**Implementing Open Source License Compliance Management (LFC194)**

# Chapter 1. Course Introduction

## Course Information

### Course Overview

In this follow-on course to [*Introduction to Open Source License Compliance Management (LFC193)*](https://training.linuxfoundation.org/training/introduction-to-open-source-license-compliance-management-lfc193/), we discuss how an open source compliance management system should be structured and implemented to be most effective.

The course will start with a quick review of what open source compliance management is. Then, you will learn why it is important to conduct an open source review and who is involved in this process and why. You will also gain an understanding of the steps necessary to achieve end-to-end compliance. In addition, you will become familiar with developer guidelines that are recommended to be followed when using open source code and contributing to open source projects and communities.

The policies and processes discussed in this course can be used to manage open source compliance activities such that the legal obligations of licenses are met in a practical and consistent way, and following industry best practices as documented and advocated by [OpenChain](https://www.openchainproject.org/) – the [Linux Foundation](https://www.linuxfoundation.org/) initiative.

These processes and practices can be scaled to fit real-world situations and can be introduced to build and implement an open source compliance management system that addresses the needs of small and medium organizations.

[+ Additional standard pages added by LF]

## Before You Begin

### Glossary

**A**

**Artifact**Artifacts are like roadmaps that software developers can use to trace the entire software development process. Artifacts might be databases, data models, documentation, scripts, and bill of materials.

**Attribution**Many open source licenses direct the respective open source software consumer to give credits to the used open source software, this is referred to as attribution statement**.**

**Audit**   
In this course audit refers to an open source audit. An open source audit is a thorough investigation into your open source components.

**B**

**Build Environment**It is usually the development environment tuned for the expected compiled output of source code. It includes the referred libraries of pre-decided version, specific operation system, integration system, CI/CD pipelines, etc.

**C**

**Compliance**Here we refer to the OSS compliance. OSS compliance is a process of meeting requirements of open source licenses of software components and attributing the respective copyright statements.

**Copyleft License**  
A license requiring that derivative works are distributed under the same terms as the original work, also called reciprocal license.

**Copyright**  
Legal protections for original works of authorship.

**D**

**Dependencies**  
It can be a library or a code snippet of functionality that is important for another program to execute.

**Derivative Work**  
A new creation/work based upon an original work that has been added to in such a way that it represents a new original work of authorship and not a copy.

**E**

**F**

**Freeware**  
A term referring to software that is distributed under a proprietary license at no or very low cost.

**G**

**H**

**I**

**Intellectual Property**  
A work or invention that is the result of creativity, such as a manuscript or a design, to which one has rights and for which one may apply for a patent, copyright, trademark, etc.

**J**

**K**

**L**

**License**  
The way a copyright or patent holder gives permission or rights to someone else.

**M**

**N**

**Notice**Open source notices refer to the licenses and copyrights under which the open source components are published.

**O**

**Obligation**Open source obligation means being morally and legally bound to the terms and conditions stated by the enforced open source license on the respective open source software.

**P**

**Patents**  
Legal protections for inventions (and plants) that are novel and non-obvious.

**Permissive Open Source License**  
Term often used to describe open source licenses with minimal conditions.

**Proprietary License**   
A license that generally has restrictions on the usage, modification, and/ or distribution of the software and often does not provide access to the source code.

**Public Domain**  
Software not protected by copyright and therefore usable by the public without requiring a license.

**Q**

**R**

**S**

**Shareware**  
Proprietary software provided to users on a trial basis, for a limited time, free of charge, and with limited functionalities or features.

**Software Bill of Materials (SBOM)**Software Bill of Materials (SBOM) is a nested inventory for software, a list of ingredients that make up software components. You can learn more about SBOM in the [*Introduction to Open Source License Compliance Management (LFC193)*](https://training.linuxfoundation.org/training/introduction-to-open-source-license-compliance-management-lfc193/) course.

**T**

**Trademarks**   
Legal protections for marks (such as words, logos, slogans, colors, etc.) that indicate product origin.

**Trade Secrets**  
Legal protections for confidential business information, usually processes or procedures.

**U**

**V**

**W**

**X**

**Y**

**Z**

### License Acronyms

**Table: Examples of License Acronyms**

| **Acronym** | **Full License Name** |
| --- | --- |
| AGPL | Affero General Public License |
| MPL | Mozilla Public License |
| LGPL | Lesser General Public License |
| GPL | General Public License |
| EPL | Eclipse Public License |
| BSD | Berkeley Software Distribution |
| CC-BY-ND | Creative Commons Attribution NonDerivative |
| CC-BY-SA | Creative Commons Attribution ShareAlike |

For a more detailed list of licenses and their acronyms, take a look at the [SPDX License List](https://spdx.org/licenses/).

[+ Additional standard pages added by LF]

## Open Source License Compliance Management

### What to Keep in Mind

Before we proceed, let’s revisit the four main concepts that we learned in the [*Introduction to Open Source License Compliance Management (LFC193)*](https://training.linuxfoundation.org/training/introduction-to-open-source-license-compliance-management-lfc193/) course.

* Rights and licensing around software  
  This was the first topic that we touched upon. Here we learned about intellectual property rights in software, how copyright both restricts and grants rights to users of the software, and that users require a license to use it.
* Open source licenses  
  This section discussed how open source licenses work and that it is important to respect third-party open source license requirements to ensure your organization can use the relevant open source software.
* Open source compliance  
  We also learned how compliance with open source licenses works and how to comply with the obligations of licenses such that your organization avoids unnecessary mistakes in the adoption, use, and deployment of software made from or containing open source.
* Code building and distribution  
  Last but not least, we reviewed specific software development activities that occur when using open source (or any other) software. Understanding what activities are required to plan, prepare, and deploy processes allows for effective management of open source licensing. The way you use open source may impact which obligations you have to meet.

With the key concepts reiterated above, it should be clear that it is necessary to comply with the open source license obligations of any open source software used in your own software or products, and to distribute them in the desired format of packaging the code. To achieve best practice license compliance, it is necessary to review the software in multiple phases and procedures through those in different roles in the organization. We will discuss those review phases, procedures, and roles in more detail in the next chapters.

### Compliance Management Overview

Compliance management is the set of actions (activities) that manage open source code and components that are distributed by an organization. Those activities are used in products, services or contributions to open source projects to ensure that all obligations of the applicable license(s) are met. Open source components are called *supplied software* in the OpenChain specification. Organizations may have similar or complementary processes in place for proprietary, license fee-bearing components and software.

At a high level, incoming open source software is brought into an organization (e.g., by the project team downloading it themselves, software delivered from a supplier, etc.). Then, compliance activity is performed in such a way that any outgoing open source components in products or services meet all the necessary legal obligations.



Compliance management includes the following important steps:

* Identifying all open source componentsIn this step, we identify the open source components included in a software product. They can be anything, such as library packages or executables or even code snippets copied from open source projects. This step also includes identifying the sub-components, code snippets, etc., used, or planned to be used, in any software from a third-party vendor.
* Identifying all licenses and license obligations   
  In this step, we identify the respective applicable license(s) and their obligations for all the open source components present in the software product.
* Conforming to all obligationsIn this step, we confirm that all license obligations are met before the distribution of the software occurs (where “distribution” may include implementations of Software as a Service accessed via a network for some licenses).
* Delivering open source compliant software   
  This step involves making the software available with open source compliance-assured articles. Such articles would include packaging Software Bill of Materials and all other documentation as directed by the license obligations along with the software distribution media.

While organizations can differ widely and have vastly different resources available (such as legal support), the same activities still need to be performed. It is expected that existing business processes can be adapted to support an open source compliance program.

To ensure everything is properly managed, the supporting requirements are as follows:

* Adequate compliance staffing and clear lines of responsibility are defined.
* Training on the organization's open source policy is available to everyone, and anyone involved in delivering products and services, including open source, has to complete said training.
* All open source compliance activities are tracked and recorded.

Achieving end-to-end compliance of software products during open source review is the ultimate goal. In the next chapters, we will learn more about the open source review, the roles and responsibilities of those involved in the process, and how open source review fits into license compliance management.

.

# Chapter 2. Open Source Review

## Introduction

### Chapter Overview

Chapter 2 explains what an open source review is and who is involved in reviewing the use of code under open source licenses (note that this is often a multiple domain activity, requiring collaboration among people with legal, engineering, operations, and management knowledge and skills). This section also discusses the processes that are necessary to complete the open source review.

### Learning Objectives

By the end of this chapter, you should be able to:

* Explain what an open source review is.
* Understand how to gather and organize a successful open source review team.
* Analyze the results generated during the different phases of the review.
* Automate some of the review activities to eliminate redundant and repetitive tasks.

## Practical Activities Undertaken

### What Is Open Source Review?

After program and product management and engineers have reviewed the proposed open source components for usefulness and quality, a review of the rights and obligations associated with the use of the selected components should be initiated.

A key element to an open source compliance program is an open source review process. During this process, an organization identifies the open source software it uses, or plans to use, and determines the rights and obligations of the software license(s).

The open source review process includes the following steps:

* Identifying open source components included in the software product.
* Determining open source license obligations.
* Providing guidance compatible with company policy and business objectives to achieve compliance with license obligations for the open source components used.

### Open Source Review Team

An open source review team includes the company representatives that support, guide, coordinate and review the use of open source.



