Implementing Open Source License Compliance Management (LFC194)

Course Introduction / Welcome

[LF Boilerplate]

CHAPTER 6: Running an Open Source Review

# Introduction

## Chapter Overview

This chapter explains the types of people who can assist with reviewing the use of code under open source licenses. This is often a multiple domain activity and it is important to understand who can be involved and how.

## Learning Objectives

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

* Understand how to staff an open source review team
* Understand how to analyze results during different phases of the review
* Understand how to automate some of the review activities

# Practical Activities Undertaken

## Open Source Review

•After Program and Product Management and Engineers have reviewed 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. This process is where a company can analyze the Open Source software it uses, or plans to use, and understand its rights and obligations

•The Open Source Review process includes the following steps:

•Analyze and understand license obligations

•Provide 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:

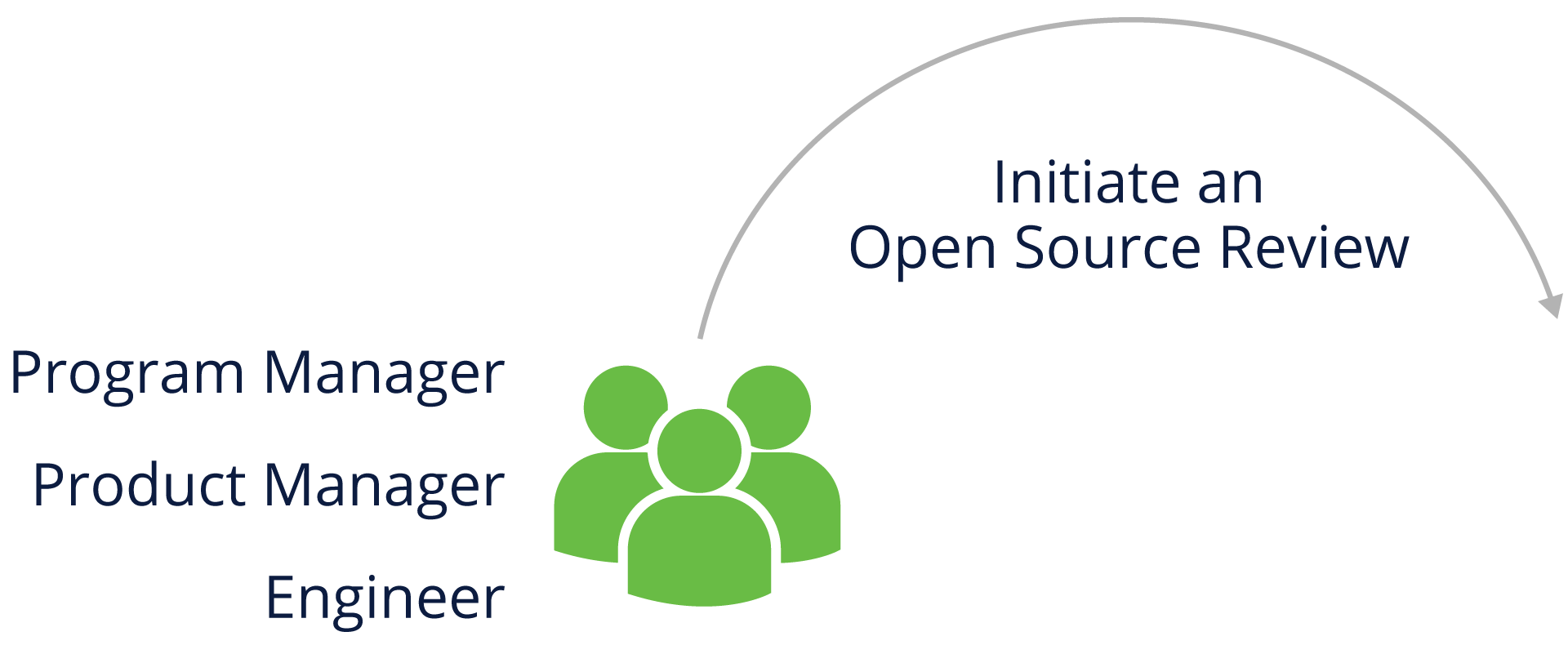
•Legal professionals to identify and evaluate license obligations and guide on complying to the terms

•Source code scanning and tooling support to help identify and track Open Source usage (Provide 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 further discussed below.

## Initiating an Open Source Review



Any individual contributor or leads of a team working with Open Source packages should be able to initiate an Open Source Review, that includes Program or Product Managers, Engineers, and Legal colleagues.

Based on type of development practices, the review initiation phase varies. Some teams or individuals involve open source package reviewers from the design phase. Some teams/individuals initiate the review post final build.

Note: The process often starts when new Open Source-based software is selected by engineering or outside vendors while initiating a new project and enhancing the existing projects.

## Analyzing Proposed Open Source Usage



The Open Source Review team should assess the information it has gathered, i.e., the bill of materials of the packages used in the project, before providing guidance for open source packages license compliance. This may include scanning the code to confirm the accuracy of the information.

