (SCA)* [24].

2.8 Mend SCA Tool

Mend is an online tool provided by GÉANT for open source licence and security compliance [25]. It is designed for in-house use by customers and does not offer direct legal consultancy. Mend **detects software components**, **their licences, and vulnerabilities**. It integrates seamlessly with development environments, building a pipeline to detect open source libraries with security or compliance issues. Mend reports severe bugs, problematic licences, new versions, and available fixes. It simplifies the management of open source libraries and helps detect and resolve compliance and security issues.

Mend creates an **inventory of software components** by detecting declared dependencies and matching them with a rich database. It provides **licence information**, warnings about outdated or risky open source libraries, and details of associated security vulnerabilities. The licence information includes licence type, risk level, patent handling, summary descriptions, and excerpts from original licence texts.

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 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. An overview of Mend usage is available in the *Mend Short Guide for End Users* [26].

Mend analyses projects in various ways. The code may be stored locally, and a Mend scan can be triggered manually whenever the development team needs to assess the effects of a recent code change (details in *Adding Project to Mend (Scan Flow)* [27]). Scanning of GÉANT software can be done through one 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 components in the Mend database to detect and describe open source or commercial components. Mend enables assessment of a GÉANT product for compliance with the IPR policy without having to review the code. Scanning populates the Mend web dashboard with project data, enabling the creation of compliance reports using data from Mend's backend database.

The web-based UI provides numerous options and panels for reviewing and analysing scans of open source software laid out as an organisation's products and projects. Each scanned product or project is displayed on a corresponding page, summarising relevant information and offering several options, providing a comprehensive view of the organisation's open source status. This page also grants access to all contained projects and libraries used by the product or project.

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

- **Product Alerts** – shows information about library (component) alerts generated for a product. The **New Versions** category indicates alerts for scanned libraries that are outdated. When such a library is detected, a new alert is generated, specifying the out-of-date library and its latest version.
- **Security and Quality** – displays the number of libraries with vulnerabilities by severity, the score of the most vulnerable library, the number of libraries with newer versions and vulnerabilities, and the count of "buggy" libraries.
- **Libraries** – provides detailed data on libraries, including name, licence, and per-product or per-project occurrences.
- **Licence Analysis** – gives an overview of licences used by product or project components, and the number of different licence types.

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

Additional information on licences is available in reports accessed through the Report menu. The Risk Report provides a view of all aspects of libraries, their licences, security, and quality. The report contains several panels and tables displaying risk-related data. Security and licence analysis information 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. As this database is continually updated, reports can change over time even without a new scan.

Mend offers insights into OSS licences, including licence type, copyright, handling of patents and royalties, linking requirements, and compliance with free and open source software norms. Mend's experts have analysed numerous licences and assigned risk scores to help developers easily assess the risks associated with a particular licence.

The primary score is the Copyright Risk Score [28], which quantifies, on a linear scale, the degree of loss of exclusive control when using a library or source code governed by that licence. The Copyright Risk Score is, therefore, more suitable for commercial organisations that want to assess their software assets and risks affecting control over them, and associated competitive advantages, than for software projects willing to share the code they developed. Low scores (green) correspond to permissive licences, while high scores (red) indicate strong copyleft licences, helping users quickly identify and assess licence risks. Licences are also evaluated in terms of copyleft (none, partial, full) and linking (non-viral, dynamic, viral). There is also a Patent and Royalty Risk Score, and a related attribute indicating whether the software under the licence is royalty-free (yes, conditional, no).

Mend can integrate with development environments and build tools, and it can be incorporated into a continuous integration (CI) pipeline, triggering scans with each commit in host repositories such as GitHub [29], GÉANT GitLab [30] and Bitbucket [31]. Integration of Bamboo [32] CI/CD software with Mend is detailed in *Automated Mend Scans with Bamboo* [33].

Mend's functionality, initially developed for commercial organisations and the assessment of IP-related risks, is gradually being adapted to include licence compatibility checks better suited for OSS projects. However, developers and project leaders still need to familiarise themselves with its features, concepts, and peculiarities.

Mend does not provide comprehensive or fully accurate licence detection. Manual correction of licences for individual components is sometimes requested by GÉANT projects, such as for libraries without clear licence or licence version information in Mend's database. Sometimes only indicative ("suspect") licences are identified and must be manually verified. Additionally, Mend's handling of multi-licences and relicensing options is not always reliable, and not all allowed licences are consistently listed. Furthermore, there is no support for software versioning and differential reports.

These limitations hinder fully automated control and alerts on component licences, although Mend is still far more efficient than manual analysis. Further decision-making and remediation are often necessary even after the analysis is complete.

This is why the Software Licence Analysis (SLA) service, described in the next section, is needed. Mend can indicate the likely compatibility of other components with the selected one, but final licence selection always requires human judgement and trade-offs.

It should be noted that other tools are available for software composition and licence analysis; some are listed in *Other Software Composition Analysis (SCA, Software Inventory) Tools* [34].

2.9 Software Licence Analysis (SLA) Service

Selecting the appropriate licence for distributing and using software is critical, as it defines the terms for sharing, modifying, and distributing the software. This process considers project objectives, developer preferences, the desired level of collaboration, the software's ecosystem (including related products, their licences, and users' expectations), and the constraints imposed by the licences of dependencies and other background and sideground IP. Choosing a licence also involves assessing the effort needed to resolve licence compatibility issues and ensuring compliance with legal, regulatory, and funding requirements. The chosen licence plays a key role in fostering a collaborative and transparent development environment, providing clarity on how others can use and contribute to the open source project.

Before requesting an SLA, we recommend tentatively selecting a standard OSI-approved license that aligns with your goals and needs, using tools like the EC's Joinup Licensing Assistant tool (*JLA – Find and Compare Software Licenses* [35]) and choosealicense.com guide for your decision. This exercise will help identify the most suitable license. However, your options may be limited by compatibility requirements with the licenses of third-party components used in your project.

As discussed in Section 2.2, the most restrictive licence is often the only one compatible with all components. However, this is not always the case, especially when multiple permissive licences are involved. Furthermore, the most suitable licence compatible with most or all components may not necessarily be among those present. These complexities explain the need for the SLA service.

The SLA service provides technical consultancy, offering a comprehensive understanding of third-party libraries used in a software project and their licences, discussing the software's licensing, and giving recommendations and validation of project documentation artefacts. This understanding is crucial for selecting the appropriate licence and ensuring compatibility among all licences involved. The service is highly recommended for software development teams looking to validate third-party licences, establish or review their project's licence, verify compliance with both the licence and the GÉANT IPR Policy, or assess the impact of potential licence changes, including changes to dependencies' licences.

It provides deeper insight into third-party library licences and their relationship to the project. The service builds on the results of prior software composition analysis (SCA), with manual checks of libraries where necessary. It includes customising the SCA tool's licence settings, selecting the project licence based on an analysis of dependencies, and checking alignment with licence requirements and the GÉANT IPR Policy, and content-related documentation artefacts. If the SCA tool is integrated into a CI pipeline, the service team works with the customer to customise the relevant settings.

A summary of the SLA service is available in *Software Reviews* [36].

2.10 Licensing Process, Decisions, and Artefacts

The typical steps for managing licences are:

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

These steps are preceded by preparatory activities and decisions, and should be followed by ongoing measures to ensure continuous licence management. Detailed information on the preparatory steps, the process itself, and long-term licence management activities within GÉANT is provided in the following sections. For further information, refer to GÉANT's Open Source Licensing and Compliance training [37] and *GN5-1 Deliverable D9.4 Open Source and Licence Support Report* [38].