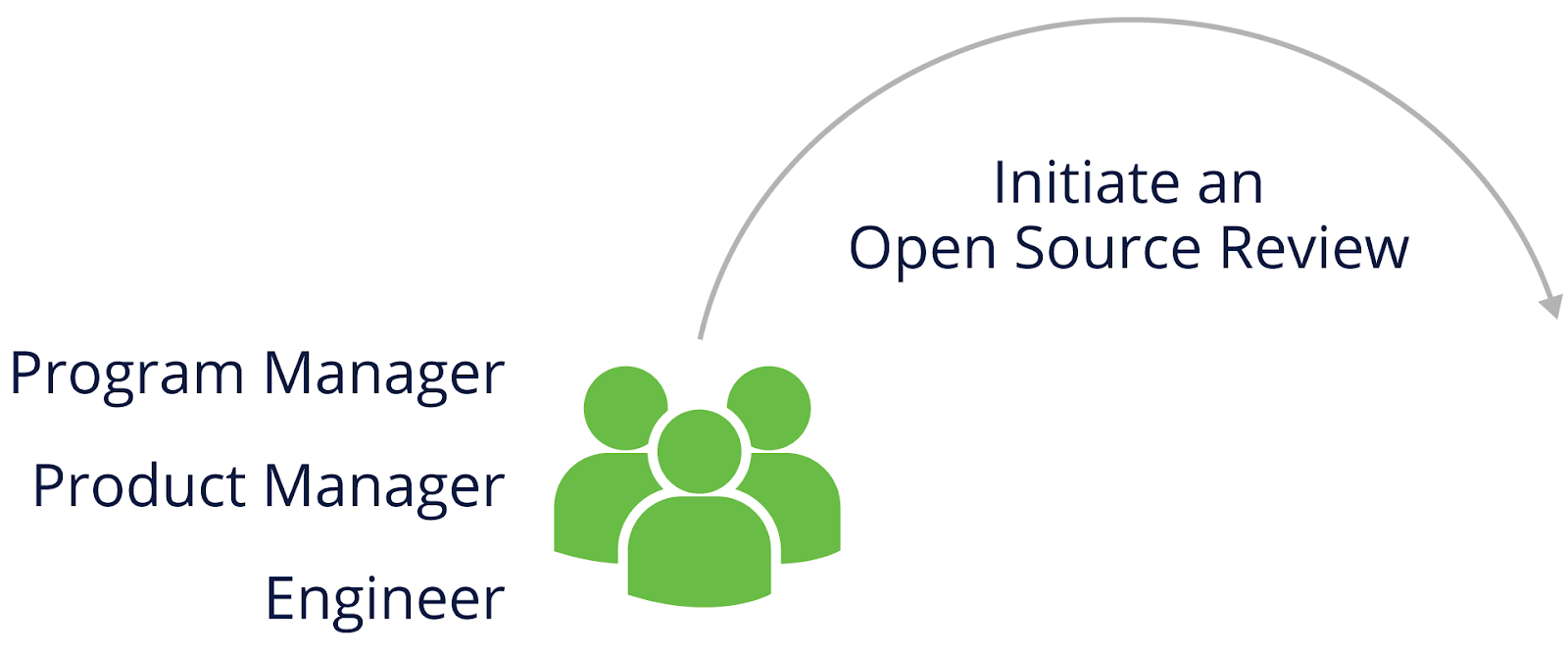
These representatives may include the following:

* Legal professionals to identify and evaluate license obligations and provide guidelines on complying with them.
* Source code scanning and tooling support to help identify and track open source usage (provide Software Bill of Materials).
* Engineering specialists working with business interests, commercial licensing, export compliance, etc., who may be impacted by open source usage.

The roles and procedures are discussed in more detail later in this chapter.

### Initiating an Open Source Review

Any individual contributor or team leader working with open source should be able to initiate an open source review. In smaller organizations or when working on small-scale projects, an individual contributor may play all the roles of a development team. On the other hand, mid-size or large companies usually require multiple people (e.g. program or product managers, engineers) to handle each activity.



Based on development practices, the review initiation phase varies. Some teams or individuals involve open source reviewers early on, in the design phase, while others postpone the review until the final build is completed.

***NOTE****: The process often starts when new open source software is selected by engineering or outside vendors while initiating new projects and/or improving existing ones.*

### Analyzing Proposed Open Source Usage

As previously mentioned, the open source review team consists of legal, open source code scanning and consultation specialists. Based on the company or project size, these roles may vary. In smaller organizations, a legal or consultation specialist can act alone as an individual contributor for all the activities, whereas in mid-size or large companies there may be multiple roles and people assigned to handle each activity individually.



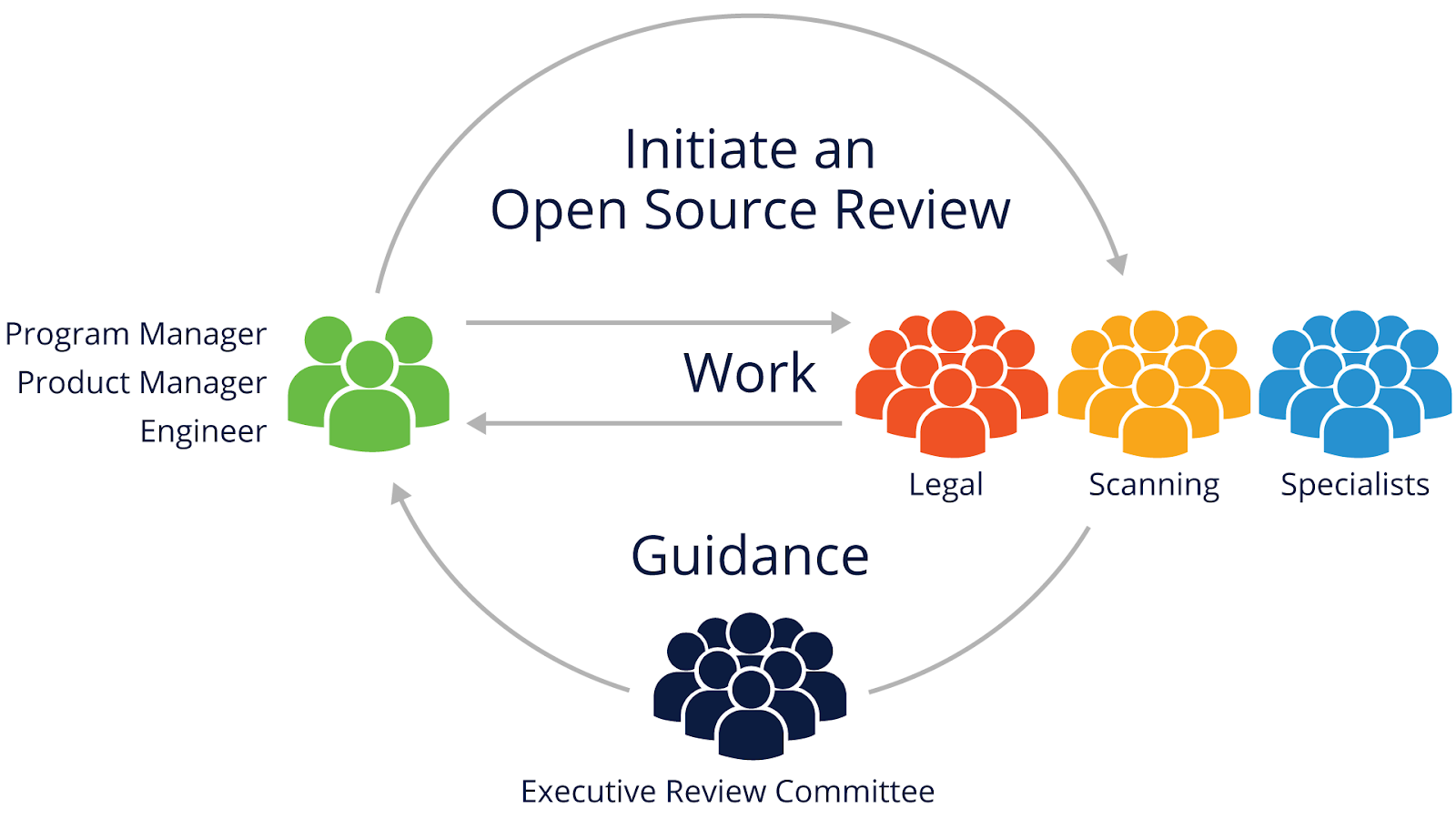
The open source review team should assess the information it has gathered (e.g., the Software Bill of Materials of the open source used in the project) before providing guidance for open source license compliance. This may include scanning the code to confirm the accuracy of the information.

Here are a few items that the open source review team should consider:

* Is the code and associated information complete, consistent, and accurate? For example, is the Software Bill of Materials accurate, are all the notices and attributions recorded as per the community standards or organization policy, etc.?
* Does the declared license match what is found in the code files, license, and notice files?
* Are the [licenses compatible](https://en.wikipedia.org/wiki/License_compatibility#Compatibility_of_FOSS_licenses) with each other for components planned to be used together?
* Are the licenses compatible with company policies (e.g., no linking of product X code to GPL licensed code)?

### Working through Open Source Review and Oversight

The image presented below visualizes this stage of the open source review process: the open source review team guides the development team (review initiators) on open source license compliance.



This open source review process may require executive oversight to resolve disagreements and approve the most important decisions.

The open source review process crosses disciplines, including engineering, business, and legal teams, and it should be interactive to ensure all these groups correctly understand the issues and can create clear, shared guidance.

### What Information Do You Need to Gather?

When analyzing open source usage, you should collect information about the identity of the open source component, its license(s), and how the open source component will be used in order to evaluate license compliance.

