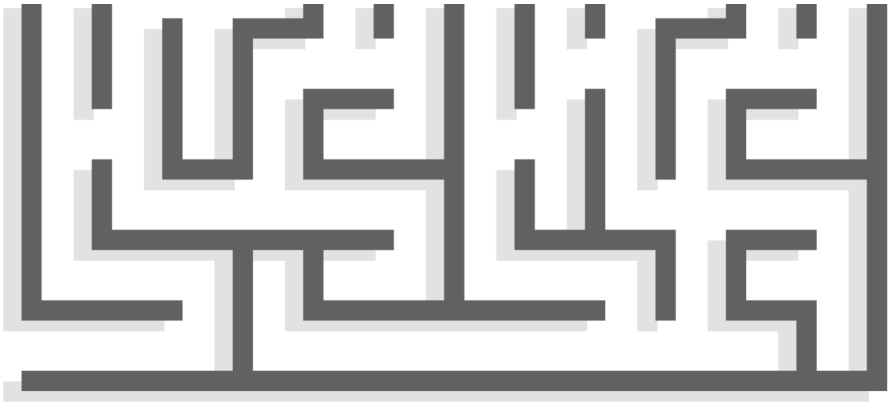
OPEN SOURCE COMPLIANCE IN THE ENTERPRISE



THE LINUX FOUNDATION

Ibrahim Haddad, PhD

This page intentionally left blank.

Ibrahim Haddad, Ph.D.

Open Source Compliance in the Enterprise

The Linux Foundation

2016

Contents

PREFACE

My involvement with open source compliance started early in my career as a software developer, and has been a part of my job directly or indirectly for two decades now. Throughout my journey working with open source software, it was difficult to find practical references on open source compliance. My interest grew in making my own experiences available so that others could possibly learn from them, and then publish their experiences, so that as an industry we can all strive towards better ways to achieve open source compliance while minimizing impact on engineering resources and product delivery timelines.

This handbook summarizes my experience driving open source compliance activities in the enterprise, and focuses on practical aspects of creating and maintaining an open source compliance program. Since most of my experience was focused in the embedded space (with C and C++ being the dominant programming languages), this emphasis comes across throughout this handbook.

I hope you find it useful in your day-to-day drive to achieve open source compliance.

Foreword

Open source has expanded not only from an idealistic movement led by individuals around software and intellectual property but from one where organizations (e.g., governments, companies, and universities) realize that open source is a key part of their IT strategy and want to participate in its development. Early success in Linux and other open source technologies has spread to all areas of technology.

More traditional organizations are also taking notice; they are making open source software a priority and using the software for strategic advantage in their operations.

Use of open source in enterprise IT has doubled since 2010.

78% of surveyed companies run their businesses on open source.

64% currently participate in open source projects.

39% plan to launch their own open source projects.

North Bridge & Black Duck
"The 2015 Future of Open Source Survey"

"Open Source First: Simply put, any solution developed using taxpayer dollars should be in the taxpayer's domain (open source). At GSA, we believe that all code we developed should be shared under an open license so others may benefit from it. In addition, we will give priority to using open source software as we design now solutions."
Office of the CIO, U.S. General Services Administration
(U.S. agency that oversees $66 billion of procurement annually)

"The development of Blockchain technology has the potential to redefine the operations and economics of the financial services industry. It emerges at an important time, as the industry strives to be leaner, more efficient, and more digital. Open source development will accelerate the innovation and help drive the scalability of this technology, and we are proud to support the Hyperledger Project."

Richard Lumb, Chief Executive, Financial Services, Accenture

"From increasing member investments to a growing, vibrant developer community, the Dronecode Project's first year has been extremely exciting. By bringing efforts together to establish a common platform and utilizing open source best practices, we're able to build the foundation for a new era of drone applications that extend from the camera to the cloud. The Dronecode 'full-stack' platform approach, combined with the hardware and software innovations of its members, will bring about a new generation of drones that are autonomous, aware of their environments, and continuously connected — an airborne Internet of Things."

Chris Anderson, CEO, 3DR

(Former Editor in Chief of Wired magazine and author of "The Long Tail")

"Open source is essential to our development process. It's a powerful approach that lets people work together to build great solutions while realizing shared benefits."

Rob Alexander, CIO, Capital One

Organizations are looking for guidance on how best to participate appropriately in open source communities and to do so in a legal and responsible way. Participants want to share their code and IP, and they need a trusted neutral home for IP assets (trademark, copyright, patents). They also need a framework to pool resources (financial, technical, etc.).

Participants need expertise to train them how to collaborate with their competitors in an effective manner. To that end, this book is geared to creating a shared understanding on the best ways to create shared value and innovation while adhering to the spirit and legal particulars of open source licensing.

This page intentionally left blank.

Chapter 1

INTRODUCTION TO OPEN SOURCE COMPLIANCE

A CHANGING BUSINESS ENVIRONMENT

Traditionally, platforms and software stacks were implemented using proprietary software, and consisted of various software building blocks that originated as a result of internal development or via third-party software providers with negotiated licensing terms. The business environment was predictable and companies mitigated potential risks through license and contract negotiations with the software vendors. It was very easy to know who was the provider for every software component. Figure 1 illustrates the major building blocks of a traditional hardware and software platform.

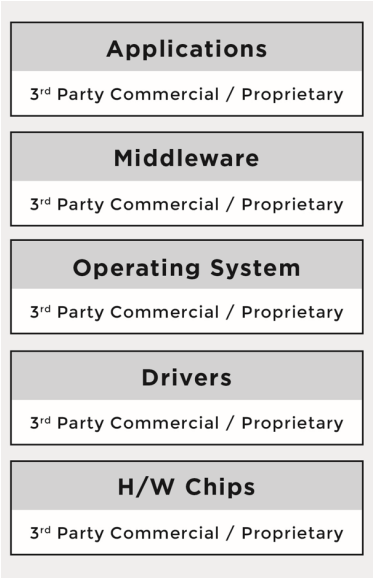


Figure 1. A simplified architecture of a traditional software platform that relies on proprietary software building blocks

Over time, companies started to incorporate open source software into their platforms and software stacks due to the advantages it offers. The reasons varied from product to product, but the common theme across industries was that open source components provided compelling features out of the box, there were meaningful economies to be gained through distributed development that resulted in a faster time-to-market, and they offered a newfound ability to customize the source code. As a result, a new multi-source development model began to emerge.

Under the new model, a product could now have any combination of:

- Proprietary code, developed by the company building the product/service
- Proprietary code, originally developed by the company under an open source license in the process of integrating and deploying open source components, but was not contributed back to the upstream open source project
- Third-party commercial code, developed by third-party software providers and received by the company building the product/service under a commercial license
- Open source code, developed by the open source community and received by the company building the product/service under an open source license.

Figure 2 (next page) illustrates the multi-source development model and the various combinations of sources for incoming source code.

Under this development model, software components can consist of source code originating from any number of different sources and be licensed under different licenses; for instance, software component A can include proprietary source code in addition to third-party proprietary source code, while software component B can include proprietary source code in addition to source code from an open source project.

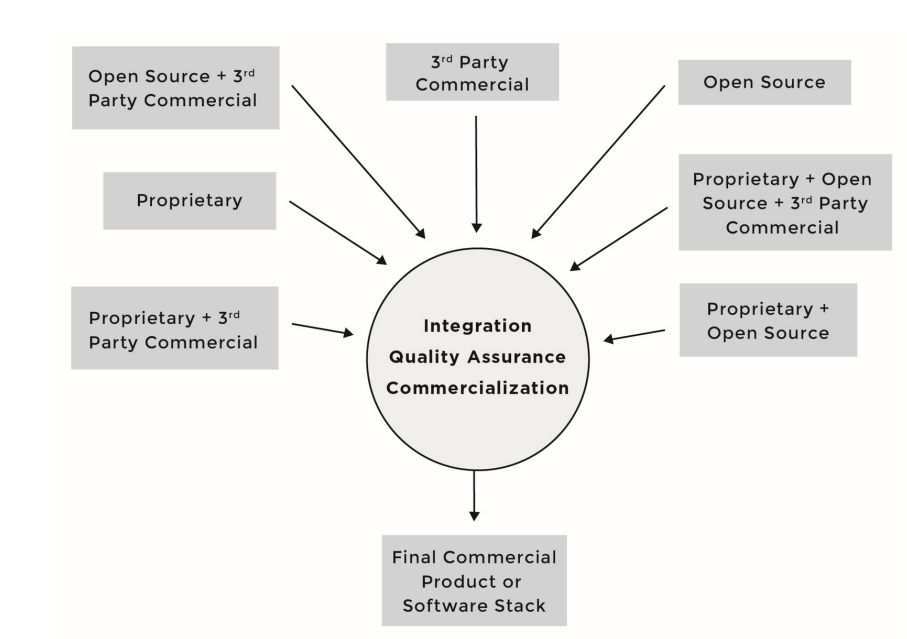


Figure 2. Multi-Source development model

As the number of open source software components grew in what were once straightforward proprietary software stacks, the business environment diverged from familiar territory and corporate comfort zones.

Figure 3 (next page) illustrates the adoption of open source software throughout the various levels of a given platform or software stack.

One of the major differences between the proprietary and the multi-source development models has been that the licenses of open source software are not negotiated. There are no contracts to sign with the software providers (i.e., open source developers or projects). Rather, the individuals who initiate the project chose a given open source license, and once a project reaches a certain scale, the licenses are virtually impossible to change. When using the multi-source development model, companies must understand the implications of tens of different licenses (and combinations of licenses) coming from hundreds or even thousands of licensors or contributors (copyright holders). As a result, the risks that companies previously managed through company-to-company license and agreement

negotiations are now managed through robust compliance programs and careful engineering practices.

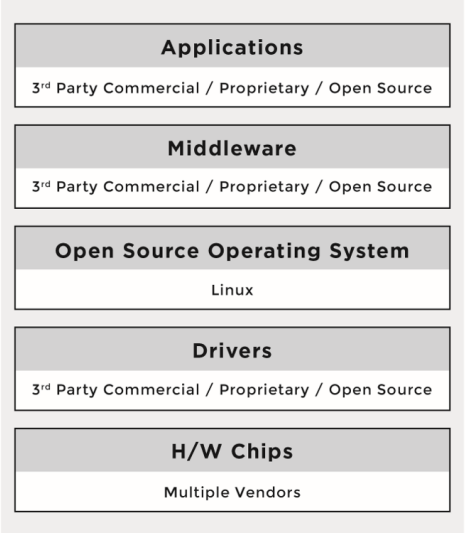


Figure 3. A simplified view of the architecture of a modern software platform, showing the proliferation of open source inside each of the software building blocks.

ENTER OPEN SOURCE COMPLIANCE

Open source initiatives and projects provide companies and other organizations with a vehicle to accelerate innovation through collaboration with the hundreds and sometimes thousands of communities that represent the developers of the open source software. However, there are important responsibilities accompanying the benefits of teaming with the open source community: Companies must ensure compliance to the obligations that accompany open source licenses.

Open source compliance is the process by which users, integrators, and developers of open source observe copyright notices and satisfy license obligations for their open source software components. A well-designed open source compliance process should simultaneously ensure compliance with the terms of open source licenses and also help companies protect

their own intellectual property and that of third-party suppliers from unintended disclosure and/or other consequences.

Open source compliance helps achieve four main objectives:

- Comply with open source licensing obligations.
- Facilitate effective use of open source in commercial products.
- Comply with third-party software supplier contractual obligations.

Benefits of Ensuring Open Source Compliance

There are several benefits to achieving open source compliance. Companies that maintain a steady-state compliance program often gain a technical advantage, since compliant software portfolios are easier to service, test, upgrade, and maintain. In addition, compliance activities can also help identify crucial pieces of open source that are in use across multiple products and parts of an organization, and/or are highly strategic and beneficial to that organization. Conversely, compliance can demonstrate the costs and risks associated with using open source components, as they will go through multiple rounds of review.

A healthy compliance program can deliver major benefits when working with external communities as well. In the event of a compliance challenge, such a program can demonstrate an ongoing pattern of acting in good faith.

Finally, there are less-common ways in which companies benefit from strong open source compliance practices. For example, a well-founded compliance program can help a company be prepared for possible acquisition, sale, or new product or service release, where open source compliance assurance is a mandatory practice before the completion of such transactions. Furthermore, there is the added advantage of verifiable compliance in dealing with OEMs and downstream vendors.

FAILURE TO COMPLY

Throughout the software development, errors and limitations in processes can lead to open source compliance failures. Examples of such failures include:

- **Failure to provide a proper attribution notice.** An attribution notice is usually provided as a text file together with the open source component that provides acknowledgement as supplied by the contributors of open source components.
- **Neglecting to provide a license notice.** A license notice is a file that includes the open source license text included in the product or stack and is typically provided with product documentation and/or within the product or application user interface.
- **Omission of a copyright notice.** A copyright notice is an identifier placed on copies of the work to inform the world of copyright ownership.
- **Failure to provide a modification notice.** A modification notice calls out modifications to the source code in a change log file, such as those required by the GPL and LGPL. An example of a modification notice is shown below:

```
/*
* Date         Author           Comment
* 10/15/2015   Ibrahim Haddad    Fixed memory leak in nextlst()
*/
```

- **Making inappropriate or misleading statements** in the product documentation or product advertisement material.
- **Failure to provide the source code.** Making source code available (including the modifications) is one of the requirements of the GPL/LGPL family of licenses.

- **Failure to provide a written offer for example when using GPL/LGPL license source code.** A written notice provides the end users of the product with information on open source software included in the product and how to download source code that is eligible to distribution. It is usually provided as part of the product documentation and also accessible from the product's user interface. A basic example of a written offer would look as such:

  > *To obtain a copy of the source code being made publicly available by FooBar, Inc. related to software used in this FooBar product, you can visit http://opensource.foobar.com or send your request in writing by email to opensource@foobar.com or by regular postal mail to:*
  >
  > *FooBar Inc.*
  > *Open Source Program Office*
  > *Street Address*
  > *City, State, Postal Code*
  > *Country*

- **Failure to provide the build scripts** needed to compile the source code (per GPL and LGPL family of licenses).

Intellectual Property Failures

Table 1 (next page) provides examples of common accidental admixture of proprietary and open source IP that can arise during the software development process leading to license compliance issues. These problems most commonly involve mixing source code that is licensed under incompatible or conflicting licenses (e.g., proprietary, third-party, and/or open source). Such admixtures may result in companies being forced to release proprietary source code under an open source license, thus losing control of their (presumably) high-value intellectual property and diminishing their capability to differentiate in the marketplace.

The intellectual property failures can lead to one or more of the following results:

- An injunction preventing a company from shipping the product until the compliance issue has been resolved
- A requirement to distribute proprietary source code that corresponds to the binaries in question under an open source license (depending on the specific case)
- A significant re-engineering effort to eliminate the compliance issues
- Embarrassment with customers, distributors, third party proprietary software suppliers and an open source community

Table 1. Examples of intellectual property failures

Problem Type	How Discovered	How to Avoid
Inserting open source code into proprietary or 3rd party code Occurs during development process when developers copy/paste open source code (aka “snippets”) into proprietary or 3rd party source code	By scanning the source code for possible matches with open source code	Offer training to increase awareness of compliance issues, open source (OS) licenses, implications of including OS code in proprietary or 3rd party code Conduct regular code scans of all project source code for unexpected licenses or code snippets. Require approval to use OS software before committing it into product repository

Problem Type	How Discovered	How to Avoid
Linking of open source into proprietary source code (or vice versa – specific to C/C++ source code) Occurs as a result of linking software components that have conflicting or incompatible licenses	With a dependency-tracking tool that allows discovery of linkages between different software components; ID if type of linkage is allowed per com-pany's OS policies	Offer training on linkage scenarios based on company compliance policy Regularly run dependency tracking tool to verify all linkage relationships; flag any issues not in line with compliance policies
Inclusion of proprietary code into an open source component Occurs when developers copy/paste proprietary source code into OS software	By scanning source code. Tool will ID source code that doesn't match what's provided by OS component, triggering various flags for Audit	Train the staff Conduct regular source code inspections Require approval to include proprietary source code in OS components

License Compliance Problems

License compliance problems are typically less damaging than intellectual property problems, as they don't have the side effect of forcing you to release your proprietary source code under an open source license.

License compliance failures may result in any (or a combination) of the following:

- An injunction preventing a company from shipping a product until source code is released.

- Support or customer service headaches as a result of version mismatches (as a result of people calling or emailing the support hotline and inquiring about source code releases).
- Embarrassment and/or bad publicity with customers and open source community.

Table 2 provides examples of the most common license compliance problems that occur during the software development process, and offers tips on how to avoid them.

Table 2. Examples of license compliance problems and how to avoid them

Problem Type	How to Avoid
Failure to publish or make available source code packages as part of meeting license obligations	Follow a detailed compliance checklist to ensure that all compliance action items have been completed when a given product, application, or software stack is released into the market
Failure to provide correct version of the source code corresponding to the shipped binaries	Add a verification step into the compliance process to ensure that you're publishing the version of source code that exactly corresponds to the distributed binary version
Failure to release modifications that were introduced to the open source software being incorporated into the shipping product	Use a bill of material (BOM) difference tool that allows the identification of software components that change across releases Re-introduce the newer version of the software component in the compliance process Add the "compute diffs" of any modified source code (eligible for open source distribution) to the checklist item before releasing open source used in the product

Problem Type	How to Avoid
Failure to mark open source code that has been changed or to include a description of the changes	Add source code marking as checklist item before releasing source code to ensure you flag all the source code introduced to the original copy you downloaded Conduct source code inspections before releasing the source code Add milestone in compliance process to verify modified source code has been marked as such Offer training to staff to ensure they update the change logs of source code files as part of the development process

Process Failures

Process failures can lead to infringement of the open source licensing terms such as the inability to meet the license obligations. Table 3 (next page) lists the most common compliance process failures that occur during the stages of the software development process, and discusses how to avoid them.

Table 3. Sample process compliance failures

Failure	How to Avoid
Failure of developers to request approval from the internal open source committee (sometimes called Open Source Review Board) to use open source software, or failure to submit a request in time	Offer training on your compliance policies and processes Conduct periodic full scans of software platform to detect any OS not corresponding to a given approval form. If OS component is found in the build system without a corresponding compliance ticket, a new ticket is auto-generated. (This assumes companies rely on specific workflow implemented in tools such as Bugzilla to track compliance of SW components.) Include compliance in performance reviews; e.g., failure to abide by the compliance policies directly affects employees' bonuses Mandate that developers file approval requests early, even if they didn't yet decide on adoption of OS code
Failure to take the open source training	Ensure completion of OS training is part of employees' professional development plan and is monitored for completion as part of the performance review process
Failure to audit the source code	Provide proper training to compliance staff Conduct periodic source code scans Ensure that auditing is a milestone in the iterative development process Provide proper level of staffing so as not to fall behind in the audit schedule
Failure to resolve the audit findings	Don't allow compliance tickets to be resolved if audit report isn't finalized. Compliance ticket is closed only if no open subtasks are attached to it

LESSONS LEARNED

In the past few years, we have witnessed several cases of non-compliance that made their way to the public eye. Increasingly, the legal disposition towards non-compliance has lessons to teach open source professionals — lessons that we will explore in following subsections.

Ensure Compliance Prior to Product Shipment/Service Launch

The most important outcome of non-compliance cases has been that the companies involved ultimately had to comply with the terms of the license(s) in question, and the costs of addressing the problem after the fact has categorically exceeded those of basic compliance. Therefore, it is really a smart idea to ensure compliance before a product ships or a service launches.

It is important to acknowledge that compliance is not just a legal-department exercise. All facets of the company must be involved in ensuring proper compliance and contributing to correct open source consumption and, when necessary, redistribution. This involvement includes establishing and maintaining consistent compliance policies and procedures as well as ensuring that the licenses of all the software components in use (proprietary, third-party, and open source) can co-exist before shipment or deployment. To that effect, companies need to implement an end-to-end open source management infrastructure that will allow them to:

- Identify all open source used in products, presented in services, and/or used internally
- Perform architectural reviews to verify if and how open source license obligations are extending to proprietary and third-party software components
- Collect the applicable open source licenses for review by the legal department

- Develop open source use and distribution policies and procedures
- Mitigate risks through architecture design and engineering practices

Non-Compliance is Expensive

Most of the public cases related to non-compliance have involved GPL source code. Those disputes reached a settlement agreement that included one or more of these terms:

- Take necessary action to become compliant.
- Appoint a Compliance Officer to monitor and ensure compliance.
- Notify previous recipients of the product that the product contains open source software and inform them of their rights with respect to that software.
- Publish licensing notice on company website.
- Provide additional notices in product publications.
- Make available the source code including any modifications applied to it (specific to the GPL/LGPL family of licenses).
- Cease binary distribution of the open source software in question until it has released complete corresponding source code or make it available to the specific clients affected by the non-compliance.
- In some cases, pay an undisclosed amount of financial consideration to the plaintiffs.