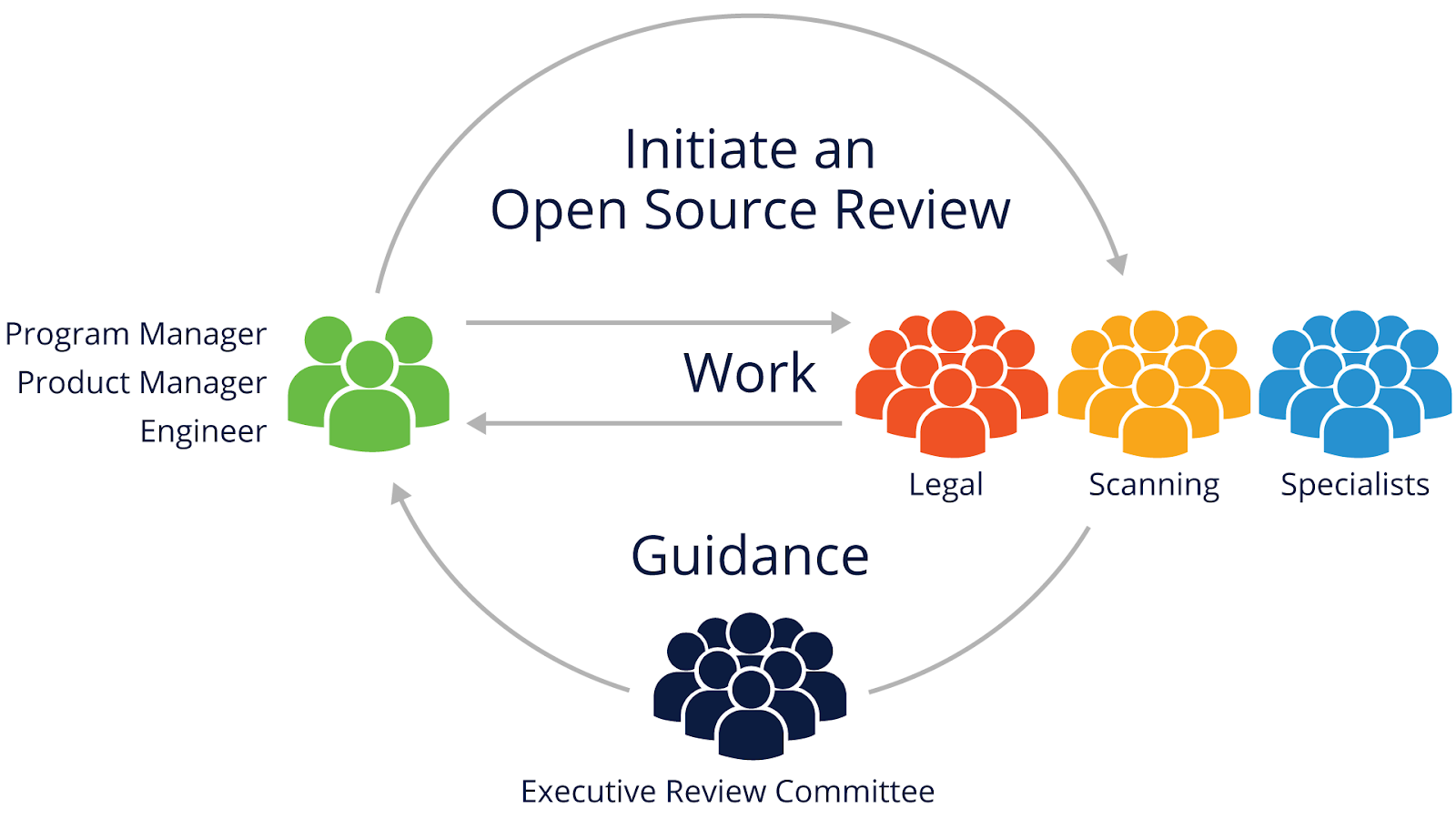
The Open Source Review team should consider:

•Is the code and associated information complete, consistent and accurate?

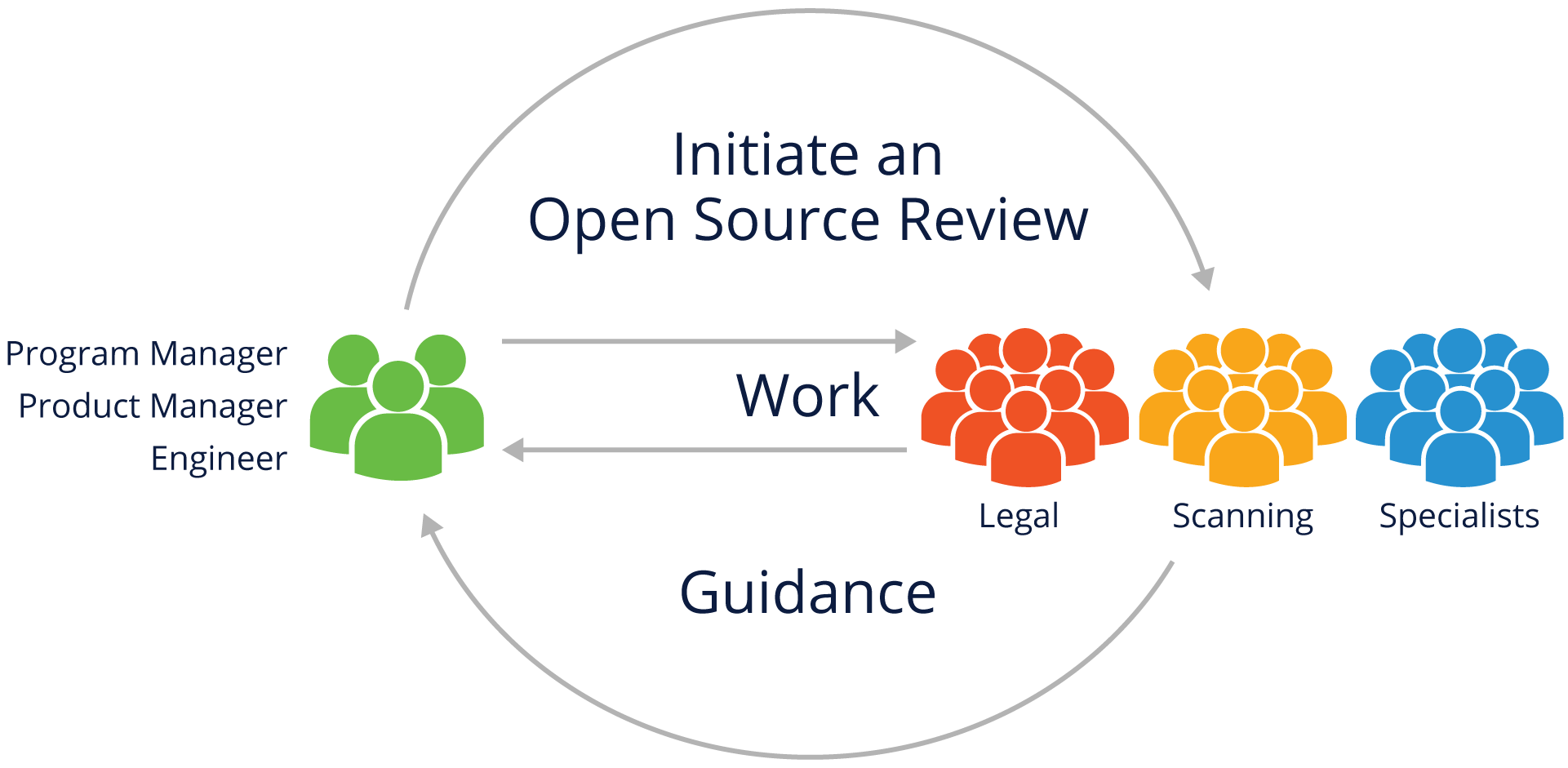
•Does the declared license match what is in the code files?