Common items to collect are the package name, the status of the community around the package, the version of the package used, the download or source code location online, the copyright owner, the license, attribution notices, a description of modifications made, a list of dependencies, the intended use in your product (for example, how it is linked and to what), the product release that will include the package, the location where the source code will be maintained, and the development team point of contact.

### Source Code Scanning Tools

Many different automated source code scanning tools can be used to streamline the open source code scanning activities.

However, the majority of tools do not offer all the features needed to identify all concerned points during open source scan. For example:

* Some tools will be specialized in capturing the licenses associated with the scanned code.
* Some tools will be better at capturing the library details of the code scanned.
* Some tools will be equipped better to identify snippets of code borrowed from other open source software packages, etc.

Since most of the solutions address only specific needs of the open source review team, solving all possible challenges all at once poses difficulties.

This is why companies select solutions/tools most suited to their specific market area and product, and many of them use both an automated tool and manual review to improve the results of analysis.

A list of freely available source code scanning tools can be found at <https://automatecompliance.org>.

## Complying with Open Source License Obligations

### Open Source Compliance: What You Need to Know

Open source license obligations vary. Licenses described as permissive licenses (discussed in more detail in the [*Introduction to Open Source License Compliance Management (LFC193)*](https://training.linuxfoundation.org/training/introduction-to-open-source-license-compliance-management-lfc193/) course) have notice or attribution obligations, whereas copyleft licenses (also discussed in LFC193) have additional obligations, including the sharing of source code.

maintain notices and provide complete attribution to the open source code used. Another basic step would be maintaining the original license text intact and bundling the same with the software deliverables before the distribution of the software.

Obligation fulfillment and distribution are discussed later in the course.

## Knowledge Check

* What is the purpose of an open source review?
  + To perform code quality review of the code from open source communities
  + To gather and analyze information regarding open source usage and to produce appropriate guidance --> Correct answer
  + To review whether using open source is a good idea for a project
  + To check that previous use of open source was a good idea
* What is the first action you should take if you want to use an open source component?
  + Choose the most popular open source community to download any open source package
  + Initiate an open source review process --> Correct answer
  + Incorporate the component into the product and distribute to customers to see if they notice any improvement
  + Name all of the open source contributors and the number of versions released
* Assessing the quality of information collected and used in an open source review:
  + Is a straightforward single-step process which requires checking what license is attached to the package as a whole (if it is found in a license or a README file then the condition is satisfied)
  + Consists of multiple steps, including checking information for completeness, consistency and accuracy --> Correct answer

# Chapter 3. End-to-End Compliance Management (Example Process)

## Introduction

### Chapter Overview

This chapter explains the processes used to manage open source compliance activities. It is designed to help small, medium, and large organizations make informed decisions regarding their compliance programs.

The processes complement, and may be impacted by, related organization policies. For example, a policy may be simply to entirely ban any Affero GPLv3 licensed code for use within any product. Another policy might be to always include all modified source code where required with the product distribution itself, such as on the distribution DVD (this removes the need of some licenses for an offer of source to be made if the source isn’t distributed). It is also common to have processes for contributions to open source projects, including creating new projects.

### Learning Objectives

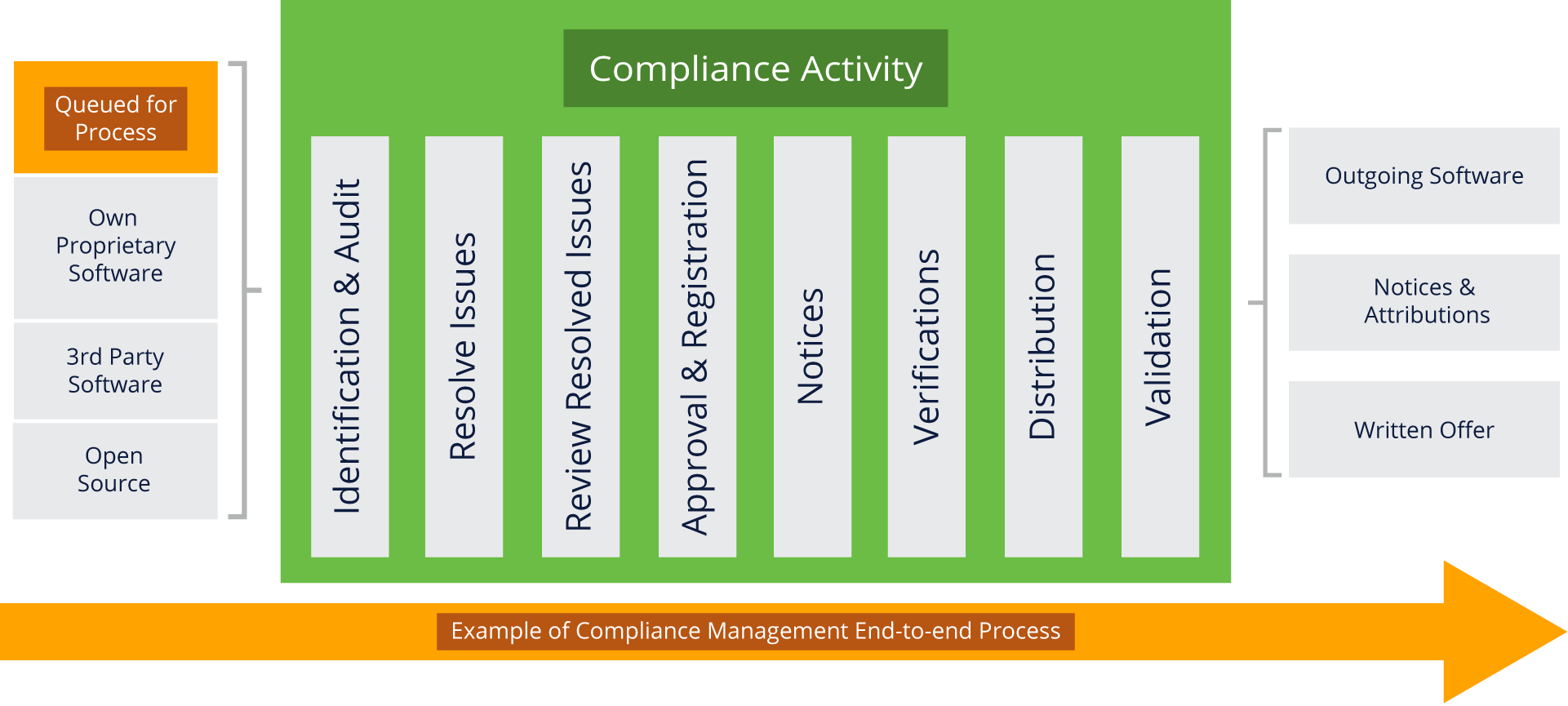
By the end of this chapter, you should be able to:

* Understand what steps are involved in end-to-end compliance based on the given example.
* List the bare minimum artifacts to be created and distributed along with the software package to meet obligations.

## Enterprise Process

### Compliance Management End-to-End Process

The image presented below shows the various steps involved in the end-to-end compliance process.



For small and medium-sized companies the end-to-end compliance checklist looks similar to this simulated example shown below.

Ongoing compliance tasks:

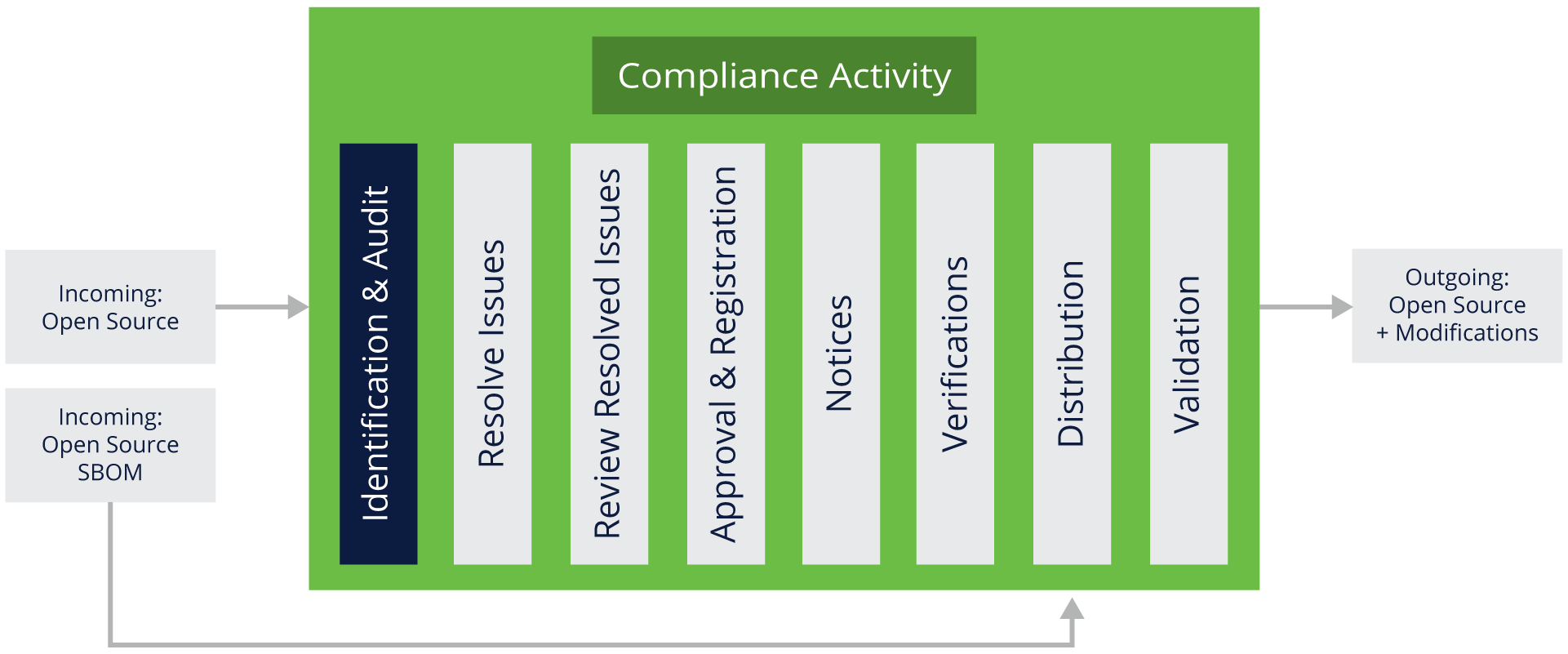
* Discover all open source components early in the procurement/development cycle.
* Review and approve all open source components used.
* Verify the information necessary to satisfy open source obligations.
* Review and approve any outbound contributions to open source projects.