Furthermore, the companies whose compliance has been successfully challenged have incurred costs that included:

- Discovery and diligence costs in response to the compliance inquiry, where the company had to investigate the alleged inquiry and perform due diligence on the source code in question
- Outside and in-house legal costs
- Damage to brand, reputation, and credibility

In almost all cases, the failure to comply with open source license obligations has also resulted in public embarrassment, negative press, and damaged relations with the open source community.

Relationships Matter

For companies using open source software in their commercial products, it is recommended to develop and maintain a good relationship with the members of the open source communities that create and sustain the open source code they consume. The communities of open source projects expect companies to honor the licenses of the open source software they include in their products. Taking steps in this direction, combined with an open and honest relationship, is very valuable.

Training is Important

Training is an essential building block in a compliance program, to ensure that employees have a good understanding of the policies governing the use of open source software. All personnel involved with software need to understand the company's policies and procedures. Companies often provide such education through formal and informal training sessions.

Chapter 2

ESTABLISHING AN OPEN SOURCE MANAGEMENT PROGRAM

An open source management program provides a structure around all aspects of open source software, including selection, approval, use, distribution, audit, inventory, training, community engagement, and public communication. This chapter provides a high-level overview of the various elements in an open source management program, surveys the challenges in establishing a new compliance program, and provides advice on how to overcome those challenges.

OPEN SOURCE COMPLIANCE PROGRAM

We'll begin this chapter with an overview of the core elements needed in a successful open source compliance program. This section, including Figure 4, will provide an overview of these essential elements.

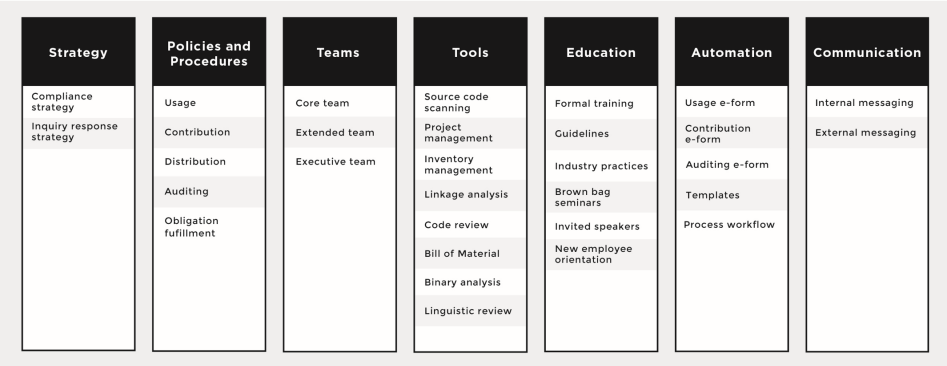


Figure 4. Essential elements of an open source management program

Compliance Strategy

The open source compliance strategy drives the business-based consensus on the main aspects of the policy and process implementation. If you do not start with that high-level consensus, driving agreement on the details of the policy and on investments in the process tends to be very hard, if not impossible. The strategy establishes what must be done to ensure compliance and offers a governing set of principles for how personnel interact with open source software. It includes a formal process for the approval, acquisition, and use of open source, and a method for releasing software that contains open source or that's licensed under an open source license.

Inquiry Response Strategy

The inquiry response strategy establishes what must be done when the company's compliance efforts are challenged. Several companies received negative publicity — and some were formally challenged — because they ignored requests to provide additional compliance information, did not know how to handle compliance inquires, lacked or had a poor open source compliance program, or simply refused to cooperate with the inquirer. None of these approaches is fruitful or beneficial to any of the parties involved. Therefore, companies should have a process in place to deal with incoming inquiries, acknowledge their receipt, inform the inquirer that they will be looking into it, and provide a realistic date for follow-up. In a later chapter, we discuss a simple process for managing open source compliance inquiries.

Policies and Processes

The open source compliance policy is a set of rules that govern the management of open source software (both use of and contribution to). Processes are detailed specifications as to how a company will implement these rules on a daily basis. Compliance policies and processes govern the various aspects of using, contributing, auditing, and distribution of open source software. Figure 5 (next page) illustrates a sample compliance process, with the various steps each software component will go through as

part of the due diligence. This process will be discussed in detail in a later chapter.

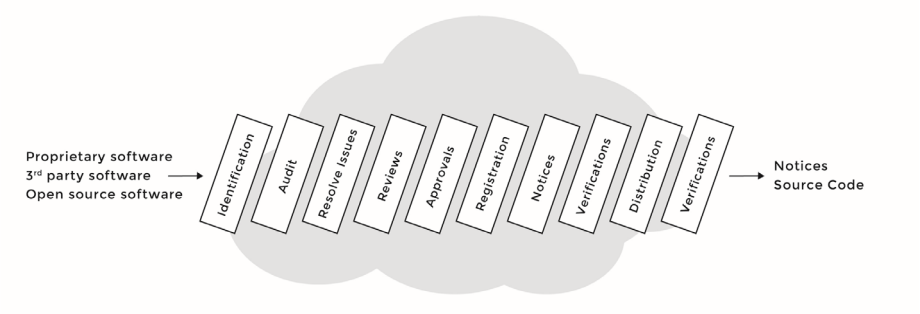


Figure 5. Sample compliance due-diligence process

Compliance Teams

The open source compliance team is a cross-disciplinary group consisting of various individuals tasked with the mission of ensuring open source compliance. The core team, often called the Open Source Review Board (OSRB), consists of representatives from engineering and product teams, one or more legal counsel, and the Compliance Officer. The extended team consists of various individuals across multiple departments that contribute on an ongoing basis to the compliance efforts: Documentation, Supply Chain, Corporate Development, IT, Localization and the Open Source Executive Committee (OSEC). However, unlike the core team, members of the extended team are only working on compliance on a part-time basis, based on tasks they receive from the OSRB. Chapter 3 provides a detailed discussion on the roles and responsibilities of individuals involved in achieving open source compliance.

Figure 6 (next page) illustrates the pair of teams involved in achieving compliance: the core team and the extended team.

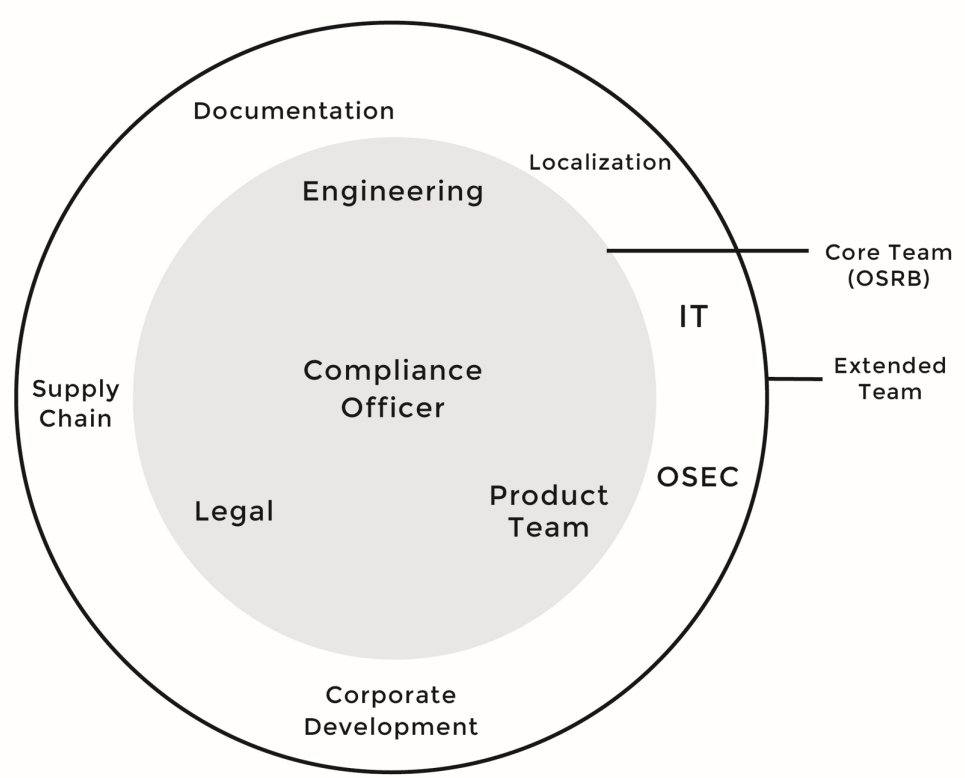


Figure 6. Individuals and teams involved in ensuring open source compliance

Tools

Open source compliance teams use several tools to automate and facilitate the auditing of source code and the discovery of open source code and its licenses. Such tools include:

- A compliance project management tool to manage the compliance project and track tasks and resources.
- A software inventory tool to keep track of every single software component, version, and product that uses it, and other related information.

- A source code and license identification tool to help identify the origin and license of the source code included in the build system.
- A linkage analysis tool to identify the interactions of any given C/C++ software component with other software components used in the product. This tool will allow you to discover linkages between source code packages that do not conform to company policy. The goal is to determine if any open source obligations extend to proprietary or third party software components. If a linkage issue is found, a bug ticket is assigned to Engineering with a description of the issue in addition to a proposal on how to solve the issue.
- A source code peer review tool to review the changes introduced to the original source code before disclosure as part of meeting license obligations.
- A bill of material (BOM) difference tool to identify the changes introduced to the BOM of any given product given two different builds. This tool is very helpful in guiding incremental compliance efforts.

Web Presence

Companies use portals in two directions: inwards, inside the company; and outwards, as a window to the world and the open source community. The internal portal hosts the compliance policies, guidelines, documents, training, announcements, and access to mailing lists. The external portal offers a public platform for the world and the open source community, as well as a venue to post source code of open source packages, acknowledgements, and other disclosures, in fulfillment of license obligations.

Education

Education is an essential building block in a compliance program, to help ensure that employees possess a good understanding of policies governing the use of open source software. The goal of providing open source and compliance training — formally or informally — is to raise awareness of open source policies and strategies and to build a common understanding around the issues and facts of open source licensing as well as the business and legal risks of incorporating open source software in products and/or software portfolios. Training also serves as a venue to publicize and promote the compliance policy and processes within the organization and to foster a culture of compliance.

Formal Training

Depending on the size of the company and the extent to which open source is included in its commercial offerings, the company can mandate that employees working with open source take formal instructor-led courses, possibly culminating in actual exams.

Informal Training

Informal training channels may include any or all of the following:

- **Brown bag seminars:** Brown bag seminars are usually presentations made during lunchtime by a company employee or an invited speaker. The goal of these seminars is to present and evoke discussions of the various aspects of incorporating open source in a commercial product or an enterprise software portfolio. These sessions can also include discussions of the company's compliance program, policies, and processes.
- **New employee orientation:** In some instances, the Compliance Officer presents on the company's compliance efforts, rules, policies, and processes to new employees as part of employee orientation, supplying new employees with necessary open source management information: who to talk to, what internal website to visit, how to sign-up for open source and compliance training, etc.

Automation

Developers who wish to use or contribute to open source software will be requested to submit online requests and get proper approvals. This process is best managed via an automated online system, commonly a bug tracker that has a specifically designed workflow to accommodate the management of open source compliance.

Messaging

Messaging, both internal and external, is an integral part of any compliance program. The single most important recommendation with respect to messaging is to be clear and consistent, whether it is internally explaining the company's goals and concerns around open source to your employees or externally toward the developer communities of the open source projects you use in your product/software stack.

COMPLIANCE CHALLENGES AND SOLUTIONS

Companies will almost certainly face challenges establishing their open source compliance program. In the following sections, we discuss some of the most common challenges, and offer recommendations on how to overcome them.

Creating a Compliance Program

The first challenge is to balance the compliance program and its supporting infrastructure with (existing) internal processes while meeting deadlines to ship products and launch services. Various approaches can help ease or solve such challenges and assist in the creation of a streamlined program that is not seen as a burden to development activities.

Proposed Solutions

EXECUTIVE SUPPORT

It is important to have executive-level commitment to the open source management program to ensure success and continuity.

LIGHTWEIGHT POLICIES AND PROCESSES

Processes and policies are important; however, they have to be light and efficient so that development teams do not regard them as overly burdensome to the development process.

Streamline open source management upon two important foundational elements: a simple and clear compliance policy and a lightweight compliance process.

MANDATE BASIC RULES

As part of putting the compliance program in place, you will need to establish some simple rules that everyone must follow:

- Require developers to fill out a request form for any open source software they plan to incorporate into a product of software stack.
- Require third-party software suppliers to disclose information about open source software included in their deliverables. Your software suppliers may not have great open source compliance practices, and it is recommended that you update your contractual agreement to include language related to open source disclosures.
- Mandate architecture reviews and code inspections for the Open Source Review Board (OSRB) to understand how software components are interrelated and to discover license obligations that can propagate from open source to proprietary software. You will need proper tooling to accommodate a large-scale operation.

- Scan all incoming software received from third party software providers and ensure that their open source disclosures are correct and complete.

INTEGRATE COMPLIANCE IN THE DEVELOPMENT PROCESS

The most successful way to establish compliance is to incorporate the compliance process and policies, checkpoints and activities as part of existing software development processes.

Long-Term Goals versus Short-Term Execution

Figure 4 described the essential elements needed for a successful compliance program. Some team members may be overwhelmed by the amount of work needed to implement such a complete program. In reality, it is not all that difficult, because you do not have to implement all elements simultaneously. The priority for all organizations is to ship products and services on time while building and expanding their internal open source compliance infrastructure. Therefore, you should expect to build your compliance infrastructure as you go, keeping in mind scalability for future activities and products. The key is thoughtful and realistic planning.

Proposed Solutions

- Plan a complete compliance infrastructure to meet your long-term goals, and then implement the pieces stepwise, as needed for short-term execution. For instance, if you are just starting to develop a product or deliver a service that includes open source and you do not yet have any compliance infrastructure in place, the most immediate concern should be establishing a compliance team, processes and policy, tools and automation, and training your employees. Having kicked off these activities (in that order) and possessing a good grip on the build system (from a compliance perspective), you can move on to other program elements.
- Establish policies and processes.
- Incorporate compliance as part of the development process.

Communicating Compliance

Communication is essential to ensure the success of compliance activities. Two types of communication activities are important to consider: internal with your organization, and external towards the developer communities of the open source projects used in your products.

Internal Communication

Companies need internal compliance communication to ensure that employees are aware of what is involved when they include open source in a commercial software portfolio, and to ensure that they are educated about the company's compliance policies, processes, and guidelines. Internal communications can take any of several forms:

- Email communication providing executive support and of open source compliance activities
- Formal training mandated for all employees working with open source software
- Brown-bag open source and compliance seminars to bring additional compliance awareness and promote active discussion
- An internal open source portal to host the company's compliance policies and procedures, open source related publications and presentations, mailing lists, and a discussion forum related to open source and compliance
- A company-wide open source newsletter, usually sent every other month or on quarterly basis, to raise awareness of open source compliance

External Communication

Companies need external compliance communications to ensure that the open source community is aware of their efforts to meet the license obligations of the open source software they are using in their products.

External communications can take one of several forms:

- Website dedicated to distributing open source software for the purpose of compliance
- Outreach and support of open source organizations: Such activities are important to help the company build relationships with open source organizations, understand the roles of these organizations, and contribute to their efforts where it makes sense
- Participation in open source events and conferences: Participation can be at various levels ranging from sponsoring an event, to contributing presentations and publications, or simply sending developers to attend and meet open source developers and foster new relationships with open source community members

Establishing a Clean Software Baseline

One of the initial challenges when starting a compliance program is to find exactly which open source software is in use and under which licenses it is available. This initial auditing process is often described as establishing a clean compliance baseline for your product or software portfolio. This is an intensive activity over a period of time that can extend for months, depending on how soon you started the compliance activities in parallel to the development activities.

Proposed Solutions

Organizations achieve initial compliance through the following activities:

- Early submission and review of open source usage requests.
- Continuous automated source code based on a predefined interval of time for all source code.
- Continual scans on the source code base, including that received from third-party software providers, to intercept source code that was checked into the code base without a corresponding

compliance ticket. Such source code scans can be scheduled to run on a monthly basis, for instance.

- Enforced design and architectural review, in addition to code inspections, to analyze the interactions between open source, proprietary code, and third party software components. Such reviews are mandatory only when a given interaction may invoke license compliance obligations.

If a company fails to establish baseline compliance, it is almost guaranteed that future revisions of the same product (or other products built using the initial baseline) will suffer from compliance issues.

To guard against such scenarios, companies should consider the following:

- Offer simple but enforced policies and lightweight processes.
- Include compliance checkpoints as part of the software development process as it moves from concept into shipping a product or software stack. Ideally, with every development milestone, you can incorporate a corresponding compliance milestone, ensuring that all software components used in the build have parallel and approved compliance tickets.
- Ensure availability of a dedicated compliance team. This topic is discussed at length in a later chapter.
- Utilize tools and automation to support efficient processing of compliance tickets. This topic is discussed in a later chapter.

Maintaining Compliance

There are several challenges in maintaining open source compliance, similar to those faced when establishing baseline compliance. In fact, many of the steps are identical, but on a smaller, incremental scale. Maintaining compliance is a continuous effort that depends on discipline and commitment to build compliance activities into existing engineering and

business processes.

Figure 7 illustrates the concept of incremental compliance, whereby you need to ensure compliance of whatever source code changes took place between the initial compliant baseline and the current versio

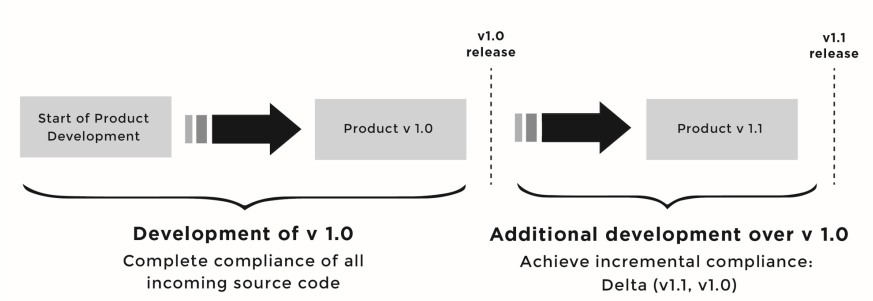


Figure 7. Example of incremental compliance

Proposed Solutions

Companies can maintain open source compliance through the following activities:

- Adherence to the company's compliance policy and process, in addition to any provided guidelines
- Continuous audits of all source code integrated in the code base, regardless of its origins
- Continuous improvements to the tools used in ensuring compliance and automating as much of the process as possible to ensure high efficiency in executing the compliance program

Institutionalization and Sustainability

Maintaining open source compliance activities is an ongoing challenge as the organization grows and ships more products and services using open source software. Companies can take several steps to institutionalize compliance within their development culture and to ensure its sustainability.

Proposed Solutions

SPONSORSHIP

Executive-level commitment is essential to ensure sustainability of compliance activities. There must be a company executive who acts as ongoing compliance champion and who ensures corporate support for open source management functions.

CONSISTENCY

Achieving consistency across the company is key in large companies with multiple business units and subsidiaries. A consistent interdepartmental approach helps with recordkeeping, and also facilitates sharing code across groups.

MEASUREMENT AND ANALYSIS

Measure and analyze the impact and effectiveness of compliance activities, processes, and procedures with the goal of studying performance and improving the compliance program. Metrics will help you communicate the productivity advantages that accrue from each program element when promoting the compliance program.

REFINING COMPLIANCE PROCESSES

The scope and nature of an organization's use of open source is dynamic — dependent on products, technologies, mergers, acquisitions, offshore development activities, and many other factors. Therefore, it is necessary to continuously review compliance policies and processes and introduce improvements.

Furthermore, open source license interpretations and legal risks continue to evolve. In such a dynamic environment, a compliance program must evolve as well.