•Does the license permit use with other components of the software?

## Open Source Review Oversight

The Open Source Review process should have executive oversight to resolve disagreements and approve the most important decisions.

## Working through the Open Source Review

The Open Source Review process crosses disciplines, including engineering, business, and legal teams. It should be interactive to ensure all those groups correctly understand the issues and can create clear, shared guidance.

## What information do you need to gather?

When analyzing Open Source usage, collect information about the identity of the Open Source component, its origin, and how the Open Source component will be used. This may include:

| ●Package name  ●Status of the community around the package (activity, diverse membership, responsiveness)  ●Version  ●Download or source code URL  ●Copyright owner  ●License and License URL  ●Attribution and other notices and URLs  ●Description of modifications intended to be made | ●List of dependencies  ●Intended use in your product  ●First product release that will include the package  ●Location where the source code will be maintained  ●Possible previous approvals in another context  ●If from an external vendor:  ●Development team's point of contact  ●Copyright notices, attribution, source code for vendor modifications if needed to satisfy license obligations |
| --- | --- |

## Source Code Scanning Tools

•There are many different automated source code scanning tools.

•All of the solutions address specific needs and - for that reason - none will solve all possible challenges

•Companies pick the solution most suited to their specific market area and product

•Many companies use both an automated tool and manual review

•A good example of freely available source code scanning tools can be found here: https://automatecompliance.org

# Knowledge Check

## Knowledge Check 5.1

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

CHAPTER 7: End to End Compliance Management (Example Processes)

# Introduction

## Chapter Overview

This chapter explains the processes used to manage open source compliance activities. It is designed to help parties from small, medium and large companies make the appropriate choices in their programs.

## Learning Objectives

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

# Subsection Title

## Compliance Management Overview

•Compliance management is a set of actions that manages Open Source components used in products. Companies may have similar processes in place for proprietary components. Open Source components are called "Supplied Software" in the OpenChain specification.

•Such actions often include:

•Identifying all the Open Source components used in Supplied Software

•Identifying and tracking all obligations created by those components

•Confirming that all obligations have been or will be met

•Small companies may use a simple checklist and enterprises a detailed process.



## Small to Medium Company Checklist (Example)

Ongoing Compliance Tasks:

1.Discover all Open Source early in the procurement/development cycle

2.Review and Approve all Open Source components used

3.Verify the information necessary to satisfy Open Source obligations

4.Review and approve any outbound contributions to Open Source projects

Support Requirements:

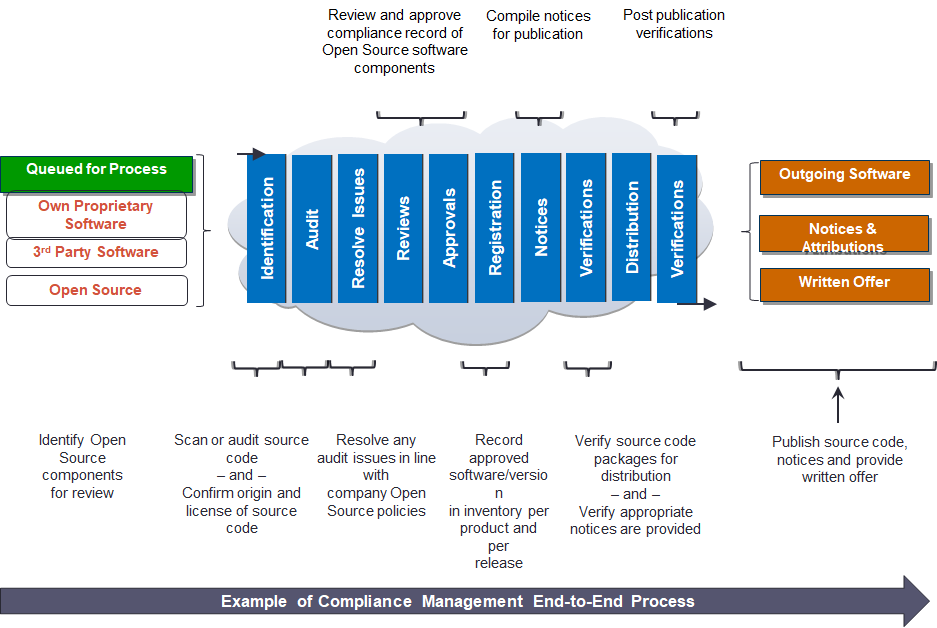
1.Ensure adequate compliance staffing and designate clear lines of responsibility