Support requirements:

* Ensure adequate compliance staffing and designate clear lines of responsibility.
* Adapt existing business processes to support the open source compliance program.
* Have training on the organization’s open source policy available to everyone.
* Track progress of all open source compliance activities.

### Identification and Audit

In our example, the first step is to identify incoming open source components using established audit processes.



**Steps to identify/audit open source components:**

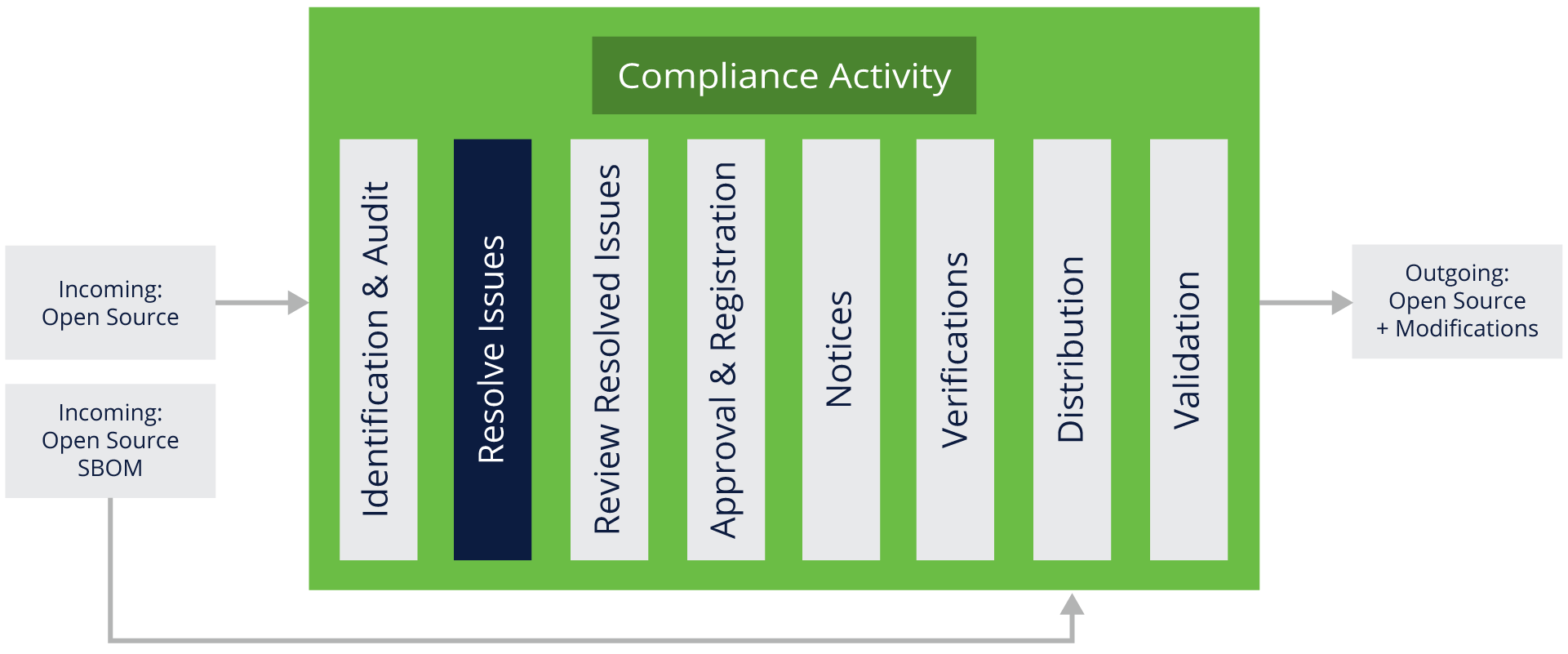
* The development team raises a request to perform identification/audit of the incoming open source components in the project.
* From the open source review team, the person trained in scanning activity will scan open source components using previously established processes. This helps to identify details of the open source components in question as well as their applicable license(s). The source may be scanned using a software scan tool.
* Scan results (understood as a list of complete or partial matches to open source shown by the scan tools, and identification of third-party party license texts) from the audit or scan are reviewed and verified to ensure the proper origin of the code (the applicable open source component used).
* Audits or scans are performed iteratively based on the software development and release lifecycles.
* An individual responsible for scanning should also perform the due diligence of third-party software and verify the list of open source components shipped within the software.
  + The above verification can be completed by referring to the Software Bill of Materials documentation (SBOMs) which should have been shared along with the third-party software or otherwise made available.
  + Confirm whether the incoming open source licenses are in compliance with the receiving organization’s open source policy.
* If open source components in the third-party software are not mentioned in the SBOM, then a manual identification of new components should be performed and added to the SBOM.

**Outcome:**

* An audit report identifying the origins and licenses of the open source code, as well as any issues that need to be addressed.
* A compliance record (created or updated) for the open source software.

### Resolving Issues

The next step is to resolve issues.



**Steps to resolve all issues identified in the audit:**

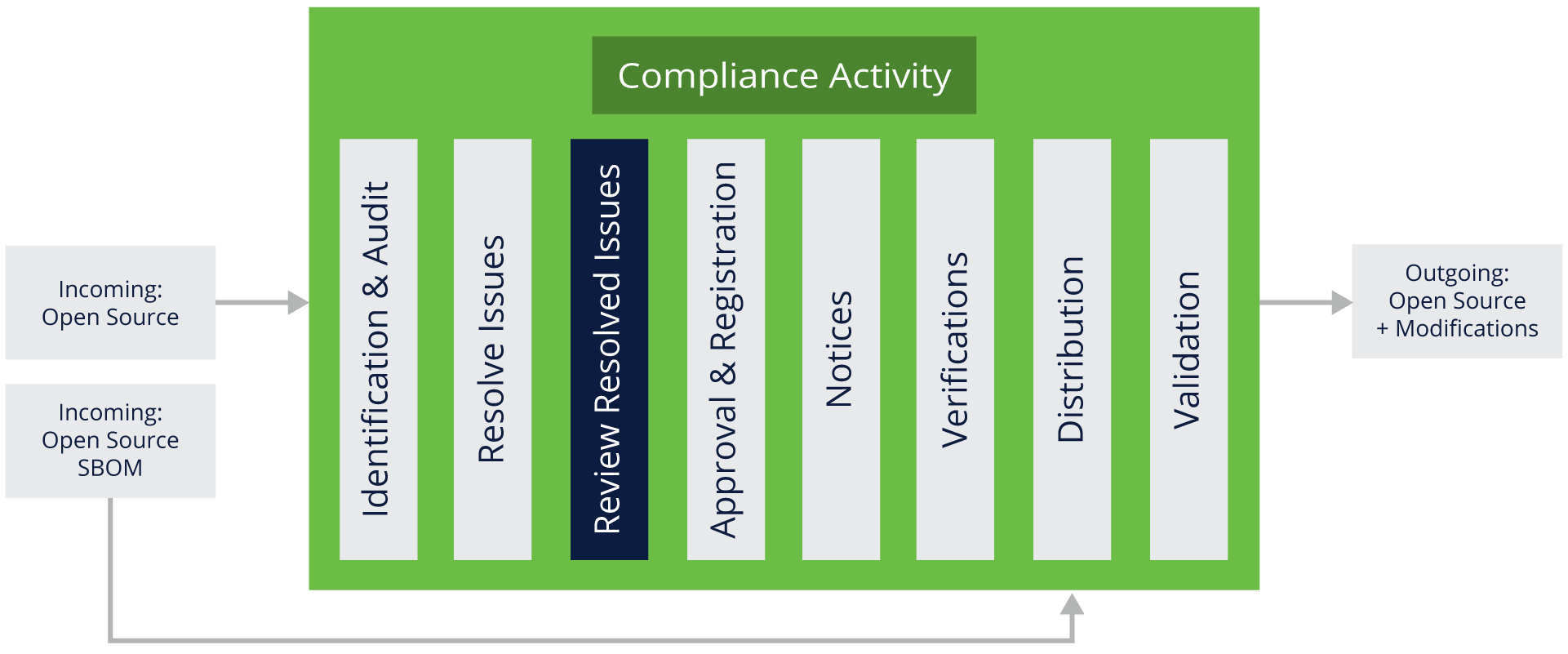
* Provide feedback to the appropriate engineers to resolve issues listed in the audit report that conflict with your open source policy.
* The appropriate engineers then conduct open source reviews on the relevant source code (as discussed earlier in the course, under the open source review topics).

**Outcome:**

* A resolution for each of the issues in the report.
* A resolution for any license conflicts.

### Performing Reviews

Next, it is recommended to review the resolved issues.