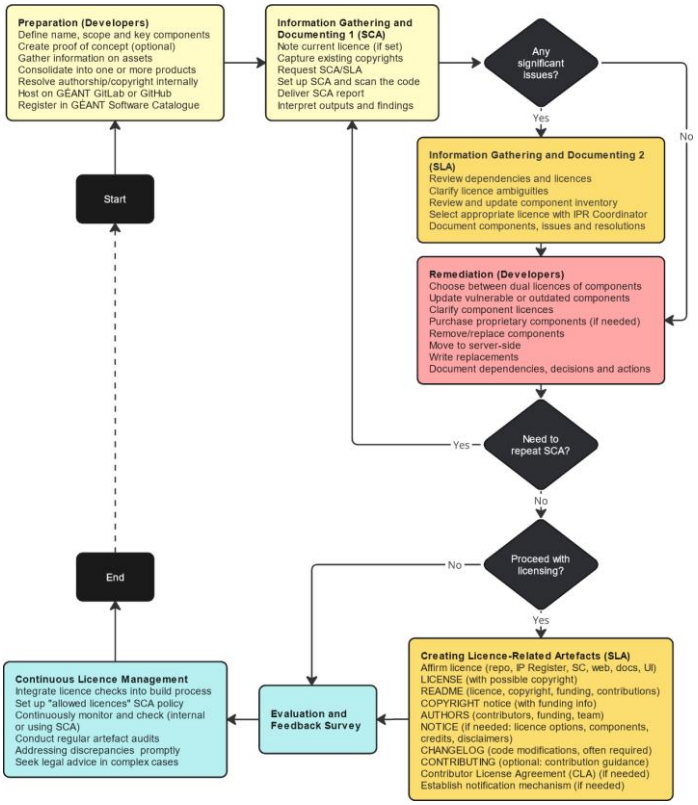

Figure 2.3: Overview of the GÉANT OSS licence management workflow

The workflow begins with an SCA request, which may be accompanied by a request for an SLA. If no significant issues arise and the current or proposed licence is clear and agreed upon with the IPR Coordinator, further analysis may be shortened or skipped. If the SLA is not requested, developers may make their own changes based on the SCA results before deciding they are ready for the final SCA, licence selection, and implementation. Changes made during remediation or significantly postponed SLA may necessitate a repeated SCA to validate changes and facilitate the resolution of remaining issues.

The licence implemented during the creation of licence-related artefacts must still be approved by the IPR Coordinator. This stage involves reviewing necessary licence and copyright-related files, with developers updating these files according to the provided guidance. At the end of the process, software developers complete a survey to provide feedback on all the steps they have taken.

Continuous licence management is carried out by the development team based on their SCA-SLA experience, with advice from the licensing team. The licensing team may adjust project configuration in Mend and configure it for the chosen allowed or prohibited licences. While the development team should perform further monitoring and adjustments after successful licensing, it is recommended to repeat the SCA-SLA process after significant changes to ensure the project remains compliant.

Preparation

- Decide on the software name, **grouping of subprojects**, and the use of available contributions.
- New projects may require a proof of concept or **prototype** to identify and validate key components.
- Gather **pre-existing information** and documentation for components, models, assets, and relevant specifications and standards.
- Consolidate project components into a **single repository project**, or **clarify their relationships** if they should remain **separate**.
- Address authorship and copyright matters internally.
- Ensure the software project is hosted on GÉANT GitLab [39] or GitHub [40].
- Register the software project in the GÉANT Software Catalogue [41].

All components of a closely interconnected software development project should reside in a single repository, ideally on GÉANT GitLab. However, some developers may opt for GitHub or mirror their GitLab project there to gain increased visibility, collaboration opportunities, redundancy, and access to additional features and integrations. Additionally, while development and complex pipelines can remain within the private GÉANT GitLab environment, the final outputs can be publicly shared on GitHub.

The software may include non-original artefacts or assets with different licences. These assets, which may not be detected by SCA tools, should be documented with their origin, copyright, and licence information as soon as they are added to the project. Methods for documenting them are covered in Section 2.11 Licences and Tracking of Documentation, Data, and Other Works. Failing to document them promptly can complicate their identification and tracking in the future.

One Project or Several Projects?

When handling multiple projects, it is essential to determine which dependencies should be included in the SCA analysis. This may depend on the relationship between components and their respective responsibilities. For instance, a project may be a subproject of the same team or a module within a larger project overseen by an external group or entity. Both the subproject and the main project may need to be analysed together with their source code and dependencies, even if stored in separate repositories.

This extended analysis is important for two reasons. First, an integrated analysis of the larger project and its subprojects provides a more comprehensive understanding of components. Second, an integrated view ensures that the larger project's licence governance is sound.

Even when contributions are perceived as part of another developer's project, contributors are not fully shielded from potential licensing disputes, especially if there is a formal relationship between the teams. To mitigate this risk, it is advisable to analyse the larger project as thoroughly as the contribution itself.

Developers should inform the licensing team of such situations and which additional components should be included in the analysis. This can be facilitated by the configuration of the project's build tool. For example, Maven dependencies with the default "compile" scope are included in all build tasks and are propagated to all dependent projects, with their dependencies transitively included in the Mend analysis. Dependencies with "provided", "runtime", and "test" scopes, which are supplied by the execution environment at runtime or during testing, are excluded from the analysis as they are considered external libraries.

In some cases, the development team may not yet have its own software and may consider using an external library or framework as the foundation for future development. The analysis of a non-GÉANT project with Mend for licences and vulnerabilities is a resource-intensive process requiring consultation with the licensing team on a case-by-case basis. Such analysis is only conducted when essential for a strategic decision.

The licensing team can also advise on grouping features or products within the project and software branding from a technical and licensing perspective. For advice on product or service development, please contact the GÉANT Marcomms Team at marcomms@geant.org.

Information Gathering and Documenting

Collect and document copyrights, licence findings, and decisions:

- **Specify the current licence** of your product (a packaged bundle of components) or project (a separate programme or component), if one is already in place.
- Record the copyrights for the contributions used.
- **Request** software composition analysis (**SCA**) **and** software licence analysis (**SLA**) **via GÉANT Jira** [42].
- Scan the project codebase, producing an **inventory of components** and their licences **through the SCA service** (the service will set up the SCA project, scan the codebase, and deliver an SCA report).
- **Interpret** the SCA outputs and findings.
- If ambiguities or complex situations arise or advanced support is needed, proceed with the **SLA service for additional assistance**.
- Further analyse and **document dependencies** and licences.
- Based on the SCA report and any additional security or vulnerability reviews, **review and update the inventory of components**, identifying:
  - Vulnerable open source components that should be removed or replaced.
  - Outdated open source libraries that need updating.
  - Confirmed licences for the components used (in-licences).
- The review may also **clarify ambiguities** or doubts:
  - The SCA tool may not properly identify a licence or may report some licences as suspect or ambiguous.
  - Information about a licence may be false, unclear, or contradictory.
  - Some licences may be recognised by multiple names.

    - Some permissive licences (BSD, Artistic, etc.) have unnumbered variants or are sometimes edited by software authors.
    - The applicability of “or later” clauses may be unclear or wrongly declared by editing the original licence text.
- Document all findings through reports, UI displays, and data exports.
- **Consult the licensing team and the IPR Coordinator**, who will help determine the appropriate licence to apply. Tools like JLA or choosealicense.com can help you in this process.
- **Select an appropriate licence** – The document *Open Source Licences Used in GÉANT* [43] provides insights into OSS licences and their relationships. Reading it improves understanding of OSS licences and helps in selection, but the final decision must be made with the IPR Coordinator after the SCA scan.
- **Document decisions** – Some licences may be further refined during remediation and decision-making, but you should record the reasoning behind licence selection, detected issues, and how they were or will be addressed. The IPR Coordinator’s reasoning and approval should also be documented.