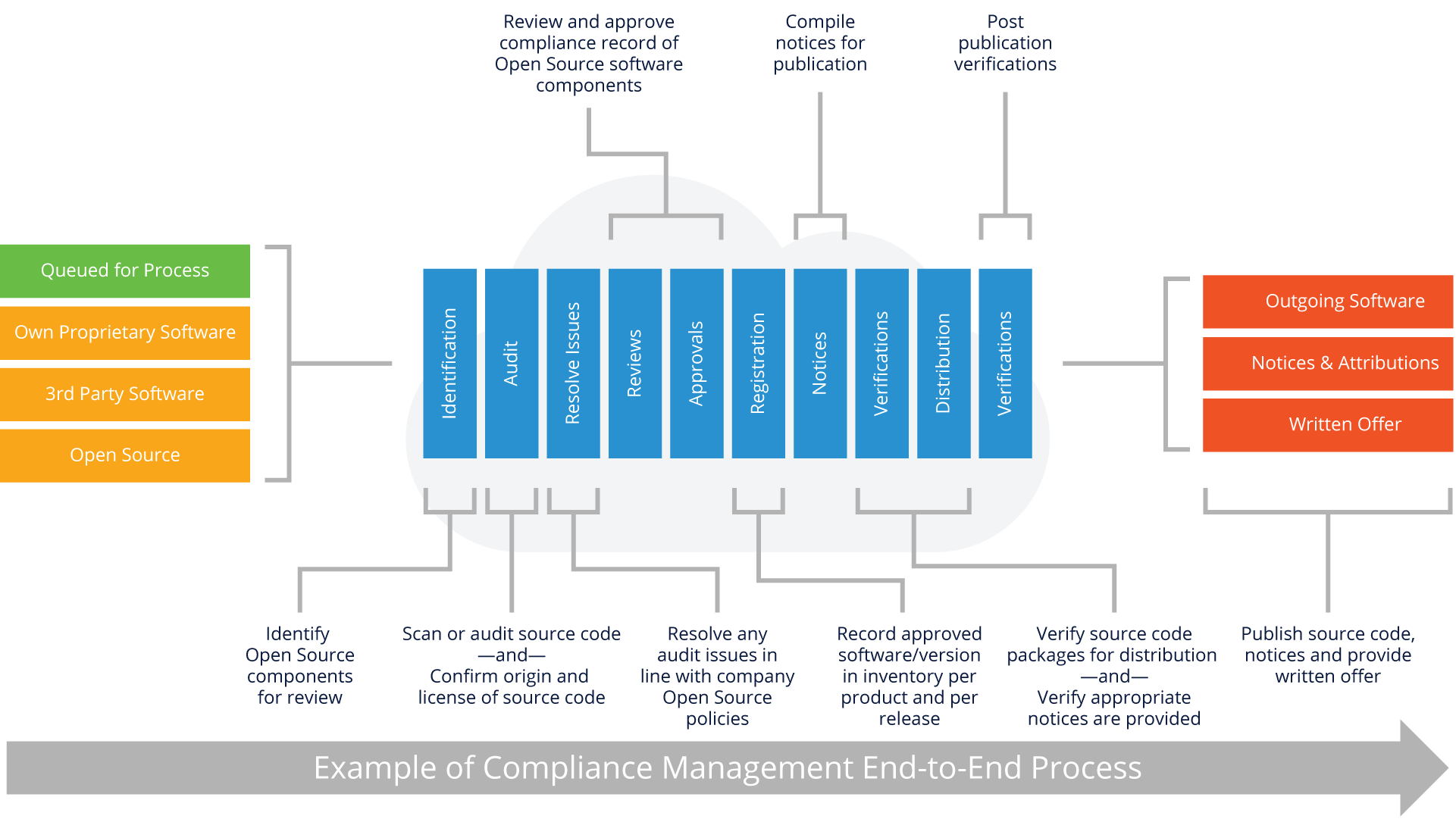
2.Adapt existing Business Processes to support the Open Source compliance program

3.Have training on the organization’s Open Source policy available to everyone

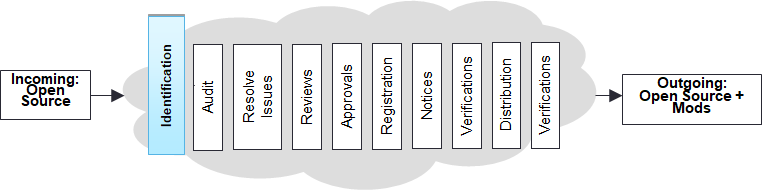
4.Track progress of all Open Source compliance activities

## Enterprise Process (Example)





## Identify and Track Open Source Usage



•Steps to Identify Open Source components:

•Incoming requests from engineering

•Scans of the software

•Due diligence of 3rd-party software

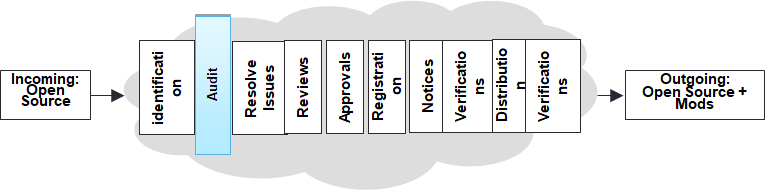
•Manual recognition of new components added to the repository

•Outcome:

•A compliance record is created (or updated) for the Open Source

•An audit is requested to review the source code with a scope a defined as exhaustive or limited according to Open Source policy requirements.