**Steps to review the resolved issues to confirm they match your open source policy:**

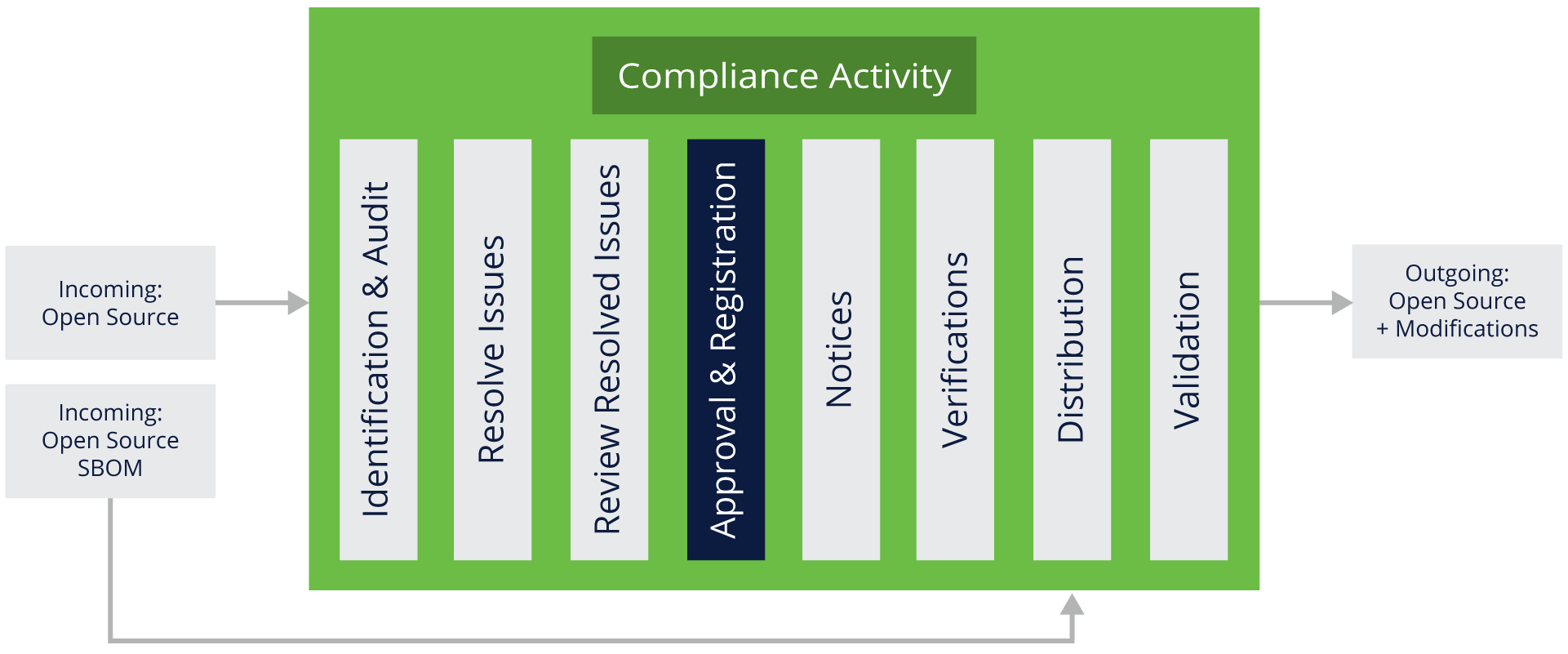
* Include appropriate authority levels in review staff.
* Conduct review with reference to your open source policy.

**Outcome:**

* Ensure the software in the audit report conforms with the organization’s open source policy.
* Preserve audit report findings and mark resolved issues as ready for the next step (i.e., Approval & Registration).

### Approval and Registration

The following step is called Approval and Registration.



**Steps for approvals and registration:**

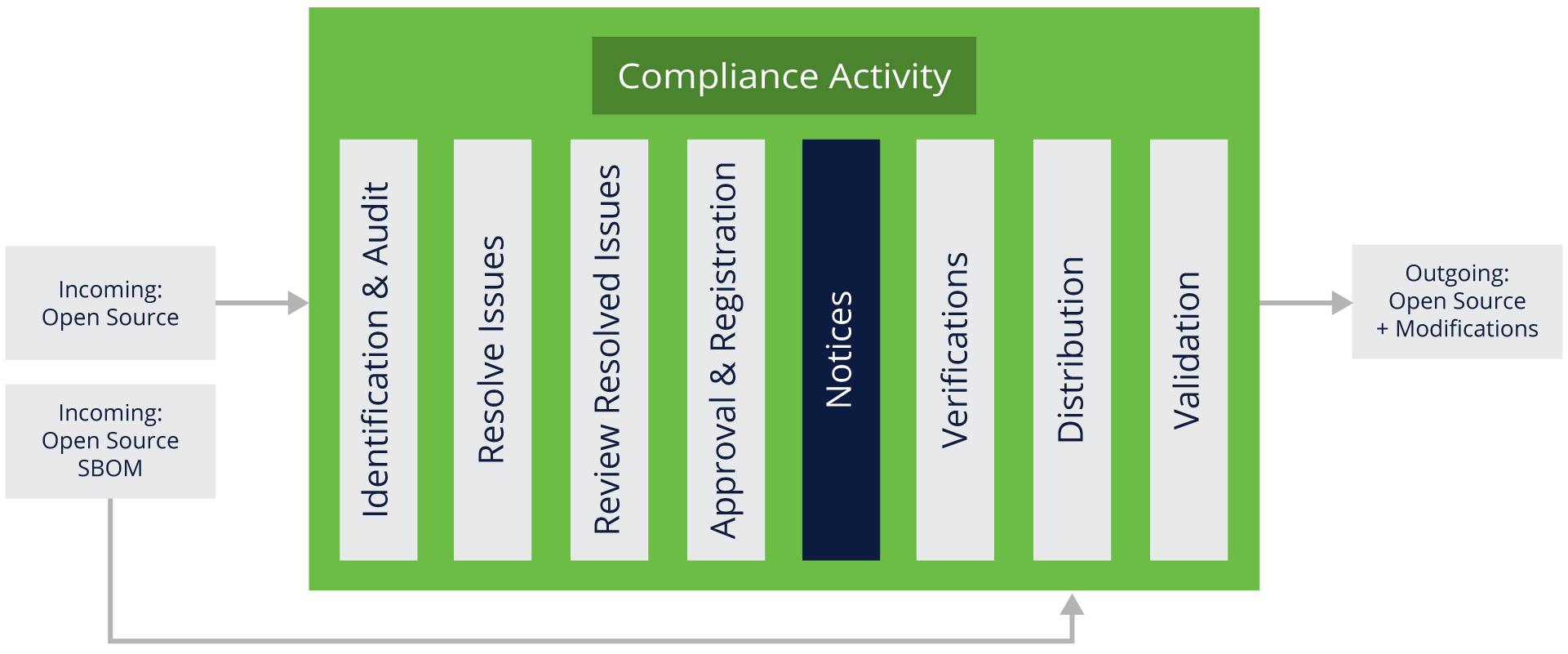
* Based on the results of the software audit and review conducted in the previous steps, the software may or may not be approved for use.
* The approval should specify versions of approved open source components, the approved usage model for the component, and any other applicable obligations under the open source license.
* Approvals should be made at appropriate authority levels. Usually, the open source review team is involved in the activities, and rarely executive decisions shall be obtained.

**Outcome:**

* Once an open source component has been approved for usage in a product, it should be added to the software inventory for that product.
* The approval and its conditions should be registered in a tracking system.
* The tracking system should make it clear that a new approval is needed for a new version of an open source component or if a new usage model is proposed.

### Notices

Notices refer to collecting and providing open source copyright and license information, a very common open source license requirement.