ENFORCEMENT

A compliance program is of no value unless it is enforced. An effective compliance program should include mechanisms for ongoing monitoring of adherence to the program and for enforcing policies, procedures, and guidelines throughout the organization. One way to enforce the compliance program is to integrate it within the software development process and ensure that some measurable portion of employee performance evaluation depends on their commitment to and execution of compliance program activities.

STAFFING

Ensure that staff is allocated to the compliance function, and that adequate compliance training is provided to every employee in the organization. In larger organizations, the compliance officer and related roles may grow to be FTEs (full time equivalents); in smaller organizations, the responsibility of open source management is more likely to be a shared and/or a part-time activity.

Chapter 3

ACHIEVING COMPLIANCE: ROLES AND RESPONSIBILITIES

A single individual, no matter how adept, can't successfully implement open source compliance across a whole organization. Figure 8 presents a breakdown of the different departments responsible for achieving open source compliance. There are two teams involved in achieving compliance: a core team and an extended team, with the latter typically being a superset of the former.

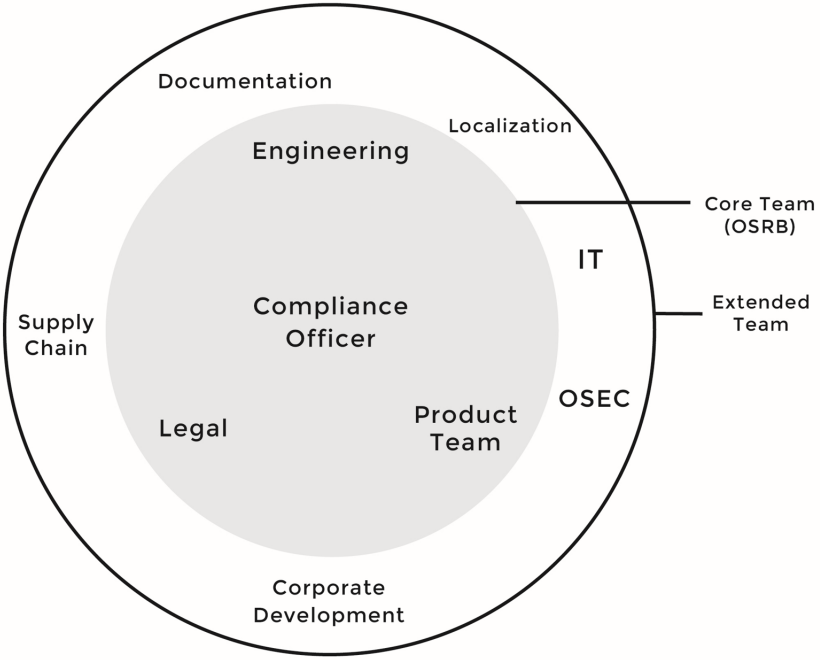


Figure 8. Individuals and teams involved in ensuring open source compliance

The core team, often called the Open Source Review Board (OSRB), consists of representatives from engineering and product teams, one or more legal counsels, and the Compliance Officer. Table 4 describes the roles and responsibilities of each participant in this core team.

The extended team, described in Table 5 (page 49), consists of various individuals across multiple departments that contribute on an on-going basis to the compliance efforts: Documentation, Supply Chain, Corporate Development, IT, Localization, and the Open Source Executive Committee (OSEC). However, unlike the core team (in substantial organizations), members of the extended team are working on compliance only on a part-time basis, based on tasks they receive from the OSRB.

Table 4. Primary roles and responsibilities of the compliance core team (OSRB)

Participant	Primary Roles and Responsibilities
Legal Representative This representation varies from a Legal counsel to a Legal paralegal depending on the task at hand.	Participate in OSRB and OSEC Review and approve usage, modification, distribution of open source (OS) software Provide guidance on licensing Contribute to and approve training Contribute to improving the OS compliance program Review and approve content of OS portals Review and approve list of obligations to fulfill Review and approve open source notices

Participant	Primary Roles and Responsibilities
Engineering and Product Team Representative In some companies, there is no distinction between engineering and product teams.	Participate in OSRB and OSEC Follow compliance policies and processes Integrate compliance practices in dev process Contribute to improving the compliance program Follow technical compliance guidelines Respond quickly to all questions Conduct design, architecture, and code reviews Prepare software packages for distribution
Compliance Officer Open source compliance officer is not necessary a dedicated resource. In most cases, the individual fulfills the role of Manager or Director of Open Source.	Drive all compliance activities Coordinate source code scans and audits Coordinate distribution of source code packages Participate in OSRB and OSEC Contribute to compliance and OS training Contribute to improving compliance program Report to OSEC on compliance activities Contribute to creation of new tools to facilitate automation, discovery of OS in dev environment

Table 5. Primary roles and responsibilities of the compliance extended team

Participant	Primary Roles and Responsibilities
Open Source Executive Committee (OSEC) Decide on open source strategy	Review and approve proposals to release IP Review and approve proposals to release proprietary source code under an open source license. This is not required if the source code was created with the assumption that it will be open sourced.
Documentation	Include open source license information and notices in the product documentation
Localization	Translate basic information in target languages about open source information related to the product or software stack
Supply Chain	Mandate third party software providers to disclose open source in licensed or purchased software components Assist with ingress of third party software bundled with and/or includes open source software
Information Technology (IT)	Provide support and maintenance for the tools and automation infrastructure used by the compliance program Create and/or acquire new tools based on OSRB requests

Participant	Primary Roles and Responsibilities
Corporate Development	Request open source compliance be completed before a merger or acquisition Request open source compliance be completed when receiving source code from outsourced development centers or third-party software vendors

OPEN SOURCE REVIEW BOARD (OSRB)

The OSRB is responsible for:

- Ensuring mutual compliance with third-party software and open source software licenses.
- Facilitating effective usage of and contributions to open source software.
- Protecting proprietary intellectual property (and consequently product differentiation) by ensuring that open source license obligations do not propagate to proprietary or third party software

On a daily basis, OSRB members are involved in the following tasks:

- Establish the Compliance End-to-End Process: The OSRB is responsible for creating the compliance end-to-end process including usage, audit, development, engagement, assurance, and compliance management. Chapter 4 provides an overview of the end-to-end compliance process.
- Create and maintain compliance policies, processes, guidelines, templates, and forms used in the compliance program.

- Review requests for the use, modification, and distribution of open source: The OSRB reviews and approves incoming requests from engineering and product teams for using open source. Chapter 6 provides a discussion on the usage process.
- Perform software audits: The OSRB performs audits on all software included in a product, which involves the following tasks:
- Run a source code scanning tool over the software base
- Analyze the results provided by the scanning tool
- Address all the hits, possible matches, and licensing conflicts flagged by the scanning tool
- Oversee the closure of all issues identified by scanning tools
- Create a final audit report and ensure that all identified issues have been closed

Depending on the size of the organization, auditing responsibilities can be assigned to the OSRB or to an independent team (auditing team) that reports to the Compliance Officer. Chapter 6 provides a discussion of the auditing process.

- Perform architectural reviews: As part of the approval process, the OSRB performs architecture review with Engineering representative to analyze the interaction between open source code, proprietary code, and third-party source code. The goal of this review is to ensure that architectural guidelines are respected and that the interactions among open source, proprietary, and third-party software are within the acceptable legal guidelines.
- Perform linkage analysis review: The OSRB also performs a linkage analysis to determine if any open source license obligations propagate to proprietary or third-party software through linking (from API calls, etc.).

- Verify the resolution of issues that deter releasing product or launching services that contain open source.
- Provide guidance on open source questions coming from company staff and engineers.
- Perform code inspections as part of the pre-distribution verification, to ensure that open source license text and copyright notices have been kept intact and that engineers have updated the change logs to reflect the changes introduced to the source code.
- Compile a list of license obligations that must be met to use the open source software in question and pass it to appropriate departments for fulfillment: Once the OSRB has approved the usage of open source in a product, as part of the approval process the OSRB compiles a list of obligations and passes it to the various other individuals and teams to ensure its fulfillment. As part of the pre-distribution process, the OSRB performs final checks before product or service releases, including verifying the fulfillment of obligations.
- Develop and offer open source and compliance training: The OSRB drives the development of open source and compliance training to ensure that employees have a good understanding of the company's open source policies and compliance practices. In addition, the OSRB should educate employees on some of the most common open source licenses and the issues surrounding using open source in commercial contexts. Training must be mandatory for all staff engaged in management and development of software using open source.
- Host and maintain the company's open source websites: The internal website, intended for employees, focuses on open source processes and policies, guidelines, training, and announcements. The external website usually exists for the primary reason of making available source code packages in fulfillment of certain compliance obligations.

- Handle compliance inquiries: The OSRB is responsible for answering any inquiries sent to the company in relation to open source compliance. Chapter 9 discusses the process of handling compliance inquiries.
- Maintain records of compliance: OSRB is responsible for ensuring that all compliance records for any given open source software component are up to date.
- Review end-user documentation to ensure that appropriate copyright, attribution, and license notices are given to consumers regarding open source included in the product or the software stack. In addition, specific to the GPL/LGPL family of licenses, provide a written offer on how to obtain the source code, when applicable.
- Recommend new tools to be used as part of the compliance infrastructure that will contribute to making the compliance work more efficient through automation.
- Sign off on product distribution from an open source compliance perspective.
- Develop community involvement policy, process, procedures, and guidelines. This responsibility is not compliance-related; however, it is listed here for completion of the list of responsibilities.

LEGAL

The Legal Counsel is a core member of the OSRB, the committee that ensures compliance with open source licenses. The Legal Counsel focuses on four essential duties:

1. Provide approval regarding the use of open source in products or services

The approval of the Legal Counsel is required when using open source in a commercial product. Typically, the Legal Counsel reviews the compliance ticket in the online tracking system (for instance, JIRA or Bugzilla), the resulting report from the source code scanning tool, and the license

information provided with the source code package. They then evaluate risk factors based on feedback provided by both engineering and the open source compliance officer. As part of this exercise, the Legal Counsel also decides on incoming and outgoing licenses of the software component in question. The incoming licenses are the licenses of all source code included in a given body of code; the outgoing licenses are the licenses under which the source code and/or object files are being made available to its recipients.

2. Advise on open source licensing

- Offer guidance about open source license obligations.
- Advise on licensing conflicts arising from incompatible or conflicting licenses primarily based on company policy, which is some cases rely on external factors such as legal opinions of relevant open source organizations.
- Advise on IP issues associated with the use of open source. This is especially the case when the company is about to release previously proprietary source code under open source license(s).
- Provide recommendations and guidance to engineering teams on open source questions and concerns.

3. Review and approve updates to end-user documentation

This form of legal support is related to ensuring that appropriate open source notices (copyright, attributions, and license notices) are provided to consumers in relation to any open source included in the product. In addition, if there is source code licensed under one of the GPL/LGPL family of licenses, a written offer needs to be provided along with information on how to obtain the source code.

4. Contribute to the startup and ongoing management of the open source compliance program

- Establish and maintain the open source policy and process.
- Handle inquiries sent to the company in relation to open source compliance.
- Provide training around open source licenses, company policies, and guidelines.

ENGINEERING AND PRODUCT TEAMS

Engineering and product teams may have one or more representatives who participates in the OSRB, tracks down all compliance-related tasks assigned to engineering, and ensures proper resolution. In parallel, engineering and product teams have several responsibilities with respect to open source compliance:

- Submit requests to use open source software: Engineering and product teams decide what external software to bring into the product baseline, including third-party and open source. Their primary responsibility from a compliance perspective is to submit a usage form for any open source that is planned for inclusion in a product or service. The form describes the intended use of the open source in question and helps construct and maintain a good record of software origin and provenance.
- Follow technical compliance guidelines: Engineering and product teams should follow OSRB technical guidelines to architect, design, integrate, and implement source code. The OSRB guidelines typically cover:
    - Common mistakes and how to avoid them
    - Rules to follow when integrating libraries and other middleware to avoid linkage issues that might arise

    - Development in kernel space versus user space (on Linux), especially with whole-platform development in embedded environments
    - Specific engineering situations that are applicable to open source compliance

- Conduct design reviews: Engineering teams should continuously conduct design reviews to discover and remedy any compliance issues in a timely manner. The Compliance Officer drives the design reviews and invites different engineering participants depending on the software component in question.
- Cooperate with OSRB: Engineering teams must respond promptly to questions asked by the OSRB and cooperate in resolving compliance tickets.
- Track changes: Maintain a change log for each modified open source component: As part of meeting the open source license obligations and depending on the open source license in questions, some licenses (such as the GPL/LGPL family of licenses) mandate that modified files carry prominent notices stating that you changed the files and the date of the change(s).
- Prepare source code packages for distribution: Engineering teams prepare the source code packages that will be made available on a public website as part of meeting open source license obligations (other source code distribution methods are discussed in a later chapter).
- Integrate compliance milestones as part of the development process: This exercise takes place in collaboration with the OSRB and the Compliance Officer.
- Undergo open source training: All engineers must take the available open source training.
- Monitor the open source projects to determine whether any bug fixes or security patches have become available, and take

responsibility for updating the open source component used in the product. The individual package owner within the organization usually performs this specific task.

COMPLIANCE OFFICER

The Compliance Officer, also called OSRB Chair or Manager/Director of Open Source, chairs the OSRB and manages the compliance program.

Ideally, the compliance officer must possess as many as possible of the following:

- Solid understanding of common open source licenses and obligations to discuss with legal counsel Knowledge of industry practices
- Knowledge and experience in establishing corporate-wide policies and processes
- Technical knowledge related to the company's products
- Historical perspective on open source
- Knowledge of community consensus and practices
- Contacts in the key open source project communities
- Contacts in the open source organizations such as the Linux Foundation, Apache Foundation, Mozilla Foundation, Software Freedom Law Center, etc.

In addition to the responsibilities pertaining to the OSRB, the Compliance Officer carries the following duties:

- Drive the compliance due diligence end-to-end process and act as the compliance program manager, ensuring all compliance-related tasks are addressed and there are no compliance issues blocking products from shipping

- Coordinate source code scans and drive all auditing issues to closure
- Participate in engineering design reviews, code inspections, and distribution readiness assessments to assure that the engineering and product teams follow all compliance processes and policies and conform to the approved OSRB usage form
- Coordinate source code distribution of open source packages (when stipulated by licenses) with engineering and product teams, including preparing and verifying a distribution checklist for each open source package
- Act as liaison between OSEC and OSRB
- Escalate compliance issues to OSEC
- Act as liaison between the engineering and product team and the OSRB and OSEC in regard to usage plan approval processes
- Report on compliance activities to the OSEC, including flagging issues that prevent shipping a product or service

OPEN SOURCE EXECUTIVE COMMITTEE

The Open Source Executive Committee (OSEC) consists of engineering, legal, and product marketing executives in addition to the Compliance Officer. The OSEC is responsible for setting open source strategy, reviewing and approving release of IP, and providing approvals to release previously proprietary source code under a specific open source license.

DOCUMENTATION

The documentation team is responsible for including written offers and any appropriate open source notices in the product documentation. Figure 9, on the next page, provides an illustration of how such notices are prepared and approved. The process starts with the compliance officer, who prepares the draft of the written offer and the notices that are to be made available once the product ships. Next, the legal counsel reviews the draft proposed by the

compliance officer, edits it, and pushes a final version to the documentation team. The last step of the process is including the final text in the product documentation.

Compliance Officer	Prepares the draft notices document based on the final software bill of material in addition to a proposal on where and how these notices should be presented.
Legal Counsel	Reviews, edits and approves proposal from the Compliance Officer.
Documentation Team	Update product documentation as approved by Legal.

Figure 9. The role of the documentation team in updating the product documentation, reflecting the presence of open source in the product

LOCALIZATION

The localization team is responsible for translating basic language that informs users of the availability of open source software in the product and directs them to the proper notices made available in English.

SUPPLY CHAIN

Supply chain (software procurement) procedures must be updated to address the acquisition and use of open source. It is highly recommended that you examine software supplied to you by third-party software providers. Supply Chain personnel are usually involved in moving software from the suppliers to your company. Supply Chain can support open source compliance activities by mandating that third-party software (and hardware) providers disclose any open source that is being delivered with their wares, and by assisting with licensing-in third-party software that is bundled with and/or integrates open source packages.

A best practice in this area is to mandate that third-party software providers

disclose any open source used in their offering, along with a statement on how they plan to meet the applicable open source license obligations. If third-party software includes open source, Supply Chain must ensure that open source license obligations are satisfied, since, after initial ingress, those obligations become your responsibility as distributor of a product or service that includes open source. It is not acceptable to point "upstream" to a supplier and to inform recipients of your code that meeting license obligations was the responsibility of the supplier instead of your own.

IT

IT provides support and maintenance for the tools and automation infrastructure employed by a compliance program. This responsibility spans the servers hosting the various tools, the tools, mailing lists, and web portals. In addition, IT may receive requests from the OSRB to develop and/or acquire tools that will be used to improve effectiveness the compliance activities.

CORPORATE DEVELOPMENT

Corporate Development is involved with open source compliance in two major scenarios: mergers and acquisitions transactions, and outsourced development.

Mergers and Acquisitions

If a company is considering a merger or is the target of an acquisition, it should structure its compliance program to offer a level of disclosure and provide representations. Company policies regarding merger and acquisition transactions need to be updated to account for open source. Corporate Development must mandate that source code be evaluated from a compliance perspective prior to any merger or acquisition to avoid surprises that might derail discussions or affect the company's valuation. For the acquiring company, comprehensive code evaluation assures accurate valuation of software assets and mitigates the risk of unanticipated licensing issues undermining future value. In addition, the acquiring company may include provisions in the purchase agreement requiring the disclosure of

open source that is subject to the transaction. Diligence practices should be updated to require open source disclosure and include guidance regarding the review of any disclosed open source and licenses.

Outsourced Development

Agreements relating to outsourced development of software should also be updated to reflect compliance procedures and to ensure that other provisions of these agreements (such as representations and warranties) are broad enough to cover the risks posed by open source. Corporate Development must mandate that all source code received from outsourced development centers must go through the compliance process to discover all open source being used and to ensure proper actions to fulfill license obligations.

Other Corporate Transactions

Corporate Development is also involved with compliance in transactions such as spin-offs and joint ventures. In some cases, the compliance due diligence may result in a decision not to proceed with the transaction, if that the compliance situation proves far from ideal.

Chapter 4

OPEN SOURCE COMPLIANCE PROCESS

Implementation of open source compliance processes can vary across organizations based on a number of factors including the underlying development processes into which compliance must fit, the size and nature of the code base, the number of products or services involved, the amount of externally supplied code, the size and organizational structure, and so on. But the core elements of compliance usually remain the same: identifying the open source in the code base, reviewing and approving its use, and satisfying obligations. This chapter focuses on the core elements of a compliance process. The result of compliance due diligence is identification of all free and open source software used in a product intended for external distribution, and a plan to meet the attendant license obligations. Figure 10 offers a high-level overview of a sample end-to-end compliance process, and illustrates the various compliance steps or phases that components containing free and open source software undergo before receiving approval for integration in an externally distributed product or service.

Figure 10. Simplified view of the compliance end-to-end process

What you see in Figure 10 is just one example, as there are many ways of organizing the compliance process to accomplish the same goals. Throughout this chapter we will examine these various phases, the inputs and outputs of each phase, and how to control software usage via the compliance process.