## Auditing Source Code



•Steps to Identify and audit Open Source licenses:

•Source code for the audit is identified

•Source may be scanned by a software tool

•“Hits” from the audit or scan are reviewed and verified as to the proper origin of the code

•Audits or scans are performed iteratively based on the software development and release lifecycles

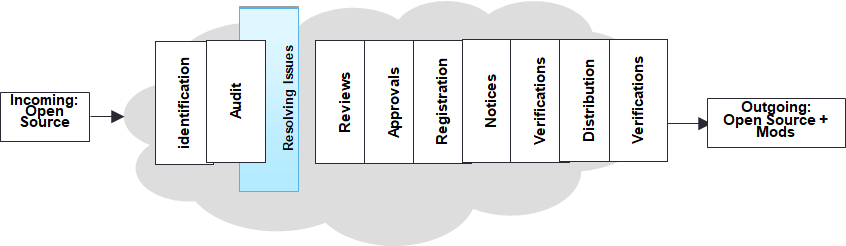
•Outcome:

•An audit report identifying:

1.The origins and licenses of the source code

2.Issues that need resolving

## Resolving Issues



•Steps to Resolve all issues identified in the audit:

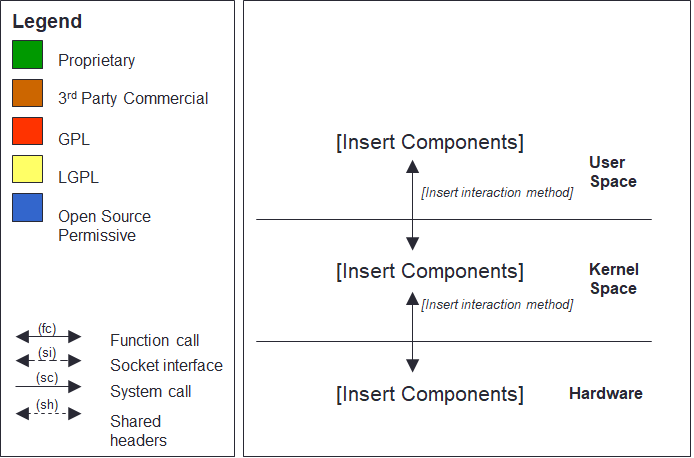
•Provide feedback to the appropriate engineers to resolve issues in the audit report that conflict with your Open Source policy

•The appropriate engineers then conduct Open Source Reviews on the relevant source code (see next page for template)

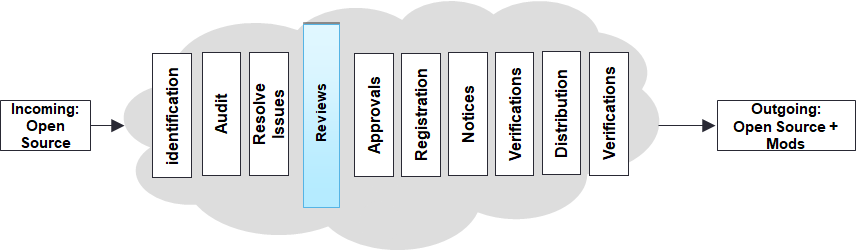
•Outcome:

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

## Architecture Review Template (Example)



## Performing Reviews



Steps to Review the resolved issues to confirm it matches 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 Open Source policies

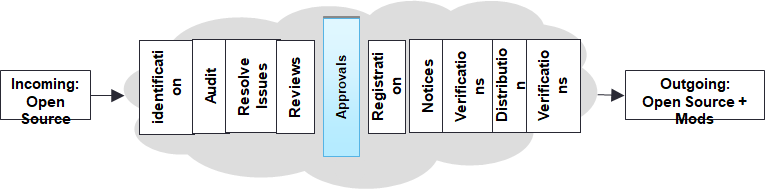
•Preserve audit report findings and mark resolved issues as ready for the next step (i.e. Approval)

## Approvals

•Based on the results of the software audit and review in previous steps, 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

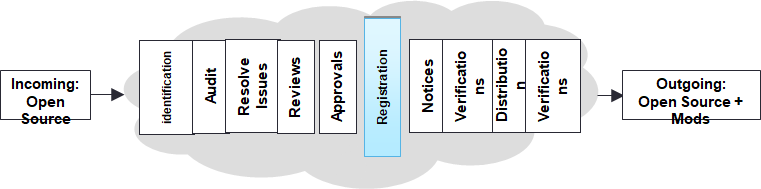


## Registration / Approval Tracking

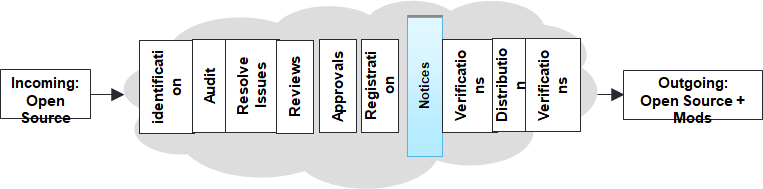
•Once an Open Source component has been approved for usage in a product, it should be added to the software inventory for that product

•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



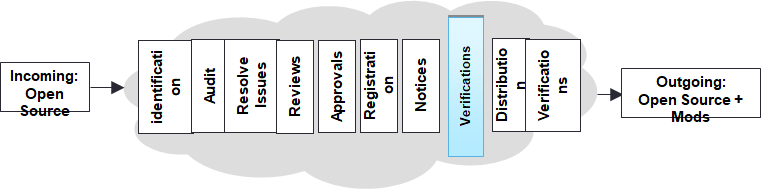
## Notices



•Prepare appropriate notices for any Open Source used in a product release:

* Acknowledge the use of Open Source by providing full copyright and attribution notices
* Inform the end user of the product on how to obtain a copy of the Open Source source code (when applicable, for example in the case of GPL and LGPL)
* Reproduce the entire text of the license agreements for the Open Source code included in the product as needed

## Pre-Distribution Verifications



•Steps to Verify that distributed software has been reviewed and approved:

•Verify Open Source packages destined for distribution have been identified and approved

•Verify the reviewed source code matches the binary equivalents shipping in the product

•Verify all appropriate notices have been included to inform end-users of their right to request source code for identified Open Source

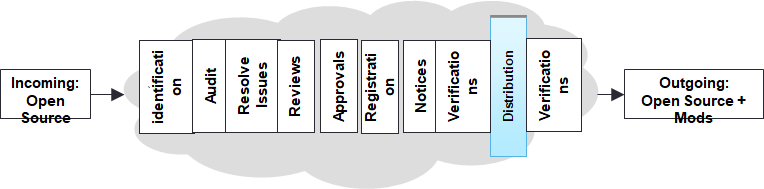
•Verify compliance with other identified obligations

•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

## Accompanying Source Code Distribution



•Steps to Provide accompanying source code as required:

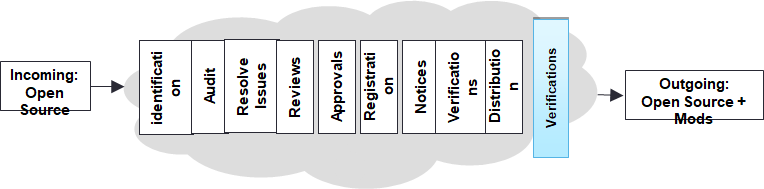
•Provide accompanying source code along with any associated build tools and documentation (e.g., by uploading to a distribution website or including in the distribution package)

•Accompanying source code is identified with labels as to which product and version to which it corresponds

•Outcome:

•Obligations to provide accompanying source code are met

## Final Verifications



•Steps to Validate compliance with license obligations:

•Verify accompanying source code (if any) has been uploaded or distributed correctly

•Verify uploaded or distributed source code corresponds to the same version that was approved

•Verify notices have been properly published and made available

•Verify other identified obligations are met

•Outcome:

•Verified Distributed Compliance Artifacts are appropriately provided

# Knowledge Check

## Knowledge Check 6.1

CHAPTER 8: Avoiding Compliance Pitfalls

# Introduction

## Chapter Overview

This chapter will describe some potential pitfalls to avoid in the compliance process:

1.Intellectual Property (IP) pitfalls

2.License Compliance pitfalls

3.Compliance Process pitfalls

## Learning Objectives

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

# Compliance Pitfalls

## Intellectual Property Pitfalls

| **Type & Description** | **Discovery** | **Avoidance** |
| --- | --- | --- |
| **Unplanned inclusion of copyleft Open Source into proprietary or 3rd 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 in conflict with the Open Source policy. | This type of failure can be  discovered by scanning or auditing the source code for possible  matches with:  •Open Source source code  •Copyright notices  Automated source code scanning tools may 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  •Conducting regular source code scans or audits for all the source code in the build environment. |

| **Type & Description** | **Discovery** | **Avoidance** |
| --- | --- | --- |
| **Unplanned linking of copyleft Open Source and proprietary source code:**  This type of failure occurs as  a result of linking software with conflicting or incompatible licenses. The legal effect of linking is subject to debate in the Open Source community. | This type of failure can be  discovered using a  dependency tracking tool  that shows any linking between  different software  components. | This type of failure can be  avoided by:  1.Offering training to engineering staff to avoid linking software components with licenses that conflict with your Open Source policies which will take a position on these legal risks  2.Continuously running the dependency tracking tool over your build environment |
| **Inclusion of proprietary**  **code into copyleft Open Source through**  **source code modifications** | This type of failure can be  discovered using the audits or scans to identify and analyze the source code you introduced to the Open Source component. | This type of failures can be  avoided by:  1.Offering training to engineering staff  2.Conducting regular code audits |

## License Compliance Pitfalls

| **Type & Description** | **Avoidance** |
| --- | --- |
| **Failure to Provide Accompanying Source Code/appropriate license, attribution or notice information** | This type of failure can be avoided by making source code capture and publishing a checklist item in the product release cycle before the product becomes available in the market place. |
| **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. |
| **Failure to Provide Accompanying 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 source code for modifications are published, rather than only the original source code for the Open Source component |

| **Type & Description** | **Avoidance** |
| --- | --- |
| **Failure to mark Open Source**  **Source Code**  **Modifications:**  Failure to mark Open Source source code that has been changed as required by the Open Source license (or providing information about modifications which has an insufficient level of detail or clarity to satisfy the license) | This type of failure can be avoided by:  1.Adding source code modification marking as a verification step before releasing the source code  2.Offering training to engineering staff to ensure they update copyright markings or license information of all Open Source or proprietary software that is going to be released to the public |