**Steps to prepare notices:**

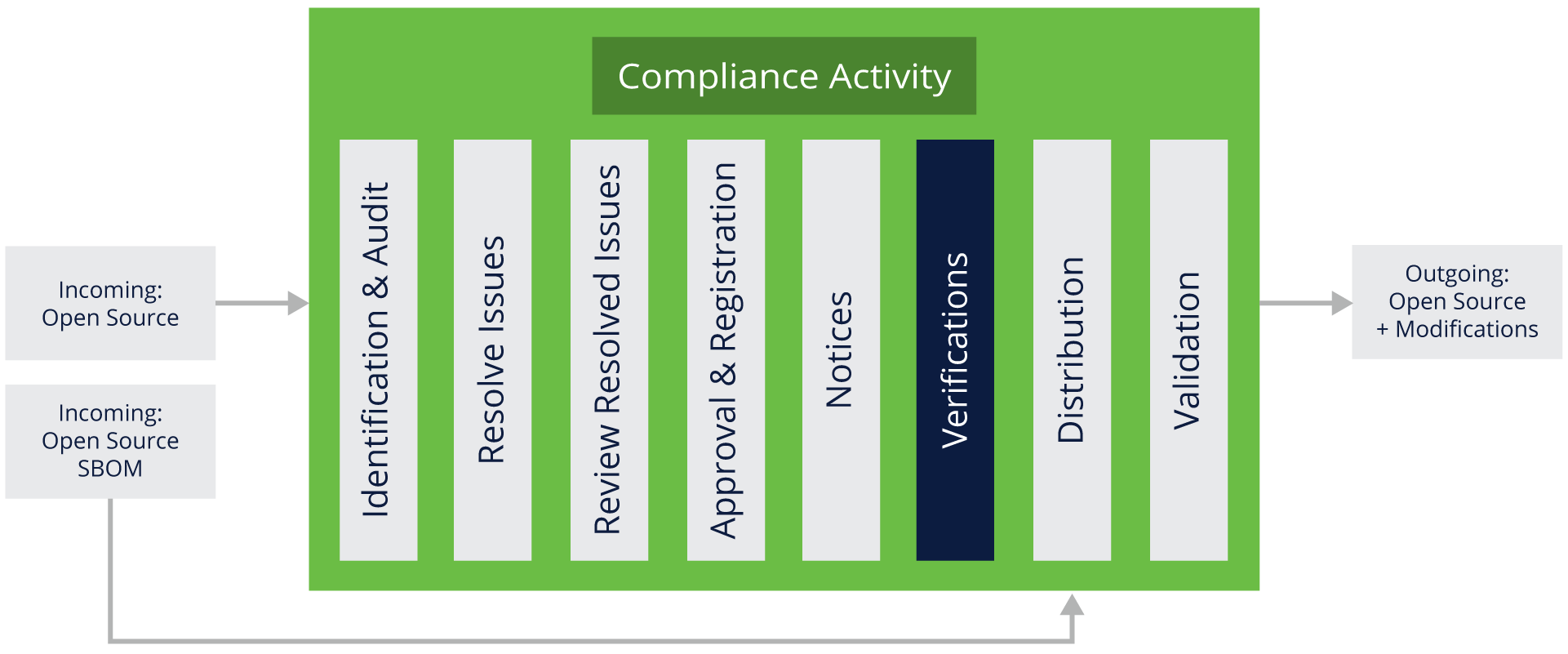
* Prepare appropriate notices for any open source component used in a product release:
  + Acknowledge the use of open source components by providing full copyright and license information.
  + Include information on how to request a copy of the open source code (when applicable, such as for GPL and LGPL requirements).
  + Reproduce the entire text of the license(s) for the open source code included in the product as needed.

**Outcome:**

* An open source notice as part of license obligation fulfillment.

### Verifications

The next compliance activity is verification.



**Steps to verify that the software to be distributed has been reviewed and approved:**

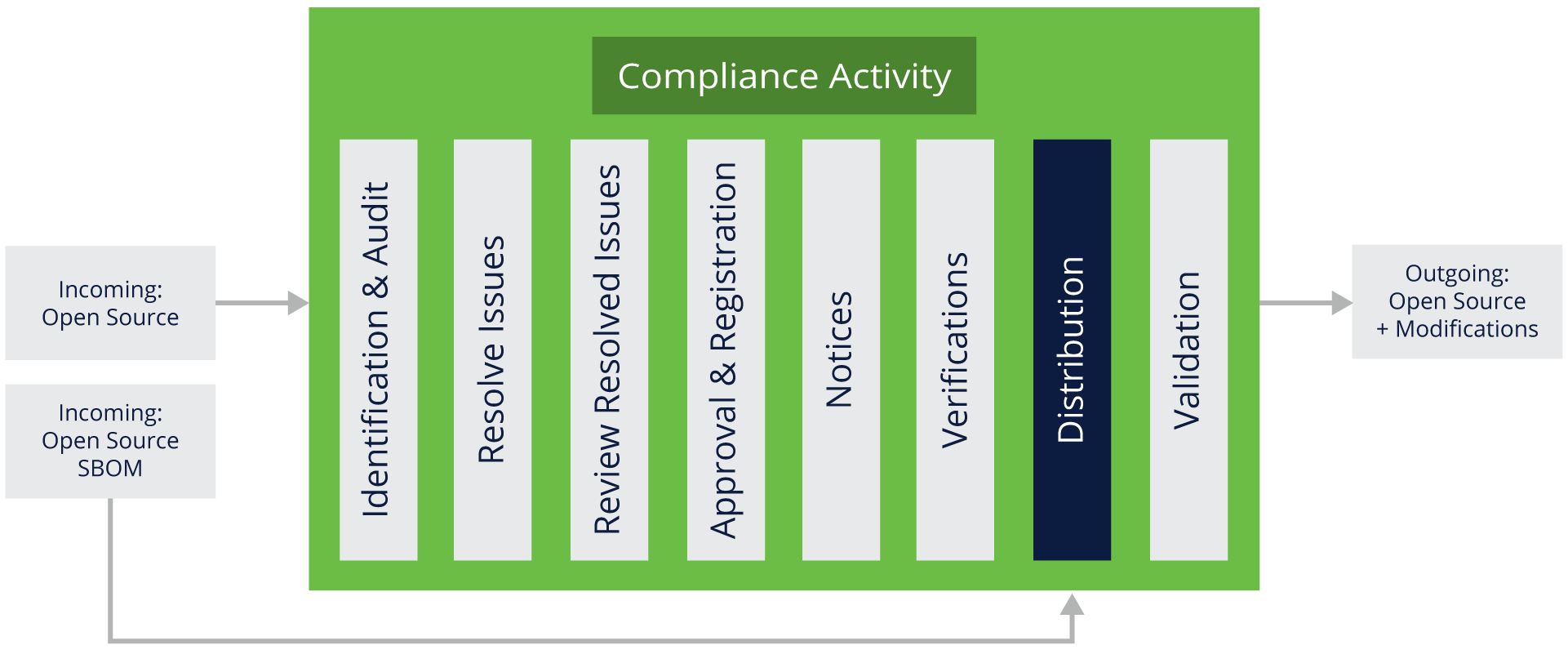
* Verify that open source packages finalized for distribution have been identified and approved.
* Verify that the reviewed source code matches the binary equivalents shipping in the product.
* Verify that instructions on how to request source code are included when required by the identified open source.
* Verify compliance with other identified obligations.

**Outcome:**

* The distribution package contains only software that has been reviewed and approved.
* Distributed *compliance artifacts* (as defined in the OpenChain specification), including appropriate notice files, are included in the distribution package or other delivery method. One example of a compliance artifact is an [SBOM](https://www.ntia.gov/SBOM). You can also learn more about SBOMs in the [*Introduction to Open Source License Compliance Management (LFC193)*](https://training.linuxfoundation.org/training/introduction-to-open-source-license-compliance-management-lfc193/) course.

### Distribution

Here, we will talk about fulfilling source code obligations, a requirement of some open source licenses.



**Steps to provide source code as required:**

* Provide source code along with any associated build tools and documentation, following your organization’s process.
* Source code is identified with labels which correspond to a specific product and version.

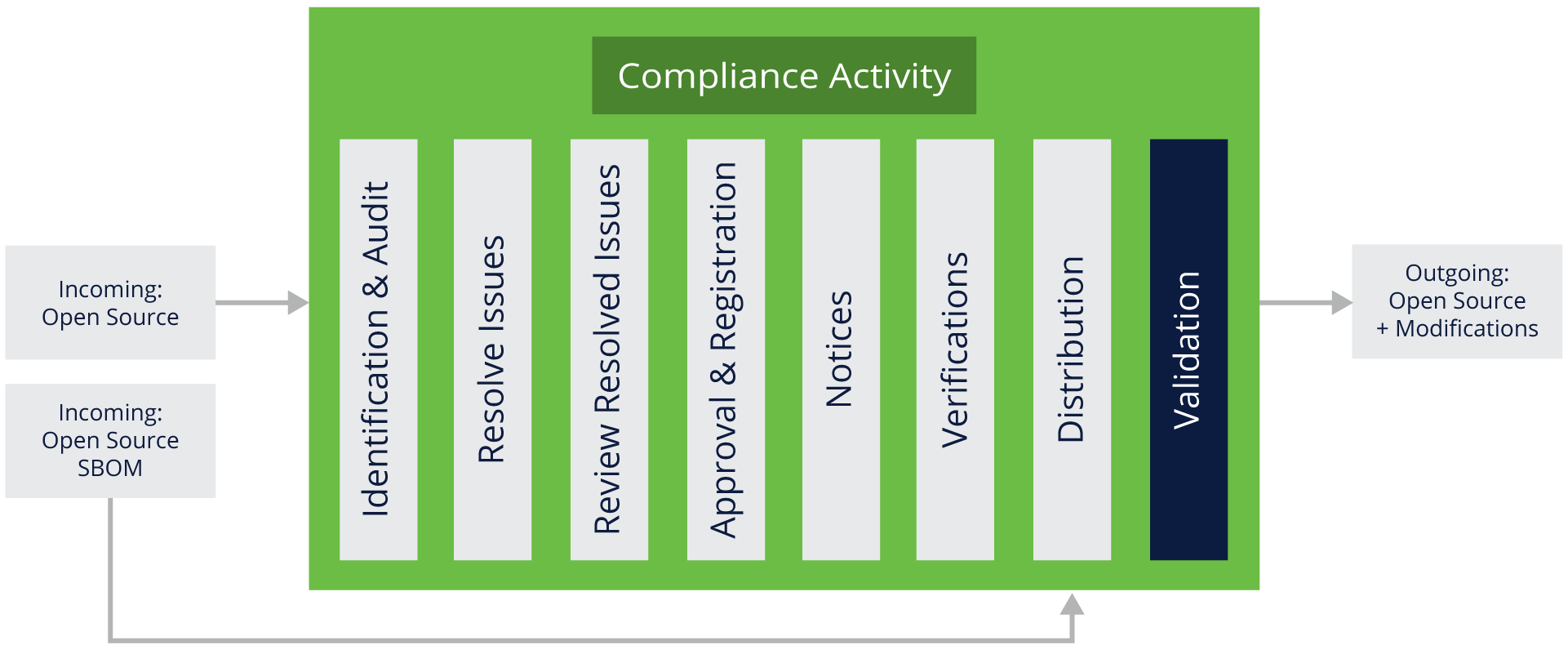
Along with the previously mentioned artifacts, we have responsibility to bundle the SBOMs shared by the third-party vendors along with our final distribution package.