For more information about the licence selection process and related recommendations, refer to the *OSS Licences and Licence Selection* guide [44] and the white paper *OSS Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* [45].

Remediation

This stage focuses on resolving licence conflicts and fulfilling obligations, which may require varying levels of effort from developers, ranging from minor adjustments to significant changes, such as removing or replacing dependencies, developing alternatives, or refactoring code. In some instances, further remediation may be required after subsequent SCA scans.

Remediation actions may include:

- **Selecting a more appropriate licence** for the product or project compatible with dependencies.
- **Implementing quick fixes**, such as:
    - Updating or replacing vulnerable open source components.
    - Updating outdated open source libraries where possible.
    - Investigating unclear component licences or seeking clarification or relicensing from authors.
    - Purchasing necessary proprietary software licences.
    - Choosing among dual-licence options for components.
- **Identifying any remaining incompatible licences**.
- **Deciding on actions** for components with such licences:
    - Removing non-essential components.
    - Replacing them with available alternatives.
    - Moving them to the server side (providing functionality via a central service).
    - Developing alternatives to meet the required functionality.
- **Accepting certain risks**.
- **Internally documenting dependencies**, their licences, features they provide, **decisions made, and remediation actions** undertaken.
- If necessary, repeat the SCA and the remediation steps listed above.

Creating Licence-Related Artefacts

To ensure OSS licence compliance, several key artefacts are required. The details of these artefacts are covered in Section 2.12.

- Provide a LICENSE file
  - Clearly state the selected OSS licence by placing a LICENSE file in the root directory.
  - Ensure the licence text matches the official version exactly. MIT, BSD, and ISC require copyright information to be included in the LICENSE file.
  - When a LICENSE file is present, header comments in source code files (with licence, copyright, and disclaimers) are generally unnecessary, except in special circumstances.
- **Declare licence options** if available and used.
  - Some licences offer options that must be explicitly stated.
  - Options may include accepting later versions of the licence or relicensing under specific licences endorsed by the original licence.
  - Licence options are typically declared in the README file.
- Declare the licence in project documentation, on the website, and in the repository UI.
  - Update project documentation to reflect the selected OSS licence.
  - Use any available repository UI features to declare the licence.
  - If integrating with other systems or package managers (e.g., npm or PyPI), you may need to specify the licence using its SPDX identifier in the project's metadata file (e.g., package.json or pyproject.toml)
- Document the project licence in the GÉANT Software Catalogue and IP Register.
  - The Software Catalogue allows for declaring the licence on the project home page.
  - The IPR Coordinator manages the GÉANT IP Register.
- Provide a copyright notice.
  - Add a copyright notice for the project in a **COPYRIGHT file**, including funding information.
  - Include copyright information for third-party components in compliance with their licences, listing 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, specifying the licences of used components if necessary.
  - Provide instructions for accessing the full licence text if not provided in the LICENSE file.
  - Briefly explain the licence's implications for users and contributors.
  - If licensing badges are available, add one to the README file to clearly communicate the project's licensing information.
- **Document code modifications** as required by the licence.
  - A history of changes may need to be documented, depending on the applied licence.
  - If the modified software or component has a **CHANGELOG file** or equivalent, extend it with details of your changes, preserving its format.
  - Check the licence text or summaries, such as those in *Open Source Licences Used in GÉANT* [46] to determine whether documenting code modifications is necessary.

- **Document dependencies**, their licences, notices, and copyright information.
  - List the licences of directly included dependencies in a dedicated file or project documentation, typically in the NOTICE file, along with each component's copyright and links to its licence text. The file may also include disclaimers and details on the project's licence, tools used, and IP used. The README can summarise this information in a more concise form.
  - In the AUTHORS file, you may credit individual contributors, acknowledge funding, and reference the GÉANT work package or project task.
  - For source code components in subfolders, store their licences, copyright, and notices within those folders.
- Document licence adherence, contribution, and updating.
  - Provide guidance for users and contributors on adhering to licensing requirements in the README, **CONTRIBUTING,** or CODE_OF_CONDUCT file. Examples include guidelines from FileSender [47] and Atom [48].
  - Outline contribution and copyright guidelines, even if external contributions are not expected.
- Prepare and apply a Contributor License Agreement (CLA) if necessary.
  - If the selected licence requires a CLA, establish one to define contribution terms and ensure mutual understanding.
  - Some licences offer suitable CLA forms.
  - Clearly outline the CLA process in the README and detail it in the CONTRIBUTING file to ensure legal clarity for all involved.
  - Place the CLA in a file named CONTRIBUTOR_LICENSE_AGREEMENT, CLA, or within broader contribution guidelines in the CONTRIBUTING file.
  - Creating a Developer Certificate of Origin (DCO) is a lightweight alternative to a CLA. By adding a "Signed-off-by: Name <Email Address>" in their Git commit, contributors self-certify that they agree to the DCO's terms, affirming they are either the original authors or have the right to submit the code, which can be used in the project under its licence.
- **Establish a licence notification mechanism** – If the licence or contribution policy may change, implement a notification mechanism for contributors and users. This could include prompts during builds, repository notifications, or updates via project notification channels. A pop-up with licence and copyright change information is also an option.
- Once licence artefacts are complete, they should be assessed by the SLA team and the IPR Coordinator.
- Complete the SLA/SCA *Evaluation and Feedback Survey* [49].

Continuous Licence Management

The aim of continuous licence management is to **integrate licence oversight 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 integrating dependency and licence checks into CI/CD toolchains. The Mend service provided by GÉANT or other tools [50] can identify dependencies and their licences or verify compliance. Setting up an "allowed licences" policy in Mend ensures licence violations are detected early.
  - Some tools, such as the License Maven Plugin [51], can generate a list of dependencies and their licences, download licence files, check, update, or remove licence headers in source files, and update (or create) the main project licence file.
  - Verify and review tool outputs, as they are not foolproof.

- Establish continuous monitoring and compliance checks.
  - Stay informed about changes to the chosen OSS licence.
  - Review the potential impact of any licence updates on your project.
  - Establish processes for continuous compliance checks, **repeating SCA and SLA as needed** for new dependencies or licences, to ensure licensing obligations are met.
- Perform regular audits of licence-related artefacts.
  - Conduct regular audits to ensure the project's licence-related artefacts are accurate and up to date.
  - Promptly resolve any discrepancies to maintain legal clarity and compliance. Delays are likely to complicate resolution.
- **Seek legal advice when necessary** – In complex licensing situations, seek advice from the IPR Coordinator to ensure proper interpretation and compliance.

GÉANT software best practice *BP-B.6: Manage Sideground IPR* [52] recommends addressing pre-existing and external IP early and periodically repeating the process.

2.11 Licences and Tracking of Documentation, Data, and Other Works

Software-related artefacts and assets distributed with software or stored in its repository generally adhere to the same open source licence. These may include data, technical documentation, configurations, and user manuals. For separate tutorials, presentations, training materials, or promotional materials which are not software artefacts, it is advisable to use the Creative Commons Attribution (CC BY) or Attribution-NonCommercial (CC BY-NC) licences. The GNU Free Documentation License (GFDL) is also noteworthy. While data is sometimes licensed under OSS licences, datasets are more commonly licensed under Creative Commons and Open Data Commons.