EFFECTIVE COMPLIANCE

The term due diligence refers to a number of concepts involving source code inspection, source code surveillance, or the performance of quality duties and system audits. In the case of open source compliance, due diligence is required to ensure the following:

- Open source software used in the product has been identified, reviewed, and approved
- Product implementation includes only approved open source components and licenses.
- All obligations related to the use of licensed material have been identified
- Appropriate notices have been provided in documentation, including attributions and copyright notices
- Source code, including modifications (when applicable), has been prepared and is available at the time the product ships
- Verification of all the steps in the process

There are great benefits to having an end-to-end compliance process that is simple and well understood within the organization. Such a process would:

- Enable organizations to benefit from open source while complying with obligations
- Move open source use from ad hoc to a standardized process
- Help manage acquisition of open source components
- Help employees understand how to work with open source in a responsible way
- Improve the relationship with developers in the various open source projects used by your organization

- Accelerate exchange of information and ideas with the project communities of integrated code through sharing of source code modifications
- Speed innovation, since the organization is able to safely adopt open source components and use them as enablers for new services and products

ELEMENTS OF AN END-TO-END COMPLIANCE PROCESS

There are ten key steps in an end-to-end compliance process (Figure 11, next page):

1. Identification of incoming source code
2. Auditing source code
3. Resolving any issues uncovered by the audit
4. Completing appropriate reviews
5. Receiving approval to use open source
6. Registering open source in the software inventory
7. Updating product documentation to reflect open source usage
8. Performing verification of all previous steps prior to distribution
9. Distributing source code packages
10. Performing final verifications in relation to distribution

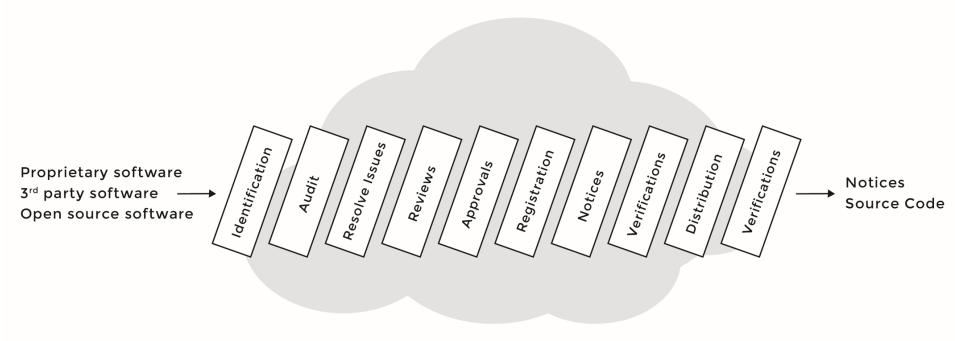


Figure 11. End-to-end compliance process

The remainder of this chapter will address each of these ten steps in detail.

Step 1 – Identification of Open Source

The goal of this initial step is to monitor the ingress and incorporation of open source in a software portfolio, either as a standalone package or embedded within third-party or company-developed software. There are several methods to identify open source used in the product:

- A request to use open source: This is the most common method for identifying the usage of open source in a product. Engineering staff or Product Management is required to inform the Open Source Review Board (OSRB; described in Chapter 3) or compliance team of the intent to use specific open source in a specific product or platform release. The submitter provides information regarding the intended use of the open source package for review and approval.

- Auditing the full platform or product code base to establish a compliance baseline, and then auditing code modules that are changed in subsequent releases.

- Third-party software provider due diligence: This involves requiring a full disclosure of open source components in products provided by third-party suppliers, with an accompanying review of the disclosure by the open source compliance team. In some cases, it will make

sense to require third-party software vendors to provide an audit of the supplied code as an additional layer of diligence. This helps ensure you are controlling the intake of open source by your product.

- Auditing proprietary (company-developed) software components: In some instances, engineers may decide to copy/paste source code from open source components and include it in proprietary software components. Therefore, it is important to also audit company-developed software components, since they may include open source code, which may lead to compliance failures if not discovered before product ship date.

- Inspect all open source components entering the organization source code repository that do not correspond to an incoming request to use open source: Relying on engineers to fill out forms announcing their intent to use open source is not always a reliable method to account for all incoming open source software. Therefore, as a backup, consider setting up a source code control system with a separate folder for open source and a notification alert any time there is a check-in to this folder. Since it is always a recommended practice to separate open source, company-developed proprietary software, and third party software in different folders in your build system, it becomes feasible to set up alerts when new code is being submitted. If a new component was submitted that does not correspond to an existing usage (open source request) form, then it is a new component and requires that a new form be filled out.

Identification phase prerequisite

One of the following conditions is met:

- An incoming OSRB form requesting using a specific open source

- Discovery of an open source being used (without proper authorization) via a platform scan

- Discovery of an open source being used as part of third party software

Identification phase outcome

- A compliance record is created (or updated) for the open source
- An audit is requested to scan the source code

Step 2 – Auditing Source Code

The second step in compliance due diligence consists of scanning the source code using automated analysis tools to discover matches with known open source projects.

The auditing personnel perform a source code scan iteratively from one release to another, to build a chain of evidence that proves what is included in the release is compliant with the various applicable open source licenses.

The goals of the audit are to:

- Update the release bill of materials to account for any open source added (or removed) since the last previous scan
- Confirm the origin(s) of the source code, including the provenance of any open source
- Flag any dependencies, code matches, and licensing conflicts

Auditing phase prerequisite

A proper compliance record (also called a ticket) is created capturing all necessary information about the usage of that specific open source and providing the location of the source code within the internal build system. In some cases, specifically when a full platform scan is done, an open source component may be scanned before having a proper compliance report. In this case, a record is created when the open source component is discovered.

Auditing phase outcome

- An audit report identifying the origins and licenses of the source code
- Change request tickets are filed against the appropriate engineering team for any issues identified during the audit that require resolution

Several actions can trigger discovery and an audit for software components (Figure 12, next page):

- A request from a developer to use an open source component
- A source code scan of the entire software stack
- Source code changes in a previously approved component
- Open source received from a third-party vendor
- Source code downloaded from the web (unknown author or license)
- Proprietary software committed into the source code repository system
- Open source added to the code repository that does not correspond to a usage form
- Using previously approved open source in a different product

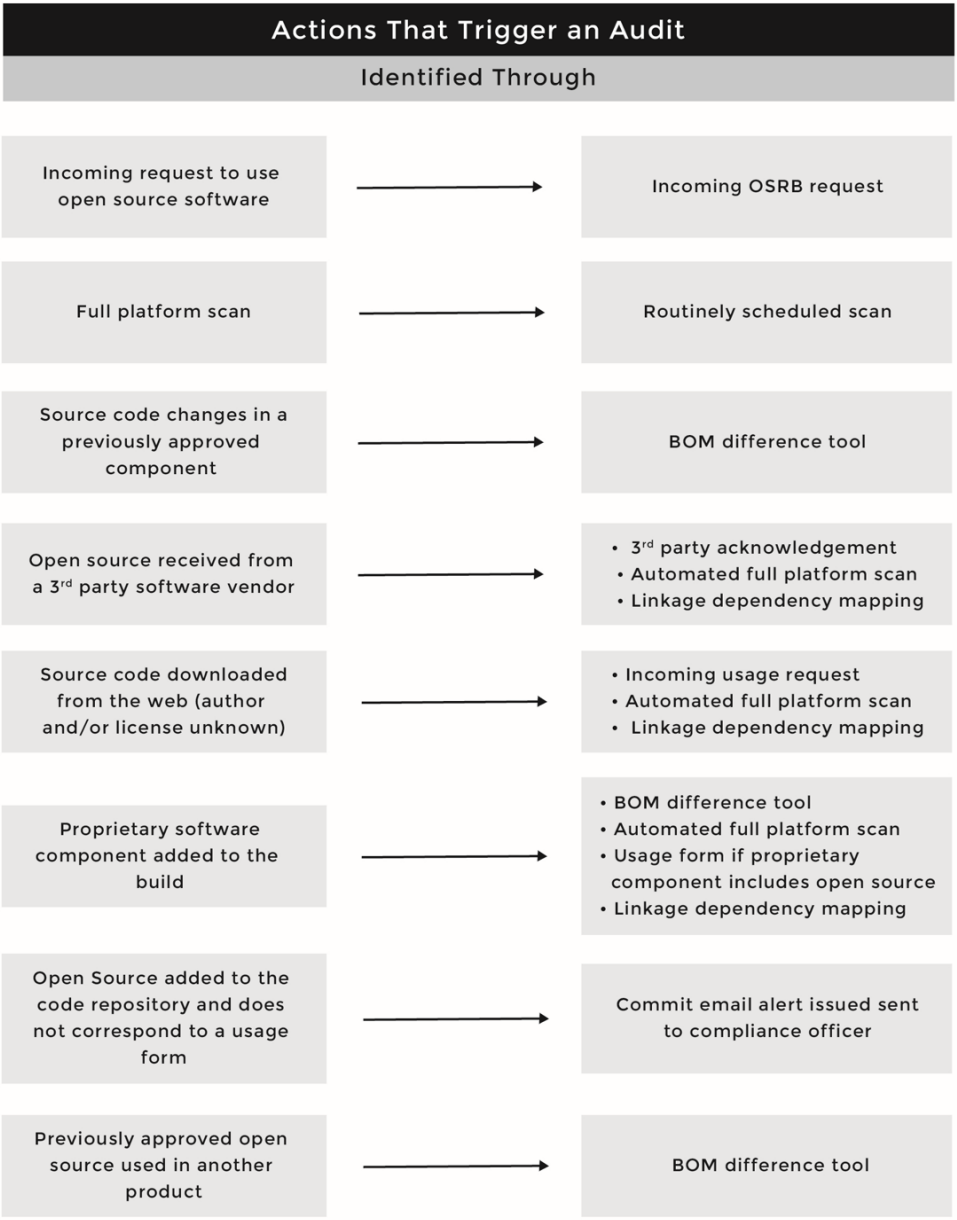


Figure 12. Methods to identify and audit incoming open source

Step 3 – Resolving Issues

In this step of the compliance due diligence, all issues identified during the auditing step are resolved. The OSRB Chair monitors closure of tickets assigned to engineers during the Audit step. Once the engineers have resolved the identified issues, the OSRB Chair should request a new audit to confirm that the resolved issues no longer exist.

Resolving Issues phase prerequisite

A source code svcan has been completed, and an audit report is generated identifying the origins and licenses of the source code. The report has flagged source code files that were not identified, or possible license conflicts resulting from mixing source code coming in under different licenses. The Compliance Officer will drive the effort to resolve these issues.

Resolving Issues phase outcome

A resolution for each of the flagged files in the report and a resolution for any flagged license conflict.

Step 4 – Reviews

Once the auditing is complete and all issues identified earlier have been resolved, the compliance ticket for a specific software component moves to the review step. Various reviews are performed, as illustrated in Figure 13 (next page), and all identified issues must be resolved. The reviewers need to understand the licenses that govern use, modification, and distribution of the software, and identify the obligations of the various licenses. For any given software components, the reviewers of the compliance ticket are:

- Internal package owner (the developer working on specific source code component)
- Source code scanning or auditing personnel

- OSRB (Open Source Review Board), which includes OSRB chair (Compliance Officer), Legal counsel, and OSRB engineering representative
- OSEC (Open Source Executive Committee)

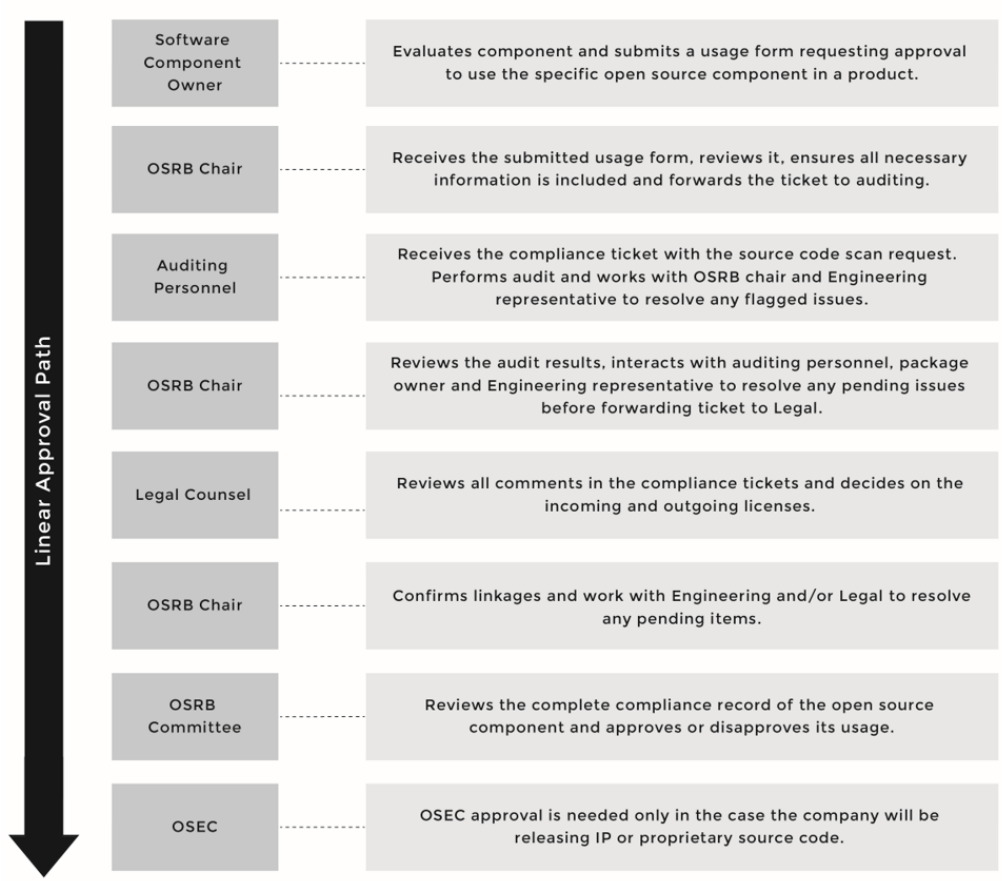


Figure 13. Reviewers of the compliance ticket and their roles

As part of this step of compliance due diligence, there are two important reviews: the architecture review and the linkage analysis review.

Architecture Review

The goal of the architecture review is to analyze any interactions between the open source, third-party, and proprietary code. The result of the architecture review is an analysis of the licensing obligations that may extend from the open source components to the proprietary components (and vice versa). The internal package owner, the OSRB engineering representative, and the open source expert usually perform the architecture review. If they identify a dependency resulting in a licensing conflict, the OSRB Chair will issue a ticket to Engineering to resolve the dependency problem by reworking the source code.

Linkage Analysis Review

The goal of a linkage analysis review is to find potentially problematic code combinations at the static and dynamic link level, such as linking a GPL or more commonly an LGPL-licensed library to a proprietary source code component. The OSRB Chair performs this review using an automated tool. Linkage conflicts are reported to Engineering to resolve.

Review phase prerequisite

Source code has been audited and all issues have been resolved.

Review phase outcome

OSRB members perform an architecture review and a linkage analysis for the specific component and mark it as ready for the next step (i.e., Approval) if no issues are uncovered.

Step 5 – Approvals

Once all reviews have been completed, the software component's compliance ticket moves to the approval step, where it is either approved for usage in the product or not. The approval comes from the OSRB (which includes, as described in an earlier chapter, Legal Counsel, an Engineering representative, and an open source expert).

For most software components, approval is granted by the OSRB once a ticket has progressed to that point in the compliance process. Once the OSRB approves the usage of an open source component, the OSRB communicates the approval to the product teams so they understand their responsibilities and begin preparations to fulfill the license obligations. If the OSRB rejects the use of the open source component, they communicate the reason for rejection to the requester, and this information is recorded as part of the compliance ticket. As a result, the open source component cannot be used in the product, although the requester can consider submitting an appeal for reconsideration by the OSRB.

Approvals phase prerequisite

All OSRB members have reviewed the compliance ticket, and the OSRB has completed the architecture review and linkage analysis.

Approvals phase outcome

Approve or deny usage of the specific component.

Step 6 – Registration

Once a software component has been approved for usage in a product or as part of a service, its compliance ticket will be updated to reflect the approval. It will also be added to the software inventory that tracks open source use and use cases.

If you follow a conservative approach with your compliance practices, you would approve open source software for a specific version and usage in a specific product or service version. If a new version of this open source software becomes available, you require a new approval to ensure that the usage model and even the license are still in line with your internal policy.

Registration phase prerequisite

The OSRB has approved the component's usage in the product.

Registration phase outcome

The component is registered in the software inventory, with the component name, version, internal owner, and the details of where the component is being used, such as product name, version, release number, etc.

Step 7 – Notices

One of the key obligations when using open source is the documentation obligation, also referred to as the notice obligation. Companies using open source in an externally distributed product or service must:

- Inform the end user how to obtain a copy of the source code that's been made available as a result of meeting the license obligations (when applicable)
- Acknowledge the use of open source by providing required copyright and attribution notices
- Reproduce the entire text of the license agreements for the open source code included in the product

If companies are non-compliant with open source license obligations, they are not licensed, and thus exposed to legal action by the copyright holder for copyright infringement, and can potentially lose the right to use and distribute the software in question. In order to fulfill documentation obligations, appropriate notices must be included with the product. In this step of the compliance due diligence, the OSRB Chair prepares the notices and passes them to the appropriate departments for fulfillment.

Notice phase prerequisite

The software component has been approved for usage and registered in the software inventory.

Notice phase outcome

The license, copyright, and attribution notices for a specific component are prepared and passed along to the appropriate departments to be included in the product documentation.

Step 8 – Pre-Distribution Verifications

The next step in the compliance due diligence is to decide on the method and mode of distribution, type of packages to distribute, and mechanism of distribution.

The goals of the pre-distribution verification are to ensure that:

- Open source packages destined for distribution have been identified and approved
- Source code packages (including modifications) have been verified to match the binary equivalent shipping in the product
- Appropriate notices have been included in the product documentation to inform end-users of their right to request source code for identified open source
- All source code comments have been reviewed and any offending or inappropriate content has been removed. This is not strictly a compliance issue; however, in some cases, an innocent comment about where the code was received can trigger a larger compliance question.

Pre-Distribution Verifications phase prerequisite

Component has been approved for usage, it has been registered in the software inventory, and all notices have been captured and sent for fulfillment.

Pre-Distribution Verifications phase outcome

- Decide on distribution method and mode.
- Ensure that all the pre-distribution verifications have been successfully completed.

Step 9 – Distribution

Once all pre-distribution verifications have been completed, it's time to upload the open source packages to the distribution website, labeled with the product and version it corresponds to (this scenario assumes that you have chosen this method to make source code available; other methods are discussed in a later chapter). Note that this action is helpful to those desiring code download but may not be sufficient by itself to satisfy license obligations. Furthermore, a recommended practice is to provide email and postal mail contact information for any compliance or open source-related questions.

Distribution phase prerequisite

All pre-distribution verifications have been checked and no issue is discovered.

Distribution phase outcome

The source code of the component in question is uploaded to the website for distribution (if that is the distribution method of choice).

Step 10 – Final Verifications

Once you upload the open source packages to the distribution website, validate that the packages have been uploaded correctly and can be downloaded and uncompressed on an external computer without errors. If you are providing a patch, ensure that it applies cleanly and that you have specified the proper version of the upstream component.

Final verifications phase prerequisite:

The source code is published on the website.

Final verifications outcome

You receive verification that the source code is uploaded correctly and accessible for download, and that it corresponds to the same version that was approved.

Chapter 5

COMPLIANCE PROCESSES AND POLICIES

For the purpose of this book, the focus of the discussion is using and integrating open source with proprietary and third-party source code in a commercial product. The discussions exclude policies and processes for using open source solely inside your organization for testing and evaluation purposes. This chapter discusses usage policy and process in addition to the base and incremental compliance process, and guidelines for achieving incremental compliance.

POLICY

The usage policy is an essential building block in a compliance program. This policy does not have to be lengthy or complicated. A simple policy can be as effective as a complex one as long as it mandates the following:

- Engineers must receive approval from OSRB before integrating any open source in a product.
- Software received from third-party vendors must be audited to identify any open source included, which will ensure license obligations can be fulfilled before product ships.
- All software must be audited and reviewed, including proprietary software components, software received from third-party providers, and open source software.
- Product must fulfill open source licensing obligations prior to customer receipt.
- Approval for one product is not approval for another deployment, even if the open source component is the same.
- All changed components must go through the approval process.

These rules ensure that any software (proprietary, third-party, open source) that makes its way into the product base has been audited, reviewed, and approved. Furthermore, it ensures that the company has a plan to fulfill the license obligations resulting from using the various software components prior to customer receipt.

PROCESS

The compliance usage process includes scanning the source code of the software package in question, identifying and resolving any discovered issues, performing legal and architectural reviews, and making a decision regarding the usage approval for a given software package.

Figure 14 illustrates a simplistic view of a compliance usage process. This figure does not demonstrate the iterative nature of such a process; a more elaborate view is provided in Figure 17 (page 89).