**Outcome:**

* Obligations to provide source code are met.

### Validation

The last step in our example is the final validation.



**Steps to validate compliance with license obligations:**

* Verify that the required source code (if any) has been provided or distributed correctly.
* Verify that the source code corresponds to the same version that was included in a software release (or “supplied software”).

**Outcome:**

* Compliance artifacts are appropriately provided.

## Knowledge Check

* Which of the following steps are considered compliance activities?
  + Identification and Audit
  + Resolving Issues
  + Performing Reviews
  + Approval and Registration
  + Notices
  + Verifications
  + Distribution
  + Validation
  + All of the above --> Correct answer
* Is it necessary to bundle third-party vendor SBOMs with our finalized source code accompanying package?
  + Yes, all third-party vendor SBOMs should be bundled
  + Yes, but only SBOMs that are used in the final delivered package should be bundled --> Correct answer
  + No need to bundle third-party SBOMs
* Can we use in a product beta release an open source software package that was rejected at the Approval and Registration step of the open source review?
  + Yes (but start working on an update to replace it in the future)
  + No --> Correct answer

# Chapter 4. Avoiding Compliance Pitfalls

## Introduction

### Chapter Overview

This chapter describes some common pitfalls to look out for and avoid in the compliance process, such as intellectual property (IP) pitfalls, license compliance pitfalls, and compliance process pitfalls.

### Learning Objectives

By the end of this chapter, you should be able to:

* Understand the common issues encountered when dealing with open source license compliance in the context of bringing products or solutions to market.

## Intellectual Property Pitfalls

### Intellectual Property Pitfalls - Examples

We will begin with the big picture, discussing some general intellectual property pitfalls. This is by no means an exhaustive list but it gives an overview of the major issues that organizations can encounter.

#### Unplanned inclusion of copyleft open source into proprietary or third party code

This type of failure occurs during the development process when engineers add open source code into source code that is intended to be proprietary, thus creating a conflict with the open source policy.

It can be discovered by reviewing and auditing the source code for possible matches with open source code or copyright notices. For example, a search for copyright statements in the source code may reveal some unexpected contributor notice, leading to an open source component and thus requiring further investigation. Automated source code scanning tools may also be used for this purpose.

This type of failure can be avoided by offering training to engineering staff about compliance issues, the different types of open source licenses and the implications of including open source in proprietary source code, and conducting regular source code scans or audits for all the source code in the build environment.

#### Unplanned linking of copyleft open source and proprietary source code

This type of failure can be discovered using a dependency tracking tool that shows any linking between different software components.

It can be prevented by offering training to engineering staff on how to avoid linking software components with licenses that conflict with their open source policy and by continuously running the dependency tracking tool over the build environment.

#### Inclusion of proprietary code into copyleft open source through source code modifications

This type of failure can be discovered using audits or scans to identify and analyze the source code that you introduced to the open source component.

It can be avoided by offering training to engineering staff and conducting regular code audits.

### License Compliance Pitfalls

We will now discuss issues specific to license compliance. Like the previous section, this is not intended to be an exhaustive list. However, it should help you understand and avoid the most common license compliance-related challenges. Each of the points discussed below assumes discovery through the manual or automated review of the open source code included in the product or service that you are planning to release.

#### Failure to provide source code/appropriate license, attribution or notice information

This type of failure can be avoided by publishing a checklist in the product release cycle before the product becomes available in the marketplace. This checklist would examine if appropriate licenses are shared, if attribution or notice files are in place, and if links or copies of source code that needs to be shared as part of the license obligation are provided.

#### Providing the incorrect version of accompanying source code

This type of failure can be avoided by adding a verification step into the compliance process to ensure that the accompanying source code for the binary version is being published or otherwise made available. It is good practice, and some licenses require that the source code should be made available for several years beyond the last distribution of the software.

#### Failure to provide source code for open source component modifications

This type of failure can be avoided by adding a verification step into the compliance process to ensure that the source code for modifications is published (as opposed to only having the original source code for the open source component).

#### Failure to mark open source code modifications

This type of failure can be avoided by adding source code modification marking as a verification step before releasing the source code, as well as by offering training to engineering staff to ensure they update the copyright markings or license information of all open source or proprietary software that is going to be released.

### Compliance Process Pitfalls

We end this chapter with an overview of some of the main failure points within the compliance processes. As with the previous sections, this is not intended to be an exhaustive list, but it does cover some of the most important areas of concern.

#### Failure by developers to seek approval to use open source

This type of failure can be prevented by conducting periodic full scans for the software platform to detect any “undeclared” open source usage, offering training to engineering staff on the company's open source policy and processes, and including compliance in the employee performance reviews.

#### Failure to take the open source training

This type of failure can be avoided by ensuring that the completion of the open source training is part of the employees’ professional development plan and that it is monitored for completion as part of the performance review.

It can be prevented by mandating engineering staff to take the open source training by a specific date.

#### Failure to audit the source code

This type of failure can be avoided by conducting periodic source code scans/audits and ensuring that auditing is a milestone in the iterative development process.

It can be prevented by enforcing periodic audits and providing proper staffing to minimize the chance of running behind schedule.

#### Failure to resolve audit findings

This type of failure can be avoided by not allowing compliance tickets to be resolved or closed if the audit report is not finalized.

It can be prevented by implementing blocks in approvals in the open source compliance process. For example, a product cannot be shipped until acceptable compliance is achieved according to the organization’s policy.

#### Failure to seek review of open source in a timely manner

This type of failure can be avoided by initiating open source review requests regardless of whether the engineering team has confirmed the adoption of the open source code.

It can be prevented through education or, if the cause is lack of resources, by increasing resources (e.g., invest in more skilled people, more frequent review meetings, tools, etc.).

#### Failure to acknowledge what open source and other third-party components are included in the product and shipped

This can be avoided by using software composition analysis (SCA) tools and tracking all OSS and other third-party software that is used to build and run the product.

#### Failure not to provide a full Software Bill of Material (SBOM) at the time when product is shipped

This can be avoided by using software composition analysis (SCA) tools and tracking all OSS and other third-party software that is used to build and run the product.

## Additional Thoughts on Compliance Pitfalls

### Ensure Compliance Prior to Product Shipment

Companies must make compliance a priority before any product (in whatever form) ships. Prioritizing compliance promotes:

* More effective use of open source within your organization.
* Better relations with the open source community and open source organizations.

### Establish Community Relationships

An organization that uses open source in a commercial product should try to create and maintain a good relationship with the open source community, and, in particular, with specific communities related to the open source projects that are used and deployed in the company’s commercial products.

Good relationships with the software communities are helpful in enhancing two-way communication: upstreaming improvements and getting support from the software developers, as well as when seeking advice on the best way to become/stay compliant or resolve compliance issues.

## Knowledge Check

* What types of pitfalls are common in open source compliance?
  + Intellectual property pitfalls
  + License compliance pitfalls
  + Compliance process pitfalls
  + All of the above --> Correct answer
* An example of a license compliance failure is failing to mark an open source software after modification. True or False?
  + True --> Correct answer
  + False
* What are the benefits of prioritizing compliance? Select all answers that apply.
  + The final product will be functional and high quality → Incorrect Answer
  + Potential compliance mistakes are addressed early --> Correct answer
  + Open source is used more efficiently and in a compliant manner --> Correct answer
  + Meaningful relationships with the open source community are established --> Correct answer
  + There are no benefits, compliance generates additional cost in money and time → Incorrect Answer

# Chapter 5. Developer Guidelines

## Introduction

### Chapter Overview

Software developers (and integrators) are central to the use of open source within an organization and therefore key to a smooth-running compliance program. This chapter provides guidelines targeted directly to developers.

***NOTE****: OpenChain additionally offers a one-page set of guidelines applicable for all developers on* [*GitHub*](https://github.com/OpenChain-Project/Reference-Material/tree/master/Open-Source-Compliance-Support-Material/Recommended-Engineering-Practices/en)*.*

### Learning Objectives

By the end of this chapter, you should be able to:

* Understand developer guidelines with respect to open source projects.
* Understand compliance process requirements.
* Understand the compliance process that applies to open source components.

## Understanding the Development Challenge

### Developer Guidelines

When using third-party code, it is imperative to select from high quality, well-supported communities that meet your requirements regarding community structure, activities, and accountability.

Don’t hesitate to seek guidance and advice from the appropriate people inside your company, such as the Open Source Program Office (OSPO), the legal department, or the office of the CTO. Make sure you are clear on who the right people are and what is the right department to talk to*—*this should be clearly specified in the organization's policy documentation. To learn more about OSPO, enroll in the [Open Source Management & Strategy](https://training.linuxfoundation.org/training/open-source-management-and-strategy/) course series offered by the Linux Foundation.

Also, check the company’s policy regarding the approval process. For example, some companies may have extensive “approve” lists for third-party code. Others may require case-by-case approvals. Most will probably have a combination of different procedures depending on the task being worked on. The company open source policy should help clarify this for you.

It is important to either finish the above action items or to follow other company guidelines before checking new code into the internal company systems. Checking the company's policy regarding the approval process typically goes beyond personal use of the code for a specified project. However, it is still encouraged considering the potential negative implications of the code accidentally being used by other teams in the future without approval if there is a process breakdown.