Software should clearly document and attribute external data sources either in the software itself or in its documentation, ensuring transparency and compliance with data licensing or usage terms. If data comes with specific licence requirements or restrictions, the software licence may be impacted, particularly if data is hardcoded, integrated into data structures, forms part of default datasets, or is included in configuration or bootstrap scripts. This applies to all data provided with the software or used for its operation. Such data should be listed alongside dependencies and included in the software's licensing analysis. If proprietary or open data licences apply, compliance with their terms is required, and the data should be referenced at least in the NOTICE file. However, if the data is widely used public reference information, such as the ISO list of country codes, acknowledgement in the documentation or project artefacts is usually sufficient, and licence analysis is generally unnecessary. Even if external data is not explicitly credited, it should be documented, with explanations on how it can be updated. This also applies to data contained in external code libraries or modules.

If data is dynamically fetched from external services or APIs during initialisation or regular use, it must be clearly referenced in the README or NOTICE file and in the project documentation. Examples include maps, environmental and sensory information, and all presentations or embedding of data from external sources. The software may also store, aggregate, or process user-created or external data, such as collaborative content, usage information, logs, harvested data, personal details from authentication services, network state, topologies or traffic, and datasets for artificial intelligence training. Documentation should clarify how such data is used and guide administrators or users on how to subscribe to or access external sources, as registration with third-party services is often required. Ideally, software should support multiple data sources, reducing reliance on specific ones. Users should be made aware of available alternatives.

Processing external or user-created data may require user consent, be subject to the service's terms of use, or depend on agreements between the service provider and the external data source. These arrangements do not affect software licensing, but their implementation should be feasible using the software. Developers should ensure data security, design for personal data protection, and include features supporting arrangements such as user consent, cookie management, and displaying privacy and data policies and terms of use.

Most OSS licences include disclaimers of warranties and liability, meaning software authors are not legally responsible for malfunctions, damages, or misuse. Regardless of this, it is important for reputation reasons that the software is reliable and secure. To facilitate this, GÉANT offers security-focused code reviews using automated analysis and expert assessments [53], related training [54], and infrastructure-level security support [55].

The use or modification of externally developed work other than software can affect software licensing, especially when it involves database models, architectural designs, development frameworks, or code generated by tools. If such work is under a specific licence, the software must comply with its terms, which may include attribution, restrictions on commercial use, or obligations for derivative works. If the software incorporates external database models, frameworks, or generated content to a significant extent, the licensing terms may apply to the entire project. This is especially relevant when using copyleft OSS licences or non-OSS licences like Creative Commons with NC (NonCommercial), ND (NoDerivatives), or SA (ShareAlike) clauses. For example, a database model may require modifications, extensions, and optimisations, making permission for derivative works crucial. Therefore, it is critical to review the licences and terms of such work before starting significant development. These works are usually copyrighted and should be documented (e.g., in a NOTICE file) and included in the code repository, preferably in a dedicated folder, regardless of whether their original or modified versions are used.

Since many of these works are unlikely to be detected by SCA tools, it is important to document and communicate their use and licences as soon as they are incorporated. This also applies to non-original software-related artefacts, such as assets, configuration files, scripts, technical documentation, and user guides. If they are distributed with the software, they should be kept in its source code repository and licensed under the same terms. If numerous such works are included, they should be annotated with easily searchable and aggregable provenance and licensing details in the software repository, recorded in the project's Software Bill of Materials (SBOM) [56], or marked with extractable metadata and comments. These details include the place of use, origin, copyright, and licence. Failure to do so when adding such assets can make their identification and tracking difficult later. This is especially important for non-original graphical or UI assets, such as images, vector graphics, JavaScript code, or UI layouts, which may not be part of external components but could be used by them, and therefore easily overlooked.

Original assets distributed with the software should be kept under the same licence and do not need to be individually tracked.

2.12 Project Licence Options

The applied licence may allow additional conditions or permissions to be added, clarifying how others can use, modify, and distribute the software. If the licence offers such options, they are explained in the licence text. Common software licence options that code owners may explicitly state include:

- **Permitting users to choose between the original licence version or any later version** – This allows users to choose between the original licence version or any later version approved by the licensor. Such relicensing can be interpreted as licence-endorsed open-ended multi-licensing. For example, a statement specifying that a particular GPL licence version is required is typically placed in the README file. You can use the whole phrase offered by the GPL: "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", or simplify it to "This software is licensed under GNU General Public License version 3 or any later version". If the licence notice ends with "version 3" or "version 3 only", only GPL 3.0 applies. If no version is specified, 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 GPL can be used, although this option is rarely used. The same applies to LGPL, with "GNU Lesser General Public License" replacing "GNU General Public License."

- **Relicensing under a different licence** – Some licences allow relicensing under a secondary licence endorsed by the original one or chosen by the licensor. For example, the EPL 2.0 notice that states "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/." means the software can only be used with EPL 2.0. However, a Secondary License may be introduced, allowing recipients to comply with either the EPL or the Secondary License, such as GPL 2.0 with Classpath-exception: "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 licence granting rights at least as broad as those under the EPL can be declared as a Secondary License. Adding the latter clause is effectively relicensing, and all copyright holders must agree to such a licence change. For MPL 2.0, its licence notice: "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/." allows most GPL licences as Secondary Licenses, unless the licensor prohibits this by adding "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** – Certain rights may be extended beyond the standard terms of the original licence, such as allowing commercial use without open sourcing modifications, providing a patent grant, or permitting the distribution of proprietary versions. These extensions should be clearly articulated, specifying that they supplement the original licence without replacing or altering its conditions. For example, Apache 2.0 states that the software is by default provided "AS IS," but additional warranties can be agreed upon in writing.

- **Placing limitations on certain uses or modifications** – Some licences may allow the addition of conditions, such as restricting non-commercial use, requiring modified source code availability, or ensuring compatibility with the original version. Any limitations should be added carefully, ensuring their compatibility with the original licence. For example, Apache 2.0 allows additional restrictive clauses, but licensors should avoid introducing restrictions that contradict or undermine open source principles.

- **Choosing the jurisdiction governing the licence** – With the EUPL, the licensor can choose the jurisdiction under which the licence is governed, specifying the applicable legal framework.

It should be noted that SCA tools cannot interpret options, additional rights, or conditions introduced in free text.

3 Complying with a Selected Licence

This section guides developers through the process after selecting a licence. It covers preparing necessary files, performing compliance checks, and reviewing adherence with the licensing team. It also provides essential information for developers seeking to address licensing issues independently. Links to GÉANT-approved example files will be provided as they become available from software projects.

A software licence must be applied to software that is not intended for internal use only. The selected licence depends on the licences of components used, and the copyright statement may vary depending on the institutions involved. When the software is provided as a service (SaaS), there is no need to declare its licence unless one of the used components is under a network-protective copyleft licence.

Developers must comply with the chosen licence and ensure licence compatibility. Failure to do so constitutes a breach and could lead to legal challenges and financial loss.

Most licences require a copy of the licence to be included, typically in the LICENSE file located in the root folder. Although the LICENSE is often the first file developers look for to learn about the licence, simply providing it is not sufficient from a legal standpoint. Even with the LICENSE file in place and the software and its dependencies aligned with the selected licence, the **licence must be clearly stated** in the documentation, particularly in the README file, especially when using licence options or offering multiple licences. Some licences only require their name or URL to appear in the documentation, but including the full licence text in a dedicated file is standard. A clear statement declaring the chosen licence for the software is necessary. For instance, the GPL family specifies this requirement in the "How to Apply These Terms to Your New Programs" section at the end of the licence text. Apache 2.0 explains this in the "APPENDIX: How to apply the Apache License to your work." Licence statements are generally placed in the NOTICE and README files.