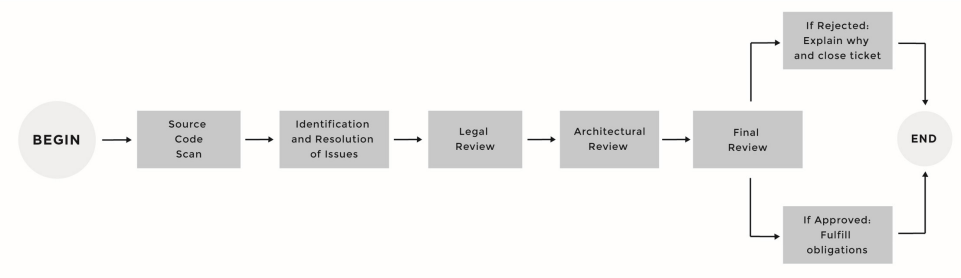


Figure 14. A sample compliance usage process

Source Code Scan

In the source code scanning phase, all the source code is scanned using a source code scanning tool. Figure 15 (page 81) illustrates some of the factors that can trigger a source code scan, which include:

- An incoming OSRB usage form, usually filled out by engineering staff. This is a simple online form that an engineer or a developer fills out, providing basic information about the source code in question. Upon the submission of the form, a compliance ticket (in a system such as JIRA or Bugzilla) will be created automatically, and a source code scanning request will be sent to the auditing staff.
- A periodically scheduled full-platform scan: Such scans are very useful to uncover open source that may have snuck into your software platform without an OSRB form.
- Changes in previously approved software components: In many cases, engineers start evaluating and testing with a certain version of an OSS component, and later adopt that component when a new version is available.
- Source code received from a third-party software provider who may or may not have disclosed open source
- Source code downloaded from the web with unknown author and/or license, which may or may not have incorporated open source in it
- A new proprietary software component entering the build system where engineering may or may not have borrowed open source code and used it in a proprietary software component

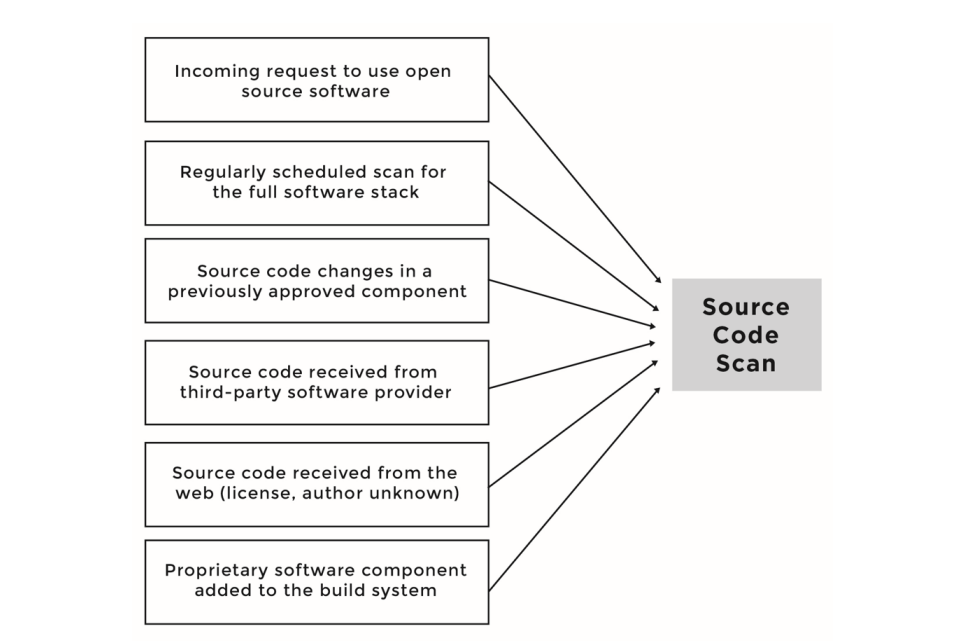


Figure 15. Events that trigger a source code scan

In preparation for legal review, the individual who runs the source code scan should attach to the compliance ticket all the license information found in the package, such as any COPYING, README, or LICENSE files, in addition to any files stating copyright and attribution notices.

Identification and Resolution

In the identification and resolution phase, the auditing team inspects and resolves each file or snippet flagged by the scanning tool.

Legal Review

In the legal review phase, the legal counsel (member of the OSRB) reviews reports generated by the scanning tool, the license information of the software component, and any comments left in the compliance ticket by engineers and members of the OSRB. If there were no issues with the licensing, the legal counsel would then decide on the incoming and outgoing licenses of the software component, and would forward the

compliance ticket into the compliance architectural phase. If a licensing issue is found, for example mixed source code with incompatible licenses, the legal counsel will flag these issues and reassign the compliance ticket to Engineering to rework the code. In some cases, if the licensing information is not clear or if it is not available, the legal counsel contacts the project maintainer or the open source developer to clarify the ambiguities and to receive a confirmation of the license under which that specific software component is licensed.

Architecture Review

In the architecture review, the compliance officer and the engineering OSRB representative perform an analysis of the interaction between the open source, proprietary, and third-party code. This is accomplished by examining an architectural diagram that identifies:

- Open source components (used “as is” or modified)
- Proprietary components
- Components originating from third-party software providers
- Component dependencies
- Communication protocols
- Other open source packages that the specific software component interacts with or depends on, especially if they are governed by a different open source license

The result of the architecture review is an analysis of the licensing obligations that may extend from open source to proprietary or third-party software components (and across open source components as well). If the compliance officer discovers any issues, such as a proprietary software component linking to a GPL licensed component, the compliance officer forwards the compliance ticket to Engineering for resolution. If there are no issues, then the compliance officer moves the ticket to the final stage in the approval process.

Final Review

The final review is usually an OSRB face-to-face meeting during which the OSRB approves or denies usage. In most cases, if a software component reaches the final review, it will be approved unless a condition has presented itself (such as the software component no longer being in use). Once approved, the compliance officer will prepare the list of license obligations for the approved software component and pass it to appropriate departments for fulfillment.

PROCESS STAGES' INPUTS AND OUTPUTS

In this section, we discuss the inputs and outputs of each of the five phases in the OSRB usage process, as illustrated in Figure 16. Please note that these phases are for illustration purposes and may not be exactly the same as the ones in your specific scenario.

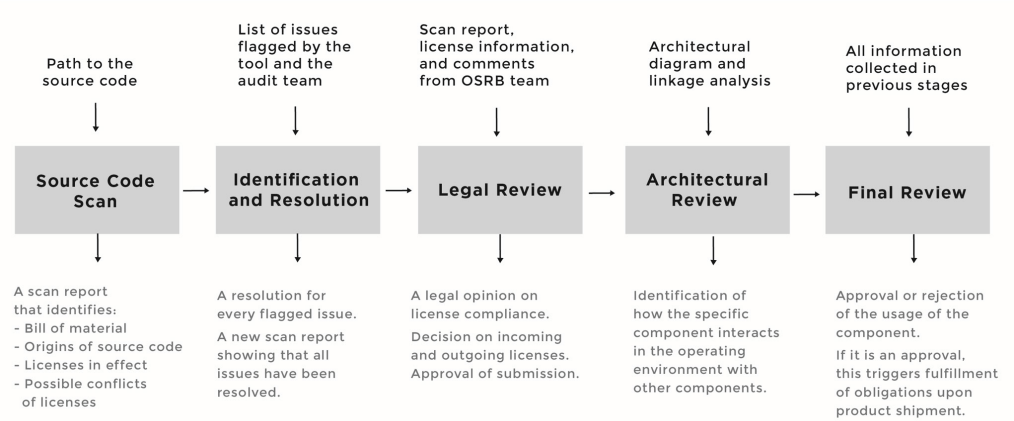


Figure 16. Inputs and outputs of the usage process

Source Code Scan Phase

Input

The input to the scan phase is the OSRB usage form that an engineer has filled out online and submitted. Table 6 (on page 97) provides details on the form. It includes all information about the open source component in question, in addition to the location of the source code in the source code repository system. Periodic full-platform scans should also take place every few weeks to ensure that no open source software component has been included into the platform without a corresponding OSRB form.

Output

The output from the scan phase is a report produced by the source code scanning tool that provides information on:

- Known software components in use, also known as the software Bill of Materials (BoM)
- Licenses in effect, license texts and summary of obligations
- License conflicts to be verified by legal
- File inventory
- Identified files
- Dependencies
- Code matches
- Files pending identification
- Source code matches pending identification

Identification and Resolution Phase

Input

The input to this phase is the report generated by the scanning tool in the previous phase. The report flags issues such as conflicting and incompatible licenses. If there are no issues, then the compliance office will move the compliance ticket forward to the legal review phase. If there are issues to be resolved, then the compliance officer creates subtasks within the compliance tickets and assigns them to the appropriate engineers to be resolved. In some cases, a code rework is needed; in other cases, it may simply be a matter of clarification. The subtasks should include a description of the issue, a proposed solution to be implemented by engineering, and a specific timeline for completion.

Output

In this phase, subtasks are closed, and the output is the resolution of all issues. The compliance officer may order a re-scan of the source code and generate a new scan report confirming that earlier issues have now been resolved. Then the compliance officer forwards the compliance ticket to the representative from the Legal department for review and approval.

Legal Review Phase

Input

When a compliance ticket reaches the legal review phase, it already contains:

- A source code scan report and confirmation that all the issues identified in the scanning phase have been resolved.
- Copies of the license information attached to the ticket: Typically, the compliance officer attaches the README, COPYING, and AUTHORS files available in the source code packages to the compliance ticket. Other than the license information, which for OSS

components are usually available in a COPYING or a LICENSE file, you need to capture copyright and attribution notices as well. This information will provide appropriate attributions in your product documentation.

- Feedback from the compliance officer regarding the compliance ticket (concerns, additional questions, etc.).
- Feedback from the engineering representative in the OSRB or from the engineer (package owner) who follows/maintains this package internally.

Output

The output of this phase is a legal opinion of compliance, and a decision on the incoming and outgoing license(s) for the software component in question. The incoming and outgoing licenses are in the plural form because in some cases, a software component can include source code available under different licenses.

Incoming and outgoing licenses

The incoming license is the license under which you received the software package. The outgoing license is the license under which you are licensing the software package. In some cases, when the incoming license is a permissive license that allows relicensing (such as BSD), companies will relicense that software under their own proprietary license. A more complex example would be a software component that includes proprietary source code, source code licensed under License-A, source code that is available under License-B, and source code available under License-C. During legal review, the legal counsel will need to decide on the incoming and outgoing license(s):

Incoming licenses = Proprietary License + License A + License B + License C
Outgoing license(s) = ?

Architecture Review Phase

The goal of the architecture review is to analyze the interactions between the open source code and third-party and proprietary code. The result of the architecture review is an analysis of the licensing obligations that may extend from the open source components to the proprietary components. The internal package owner, the OSRB engineering representative, and the Compliance Officer usually perform the architecture review. If they identify a dependency resulting in a licensing conflict, the Compliance Officer will issue a ticket to Engineering to resolve the dependency problem by reworking the source code.

Input

Source code has been audited and all issues have been resolved.

Output

OSRB members perform an architecture review for the specific component, and mark it as ready for the next step (i.e., Final Approval) if no issues were uncovered.

Final Approval Phase

Input

The input to this phase is the complete compliance record of the software component, which includes the following:

- A source code scan report generated by the scanning tool
- The list of discovered issues, information on how they were resolved, and who verified that these issues were successfully resolved
- Architectural diagrams and information on how this software component interacts with other software components

- Legal opinion on compliance, and decision on incoming and outgoing licenses
- Dynamic and static linkage analysis, if applicable in an embedded environment (C/C++).

Output

The output of this phase is a decision to either approve or deny the usage of the software component.

DETAILED USAGE PROCESS

Of course there are many possible circumstances that can affect compliance procedures. Figure 17 (next page) provides a detailed process that highlights several possible scenarios, and how to move from one step to another in the compliance process. We then discuss eight possible scenarios. These scenarios are not mutually exclusive and are not the only possible scenarios; however, they are used for illustration and discussion purposes.

Scenario 1: The source code scanned is 100% proprietary

Scenario 2: The source code scanned includes code with incompatible licenses

Scenario 3: Issue with linkages identified during architectural review

Scenario 4: A source code package is not used anymore

Scenario 5: Due diligence identified IP that will be released to meet license obligations

Scenario 6: Verification step has identified an issue that needs to be resolved

Scenario 7: Source code is approved for use

Scenario 8: Source code is rejected

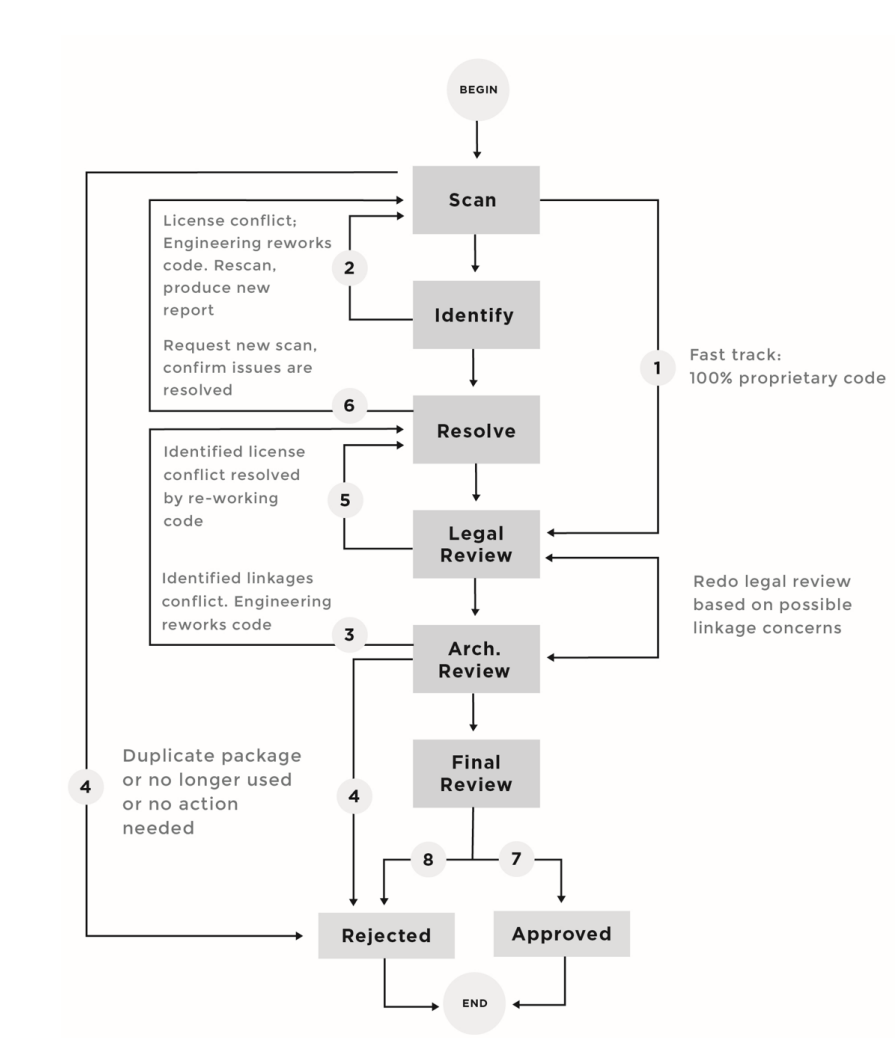


Figure 17. Specific compliance scenarios

Scenario 1: Source code is 100% proprietary

The scanned software component contains 100% proprietary code, and no open source code is declared or identified. In this case, we assume the fast track, and the compliance ticket for that specific component will be forwarded for legal review. Legal counsel attaches a license to this proprietary component, and forwards it to the compliance officer to perform architectural and linkage analysis.

Scenario 2: Incompatible licenses

The scanned software component includes source code that originated from multiple sources with incompatible licenses. Another example would be a software component with a mix of proprietary source code and source code licensed under the GPL. In this scenario, the scan report will be attached to the compliance ticket and assigned to the developers who internally own that software component, with a request to rework the code by removing the GPL source code from the proprietary software component. Once the developer reworks the code, the software component will be scanned again to verify that the GPL code has been removed before the ticket proceeds for legal review.

Scenario 3: Identified issue with linkages

In this scenario, the compliance ticket has passed legal review and is now in the architectural and linkages review. The compliance officer discovers a linkages issue. In this case, the compliance officer moves the compliance ticket back into the resolution phase, assigning it to a developer to resolve the linkage issue.

Scenario 4: Source code no longer used

In this scenario, Engineering decides that a software component is not going to be included in the product while the software component is in transit through the compliance process. As a result, its compliance ticket is closed (rejected). The next time this component is going to be used, it must re-enter the compliance process and progress as approved before it is

integrated in the product or service source code repository.

Scenario 5: IP at risk of requiring release

In this scenario, legal review uncovered that closely-held intellectual property has been combined with an open source code package. Legal counsel will flag this and reassign the compliance ticket to engineering to remove the proprietary source code from the open source component. In the event that engineering insists on keeping the proprietary source code in the open source component, the OSEC will have to release the proprietary source code under an open source license.

Scenario 6: Unresolved issue found

In every case, when an OSRB member discovers a compliance issue in the software component, the component goes through the same life cycle:

- Engineering fixes the identified issue.
- Auditing team re-scans the software component and provides a new scan report.
- Legal examines the new audit report.
- Compliance office ensures that there are no open issues in the architecture and linkage analyses.

Scenario 7: Source code is approved

Once a software component has received the audit, legal, and compliance approvals, it will be reviewed during an OSRB meeting. If nothing has changed in its status — that is, it is still in use, is the same version, and has the same usage model (Figure 18, next page) — the compliance officer will:

- Update the software inventory to reflect that the specific OSS software component version x is approved for usage in product y, version z.

- Issue a ticket to the documentation team to update end user notices in the product documentation, to reflect that open source is being used in the product or service.
- Trigger the distribution process before the product ships.

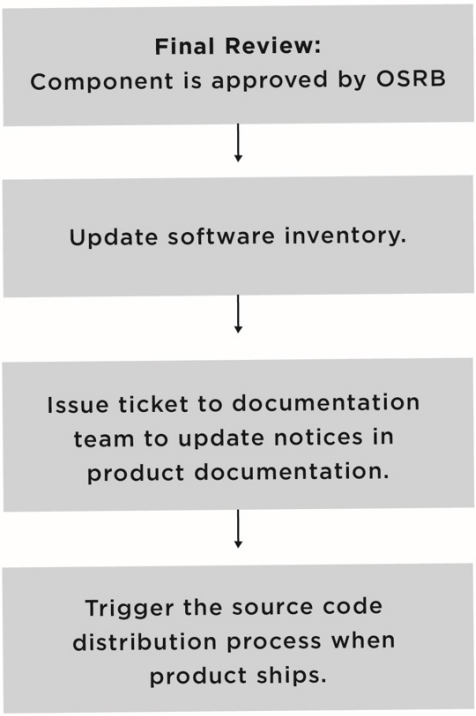


Figure 18. Steps accomplished after the OSRB approval

Scenario 8: Source code is rejected

In this scenario, the OSRB decides to reject usage of a specific software component. There are several reasons that might lead to such a rejection:

- The software component is no longer used.
- There are linkage issues that cannot be resolved easily. The decision is to stop development and design a better solution.

- There are license incompatibility issues that cannot be resolved easily. The decision is stop development and design a better solution.
- There are intellectual property issues preventing the use or release of the specific component.
- Other reasons: each depends on the specifics of the software component in question and its usage model in the final commercial product or service.

INCREMENTAL COMPLIANCE PROCESS

Incremental compliance is the process by which compliance is maintained when product features are added to a baseline version that has already completed initial compliance. (Initial compliance, also called baseline compliance, happens when development starts, and goes until the release of the first version of the product.) Incremental compliance requires a comparatively small effort in comparison to the efforts involved in establishing baseline compliance.

Figure 19 illustrates product development and incremental compliance.

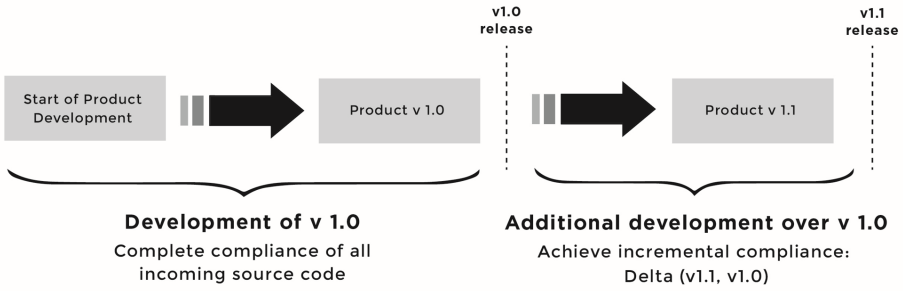


Figure 19. Incremental compliance

In this example, the compliance team identifies all open source included in the software baseline (here called v 1.0), and drives all of the open

source components through the compliance end-to-end process. Once the product ships, development begins on a new branch that includes additional features and/or bug fixes — in this example, v 1.1.