## Compliance Process Failures

| **Description** | **Avoidance** | **Prevention** |
| --- | --- | --- |
| **Failure by developers to seek approval**  **to use Open Source** | This type of failure can be  avoided by offering training to Engineering staff on the  company’s Open Source policies and  processes. | This type of failure can be  prevented by:  1.Conducting periodic full scan for the software platform to detect any “undeclared” Open Source usage  2.Offering training to engineering staff on the company's Open Source policies and processes  3.Including compliance in the employees performance review |
| **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 employee’s  professional development plan and it is monitored for completion as part of the performance review | This type of failure can be  prevented by mandating  engineering staff to take the Open Source training by a specific date |

| **Description** | **Avoidance** | **Prevention** |
| --- | --- | --- |
| **Failure to audit**  **the source code** | This type of failure can be avoided by:  1.Conducting periodic source code scans/audits  2.Ensuring that auditing is a milestone in the iterative development process | This type of failure can be  prevented by:  1.Providing proper staffing as to not fall behind in schedule  2.Enforcing periodic audits |
| **Failure to resolve**  **the audit findings**  **(analyzing the**  **"hits" reported**  **by a scan tool or audit)** | This type of failure can be avoided by not allowing a compliance ticket to be  resolved (i.e. closed) if the audit report is not finalized. | This type of failure can be  prevented by implementing blocks in approvals in the Open Source compliance process |
| **Failure to seek review of Open Source in a timely manner** | This type of failure can be avoided by initiating Open Source Review requests early even if engineering did not yet decide on the adoption of the Open Source  source code | This type of failure can be prevented through education |

## 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

## Establishing Community Relationships

As a company that uses Open Source in a commercial product, it is best to create and maintain a good relationship with the Open Source community - in particular, with the specific communities related to the Open Source projects you use and deploy in your commercial products.

In addition, good relationships with Open Source organizations can be very helpful in advising on the best way to be compliant and also help out if you experience a compliance issue.

Good relationships with the software communities may also be helpful for two-way communication: upstreaming improvements and getting support from the software developers.

# Knowledge Check

## Knowledge Check 7.1

CHAPTER 9: Developer Guidelines

# Introduction

## Chapter Overview

## Learning Objectives

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

# Subsection Title

## Developer Guidelines

•Select code from high quality, well supported Open Source communities

•Seek guidance

•Request formal approval for each Open Source component you are using

•Do not check un-reviewed code into any internal source tree

•Request formal approval for outside contributions to Open Source projects

•Preserve 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

•Do not re-name Open Source components unless you are required to under the Open Source license (e.g., required renaming of modified versions)

•Gather and retain Open Source project information required for your Open Source review process

## Anticipate Compliance Process Requirements

•Include time required to follow established Open Source policy in work plans

•Follow the developer guidelines for using Open Source software, particularly incorporating or linking Open Source code into proprietary or third-party source code or vice versa

•Review architecture plans and avoid mixing components governed by incompatible Open Source licenses

•Always update compliance verification - for every product

•Verify compliance on a product-by-product basis: Just because a Open Source package is approved for use in one product does not necessarily mean it will be approved for use in a second product

•And for every upgrade 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 a Open Source package, make sure that the license of the new version is the same as the license of the older used version (license changes can occur between version upgrades)

•If an Open Source project’s license changes, ensure that compliance records are updated and that the new license does not create a conflict

## Compliance Process Applies to all Open Source components

•In-bound software

•Take steps to understand what Open Source is included in software delivered by suppliers

•Evaluate your obligations for all of the software that will be included in your products

•Always audit source code you received from your software providers or alternatively make it a company policy that software providers must deliver you a source code audit report for any source code you receive

What is Intellectual Property?

* •What type of material does copyright law protect?
* •What copyright rights are most important for software?
* •Can software be subject to a patent?
* •What rights does a patent give to the patent owner?
* •If you independently develop your own software, is it possible that  
  you might need a copyright license from a third-party for that software?  
  A patent license?

Introduction to Open Source Licenses

* •What is an Open Source license?
* •What are typical obligations of a permissive Open Source license?
* •Name some permissive Open Source licenses.
* •What does license reciprocity mean?
* •Name some copyleft-style licenses.
* •What needs to be distributed for code used under a copyleft license?
* •Are Freeware and Shareware software considered Open Source?
* •What is a multi-license?
* •What information may you find in Open Source Notices, and how may the notices be used?

Introduction to Open Source Compliance

* •What does Open Source compliance mean?
* •What are two main goals of a Open Source Compliance Program?
* •List and describe important business practices of a Open Source Compliance Program.
* •What are some benefits of a Open Source Compliance Program?

Key Software Concepts for Open Source Review

* •What is incorporation?
* •What is linking?
* •What is modification?
* •What is translation?
* •What factors are important in assessing a distribution?

Running a Open Source Review

* Open Source Review
* Initiating a Open Source Review
* What information do you need to gather?
* Open Source Review Team
* Analyzing Proposed Open Source Usage
* Source Code Scanning Tools
* Working through the Open Source Review
* Open Source Review Oversight

End to End Compliance Management (Example Process)

* Example Small to Medium Company Checklist
* Example Enterprise Process
* Identify and Track Open Source Usage
* Auditing Source Code
* Resolving Issues
* Architecture Review (Example Template)
* Performing Reviews
* Approvals
* Registration / Approval Tracking
* Notices
* Pre-Distribution Verifications
* Accompanying Source Code Distribution
* Final Verifications

Avoiding Compliance Pitfalls

* Compliance Pitfalls
* Intellectual Property Pitfalls
* License Compliance Pitfalls
* Compliance Process Failures
* Ensure Compliance Prior to Product Shipment
* Establishing Community Relationships

Developer Guidelines

* Anticipate Compliance Process Requirements
* Compliance Process Applies to all Open Source components