While licence and copyright information usually need not be in every source file header, the licence and funding should be declared and placed in a manner that allows easy access. If the software has a website, the licences should also be stated there. The source code repository may also offer a method for specifying the licence. GitHub and GitLab provide features to declare the licence through the repository's user interface, and GitHub can automatically detect the licence from the LICENSE file in the root folder.

Software authors may choose to offer their work under several alternative licences through dual- or multi-licensing. However, when such software is used in another project, only one of the available licences should be applied, and it should be compatible with the project's licence. All component licences must be compatible with those offered for the main project.

The minimal set of typical documentation files in a software project usually includes a README for project information, a LICENSE for licensing details, and a CONTRIBUTING file for contribution guidelines. The applied licence may also require a CHANGELOG or CHANGES file to track project updates. These files may use markdown, with names ending with ".md". If so, they can be edited using online markdown editors such as Dillinger [57] or StackEdit [58].

The following sections detail compliance requirements, covering the placement of information, compliance with the licences of used code, README, COPYRIGHT, AUTHORS, NOTICE and CHANGELOG files, and copyright and licence notices in the source code.

Good templates and reference examples for well-designed files containing licence information, copyright, and other assets are provided in the repositories of some of the projects that underwent SLA, and a few external ones are available [59]. Each section includes a brief overview of the document type, its purpose, one or more templates, and relevant examples.

An AI chatbot can be used to generate an initial version of a template-based artefact by copying the template into an editor, adapting it if needed, uploading it to the chatbot in use and requesting it to populate the template using an existing artefact (online or attached), project website, or details provided in the prompt. The generated text must be validated and refined as needed.

REMEMBER: using AI prompts can create copyright issues, always creatively transform prompts and verify AI tools generated content.

3.1 Placement of Information

The README file serves as the primary entry point for users and is highlighted by platforms like GitHub, which displays its content when the project's root folder is opened. It should summarise key information, including copyright and licence. Hyperlinks can direct users to other files and external resources, such as the full licence text, project website, or funding information. Online platforms like GitLab, GitHub, and the GÉANT Software Catalogue manage and display information such as the licence via their user interface. In some cases, registration in specialised registries, like research software registries or the WordPress plugin directory, may also be necessary. These platforms often extract relevant information automatically from files like README or COPYRIGHT, and present it in the project profile or metadata.

Creating, reviewing, and updating the copyright statement is essential. Typically, a COPYRIGHT file should state that the software is copyrighted by GÉANT, NRENs, or other organisations. The copyright holder should also be named in the README and, if present, the NOTICE file, and sometimes within the LICENSE file if there is a placeholder for copyright in the licence text or in the headers of source files. Including the copyright statement in the LICENSE file is standard for MIT, BSD, and ISC licences.

The licence typically requires its full text to be included with the project, usually in the LICENSE file. Although the licence text may describe available licensing options, it is not usually modified to reflect the specific choices made in a project. Instead, a statement confirming the applied licence should be included in the README and, if present, the NOTICE file, along with any applicable licence options. The NOTICE file should also contain disclaimers, if needed. For multi-licensed projects, clearly list all licences, explain the terms of choice in the README and LICENSE, and include the full text of each licence (inline in LICENSE or in separate files). Always ensure the code repository's licence information also reflects the applicable licence or licences.

Contributors are listed in the AUTHORS file, which names individuals or organisations contributing to the software, possibly categorising them (e.g., code, documentation, testing). The source code repository, if used regularly, tracks contributions and can provide information for the AUTHORS file. COPYRIGHT, NOTICE, or CONTRIBUTORS files may recognise the project team and individual contributors, but it is best to use the AUTHORS file for data on individuals, unless this is already documented elsewhere.

Funding information may appear in the README, NOTICE, AUTHORS, and COPYRIGHT files, and may include logos or links to sponsor or grant provider websites. It should clearly state whether funding has influenced the project's direction or goals, as is often the case with developments in GÉANT. EC funding information is obligatory if the code was developed using EC funds. Funders should also be acknowledged in detailed documentation.

Contribution guidelines should be documented in the README or CONTRIBUTING files, specifying how contributors can engage with the project, submit changes (including committing, branching, testing, and releasing), and follow the project's coding standards and licence. They may include links to templates, coding style profiles or guidelines, and instructions for different types of contributions.

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

Modules in subfolders may have their own licences. Each subfolder with a different licence from the main project should include a separate LICENSE file, along with necessary attribution or copyright notices. This information is crucial for subprojects or folders differing from the main project. Other important information for subprojects should be provided in their respective folders, similar to the main project. The README, COPYRIGHT, and NOTICE files of unmodified components should remain unchanged.

3.2 Compliance with Licences of Used Code

In addition to adhering to your project's licence requirements, it is essential to meet the obligations of licences governing dependencies or reused code, including copyright and patent rules. This requires ongoing attention throughout development. Key actions include:

- Extending README, COPYRIGHT, and NOTICE files to explicitly declare and credit dependencies or reused code, clearly stating their licences.
- Retaining existing licence and copyright files and notices to ensure full original documentation and compliance with respective licences.
- Attributing and documenting any modifications to reused code, updating modification history and the contributor list as necessary.
- Staying informed about updates to licences for used code, as they may impact compliance requirements and require adjustments.

Eclipse- and Mozilla-licensed dependencies (direct or transitive) require explicit reference in project documentation.

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

- Including the original copyright notice.
- Providing a copy of the Apache licence.
- Describing significant changes made to the original code.
- Maintaining a NOTICE file with attribution notes (original or added ones).

3.3 README File

The README file should contain basic information about the software, including the licence and copyright, typically in one short line each. It should also mention the origin or motivation for the development, credit GÉANT, and refer to COPYRIGHT, LICENSE, and other files for 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 seem obvious to them, and key features.
- Scope, supported settings and environments, requirements or constraints of the application, which may not be apparent to new users.
- Installation and configuration.
- Usage.
- Documentation.
- Roadmap and known issues.
- Community contributions.
- Funding, acknowledgements, dependencies, and tools used.
- Software licence and copyright.

The licence notice and details on licence options (such as “or later”), also possibly included in the NOTICE file, should be mentioned in the README. Markdown examples for stating the licence are:

```
## Licence: [GPLv3 or later](https://www.gnu.org/licenses/gpl-3.0.html)
```

or, in a slightly longer format:

```
## Licence

This project is licensed under the MIT License - see the [LICENSE](LICENSE) file for details.
```

or:

```
## Licence

This software is licensed under the GPLv3 or later. For more details, see the [LICENSE](LICENSE) file.
```

Since README files typically appear on repository home pages, the copyright statement and the EU emblem with the accompanying text about funding should be included. Guidelines for formulating these are detailed in the COPYRIGHT section.

The list of direct dependencies in the README can be shorter than the one in the NOTICE by not including transitive dependencies. Both lists must be maintained. The README list could also include details on tools, libraries, and other software used, as illustrated in the markdown example:

```
## Dependencies

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

- [Library A](https://librarya.com) (MIT) - Used for data processing.

- [Framework B](https://frameworkb.io) (Apache 2.0) - Provides core functionalities.

- [Tool C](https://toolc.org) (GPL 2.0) - Assists in automating tasks.
```

Listing frameworks and tools in the README helps describe the software’s operational environment, supporting users, and contributors who may wish to reproduce the setup or use the same tools.

A helpful README template is available at *Make a README* [60], which provides exemplary markdown and detailed recommendations in the section *Suggestions for a good README*.

3.4 COPYRIGHT File

Software is protected by copyright law. Various OSS licences impose specific requirements under which software developers grant other users specific rights while retaining copyright. Therefore, developers must clearly indicate the copyright.