Several challenges arise with incremental compliance. Specifically, you must correctly identify the source code that changed between version 1.0 and version 1.1, and verify compliance on the delta between the releases:

- New software components may have been introduced.
- Existing software components may have been retired.
- Existing software components may have been upgraded to a newer version.
- The license on a software component may have changed between versions.
- Existing software components may have code changes involving bug fixes or changes to functionality and architecture.

The obvious question is “How can I keep track of all of these changes?”

The answer is simple: a bill of material difference tool (BOM diff tool), as discussed in Chapter 7. Briefly, for the purpose of this discussion, the tool provides you the delta between two BOMs for the same product or service. Given the BOM for product v1.1 and the BOM for v1.0, we compute the delta, and the output of the tool is the following:

- Names of new software components added in v1.1
- Names of updated software components
- Names of retired software components

Knowing this information, achieving incremental compliance becomes a relatively easy task:

- Enter new software components into the compliance end-to-end process.
- Compute a line-by-line diff of the source code in changed software components, and decide if you want to scan the source code again or rely on the previous scan.
- Update the software registry by removing the software components that are not used anymore.

Figure 20 (next page) provides an overview of the incremental compliance process. The BOM file for each product release is stored on the build server. The BOM diff tool takes two BOM files as input, each corresponding to a different product release, and computes the delta to produce a list of changes as previously discussed. At this point, the compliance officer will create new compliance tickets for all new software components in the release, update compliance tickets where source code has changed and possibly re-run them through the process, and finally update the software registry to remove retired software components from the approved list.

OSRB USAGE FORM

Completing the OSRB usage form (also called a request form) is one of the most important steps when bringing open source software into a company (ingress), and should be taken very seriously. Developers fill out the online form requesting approval to use a given open source component. The form comprises several questions that will provide necessary information for the OSRB, allowing it to approve or disapprove the usage of the proposed open source component. Table 6 (page 97) highlights the information requested in an OSRB usage form. Usually, these values are chosen from a pull-down menu to make the data entry efficient.

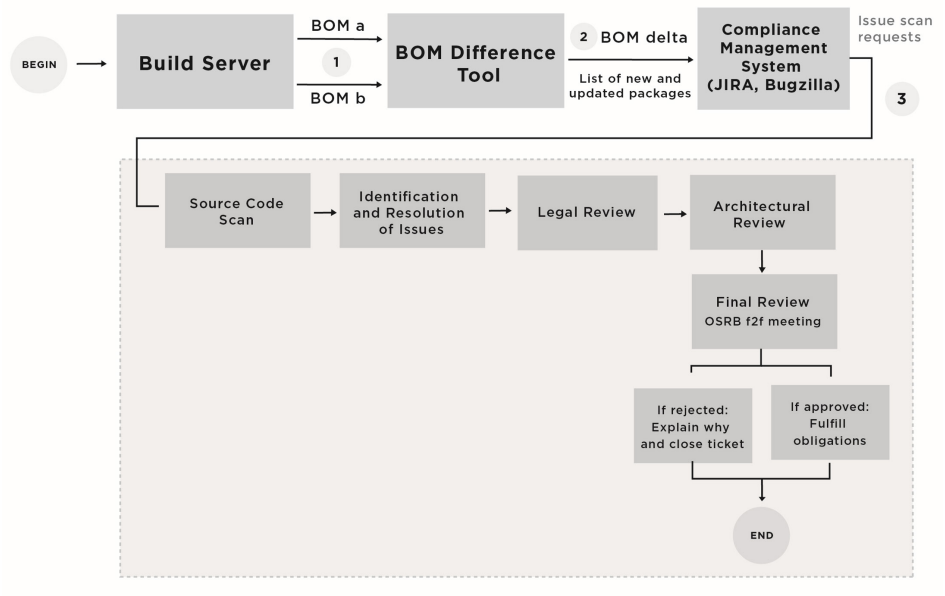


Figure 20. Example of an incremental compliance process

Note on Downloaded Open Source Packages

It is vital to archive open source packages downloaded from the web in their original form. These packages will be used in a later stage (prior to distribution) to verify and track any changes introduced to the source code by computing the difference between the original package and the modified package. If a third-party software provider uses open source, the product team integrating that code into the product must submit an OSRB usage form describing the open source to be used. If the third-party software provider provides only binaries, not source code, then the product team and/or the software supplier manager who manages the relationship with the third-party software provider must obtain a confirmation (for instance, a scan report) that there is no open source in the provided software.

Table 6. Information requested as part of the OSRB usage form

Field	Description
Submitter Information	Company ID of employee submitting the form (facilitating retrieval of employee name, phone, email, manager, location, and team from a company directory)
OSS Code Information	Package name and version Software class: open source, internally developed, third-party shrink-wrap Website URL Description License name and version License website URL Software category: OS/kernel, driver, middleware, library, utility, other (explain), etc. Benefits of using the OSS component Alternatives to using the component/package Consequences of not using the software Location of the software in the SCMS
Use Case	Internal (tools, IT, etc.) Distributed as part of a product Enabling an outward-facing service
Modification	Modified (Y/N)? Includes company IP? Exposes IP?

Note on Architecture Diagram

The architectural diagram illustrates the interactions between the various software components in an example platform. Figure 21 provides an example architectural diagram that shows:

- Module dependencies
- Proprietary components
- Open source components (modified versus as-is)
- Dynamic versus static linking
- Kernel space versus user space
- Shared header files
- Communication protocols
- Other open source components that the software component in question interacts with or depends on, especially if it is governed by a different open source license

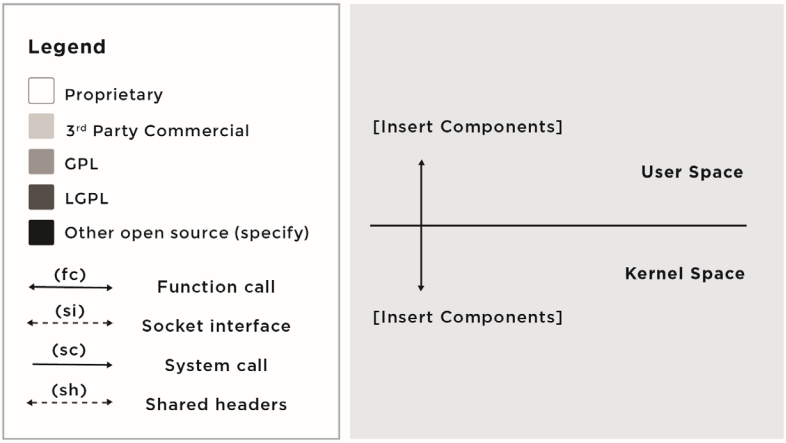


Figure 21. Template for architectural diagram (applies to an embedded environment that relies on C or C++)

Rules Governing the OSRB Usage Form

There are several rules governing the OSRB usage form. Here are a few.

- The form applies only to the usage of open source in a specific product and in a specific context. It is not a general approval of the open source component for all use cases in all products.
- The form is the basis of audit activity and provides information the OSRB needs to verify if the implementation is consistent with the usage plan expressed in the form, and with the audit and architectural review results.
- The form must be updated and resubmitted whenever the usage plans for that specific open source component changes.
- The OSRB must approve the form before engineering integrates the open source into the product build.
- The OSEC must approve the usage of any open source package where licensing terms require granting a patent license or patent non-assertion.

AUDITING

Good audit practices ensure a thorough understanding of the provenance of all software that will be deployed as part of a product or service. With this understanding comes the ability for your organization to meet its open source software license obligations. The auditing policy is usually simple and straightforward: All source code included in the portfolio/stack must be audited and have an audit report attached to its compliance ticket. The audit process consists of the workflow that requests follow after engineering has submitted an OSRB usage form for a specific software component.

The audit process consists of the following phases (Figure 22):

Receive the OSRB usage form, which includes the location of the source code to audit.

- Perform a scan of the source code.
- Perform analysis of components flagged by the scanning tool.
- Produce final auditing report.

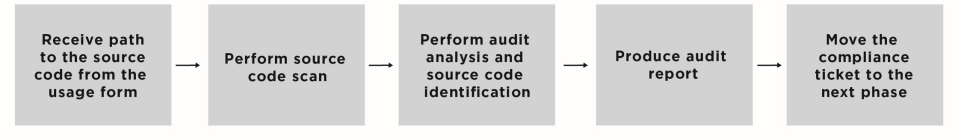


Figure 22: Basic audit process

SOURCE CODE DISTRIBUTION

The goals of the source code distribution process and policy are to ensure that:

- Customers who buy the product or use the service containing open source software are informed of their rights to receive the source code when applicable
- The source code distributed is the correct version, corresponds to the binary version included in the product transmission of the software to the distribution site, and is labeled appropriately

Distribution Incentives

There are three major business incentives to distribute open source code: Meeting license obligations, contributing enhancements to an open source project, and creating and contributing code to a new open source project.

Meeting License Obligations

In this instance, your organization has incorporated open source into a product or service, and due to the open source component license, you have the obligation to make source code available, including any modifications to it. This is informally considered a one-way distribution, versus a two-way that involves full community interaction.

Contributing modifications to an existing open source project

In some cases, the open source license does not include an obligation to make your modifications available for license compliance purposes. However, you may consider releasing your modifications and possibly upstreaming them to reduce your technical debt, or, in other words, the cost of maintaining those modifications.

Creating a new open source project

Organizations may have a business need to create a new open source project and contribute source code to it. This case is different from contributing source code (in the form of bug fixes or new feature implementation) to an existing open source project.

Distribution Policy and Process

The goal of a distribution policy is to govern the process of supplying source project code and to provide guidelines on the various logistical aspects of meeting open source license obligations regarding the availability of open source code. The distribution policy applies to any software package where the license requires redistributing source code, and it covers the publication process, publication methods, modes, and checklists.

Prior to triggering the process, you need to decide on the method and mode of providing source. Subsequently, the process begins with preparing source code for external distribution, following a pre- distribution checklist, ensuring source code package availability, and then completing the post-distribution checklist.

Figure 23 illustrates a sample distribution process. It includes:

- Deciding on the source code supply method
- Deciding on a distribution mode
- Preparing the source code packages for external distribution
- Completing a pre-distribution checklist to ensure that all prior steps have been successfully completed and that the source code packages are ready for external distribution
- Executing the distribution
- Completing a post-distribution checklist to capture any possible errors that took place as part of the distribution process

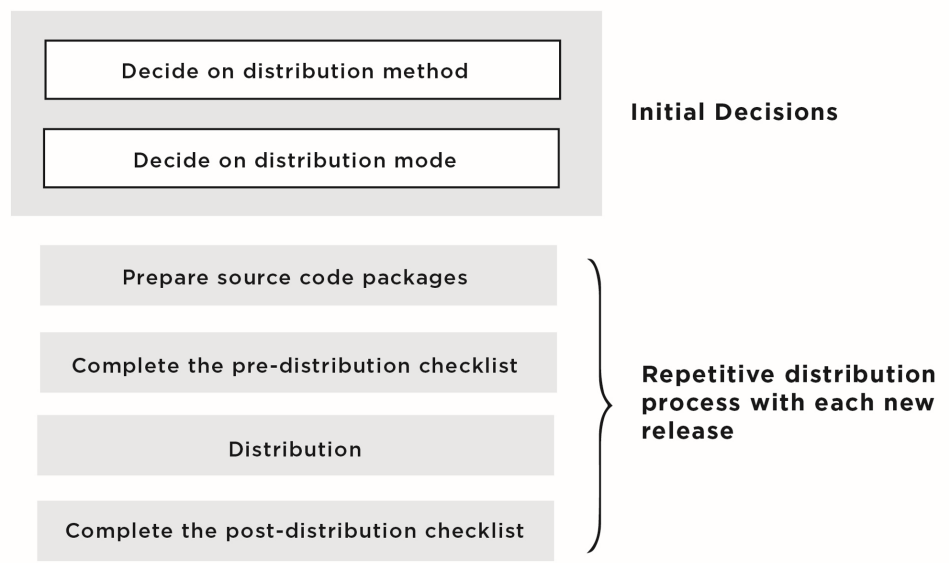


Figure 23. Sample publication process

Distribution Methods and Modes

There are three main distribution methods for making source code packages available.

Instant Compliance Method

Following this distribution method, you provide the code when or shortly after your product or software stack ships, and make it made available to anyone who wishes to get it, typically via a website download. This is usually the preferred method of distribution for developers (and compliance enforcers), as it gives them direct access to the source code without having to pass the eligibility requirement (i.e., they don't need to buy the product to be eligible to receive the source code). In some cases, the instant compliance method can be accomplished by including the source code packages on the product/device in a media directory.

There are two drawbacks to this method of distribution. Firstly, it requires a lot of effort to package all source code and make it available on the website by the date the product ships, at a time when all hands are on deck getting the product out the door. Secondly, you will be building up an expectation that future source code distributions will use this method as well. This is a very high expectation to meet every single time.

Online Supply Method

Following this distribution method, you provide exclusive access to your customers only, as they are the only entities eligible to receive the source code. This method is best managed via a secure website that requires a certain authentication on the part of the client to access and download the source code packages.

On-demand Compliance

This distribution method is a variation of the online supply method in which you rely on the written offer (in the case of the GPL/LGPL family of licenses) to communicate to your clients how they can request or access the source code. Some organizations prefer to have a written request, sent

to a corporate email address or a postal address (specified in the written offer), the result of which is that the client receives a copy of the source code, after the verification of their eligibility. This compliance method often gives the organization additional buffer time to finish packaging the source code after the product has shipped. However, generally speaking, it is not regarded as a preferred method of distribution, due to the overhead it poses in verifying the eligibility of the people asking to access the source code and the resources required to fulfill the requests. In addition, with the specific cases of the GPL/LGPL family of licenses, the written offer to access the source code must be valid for three years. Therefore, you will need to administer the distribution of the code for at least three years from the date you have last shipped your product. If you opt for providing the source code on a CD-ROM, this will introduce some additional cost to you and additional verification steps to ensure the correctness of the process of burning the CD-ROM containing the source code packages.

Distribution Checklists

There are many checkpoints for validating the open source package before it makes its way to the website for customer and/or public consumption. And additional validation is required after source code becomes publicly available. Below we outline the pre- and post- distribution process.

Pre-Conditions for Distribution

The following is a list of conditions that should be met before source code packages are ready for distribution (aka distribution hygiene):

- The open source package has been approved by OSRB for usage that corresponds to what was declared in the usage form.
- The product containing the open source packages is ready to ship or has shipped already.

- If you are making GPL/LGPL licensed code available, ensure you are providing and documenting the modifications you have introduced.
- You've performed a linguistic review. Although this is not compliance-related, there have been issues in the past related to future product code-names used, obscene or vulgar language, and references to individuals, email addresses, and/or internal URLs left in the code.

Pre-Distribution Checklist

The following is a sample checklist to follow before publishing or distributing source code:

- Verify that the modifications that were introduced to the open source package are documented and included in the open source release notes as part of the change log. Ensure that each modified source code file contains an additional entry for a copyright notice, disclaimer, and a generic "change log" entry.
- Confirm that all contents of the source code package have been reviewed by Engineering and confirmed by OSRB.
- Ensure that the open source package compiles on a non-corporate machine. It is often the case when compiling packages on a company-configured machine that environment and compiler settings are all preconfigured and set. However, when you try to compile the package on another system, the compilation settings, makefile options, include paths, etc. may break. The goal with this step is to ensure that the open source package you are about to distribute compiles on a vanilla, end-user system.
- Update the product manual to:
  - Mention that the product includes open source software.
  - Include a listing of all licenses that correspond to the different open source software included in the product.

    - Offer proper copyright and attribution notices.
    - Indicate how to access the code of the open source package (written offer) either through a web page download or by contacting your company via email or postal mail at a specified address provided in the product manual.
- Verify that the written offer is sufficient to cover all parts of the source code that require such an offer (principally, code licensed under any of the GPL/LGPL family of licenses).
- Perform a linguistic review to ensure that there are no inappropriate comments in the source code. Some companies forget to go through a linguistic review and when/if their product is hacked, they face embarrassment from exposure of inappropriate comments left in the source code. Another important reason to perform a linguistic review is to ensure that the source code and comments do not make reference to future product code names or capabilities.
- Ensure that existing license, copyright, and attribution notices are not disturbed.
- Verify that the source code to be distributed corresponds to the binary that goes with the product, that the source code builds into the same library distributed with the product, and that build instructions are included in the source distribution (derived binaries are usually identical except for time/date stamp).
- Verify that the package adheres to the linkage relationships and interactions defined in the OSRB usage form. For instance, if the developer declared that they will dynamically link that component to an LGPL licensed library, then we need to verify they've done so, and have not used a static linkage method instead. This is verified by using the linkage dependency-mapping tool.

- Ensure inclusion of a copy of the license text, if not already present, in a LICENSE file in the source code root folder of the open source package.
- If the source code package requires special build tools or environment, then include the details in a README file or similar.

Post-Publication Checklist

The following is a sample checklist to go through after publishing source code, to validate the source code packages being made available:

- Source code packages have been successfully uploaded to the website and can be downloaded on an external computer.
- Source code packages can be uncompressed on an external computer without errors.
- Source code packages can be compiled and built on an external computer without errors.

Written Offer

Below is an example of a written offer to provide the source code:

> *To obtain a copy of the source code being made publicly available by FooBar, Inc. ("FooBar") related to software used in this FooBar product ("Product"), you should send your request in writing to:*
>
> *FooBar Inc.*
>
> *Attention: Open Source Compliance Inquiries*

Street Address
City, State, Postal Code
Country

FooBar makes every possible effort to make the source code publicly available at http://opensource.foobar.com ("Website") within reasonable business delays. Before sending your written request, please check the Website, as the source code may already be published there.

Alternatively, if you prefer receiving requests via email and not via postal mail, the wording of the written offer would change slightly to:

To obtain a copy of the source code being made publicly available by FooBar, Inc. ("FooBar") related to software used in this FooBar product ("Product"), you should send your request in writing to opensourcecompliance@foobar.com.

FooBar makes every possible effort to make the source code publicly available at http://opensource.foobar.com ("Website") within reasonable business delays. Before sending your written request, please check the Website, as the source code may already be published there.

Chapter 6

RECOMMENDED PRACTICES FOR COMPLIANCE PROCESS MANAGEMENT

This chapter highlights some of the recommended practices and various considerations when integrating open source in commercial products. It is divided into three parts:

- Recommended practices that map to the various steps within an open source compliance management end-to-end process.
- Compliance considerations in relation to source code modifications, notices, distribution, software design, usage, linkages, and code mixing.
- Recommended practices related to the various building blocks in an open source compliance program.

COMPLIANCE PROCESS

As a refresher, the compliance management process includes the various steps a software component goes through before it is approved for inclusion in a product software stack. The process starts by identifying the various software components integrated into the product's build system, and ends by compiling a list of resulting license obligations.

The following sections provide recommended practices for processing compliance requests. The recommended practices map directly to the steps illustrated in the compliance process shown in Figure 24 (next page).

Identification Phase

In the identification phase of a compliance process, organizations identify all of the components or elements entering the build system, origin, and license information. There are three main sources for incoming source code:

- Proprietary software created by internal developers, which may include snippets of open source or which integrates open source at the component level, with dependencies on or links to open source code
- Third-party software developed by independent providers or consultants and made available under a commercial or open source license. This software category may include snippets or dependencies as above.
- Open source software developed by members of an open source project

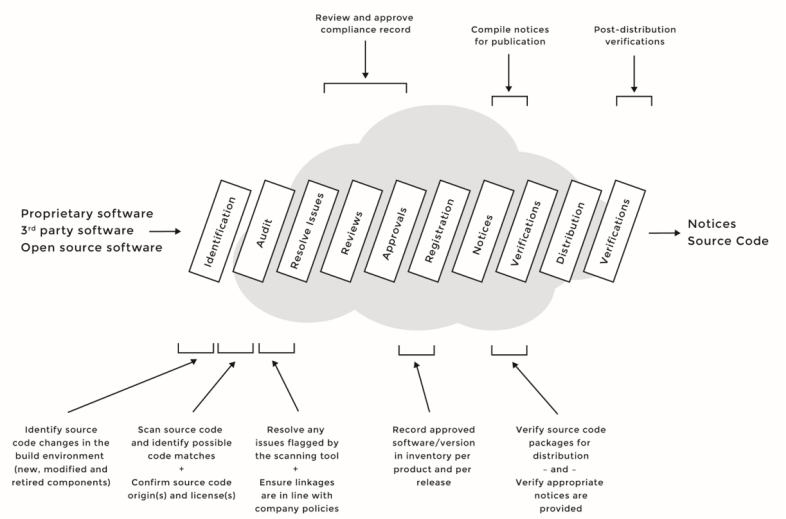


Figure 24. Compliance end-to-end management process

It is recommended to identify all the incoming software components and run them through the compliance process.