Quite often a developer or development team will be in a situation where contributing code back to a third-party project makes sense (e.g., a bug fix, so that it isn’t necessary to keep making the same fix whenever a new version of the open source is released and used). In these situations, you should also check your company’s open source policy (or relevant documentation) to make sure that your intended contribution fits into the company’s approach and that of the community in question.

Regardless of whether you are taking code into your company or sending code outside of your company, it is important to be mindful and careful of license information. It should be provided in all cases to ensure clarity. Be careful not to remove or inadvertently change existing open source license information or license texts in third-party code. This applies whether it is an inbound or outbound contribution. The clarity in the licensing is critical to ensuring that things run smoothly.

Wherever possible, do not rename open source components unless there is a specific requirement to do so for your particular use case or because of a requirement in a license. The reason for this is simple: if and when you rename the code, you increase the risk of misinterpretation around the license and the chance of someone in the future failing to understand the provenance of this code.

Adjacent to this, you should also gather and retain attribution information according to the licenses about the third-party open source projects you are using. This will allow you and your teams to quickly return to the projects and engage with them to receive the latest code and the latest available community support.

### Anticipate Compliance Process Requirements

The overhead of using third-party code in a compliant way is not high, but it does exist. It is useful to ensure that you budget time and resources for addressing your company’s open source policy process requirements in the development work plans. This type of activity is part of the (small) cost of having access to open source or proprietary code from third-parties.

Issues you may want to spend time on include considering the impact of linking or incorporating third-party code into your code or other third-party code. Understanding license implications early on can save you a lot of time later. This can be part of your architectural planning cycle.

You will also want to ensure that the third-party code approvals previously obtained via your company’s policy are still valid with this new project. Sometimes an approval is dependent on a specific product or service. Checking this now will save time later.

The same applies to every upgrade to the packages being used. It is important for security reasons to be current, but it must be a decision and approval explicitly included in processes to avoid unforeseen complications later. These can range from unexpected behavior to changes in the licensing of the code.

### Compliance Process Applies to All Open Source Components

Your final validation is an important step aiming to prevent releasing code with compliance issues that will be more difficult and/or costly to fix later. You always have the obligation to ensure that you are in compliance with all OSS component licenses that you have obtained from your suppliers and that you include in your product what your suppliers are providing. You also need to be aware of any obligations that transfer onto the receiver as the third-party code arrives.

Having some form of inbound code audit is extremely useful to get this done. This burden does not have to rest on your team. For example, some companies and some project teams have a policy that suppliers must include a source code audit report to make your decision-making processes quicker.

## Knowledge Check

* Which of the following are some general guidelines that developers can practice when working with open source? Select all answers that apply.
  + Select code from high quality open source communities --> Correct answer
  + Seek guidance --> Correct answer
  + Ignore all guidelines, and just contribute good code → Incorrect Answer
  + Preserve existing licensing information --> Correct answer
  + Gather and retain open source project information for your review process ---> Correct answer
* Should you remove or alter open source license header information?
  + Yes, as it is now part of your software
  + Yes, that’s the best way to keep the source file short and tidy
  + No, existing license information should be preserved and additional header information can be added for modifications or additions to source code --> Correct answer
* Can a new version of a previously-reviewed open source component create new compliance issues? Select all answers that apply.
  + Yes, if there is a change in the open source license for the new version of the open source component --> Correct answer
  + Yes, if new dependencies are introduced with new versions which creates additional open source obligations --> Correct answer
  + No, authors are not allowed to change licenses or copyright notices for future versions of existing open source components. → Incorrect answer
  + None of the above → Incorrect answer

# Chapter 6. Bringing Things Together

## Introduction

### Chapter Overview

This chapter summarizes what was presented in the previous sections and discusses next steps in your open source license compliance management journey.

### Learning Objectives

By the end of this chapter, you should be able to:

* List key takeaways and challenges as related to open source license compliance management.
* Use additional resources to further deepen your understanding of open source license compliance management.

## Key Takeaways and Challenges

### Takeaways

The key takeaways are as follows:

* We first discussed what open source license compliance is and what is needed to achieve open source license compliance. Then we talked about the outcomes of compliance software including identifying all open source components used in the software and licenses applicable to them, meeting the license obligations, and delivering the software package bundled with the compliant artifacts.
* We also learned about the roles and responsibilities involved in the Open Source Review processes and took a brief walkthrough of how such a team would be placed in a medium-to-large-scale company. Then, we talked about how individual contributors often wear many hats in smaller companies.
* We talked about eight compliance activities based on a hypothetical example of an end-to-end compliance process for a medium-scale organization. We studied the steps involved in each activity and the possible outcomes of those steps.
* We then focused on potential compliance pitfalls and discussed different ways to address them and minimize their risks.
* Last but not least, we introduced some developer guidelines with respect to the open source projects, and how to adhere to compliance requirements of the open source projects and components we are working with.

### Challenges

Challenges we may have to be aware of include the following:

* Knowing how to identify open source components when they are not visible to the naked eye. For example, when OSS license components are not listed on the package except for copyrights, author name(s) and/or inheritance notices.
* Understanding how to accommodate and identify the roles and responsibilities of open source review teams, and how such teams work in small-scale companies as opposed to midsize and large-scale organizations.
* Understanding that there is no universally applicable set of guidelines available for identifying and addressing compliance pitfalls. This is because various regional and company legal interpretations for compliance and risks exist that need to be taken into consideration.

But there’s also good news! Some of these challenges are being handled by existing communities. For example:

* [OpenChain](https://www.openchainproject.org/) has released and is further planning to release Playbooks which should help you understand and work through common challenges seen in the area of open source compliance.
* [TODO Group](https://todogroup.org/) also has been busy at creating and sharing various OSPO-related materials that can help you better understand OSPO requirements and identify the skill sets necessary to address some of the challenges we’ve discussed in this course and many more.

## What’s Next

### Additional Resources

Here is what you can do next:

* Join the [OpenChain channel(s) of your interest](https://lists.openchainproject.org/g/main).
* Get familiar with the [OpenChain reference pages and GitHub](https://github.com/OpenChain-Project/Reference-Material).
* Enroll in courses that talk about [open source best practices](https://training.linuxfoundation.org/full-catalog/?_sft_topic_area=open-source-best-practice).
* Read the [*“Minimum Elements For a Software Bill of Materials (SBOM)”*](https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom)article by National Telecommunications and Information Administration.

## Knowledge Check

* Which community provides freely available resources to help with open source compliance?
  + OpenChain -> Correct answer
  + OpenDaylight
  + OssChain
  + OpenView
* What is the minimum number of people required to staff a compliance program?
  + One person is sufficient to complete all tasks
  + Five people are needed: an engineer, a product manager, a program manager, a lawyer, and an executive manager
  + 10% of the number of developers
  + Depends on the organization’s size and the number of product(s) --> Correct answer
* There is one universal training available to identify compliance pitfalls. True or False?
  + True
  + False --> Correct answer

## Course Completion

### Final Exam

* What additional information is important when reviewing an open source component from a third-party?
  + That they offer a commercial version of the software
  + That they are being paid to make the software available
  + That they provide relevant copyright, attribution notices and source code modifications details --> Correct answer
  + No additional information is necessary
* Which of the following is an example of an intellectual property failure:
  + Not being able to find an appropriate open source code, even though it must exist somewhere on the Internet
  + Mixing proprietary code and open source code, which may result in making proprietary software available to the general public despite the company's policy --> Correct answer
  + Not being able to write the software
* Name some important steps in a compliance process. Select all answers that apply.
  + Follow developer guidelines, especially for any open source code included in or linked to proprietary code --> Correct answer
  + Review and approve all open source early in the cycle --> Correct answer
  + Review architecture and avoid mixing components governed by incompatible licenses --> Correct answer
  + Verify OSS compliance for every product and every version prior to release --> Correct answer
  + Review OSS compliance for new versions of OSS --> Correct answer
* What information helps identify who is licensing the software?
  + Footer with details indicating where the software was downloaded from
  + Copyright notices, attribution, and source code --> Correct answer
  + Direct information about the licensor will not be available
* What risks should you address with software supplied to you that you use and incorporate into your products and solutions?
  + License compliance for any disclosed open source embedded in the supplied software
  + The potential for creating license conflicts by integrating said software with other open source or proprietary software
  + Undisclosed or unknown open source included in the software
  + All of the above --> Correct answer
* What is an example of a compliance process failure:
  + Not taking minutes for the open source review meeting
  + Auditors waiving all the red-flagged items in a compliance report to allow the software to be released → Correct Answer
  + Software not passing all of the test cases
  + Allowing software to be released even though auditors deemed one of the key features was not properly tested
* What are the benefits of maintaining a good community relationship?
  + You can enjoy a beer together on the project budget
  + You can better assess how to comply with the open source license requirements, and have a better two-way communication with regard to contribution and use of the open source --> Correct answer
  + You can step up your marketing efforts for the organization/product
* What should you do if you have a question about using open source?
  + Nothing, focus on delivering the code because only the project lead needs to know more about open source
  + Initiate an open source review process or contact the open source review team --> Correct answer
  + Directly contact the contributors of the open source community
* What kind of information should you collect for an open source review?
  + Test results showing that the code works as expected
  + The count of lines of codes in the software package and year of release of the software package
  + The package name, version, download URL, license, description and intended use in your product --> Correct answer
  + No additional information is required
* In which stage of an end-to-end compliance do we bundle SBOMs shared by the third-party vendor along with our final distribution package of software?
  + Distribution stage --> Correct answer
  + Final validation stage
  + At the stage of preparing notices and attribution
  + We do not need to bundle third-party open source SBOM with our final distribution package