Copyright management within the project is expressed through a copyright statement, sometimes embedded in the LICENSE file. For short licences such as MIT, BSD, and ISC, the copyright statement is often part of the LICENSE file, typically placed at the beginning. In other cases, it appears in a separate COPYRIGHT file to preserve the integrity of the LICENSE file.

Not all software developed within the GÉANT project correctly indicates copyright, which must be addressed when working on software licensing. While core copyright information can be provided on the project website, in the LICENSE file, and in the README, it is advisable to create a dedicated COPYRIGHT file with more comprehensive details for software created or modified within GÉANT.

The COPYRIGHT file should be placed in the project's root folder, or in the component's root folder if different. It should reference the GÉANT project and specify the years during which the work was carried out, stating that the software is copyrighted by GÉANT and other contributing organisations. Additionally, it should indicate whether any code was reused, adapted, or relied upon from other sources. If contributions come from NRENs that were created independently of GÉANT, or include adapted code from other projects, the original copyright holders should be acknowledged, with their copyrights preserved if present.

If you are creating a COPYRIGHT file in markdown format, Dillinger, StackEdit, or other online markdown editors can be helpful. However, this is unnecessary if the copyright information is short and simple.

An example of a COPYRIGHT.md file with a disclaimer that can be adapted to your needs is provided below:

> Copyright (c) 2024-2025 GÉANT Association on behalf of the GÉANT 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 project, a project that has received funding from the Horizon Europe research and innovation programme.
>
> 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 the GÉANT IPR Policy.
>
> ![Co-funded by the EU](docs/assets/EU_logo.png)
>
> **The GÉANT project is funded by the Horizon Europe research and innovation programme.**

The COPYRIGHT file begins with a copyright statement. It should follow the format “Copyright (c) <Year Range> <Copyright Holder>”, where <Year Range> refers to the year or range of years the software was developed or updated, and <Copyright Holder> is the name of the organisation. For GÉANT, include the project phase in which the licensed software was developed, for example “Copyright (c) 2024-2025 GÉANT Association on behalf of the GÉANT project”.

Additional copyright statements approved by the IPR Coordinator, such as those for contributions independently developed by other partners like NRENs, should be added after the GÉANT Association copyright. Contributors from GÉANT cannot claim copyright. Contact the IPR Coordinator with any questions about copyright.

The template includes a disclaimer of warranty and a limitation of liability clause. If the software is not open source and you wish to retain all rights under copyright law, include the phrase “All rights reserved” instead. Even without this statement, the work is automatically protected by copyright, meaning no one may reproduce, distribute, or adapt any part of the work without permission.

When the code has been developed with EC funding, the EU emblem must be included. To use it, copy one of the images provided with the artefact templates [59] to your own location. These have a white background for consistent display and are available in three sizes. The smallest is sufficient for web use. A good practice is to place the logo in the project repository (e.g., under docs/assets/) and reference it locally. The image URL used in templates is illustrative only, as its location may change in the future, resulting in broken links. It is also a large file with a transparent background, which may render poorly in dark themes or dark mode.

In addition to the COPYRIGHT file, the EU emblem and funding statement must be prominently displayed in the README, project documentation, the application’s “About” page or pop-up, as well as in all public or participant-facing communication materials, such as printed or digital products or websites. The same is recommended for NOTICE and AUTHORS, if present. Their use must follow the guidelines outlined in the GÉANT IPR Policy [61], summarised as follows:

- Minimum emblem height is 1 cm.
- It must include the phrase “Co-funded by the European Union”.
- Alongside the EU emblem, funding information must be provided.
- The EU emblem should be used in conjunction with the name of the programme or fund.
    - For the GN5 project: “The GÉANT project is funded by the Horizon Europe research and innovation programme.”
    - For GN3 and GN4: “The GÉANT project is funded by the Horizon 2020 research and innovation programme.”
    - For developments spanning multiple programmes: “Horizon 2020 and Horizon Europe research and innovation programmes.”
    - Including the project phase, such as “GÉANT project (GN5-2)” is recommended for clarity and traceability.
    - The “Project IP was generated” paragraph in the COPYRIGHT file may need to be updated accordingly, e.g., to “GÉANT (GN4) project” and “Horizon 2020 research and innovation programme.”
- Use a non-serif font without effects.
- The text should be proportionate to the emblem’s size.
- The text should be in EU blue, black, or white, depending on the background.
- Ensure the text does not overlap with the emblem.

Figure 3.1 presents an example of the EU emblem with a reference to GÉANT:

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

Wherever used, the funding statement should consistently reference the GÉANT project as described above and be placed below the EU emblem.

The EU emblem and funding statement must be prominently displayed in the COPYRIGHT and README files, as well as in all public or participant-facing communication materials, such as printed or digital products or websites. The same is recommended for NOTICE and AUTHORS, if present.

3.5 AUTHORS File

Open source projects typically result from the collaborative efforts of various developers or teams, and it is important to acknowledge their contributions.

An AUTHORS (or CONTRIBUTORS) file is a simple way to credit individual contributors and reference the GÉANT work package or project task. This file should be placed in the project's root folder.

You can also include funding acknowledgements as required by the contract or programme. If individual contributor details are not provided, funding credits may be included in the COPYRIGHT file. Some information in the COPYRIGHT and AUTHORS files can also be summarised in the README file and the project's standalone website, as advised by the GÉANT Marcomms Team. However, GÉANT funding credits are not required for software descriptions on GÉANT-branded websites or internal collaboration platforms.

Here is a markdown template for an AUTHORS file:

<Project Name> has been created by the GÉANT <GNx-y> WP<a> Task <b> <Recognisable Task Name>.

Developers

- [<Author 1 Name>](<Author 1 Contact, hopefully working as a link>): <Author 1 Role>

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

Other Contributors

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

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

Funding

- The [GÉANT project](https://geant.org/projects/) is funded by the Horizon Europe research and innovation programme.

To indicate the project's GÉANT origin, you may begin with the statement provided at the start of the above snippet.

When listing individuals, include their email addresses or other contact details (e.g., repository identifiers of contributors, or ORCID identifiers), and a brief description of their contributions or roles, such as "Implemented the core functionality" or "Designed the user interface." This not only acknowledges contributors, but also encourages new ones and facilitates collaboration. Be sure to include individuals significantly involved in the project, even if they did not directly author the code. Developers and contributors should be clearly distinguished from copyright holders listed in the COPYRIGHT file. You may also specify time periods, such as "2021:" and "2020 – present:". Ensure this section is regularly updated as new contributors join the project or make significant contributions.

After funding information, add other acknowledgements as needed. If the project is part of a larger ecosystem or depends on third-party libraries, it is good practice to acknowledge these dependencies either in the README or a separate NOTICE file.

3.6 NOTICE File

The project licence may require a NOTICE file, or you may be modifying a project with an existing one, in which case you should add your entries. Mention all the libraries and frameworks your project uses, and provide links to their respective websites or repositories. Declare and credit any other IP used, along with its licence and licence options.

The Apache License 2.0 recommends including copyright information at the beginning of the NOTICE file, alongside the standard Apache License 2.0 boilerplate text that references the license.

Below is a short markdown example of a NOTICE file listing 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 Name>
>
> - Version: <Dependency Version Number>
>
> - URL: <Dependency URL>
>
> - Licence: <Dependency Licence>
>
> - Copyright (c) <Dependency Year Range> <Dependency Copyright Holder>
>
> We thank all open source developers who contributed to these dependencies.