Source Code Auditing

There are three core recommended practices for source code auditing or scanning:

Scan all source code

Scan every bit of source code incorporated into products and services, because development teams may have introduced open source into proprietary or third-party source code. Furthermore, development teams may have made modifications to open source components, triggering the need for additional due diligence and potential additional obligations. Therefore, it is critical to audit and identify all source code included in a product.

Scan newer versions of previously approved packages

Sometimes a previously approved package is either modified and used again (in the same or in a different context) or used as-is or with modifications in a different product or service, or a new version is downloaded and applied in the software stack. Since compliance is verified on a product-by-product, service-by-service basis, approving for use in one case does not necessarily serve for all cases.

As a rule, each time developers modify a previously approved component or plan to use a previously approved component in a different context, component source code should be rescanned, and the component should pass the approval process again.

Ensure review and approval of each new version of open source components

License changes can occur between version upgrades. When developers upgrade versions of open source packages, make sure that licenses of new versions are the same as those employed with older versions.

There is an open source adage that goes “release early and release often.”

The open source development model encourages frequent releases, starting on project day one, to give users opportunities to experiment and report bugs. The goal is to make quality assurance activities a regular part of the development process.

"Scan early and often" follows the same spirit. Scanning source code early in the development process and continuing to do so regularly ensures that compliance efforts are not lagging behind the development efforts. Organizations should also create a list of conditions that define when a new scan is required, to make the process more efficient.

The "scan early and often" approach has several advantages:

- It aids in the discovery of compliance issue very early in the process.
- It accelerates providing solutions to discovered problems within acceptable timeframes without posing a serious threat to the delivery timeline.
- It improves the efficiency of processing incremental, scans since it reduces the delta of source code that needs to be scanned from previous source code scans.

Resolving Issues

When source code is scanned and compliance issues are discovered and flagged, there are a number of ways to resolve issues:

- When in doubt with the scan results, discuss with Engineering (interview developers responsible for specific software components in question).
- Inspect and resolve each file or snippet flagged by the scanning tool; matching source code may come from sources that surprise you.
- Identify modifications to open source. Ideally, you should not rely on engineers to remember if they made code changes (let alone

document them). You should rely on your build tools (SCM, build automation, etc.) to identify code changes, who made them, and when.

- If, for instance, the source code scanning tool identifies the use of unauthorized GPL licensed source code (a snippet) within a proprietary component, this will be reported to Engineering with a request for correction. It is highly recommended to re-scan the source code after Engineering has resolved the issue, to confirm removal of the problem source code and its replacement with appropriate and compatible code.
- In preparation for the legal review, it is best to provide attorneys with all discovered licensing information for specific components:
  - The source code audit report generated by the scanning tool
  - The COPYING, README, or LICENSE files for open source components
  - The licensing agreement for software components received from a third-party software provider

Reviews

There are different types of reviews that occur as part of a compliance process. In this section, we discuss architecture review and linkage analysis review.

The architecture review is an analysis of the interaction among open source, proprietary, and third-party software components. Companies often conduct architecture reviews with the architect responsible for the product in question, plus developers responsible for the various key software components.

The goal of this review is to identify:

- Components that are open source (used as-is or modified)
- Proprietary components
- Third-party components under a commercial license
- Component dependencies
- Communication protocols among components and subsystems
- Dynamic versus static linking (discussed in the following section)
- Components deployed in kernel space (drivers, etc.) versus user space (libraries, middleware, applications)
- Components that use shared header files
- Other open source that the specific software component interacts with or depends on, especially if it is governed by a different open source license

The result of the architecture review is an analysis of the licensing obligations that may extend from open source to proprietary or third-party components.

Approvals

As part of the approval step in the compliance process, there are two main recommended practices:

- Verify that all subtasks related to the compliance ticket have been completed and closed before approving the compliance ticket. It's easy to forget subtasks or pending subissues, but doing so may lead to prematurely closing a compliance ticket even though open issues remain.
- Record a summary of discussions that lead to approval or denial. Such documentation can prove very useful when attempting to determine on which basis approval was provided for a given component and how issues were resolved.

Notices

Organizations using open source in products and services need to:

- Acknowledge the use of open source by providing full copyright and attribution notices.
- Inform end users how to obtain a copy of the open source code (when applicable, for example in the case of GPL and LGPL licensed source code).
- Reproduce the entire text of the license agreements for the open source code included in the product.

Some recommended practices in this area include:

- Collect attribution and license notices incrementally, as open source is approved for inclusion. Following this method, the required notices file will always be up to date and will include lists of all open source, license information, copyright, and attributions notices.
- Use clear language in the written offer and be inclusive of all open source included in the product.
- Ensure that the end users of the product know how to locate this information, whether on the product itself, in the product documentation (user manual or CD-ROM), and/or on a website.

Verifications

It is very helpful and efficient to develop, maintain, and evolve checklists that cover the verification steps that the compliance team follows, both to ensure consistency and to ensure that no verification steps are overlooked. Examples of pre-distribution verifications include:

- Open source packages destined for distribution have been identified and approved
- Inappropriate comments have been removed from the source code packages (this is not strictly a compliance issue; however, comments may reveal a compliance issue that is not as visible)
- Source code packages made available (including modifications) match the binary(ies) shipping in the product or software stack
- Appropriate notices have been included in the product documentation, in addition to the availability of a written offer to inform end users of their right to request source code for identified open source (when applicable)

Once open source packages are uploaded to the distribution website (and/or stored on equivalent media), your work is not complete. You still need to verify that:

- Packages have been uploaded correctly
- Packages can be downloaded and uncompressed on an external computer without error
- Included packages compile/build properly
- Developers did not leave comments about future products, product code names, mention of competitors, or any inappropriate comments

TOOLS AND AUTOMATION

Tools are an essential element in a compliance program in that they can help organizations perform compliance activities efficiently and accurately. Many tools can prove very useful in an open source compliance program:

- Source code scanning and license identification tools
- Project management tools
- Bill of material difference tools
- Linkage analysis tools

In the following subsections we provide basic information about such tools and how their use can contribute to compliance activities. In the market, there are multiple commercial/proprietary and open source tools that provide the various functionalities described below.

Source Code Identification Tools

Source code and license identification tools provide recognition and analysis capabilities to assist users in identifying the origin of source code and licenses associated with open source software components.

- Antelink Reporter: http://www.antelink.com/
- Black Duck Protex: https://www.blackducksoftware.com/products/protex
- The Black Duck Hub: https://www.blackducksoftware.com/products/hub
- FOSSology: http://www.fossology.org/projects/fossology
- nexB DejaCode: http://www.nexb.com/products.html
- Open Logic Exchange: http://www.openlogic.com/products-services/openlogic-exchange
- Palamida Enterprise: http://www.palamida.com/products/enterprise
- Protecode Enterprise: http://www.protecode.com/our-products/
- WhiteSource: http://www.whitesourcesoftware.com

Project Management Tools

A project management tool is essential to managing and tracking compliance activities. Some companies use bug tracking tools (see list below) already in place with a customized compliance workflow; other companies rely on specific project management tools or even in-house solutions. Whatever your preference, tools should reflect the workflow of compliance processes, facilitating moving compliance tickets from one phase of the process to another, providing task and resource management, time tracking, email notifications, project statistics, and reporting capabilities.

Example bug-tracking tools commonly employed for compliance:

- Bugzilla: https://www.bugzilla.org/
- IBM Rationale ClearQuest: http://www.ibm.com/software/products/en/clearquest/
- JIRA: https://www.atlassian.com/software/jira
- Redmine: http://www.redmine.org/
- Bugzilla: https://www.bugzilla.org/

Software Bill of Material (BOM) Difference Tools

The goal of a Software BOM difference tool is to compute the difference between two BOMs and produce a list of changes. Such a tool enables efficient incremental compliance when facing newer versions of an existing base code (for instance, going from release 1.1 to 1.2). The inputs to a BOM difference tool are two BOM files that represent the list of components available on two different versions of a product or service code base. The output of the BOM difference checker documents the list of new components, retired components, and modified components.

BOM management tools are plentiful in the world of physical manufacturing, but less so for managing use of open source software. In this author's experience, BOM difference tools that support open source management

processes are usually home-grown and/or built as mash-ups of existing tools and capabilities. Depending on the form and format of the bill of materials, it is possible to use command-line diff tools, productivity tools (spreadsheets, etc.), directory comparison tools, and reports from build and continuous integration tools, plus scripting "glue," to create web-based BOM version comparisons. Figure 25, created for illustration purposes, shows the sample output of a homegrown BOM difference tool.

BOM Difference Report

Package	Image	Owner	Version	Submission
adapterbase	none	chad.everett@acme.co.nz > richardburton@acme.co.nz	1.0.0	23
adec-omxump3-msm7x25	customization	pamela.anderson@acme.co.nz	1.0.0	1 > 2
amazonservice	luna	una.due@acme.co.nz	1.0.0	4 > 5
audiod	luna	damien.lewis@acme.co.nz	1.0	111 > 114
audiod-config	luna	damien.lewis@acme.co.nz	1.0	70 > 72
browser-adapter	cust	chris.reeve@acme.co.nz	1.0.01	118 > 131
browser-service	cust	chris.reeve@acme.co.nz	1.0.01	126 > 128
bzip2	rockhopper	> eric.idle@acme.co.nz	1.0.2	1
camd-omap	any	itzaak.periman@acme.co.nz	1.0.0	S > 7
cifs	rockhopper	> eric.idle@acme.co.nz	3.0.23c	1
clam	any	sean.connery@acme.co.nz	1.0.0	7 > 8

Figure 25. Example BOM difference report

Linkage Analysis Tool

The goal of the dependency checker is to flag problematic code combinations at the dynamic and static link level, specific to C and C++ programming languages. The tool identifies a linkage conflict between the license of the binaries and the license of the libraries it links to, based on predefined license policies that the user of the tool has already defined.

There are many tools that can be used together to fulfill the function of dependency checking, including the Source Code Identification Tools listed above, many static analysis tools, reporting tools from build and integration suites, and many home-grown code analysis tools. The main requirements for dependency mapping are the abilities to:

- Identify linking among binaries and libraries
- Identify licenses for binaries and libraries

- Connect with license scanning tools or consume the output thereof
- Configure to match company policy preferences to flag linkages that violate those policies (e.g., linking with GPL-licensed code)

In this author's experience, Dependency Mapping tools, much like BOM tools, are usually home-grown and/or built as mash-ups of existing tools and capabilities.

One off-the-shelf open source tool of this type is The Linux Foundation Dep-Checker (http://git.linuxfoundation.org/dep-checker.git/).

CHAPTER 7

MANAGING COMPLIANCE INQUIRIES

This chapter presents guidelines for handling compliance inquires. These guidelines aim to maintain a positive and collaborative attitude with requesters while investigating allegations and ensuring proper actions when violations actually occur.

Several organizations have received negative publicity and/or have been subject to legal action after ignoring requests to provide compliance information; did not know how to handle compliance inquires; lacked or had a poor compliance program; or simply refused to cooperate, thinking (incorrectly) that license terms were not enforceable. Today, best practices inform us that none of these approaches are beneficial to any party involved. Therefore, companies should not ignore compliance inquiries — rather, they should acknowledge receipt of inquiries, inform inquiring parties of pending response, and provide an estimated date for follow-up.

Compliance inquiries can include requests for:

- Access to source code in accordance with a written offer to provide source code licensed under GPL, LGPL, or other licenses
- Access to source code for an undisclosed component that was discovered in a product
- Verification of whether a specific open source component is used in a product or service
- Update to an out-of-date attribution or copyright notice
- Providing files missing from open source packages made available as part of license obligations

Companies usually receive compliance inquiries through a dedicated email address that they advertise in their written offer or as part of their open source notices.

RESPONDING TO COMPLIANCE INQUIRIES

This section introduces a method for responding to compliance inquiries. Figure 26 presents a sample inquiry response process that illustrates the steps a compliance inquiry goes through, from receiving the inquiry until its closure.

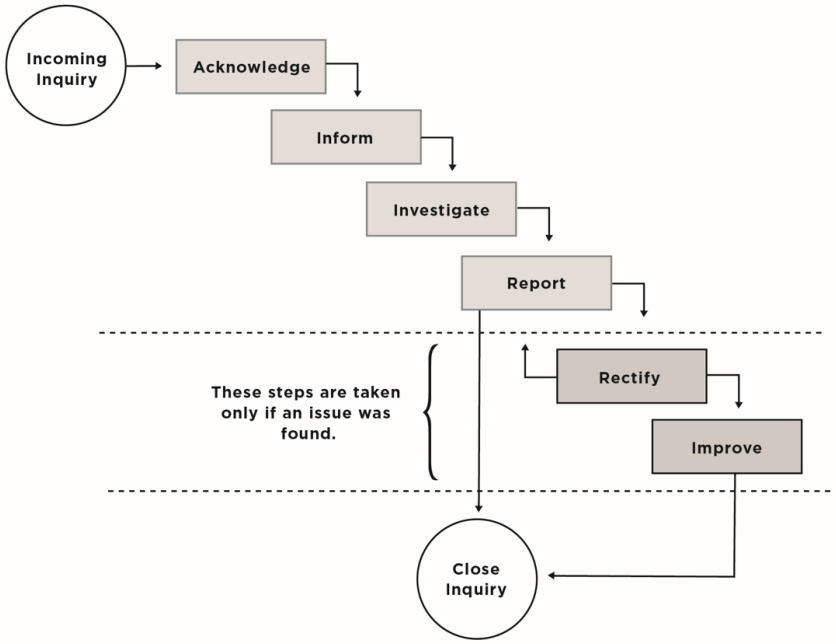


Figure 26. Process of responding to compliance inquiries

Acknowledge

Once you receive the compliance inquiry, you should respond immediately, confirming receipt and committing to investigate by a specific date.

It is important to understand the inquirer's identity and motive and to verify whether the complaint is justifiable, accurate, and current. Realize that inquirers often don't fully understand licenses, leading to mistaken assumptions and submissions. If an inquiry is missing information, request additional clarification, such as:

- The name of the affected product(s) or service(s) or the exact code of concern
- The reason why a violation is believed to exist
- The name of the project code and license that may have been violated
- A link to the project site

Inform

It is recommended to maintain an open a dialog with inquirers. Always highlight your open source compliance practices and demonstrate historical good faith efforts toward compliance. Inform inquirers about your compliance program and practices, and assure them that you will investigate their question. It is also advisable to send updates of your internal investigation as they become available.

Investigate

In this step, investigating reported allegations, you should refer to the compliance record for the component in question, review it, and verify if and how the compliance record compares with the inquiry.

Report

After concluding the internal investigation within an acceptable period of time, and creating an internal record of the findings, you need to inform the inquirer of the results.

Close Inquiry

If the compliance inquiry was a false alarm, you can close the compliance inquiry ticket without any further action (other than informing the inquirer of that resolution).

Rectify

If the investigation uncovers an actual compliance issue, you should report back to the inquirer confirming that fact, with assurances that your organization will take all the necessary steps to bring your product or service back to compliance, specifying a date by which you expect to complete this task. It is your responsibility to resolve the issue with the inquirer, while being collaborative and showing good will. You need to show that you understand the obligations under the applicable license, communicate how — and when — you will meet the obligations.

Once you fix the problem, you should notify the inquirer immediately, and invite them to verify the solution.

Improve

If there was a compliance issue, you should call for an OSRB meeting to discuss the case, learn how this non-compliance occurred, and improve existing process and practices to ensure that such errors do not happen again.

General Considerations

- Treat all inquiries as formal inquiries, and work under the assumption that any information you disclose as part of the interactions with the inquirer can become public.
- Consider how your existing open source compliance efforts would measure up in an enforcement action, and work to improve your processes.

CHAPTER 8

OTHER COMPLIANCE-RELATED PRACTICES

This chapter highlights compliance best practices and various considerations outside of the actual compliance process.

EMPLOYEE APPRAISAL

There are four challenges that all companies face with regard to engineering and compliance enforcement:

- Ensuring engineers consistently fill out request forms for each open source component they want to use
- Requiring engineers to respond in a timely fashion to compliance tickets
- Verifying engineers are following the guidelines set by the OSRB
- Mandating engineers to take your internal open source compliance training

A practice that has proved to be effective in helping companies face these four challenges is to include open source and compliance metrics as part of employee performance reviews. As a result, part of the developers' yearly bonus will depend on the extent to which they have followed the compliance policies and procedures. Reviews may evaluate performance on whether employees:

- Fill out OSRB forms for each open source component they use
- Respond to compliance tickets without significant delays
- Complete the open source and compliance training within a time limit set by the manager
- Use open source within the guidelines that the OSRB has set and do not cause a compliance violation

In turn, to use compliance as a factor in employee performance reviews, the OSRB must track these issues for each developer:

- Components that were included in the software BOM that don't have a corresponding approval
- Response time to compliance tickets
- Course completion
- Compliance violations reported to the executive team

WEB PORTALS

Some companies maintain both an internal and an external open source web portal. The internal portal hosts compliance policies, guidelines, training material, announcements, and access to related mailing lists. The external portal offers a consistent means of posting source code of open source packages they use, in fulfillment of license obligations.

MESSAGING

The single most important recommendation with respect to messaging is to be clear and consistent, whether internally — explaining company goals and concerns around open source, or externally — facing community participants. Having a community-facing site is particularly important when responding to compliance inquiries.

TRAINING

The goal of open source and compliance training is to raise awareness of open source policies and strategies and to build a common understanding of the issues and facts of open source licensing. Training may also cover the business and legal risks of incorporating open source in products. It also serves as a way to publicize and promote an organization's compliance policies and processes, and to promote a culture of compliance.

There are formal and informal training methods. Formal methods include instructor-led training courses where employees have to pass a knowledge exam to pass the course. Informal methods include webinars, brown bag seminars, and presentations to new hires as part of the new employee orientation session.

Informal Training

Brown Bag Seminars

Brown bag seminars are usually presentations offered during lunchtime by either a company employee (in-house legal counsel, open source expert, compliance officer, etc.) or an invited speaker (most commonly a high profile open source developer). The goal of these seminars is to present and elicit discussions about the various aspects of incorporating open source in products or software stacks. These sessions can also include discussions of the company's compliance program, policies, and processes.

New Employee Orientation

In some instances, the Compliance Officer presents on organization compliance efforts, rules, policies, and processes to all new employees as part of the new employee orientation session. On their first day, new employees would receive a 30-minute training on open source and compliance. As a result, the new employees will have all the necessary information they need, such as who are the internal subject matter experts, what intranet resources exist, and how to sign up for open source and compliance training.

Formal Training

Depending on the size of the organization and the extent to which open source is used in commercial offerings, the organization can mandate that employees working with open source take formal instructor-led courses and be tested on their subject-matter proficiency.

SOURCE CODE MODIFICATION CONSIDERATIONS

It is strongly recommended to publish an internal-only set of guidelines in plain, non-legalistic language that establishes basic rules for modifying existing source code. For example:

- Source code modifications that will remain proprietary must not be made within an open source package, especially one that has derivative work obligations (e.g., GPL or LGPL).
- Proprietary source code must not link to an open source library that has a derivative work obligation. Companies usually request formal OSRB approval for such action.
- Ensure that any modifications to source code are documented in compliance with the open source license prior to distribution.
- All modifications to open source code modules shall be captured in the revision history of the module (change log file).

NOTICES CONSIDERATIONS

One of the key obligations when using open source is to ensure clear and accurate documentation of copyright, attribution, and license information, and the availability of a written offer (for GPL/LGPL licensed source code). The sum of all of these documentation obligations is often referred to as open source notices.

Companies using open source in their offerings must acknowledge the use of open source by providing full copyright attribution, and, in most cases, reproducing the entire text of the licenses of the open source software included in the product or service. Therefore, companies must fulfill documentation obligations by including copyright, attribution, and license notices text in the documentation of every product they ship and service they provide.

There are two primary options for fulfilling documentation obligations requirements:

- Display the open source notices on the product itself. This is a viable option if the product has a user interface that allows the user to interact with it and pull up or display licensing information. An example of this option is a cell phone or a tablet.
- Include the open source notices in the product manual or any kind of documentation accompanying the product.