**Check Knowledge section:**

Chapter 6: Running a Open Source Review

* What is the purpose of an Open Source Review?
  + The purpose is to perform Code quality review of the codes from open source communities.
  + To gather and analyze information regarding Open Source usage and to produce appropriate guidance. --> Correct answer
* What is the first action you should take if you want to use Open Source components?
  + First action should be to choose the most popular open source community to download any open source package.
  + Initiate an Open Source review process. The method for initiating this process may vary by company, but should be open to those who are involved in using Open Source in development. --> Correct answer
* What should you do if you have a question about using Open Source?
  + Initiate a Open Source review process or contact the Open Source review team. The process should be flexible enough so that Open Source users in your organization have access to guidance. --> Correct answer
  + Directly contact the contributors of the Open source community.
* What kinds of information might you collect for an Open Source review?
  + The package name, version, download URL, license, description and intended use in your product is a good starting point. The precise level of detail you will need depends on your organization and intended use case. --> Correct answer
  + Name of all the open source contributors and the number of versions released.
* What information helps identify who is licensing the software?
  + Direct information about the Licensor will not be available.
  + The copyright notices, attribution and source code normally helps to identify who is licensing the Open Source software. --> Correct answer
* What steps can be taken to assess the quality of information collected in an Open Source Review?
  + it is a straight forward single step process to check what license is attached to the package as a whole. if it is found in a license or readme file then the condition is satisfied.
  + Check information for completeness, consistency and accuracy. This process may be assisted by support teams, including teams that run code scanning tools to scan for undisclosed Open Source usage. --> Correct answer

Chapter 7: End to End Compliance Management (Example Process)

* What is involved in compliance due diligence?
  + Identification
  + Audit source code
  + Resolving issues
  + Performing reviews
  + Approvals
  + Registration/approval tracking
  + Notices
  + Pre-distribution verifications
  + Accompanying source code distribution
  + Verification
  + Product Manual → In correct option
  + All of the above except product manual--> Correct answer (Multiple response option, above all together)
* What does an architecture review look for?
  + Architecture reviews look for the individual module communication in a program.
  + Architecture reviews examine the relationships between Open Source components and company software.. --> Correct answer

Chapter 8: Avoiding Compliance Pitfalls

* What types of pitfalls can occur in Open Source compliance?
  + It only causes monetary or effort loss.
  + Pitfalls can occur under the following categories: IP failure, license compliance failure, and compliance process failure. --> Correct answer
* Give an example of a license compliance failure.: *An example of license compliance failure would be a failure to mark an open source software after modification or to properly list the open source software components in the software or to make the complete and corresponding source code available.* 
  + True --> Correct answer
  + False
* Give an example of a compliance process failure.: *An example of compliance process failure would be a failure in the process related to audit, review, or approving the open source software. Auditors "waived" all the red-flagged items in a report, or that the review and approval process takes too long.*
  + True → Correct Answer
  + False
* What are the benefits of prioritizing compliance?
  + There are no benefits, it is just a additional cost in money and time → Incorrect Answer
  + The benefits of prioritizing compliance are that you become more efficient in your use of Open Source, and that you build a better relationship with the open source community. --> Correct answer
* What are the benefits of maintaining a good community relationship?
  + The benefits of maintaining a good community relationship are that you can better assess how you can comply with the Open Source license requirements, and you have a better two-way communication with regard to contribution and use of the Open Source. --> Correct answer
  + Better marketing for the organisation/product in the Business world.

Chapter 9: Developer Guidelines

* Name some general guidelines developers can practice when working with Open Source.
  + Select code from high quality Open Source communities
  + Seek guidance
  + Preserve existing licensing information
  + Gather and retain Open Source project information for your review process --> List of Correct answer (Multiple response)
* Should you remove or alter Open Source license header information?
  + No – existing license information should be preserved, additional header information can be added for modifications or additions to source code --> Correct answer
  + Yes, that’s the best practice to minify the source file
* How can a new version of a previously-reviewed Open Source component create new compliance issues?
  + A change in the Open Source license for the new version of the Open Source component
  + New dependencies introduced with new versions which create additional Open Source obligations. These dependencies may be embedded in the Open Source distribution or they may be dependencies resolved at build time. --> List of Correct answer (Multiple response)
  + None of the above → Incorrect answer
* What risks should you address with in-bound software?
  + License compliance for any disclosed Open Source embedded in the in-bound software
  + The potential for creating license conflicts by integrating inbound software with other Open Source or proprietary software
  + Undisclosed or unknown Open Source included in the in-bound software
  + All of the above --> Correct answer

**Glossary**

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

Dependencies:

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

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

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.

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

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

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

Trademarks: Legal protections for marks (such as words, logos, slogans, colors, etc.) that are connected to product identification

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

**Examples of License Acronyms**

| 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 |  |
| CC-BY-SA | Creative Commons Attribution ShareAlike |

You can find a more detailed list of licenses and their acronyms: https://spdx.org/licenses/

**Additional Reference:**

<Link to join Openchain channels of interest> :

<https://lists.openchainproject.org/g/main>

<Link to Openchain reference pages and github for readable doc> : <https://github.com/OpenChain-Project/Reference-Material>

<Link to LFC open source courses page> (We have free and paid courses available)

<https://training.linuxfoundation.org/?s=Open+source&filter=training>

* What additional information is important when reviewing a Open Source component from an outside vendor?
* No additional information necessary.
* Development team's point of contact in case you need to follow up with future Open Source issues. You may also want to obtain copyright and attribution notices, and source code for vendor modifications if these are needed to satisfy license obligations for Open Source licenses governing the third-party software. --> Correct answer
* Give an example of an intellectual property failure.: *An example of IP failure would be commingling (blend) of proprietary code and open source code, which may result in making proprietary software available to the general public despite the company's preference.*
* False
* True --> Correct answer
* Name some important steps in a compliance process.
  + Follow developer guidelines, especially for any Open Source code included in or linked to proprietary code
  + Review and approve all Open Source early in the cycle
  + Review architecture and avoid mixing components governed by incompatible licenses
  + Verify OSS compliance for every product and every version prior to release
  + Verify OSS compliance for every product and every version prior to release
  + Review OSS compliance for new versions of OSS --> List of Correct answer (Multiple response, above all options together constitute correct answer)
  + Documentation of efforts involved → Incorrect option