The content and format of a NOTICE file may vary depending on the project and the licensing requirements of its dependencies. Licences often require a notice stating that <Project Name> is under the chosen licence, with information about applied licence options, secondary licences, exceptions, additional permissions, or disclaimers. Information about these possibilities is usually provided within or below the licence text; the offered boilerplate text typically needs to be adapted and placed at the start of the NOTICE file. Some licences require preserving content from the existing NOTICE files of dependencies. If a NOTICE file is unnecessary, the above information can be provided at the end of the README file. It may be repeated in the README, omitting disclaimers for brevity.

Using Mend to scan the project helps identify overlooked dependencies. However, transitive dependencies listed in Mend reports should not be included in the lists described above.

It is your responsibility to review and comply with the respective licences and attribution requirements for these components, as project dependencies and licence conditions evolve. Acknowledging contributions, dependencies, and tools fosters a collaborative open source community.

3.7 CHANGELOG File

A well-maintained CHANGELOG enables users and developers to track a project's development history, making it easier to follow updates and modifications. It also aids in troubleshooting and identifying potential issues during upgrades. Even if not required by the licences, documenting the history of changes in the CHANGELOG is considered good practice.

Some licences require maintaining and providing a software change history, typically in a file named CHANGELOG, CHANGES, or HISTORY, located in the root directory. Certain licences have strict naming or content requirements. The history should summarise new features, significant changes, additions, bug fixes, and other modifications chronologically, serving as a record of changes made to the software over its releases and time. One common approach to generate the history is by concatenating release notes, which, in turn, can be derived from commit notes, release plans, and internal work documentation.

Entries should be organised with the latest changes at the top, each typically corresponding to a single release. Include release or version numbers or tags (e.g., "1.2.3") as described at *Semantic Versioning* [62] and indicate release (or change) dates. For unreleased changes, consider using "Unreleased" as the version until the release occurs. Summarise changes using bullet points or brief paragraphs, optionally grouping them by type under headings such as "Added," "Changed," "Deprecated," "Removed," "Fixed," and "Security." Alternatively, use indicative prefixes, such as "Added:" for each bullet point. Emphasise significant updates visually.

Systematic tracking of changes through commit notes is extremely helpful. Frequent commits, development in branches, and clear commit notes support this, but the information in the repository may be too specific. It is also important to tag releases with version numbers for easy identification. Other useful sources for tracking changes include release notes, lists of planned features and modifications, developer tasks, and fixed issues. The history should remain meaningful throughout the software's lifetime, so include the reason or context for changes where useful. Change-related pull requests and code reviews may also provide additional context.

Use a consistent format for each entry, such as the following markdown example:

```
## [1.2.3] – 2025-08-01
### Added
- New feature 1
- New feature 2
### Changed
- Update of an existing feature or its implementation
### Fixed
- Bug fix 1
- Bug fix 2
```

Version numbers in markdown should link to the corresponding release tags. Descriptions of changes may reference relevant commit IDs and code files for more detailed information. Markdown linking to specific commits or issues related to changes provides access to further details:

- New feature 1 (#123)
- Bug fix 1 ([<Commit Hash>])

Developers should follow the recommended structure for each version entry and keep the CHANGELOG file up to date with each release. This results in clear, useful records that help users and developers understand the project's history and progress. The Common Changelog [63] outlines a suitable markdown format and provides guidance for writing concise entries. Tools like Keep a Changelog [64] can be integrated with the CI/CD pipeline to automate parts of this process.

The project's documentation should explain that users and contributors can access the CHANGELOG file for release notes, and possibly other artefacts such as the project roadmap.

3.8 Copyright and Licence Notices in Source Code

Placing copyright and licence notices in source code file headers ensures proper attribution and compliance with licence requirements, and may improve transparency for contributors. However, this practice is declining due to redundancy with centralised LICENSE files, maintenance complexity, interaction with version control systems and automation tools when the information is updated, as well as the preference for cleaner and more readable code.

A simple copyright line and licence indicator can be placed in the header of each source file, as shown below. While not mandatory, these notices are helpful if individual files are likely to be accessed, reused, or modified independently, and there are concerns that users or modifiers may overlook the project-level copyright and licence information.

```
// Copyright (c) <Year Range> <Copyright Holder>
// Licensed under the <Licence Name, e.g., "GPLv3 or later">. See
LICENSE file for details.
```

Rare exceptions where a copyright header is required are generally tied to projects' licensing policies rather than licence requirements. For instance, Mozilla's source repositories require appropriate header text for new code [65].

3.9 Checklist for Preparing and Validating Project Files

This checklist provides key steps for preparing and validating project files, based on prior instructions and common issues:

- README
  - After the title with the software name, optionally list key attributes in bullet points.
  - Use subtitles to structure the content.
  - Briefly describe software purpose, motivation, usage context, and prerequisites.
  - Provide brief installation, configuration, and usage instructions. Screenshots are encouraged. Include key notes on user management, integration, and security.
  - Clearly state the licence in one line, noting if options like “or later” apply. Reference the LICENSE file or official licence URL.
  - Include copyright in a single line or paragraph, and refer to the COPYRIGHT file.
  - Mention the software’s origin or motivation.
  - Credit GÉANT and relevant parties, such as NRENs or external partners.
  - Include the EU logo and required text.
  - To keep the file short, refer to other files or documentation for further details.
- LICENSE
  - Confirm the licence with the GÉANT IPR Coordinator after ensuring component compatibility.
  - Copy the full licence text from the official source. Some licences include copyright placeholders to be filled.
- COPYRIGHT
  - Verify copyright and IPR details with the GÉANT IPR Coordinator.
  - Include the EU logo and required text. Host the logo as a static image in a controlled location.
- AUTHORS (Optional)
  - List authors and contributors, including contact details (email, ORCID, or repository ID). Optionally include their institutions and roles.
  - Optionally credit the GÉANT project phase, work package, or task while keeping copyright holders in the COPYRIGHT file.
- NOTICE (Optional)
  - Not necessary if all licence-mandated information is in the README.
  - Acknowledge third-party libraries, listing components, their licences, URLs, and copyrights.
  - Optionally mention tools used in the project.
  - Include mandatory notices for third-party components when required (e.g., for Apache 2.0 licensed components).
- CHANGELOG (Recommended, often mandatory)
  - Maintain a concise history of changes. This is a requirement for many licences and is good practice.
  - Use a standard format.
  - For the first version, simply note it as the initial public or production release.

Refer to *Templates and Examples for Software Project Artefacts* [66] for guidance. A more detailed checklist is available with further instructions [67].

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 [68]
- Webinar: License Dependencies Analysis with WhiteSource [69]
- Training course: Introduction to Open Source Licensing and Compliance 2023 [70]
- Infoshare: Software Licences Management in GÉANT [71]

4.3 Further Reading

- GÉANT IPR Policy [72]
- *OSS Licences and Licence Selection* [73] – introductory guide
- *OSS Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* [74] – project-oriented white paper for GÉANT participants, providing guidance on licence selection, with an appendix describing the OSS licences most frequently used by analysed projects
- *Software Licence Selection and Management in GÉANT* [75] – maintained online version of this guide
- *Open Source Licences Used in GÉANT* [76] – descriptions and compatibility of commonly used licences in GÉANT
- *Templates and Examples for Software Project Artefacts* [77] – per-artefact Markdown templates, and links to examples from various projects
- *Reference Information about OSS Licences and Tools* [78] – a comprehensive catalogue of thematically organised resources and pointers
- GÉANT software best practice *BP-B.6: Manage sideground IPR* [79]
- OSI Approved Licenses [80]