Some companies opt for both options when possible, in addition to maintaining these notices on a given website (optional, but also often adopted, and it's low maintenance — basically just hosting the notices file on the website). The important takeaway from the notices considerations is to ensure that all open source notice requirements are satisfied prior to a product distribution or service launch.

DISTRIBUTION CONSIDERATIONS

Generally speaking, companies want to ensure that any source code subject to open source distribution obligations is compliance-ready prior to product shipment. By thoroughly integrating compliance practices into the development cycle, distribution considerations can be greatly simplified and streamlined.

USAGE CONSIDERATIONS

The following sections address considerations and caveats for using open source in a fully compliant manner.

Clean Bill of Materials (BOM)

Ensure that any inbound software does not contain undeclared open source. Always audit source code upon receipt from providers; alternatively, make it a company policy that software providers must deliver source code audit reports for code they supply.

OSRB Form for Each Open Source Component

Fill out an OSRB usage request form for each open source component in use. Avoid using any open source without explicit OSRB approval.

Understand the Risks During Mergers and Acquisitions (M&A)

Understand the open source code in use and its implications as part of the due diligence performed prior to any corporate transaction.

Retired Open Source Packages

If an approved open source package is no longer in use, engineers must inform the OSRB to update the open source inventory; alternatively, the OSRB will discover that the package is not used anymore when they run the BOM diff tool.

Major Source Code Changes

If an approved package went through a major change, inform the OSRB to re-scan the source code; alternatively, the OSRB will discover that the package has been modified when they run the BOM diff tool. A major change in the design or implementation often impacts architecture, APIs, and use cases, and in some cases may have an impact on the compliance aspect.

Reference Original Source Code

Document the URL from which you downloaded the open source package in addition to saving an original copy of the downloaded package.

Upgrading to Newer Versions of open source

Ensure that each new version of the same open source component is reviewed and approved. When you upgrade the version of an open source package, make sure that the license of the new version is unchanged from the prior version, as license changes can occur between version upgrades. If the license changed, contact the OSRB to ensure that compliance records are updated and that the new license does not create a conflict.

Compliance Verification Golden Rule

Compliance is verified on a product-by-product, service-by-service basis: Just because an open source package is approved for use in one context does not necessarily mean it will be approved for use in a second one.

Copy/Paste

Avoid using source code snippets, and avoid copying/pasting open source code into proprietary or third-party source code (or vice versa) without prior documented OSRB approval. Such actions have serious implication on compliance.

Mixing Source Code with Different Licenses

Avoid mixing different open source licenses in a derivative work, as many open source licenses are incompatible with one another. It is highly recommended to seek legal support from your Counsel on this topic.

Source Code Comments

Do not leave inappropriate comments in the source code (private comments, product code names, mention of competitors, etc.).

Existing Licensing Information

Do not remove or in any way disturb existing copyrights or other licensing information from any open source components that you use. All copyright and licensing information must remain intact in all open source components, unless you are completely certain the license allows it to be changed.

ATTRIBUTION CONSIDERATIONS

Companies that include open source in a product need to provide required attribution to the end user. This section provides guidelines of how to fulfill open source attribution obligations.

Attribution Types

Open source attribution requirements differ from license to license, but can generally be grouped into four categories:

Full License Text

A verbatim copy of the full license text is required for almost all open source licenses.

Copyright Notices

A verbatim copy of the copyright notices is required for many open source licenses.

Acknowledgments Notices

Some open source licenses explicitly require author attribution. In most cases, open source projects maintain a file called AUTHORS that includes the list of contributors; you can use this information as part of the attribution notice.

Information on Obtaining the Source Code

Most licenses with a source code redistribution obligation require that either the source code accompany the product or that the user receive a written offer with details on how to obtain the source code. The GPL and LGPL are examples of licenses in this category.

Presentation of Attributions

For each product or service containing or using open source, the attributions must be included in published user documentation (such as the product manual) distributed in printed or electronic form, such as a CD or a download from a website. If products or services possess a graphical user interface or a command line administrative interface, you can also provide the option to display the attributions via that UI. For product updates such as over-the-air (OTA) updates for cell phones, the attributions must also be revised when the product update includes new or updated open source components.

SPECIFIC LICENSE OBLIGATIONS

"Must include a copy of the license in documentation available to the end user"

The license of the open source component in question must be included in the user documentation for all products using this open source.

RECOMMENDATIONS

- In some instances, such as with mobile phones or tablets, manufacturers are able to provide the notices on the actual device via a web browser or a PDF viewer (i.e., licensing text is available on the device either in HTML or PDF format).
- For products with a user accessible file system, it is recommended that the license is included in the file system with a filename LICENSE to make it stand out and to be similar to the open source license filename.

- For product updates, license information must also be updated. For instance, when a new software release becomes available, the updated release must include an update license information file to reflect any open source changes introduced in the new release. Changes may include:
  - New open source used
  - Deprecated/removed open source
  - Open source upgraded to a new version, which may require updating the attribution/copyright notices, and, in some rare cases, updating the license

"Must include copyright notices in documentation available to the end user"

The license of the open source component in question may require including copyright notices in the product document available to the end user.

RECOMMENDATIONS

- For all products, copyright information must be included in printed documentation (such as a user manual).
- If the use case includes a graphical user interface, the end user should be able to view the copyright information from an ABOUT or a LICENSE screen.
- If the product has a user-accessible file system, the copyright information should be included in the file system in a file containing, for instance, all the copyright notices for all open source used in the product.
- For products updates, the copyright information must also be updated.

"Advertising materials may need special acknowledgments"

This advertising clause from the original BSD license is written as follows:

All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.

Where applicable, all marketing and advertising material (including web-based, magazines, newspapers, flyers, etc.) must display the acknowledgement.

GENERAL GUIDELINES

You're probably already familiar with some of the guidelines that apply to open source licenses, such as not using the name of the open source project for endorsement, marking the source code modifications you have introduced, and preserving the original licensing, copyright, and attribution information. The following sections expand on these general guidelines in more detail.

No Endorsing or Promoting

You cannot use the name of the open source project, authors, or contributors in any marketing, advertising, or documentation (hard copy, digital, or on the web) without prior written permission.

Source Code Modifications Markup

When redistributing modified open source code, your modifications need to be clearly marked as such, including a copyright line for those modifications (company, year) while preserving the existing copyright lines.

Some companies elect a different approach — providing the original open source code along with the company's contributed patch files that apply against the original open source code. Following this approach, the company's modifications are clearly separated from the original open source code.

Preserving Original License, Copyright, and Attribution

Whenever you are redistributing open source code, with or without modifications, you must preserve the original licensing information, copyright lines, and other attributions.

Source Code Comments

Do not leave any inappropriate comments in the source code, such as private comments, product code names, mention of competitors, etc.

Existing Licensing Information

Do not remove or in any way disturb existing open source licensing copyrights or other licensing information from any open source components that you use. All copyright and licensing information is to remain intact in all open source components.

Chapter 9

SCALING OPEN SOURCE LEGAL SUPPORT

Open source compliance is often more of an operational and logistical challenge than a legal challenge. Achieving compliance requires the proper policies and processes, training, tools, and proper staffing that enable an organization to effectively use open source and contribute to open source projects and communities, all while respecting copyrights of their respective holders, complying with license obligations, and protecting the organization's intellectual property and that of its customers and suppliers.

However, legal counsel plays an indispensable role in supporting the open source compliance programs and core teams that most organizations create to ensure proper compliance. In this chapter, we look closely at the role of the Legal Counsel in ensuring open source compliance, and offer practical advice that a Legal Counsel can provide to the software development team. Such practical advice will enable software developers to make day-to-day decisions related to open source licenses without having to go back to Legal Counsel for every single question.

PRACTICAL LEGAL ADVICE

Practical advice from Legal Counsel to software developers may include:

- **License Playbooks:** Easy-to-read, digest-form summaries of open source licenses intended for software developers
- **License compatibility matrix:** A grid to help determine whether License-A is compatible with License-B. Software developers can use such a matrix as they merge incoming code from different projects under different licenses into a single body of code.
- **License classification:** An easy way to understand the different licenses, and the course of action needed when using source code provided under these licenses

- **Software interaction methods:** A guide to understanding how software components available under different licenses interact, and if the method of interaction is allowed per company compliance policies
- **Checklists:** A consistent, foolproof way to remember what needs to be done at every point in the development and compliance processes

In the following sections, we examine these five pieces of advice, provide examples, and discuss how they help software developers working with open source.

LICENSE PLAYBOOKS

License playbooks are summaries of commonly used open source licenses. They provide easy-to-understand information about these licenses, such as license grants, restrictions, obligations, patent impact, and more. License playbooks minimize the number of basic questions sent to Legal Counsel and provide developers with immediate legal information about these licenses.

Figure 27 (next page) provides an example license playbook for the GPL v2. Please note that this playbook is provided for illustration purposes only and its content should not be considered definitive.

SAMPLE PLAYBOOK FOR THE GPL v2.0

License Grant:

1. Copy and distribute verbatim copies of source code

2. Modify source code and copy and distribute copies of modified source code

3. Copy and distribute copies of object code

License Limitations:

Modified source code

Mark that source code has been changed (change log)

All source code

Mark with copyright notice

Mark with disclaimer of warranty

Keep intact with all other notices

Object code

Accompany with source code

Written offer to provide source code upon request

License Obligations:

Must include a copy of the license in documentation available to the end-user

Must inform user of how to obtain the source code and that it is covered by GPL v2

Must redistribute source code [including modifications, if any]

When redistributing source code, must include a copy of the license

Source code modifications must be clearly marked as such and carry a

Figure 27. Example license playbook for GPL v2 (for illustration purposes only)

LICENSE COMPATIBILITY MATRIX

License compatibility is the determination of whether a software component and its license are compatible with one or more other components and their licenses (i.e., that their licensing terms do not conflict). Compatibility also

addresses the appropriate licenses for works that combine two or more licenses (combined outlicensing).

License compatibility challenges can arise when combining diverse open source software components, in source and/or object form, that are distributed under licenses with incompatible terms. The result of such combination is a licensing chimera, an aggregation of software components that for purely legalistic reasons cannot be redistributed.

An example of licensing incompatibility can be found in attempting to combine code distributed under the Apache version 2 license with software under the GNU GPL version 2.0 (due to patent termination and indemnification provisions not present in the older GPL license). An example of license compatibility is combining code licensed under the X11 license, which is explicitly compatible with the GPL version 2.

Figure 28 illustrates the creation of a single source component that originated from multiple sources under different licenses. In this scenario, you must ensure the sources have compatible license terms that allow you to join them in a binary or an object file without any conflict.

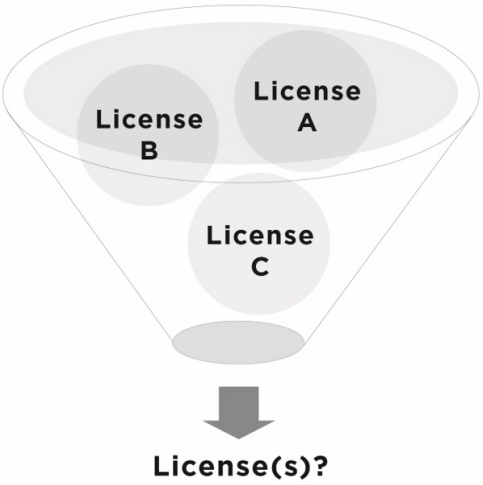


Figure 28. Combining source coming under different licenses into a single binary

License compatibility is an area where development teams need detailed guidance from Legal Counsel and should never be left to draw their own conclusions. Such guidance can be provided via a License Compatibility Matrix that covers most popular licenses. An example matrix is provided in Table 7.

Table 7. Example license compatibility matrix (for illustration purposes only)

	License-A	License-B	License-C	License-D	License-E	License-F	License-G
License-A	✔				✔	✔	
License-B		✔					
License-C			✔				
License-D		✔		✔			✔
License-E					✔		
License-F			✔				
License-G	✔						✔

When development teams need to combine code under different open source licenses, they can refer to this matrix to determine if joining the software components in question creates a licensing conflict. When a new or novel license is encountered that isn't included in the matrix, that license should be analyzed by the Legal Counsel, who should update the table accordingly.

LICENSE CLASSIFICATION

In an effort to reduce the number of questions received by Legal Counsel and to increase license and compliance process education, some companies opt to classify the most-used licenses in their products under a handful of categories. Figure 29 (next page) presents an example license classification, in which most-used licenses are divided into four categories.

Pre-approved Licenses

Permissive open source licenses often fall under this category. Source code available under these licenses may be pre-approved for use by developers without having to go through the approval process with their manager and/or legal counsel. Such pre-approvals usually also require the developer to capture any notices and to make sure they are sent to the documentation team.

Licenses Requiring Manager Approval

Manager approval is required for components distributed under these licenses, since in addition to notices fulfillment (publishing license text, attribution notice, copyright notice, etc.), you have the obligation to release any source code modifications.

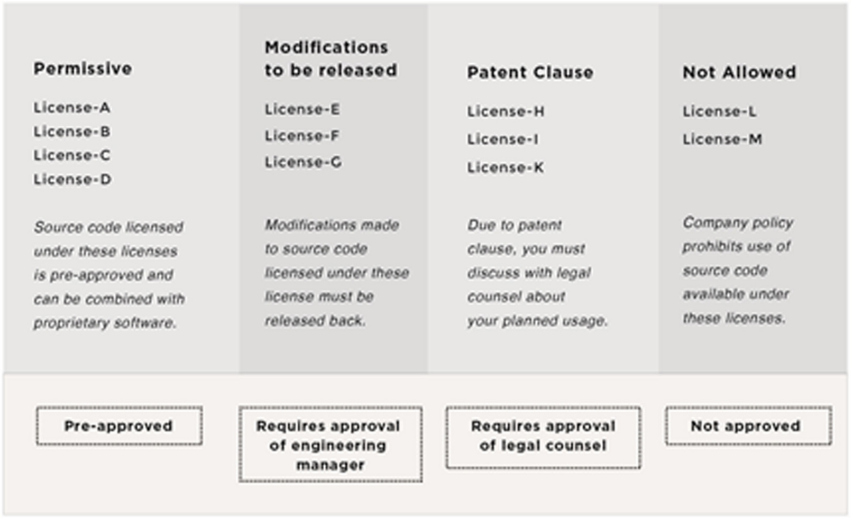


Figure 29. Example license categories (for illustration purposes only)

Licenses Requiring Legal Counsel Approval

Source code available under these licenses requires legal review and approval. This usually applies to licenses that have a patent clause.

Prohibited Licenses

Some companies flag certain licenses as “not allowed” — usage not allowed by company policy.

How can classifying licenses be helpful?

The above license categories are a way to classify licenses to make it easier for developers to know the proper course of action when integrating code under these licenses. Furthermore, it makes it easy to create an association between a license and what needs to be done. Table 8 shows one easy way developers can remind themselves of the proper actions associated with various licenses.

Table 8. A simple how-to for license classifications

Which License	Action
License A	Use with no problem
License E	Get my manager’s approval
License I	Consult with Legal
License M	Can’t use this source code
Other	Ask my manager for course of action

Please note that these different scenarios are provided for illustration purposes only. You can set up a different classification model with different actions depending on your organization’s policies and guidelines.

SOFTWARE INTERACTION METHODS

As part of the compliance process, there is usually an architecture review, the goal of which is to understand how any specific software component interacts with any other software component, and the method of interaction. Architecture review should identify:

- Components that are open source (used “as is” or modified)
- Proprietary components

- Components originating from third-party software providers (both open source and proprietary)
- Component dependencies
- Use of shared header files
- Component run-time context (kernel/drivers/modules, middleware, libraries, applications, etc.)
- Inter-component dependencies beyond APIs (s/w buses, IPCs, web APIs, etc.)
- Inter-language bindings

Tables 9 and 10 (next page) provide additional information that Legal Counsel can provide to software developers. The tables illustrate which licenses can dynamically or statically link to which others, while respecting company policies.

Table 9. Sample dynamic linkage matrix

Can Dynamically Link To	License-A	License-B	License-C	License-D
License-A	✔	✔	✔	✔
License-B		✔		✔
License-C	✔		✔	
License-D		✔	[Requires Pre-Approval]	✔

For example, looking at Table 9, source code licensed under License-B can dynamically link to source code license under License-D. However, source code licensed under License-C cannot dynamically link to source code licensed under License-B. Also, note that linkages may not always be reciprocal between licenses.

Similarly, looking at Table 10, source code licensed under License-A can statically link to source code license under License-C. However, source code licensed under License-A cannot statically link to source code licensed

under License-B. Some linkage combination may be allowed on a case-by case basis, which is why certain combinations note "[Requires pre-approval]."

Table 10. Sample static linkage matrix

Can Statically Link	License-A	License-B	License-C	License-D
License-A	✔		✔	
License-B		✔	[Requires Pre-Approval]	
License-C	✔		✔	
License-D	[Requires Pre-Approval]			✔

In the event that the architecture review reveals any linkage issue (i.e., a static or dynamic linkage that does not follow company policy as defined in the linkage matrices), then the person responsible for driving the architecture review (usually the compliance officer) would notify the software developer responsible for that software component and request a correction.

CHECKLISTS

Most companies establish checklists that are used within the development process at every major milestone. When it comes to open source compliance, several checklists can be developed and used before committing new external open source code to the product's source code repository. One example is the following checklist, used before making source code available on an external website:

- All source code components have a corresponding compliance ticket.
- All compliance tickets have been approved by engineering and legal.
- All compliance tickets are clear of any unresolved subtasks attached to them.

- Notices for all of the software components have been sent to the Documentation team and included in product documentation.
- Legal has approved the written offer notice and overall compliance documentation.
- Source code packages have been prepared and tested to compile on a standard development machine.
- Source code provided is complete and corresponds to the binaries in the product.

Such checklists minimize the probability of error and ensure that everyone involved in open source management is aware of what needs to be done before moving to the next step in the process.

CONCLUSION

Software developers need to be educated about the licenses on the various open source components they integrate and employ. Having Legal Counsel provide this education in a very practical way is extremely helpful, as it allows software developers to have access to documented practical advice that will help answer most of their daily legal-related questions. This practical advice usually revolves around:

- Inclusion of open source components into proprietary or third-party source code or vice versa
- Linking open source components into proprietary or third-party source code or vice versa
- Interaction methods between various software components (proprietary, third-party, open source)
- License obligations that must be met when using open source components

Open source compliance is easy to achieve once you have built up your compliance program, created a compliance policy and process, established staffing to ensure execution, and enabled your team with various tools to assist in the compliance automation aspect.

ABOUT THE AUTHOR

Ibrahim Haddad (Ph.D.) is Vice President of R&D, and the Head of the Open Source Group at Samsung Research America, a wholly owned R&D subsidiary of Samsung Electronics Co. Ltd., South Korea. He is responsible for overseeing Samsung's Open source strategy and execution, internal and external collaborative R&D projects, participation in key open source development projects, and representing Samsung in various open source foundations and open standards organizations.

Prior to joining Samsung, Haddad was a member of the management team at The Linux Foundation responsible for technical and legal compliance projects and initiatives. Haddad's career started at Ericsson Research where he spent five years focusing on advanced research for system architecture of wireless IP networks and on furthering the adoption of Linux and Open source software in carrier grade environments. He then joined Motorola as Technical Director managing the Open Source Technology Group and contributing to Motorola's Open source initiatives. After Motorola, he ran the Open Source function at Palm as Director of Open Source responsible for the webOS open source strategy and compliance. He later supported Hewlett Packard in a consulting role with open sourcing webOS to become the open webOS project.

Haddad graduated with Honors from Concordia University (Montréal, Canada) with a Ph.D. in Computer Science. He completed his B.Sc. and M.Sc. (both in Computer Science) at the Lebanese American University. He is a Contributing Editor to the Linux Journal, Co-Author of two books on Red Hat Linux and Fedora, and Technical Editor for four books on Linux System Administration, Fedora Linux and Ubuntu Linux. He is known for his writing and speaking on topics ranging from open source legal compliance to using open source as a business strategy and an R&D tool to drive collaboration
and innovation.

Haddad is fluent in Arabic, English, and French.

Twitter: @IbrahimAtLinux

THE LINUX FOUNDATION

The Linux Foundation promotes, protects and standardizes Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about The Linux Foundation, please visit us at linuxfoundation.org.