4.4 Services

- GÉANT Software Licence Management (homepage) [81]
- Jira requests for SCA and SLA [82]
- *Software Reviews* [83] – GÉANT Software Development Support
- GÉANT Mend (login via GÉANT SSO; special permissions may apply) [84]
- *Accessing Mend and visibility levels* [85] – on the visibility of Mend scan results
- *Mend short guide for end users* [86] – includes interpretation of Mend reports
- The Risk Report [87] – short end-user guide on Mend risk report
- GÉANT GitLab [88]
- GÉANT Software Catalogue [89]

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
DCO	Developer Certificate of Origin
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
ICT	Information and Communication Technology
IP	Intellectual Property
IPR	Intellectual Property Rights
ISC	Internet Software Consortium
ISO	International Organisation for Standardisation
JLA	Joinup Licensing Assistant
LGPL	GNU Lesser General Public License
MIT	Massachusetts Institute of Technology
MPL	Mozilla Public License
NC	NonCommercial
ND	NoDerivatives
NREN	National Research and Education Network
ORCID	Open Researcher and Contributor ID
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
SaaS	Software as a Service
SBOM	Software Bill of Materials
SCA	Software Composition Analysis
SLA	Software Licence Analysis
UI	User Interface
WP9	Work Package 9 Operations Support
WP9 Task 2	WP9 Task 2 Software Governance and Support

References

[1] https://resources.geant.org/wp-content/uploads/2022/09/GEANT-_IPR_Policy_2022.pdf
[2] https://resources.geant.org/publications/intellectual-property/
[3] Community Edition hosting most projects: https://gitlab.software.geant.org/public
Ultimate Edition hosting selected projects: https://gitlab.geant.org/
[4] https://www.tiny.cloud/software-evaluation-criteria-checklist/
[5] https://www.redhat.com/en/resources/open-source-project-health-checklist
[6] https://technical-reference.readthedocs.io/en/latest/quality/software-checklist.html
[7] https://resources.geant.org/wp-content/uploads/2024/04/GN5-1_Software-Licence-Selection-and-Management-in-GEANT.pdf
[8] https://resources.geant.org/wp-content/uploads/2024/04/GN5-1_D9-4_Open-Source-and-Licence-Support-Report.pdf
[9] GÉANT IPR Policy 2022 – See [1]
[10] https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant/jla-find-and-compare-software-licenses
[11] JLA – Find and Compare Software Licenses – See [10]
[12] https://wiki.geant.org/display/GSD/Reference+information+about+OSS+licences+and+tools
[13] https://wiki.geant.org/display/GSD/OSS+licences+and+licence+selection
[14] https://wiki.geant.org/pages/viewpage.action?pageId=1036025888
[15] GÉANT IPR Policy 2022 – See [1]
[16] https://opensource.org/license
[17] https://www.gnu.org/licenses/license-list.html
[18] https://wiki.geant.org/display/GSD/Software+Reviews
[19] https://jira.software.geant.org/servicedesk/customer/portal/2/create/55
[20] OSS Licences and Licence Selection – See [13]
[21] Open Source Licences Used in GÉANT – See [14]
[22] https://geantprojects.sharepoint.com/sites/plm
[23] Software Reviews – See [18]
[24] https://wiki.geant.org/pages/viewpage.action?pageId=599785535
[25] https://www.mend.io/sca/
[26] https://wiki.geant.org/display/GSD/Mend+short+guide+for+end+users
[27] https://wiki.geant.org/pages/viewpage.action?pageId=240844905
[28] https://docs.mend.io/bundle/sca_user_guide/page/understanding_risk_score_attribution_and_license_analysis.html#Risk-Score-Attribution
[29] https://github.com/
[30] GÉANT GitLab – See [3]
[31] https://bitbucket.software.geant.org/repos?visibility=public
[32] https://bamboo.software.geant.org/
[33] https://wiki.geant.org/pages/viewpage.action?pageId=219938818
[34] https://wiki.geant.org/display/GSD/Reference+information+about+OSS+licences+and+tools#ReferenceinformationaboutOSSlicencesandtools-Othersoftwarecompositionanalysis(SCA,softwareinventory)tools
[35] JLA – Find and Compare Software Licenses – See [10]
[36] https://wiki.geant.org/display/GSD/Software+Reviews
[37] https://e-academy.geant.org/moodle/course/view.php?id=214
[38] GN5-1 Deliverable D9.4 Open Source and Licence Support Report – See [8]

[39] GÉANT GitLab – See [3]
[40] GitHub – See [29]
[41] https://sc.geant.org/
[42] https://jira.software.geant.org/servicedesk/customer/portal/2/create/55
[43] Open Source Licences Used in GÉANT – See [14]
[44] OSS Licences and Licence Selection – See [13]
[45] https://wiki.geant.org/pages/viewpage.action?pageId=633275197
[46] Open Source Licences Used in GÉANT – See [14]
[47] https://github.com/filesender/filesender/blob/development/CONTRIBUTE.md
[48] https://github.com/atom/atom/blob/master/CONTRIBUTING.md
[49] https://wiki.geant.org/display/G52W9T2/Evaluation+Survey+-final
[50] Other Software Composition Analysis Tools – See [34]
[51] https://github.com/mojohaus/license-maven-plugin
[52] https://wiki.geant.org/display/GSD/BP-B.6%3A+Manage+sideground+IPR
[53] https://wiki.geant.org/display/GSD/Software+Reviews
[54] https://wiki.geant.org/display/GSD/Secure+Code+Training
[55] https://security.geant.org/
[56] https://www.mend.io/blog/guide-to-standard-sbom-formats/
[57] https://dillinger.io/
[58] https://stackedit.io/
[59] https://wiki.geant.org/display/G52W9T2/Templates+and+Examples+for+Software+Project+Artefacts
[60] https://www.makeareadme.com/
[61] GÉANT IPR Policy 2022 – See [1]
[62] https://semver.org/
[63] https://common-changelog.org/
[64] https://keepachangelog.com/
[65] https://www.mozilla.org/en-US/MPL/headers/
[66] Templates and Examples for Software Project Artefacts – See [59]
[67] https://wiki.geant.org/display/G52W9T2/Software+Artefacts+Checklist
[68] https://e-academy.geant.org/moodle/course/view.php?id=214
[69] https://e-academy.geant.org/moodle/course/view.php?id=220
[70] https://e-academy.geant.org/moodle/course/view.php?id=478
[71] https://wiki.geant.org/pages/viewpage.action?pageId=633276866
[72] GÉANT IPR Policy 2022 – See [1]
[73] OSS Licences and Licence Selection – See [13]
[74] Open Source Software Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations – See [45]
[75] https://wiki.geant.org/pages/viewpage.action?pageId=1032978710
[76] Open Source Licences Used in GÉANT – See [14]
[77] Templates and Examples for Software Project Artefacts – See [59]
[78] Reference Information about OSS Licences and Tools – See [12]
[79] Manage Sideground IPR – See [50]
[80] https://opensource.org/license
[81] https://wiki.geant.org/display/GSD/Software+Licence+Management
[82] https://jira.software.geant.org/servicedesk/customer/portal/2/create/55
[83] https://wiki.geant.org/display/GSD/Software+Reviews
[84] https://app-eu.whitesourcesoftware.com
[85] https://wiki.geant.org/display/gn51wp9t2/Accessing+Mend+and+visibility+levels
[86] https://wiki.geant.org/display/GSD/Mend+short+guide+for+end+users
[87] https://docs.mend.io/bundle/sca_user_guide/page/the_risk_report.html
[88] Community Edition hosting most projects: https://gitlab.software.geant.org/public
Ultimate Edition hosting selected projects: https://gitlab.geant.org/
[89] https://sc.geant.org/