OPENCHAIN

Reference Training Slides

Open Source Training for OpenChain 2.1 (ISO/IEC 5230:2020)

These slides follow US law. Different legal jurisdictions may have different legal requirements.

These slides do not contain legal advice

What are the OpenChain Reference Slides?

- The OpenChain Project defines the key requirements of a quality open source compliance program.
- The requirements are described in the International Standard for open source compliance: DIS 5230 (ISO Number Pending).
- These reference training slides help companies meet the requirements of the International Standard.
- These slides help companies satisfy the requirements of the Specification Section 2.0. They can also be used for general compliance training.

Learn more at: https://www.openchainproject.org

Contents

1. What is Intellectual Property?
2. Introduction to Open Source Licenses
3. Introduction to Open Source Compliance
4. Key Software Concepts for Open Source Review
5. Running a Open Source Review
6. End to End Compliance Management (Example Process)
7. Avoiding Compliance Pitfalls
8. Developer Guidelines

CHAPTER 1

What is Intellectual Property?

What is “Intellectual Property”?

- Copyright: protects original works of authorship
  - Protects expression (not the underlying idea)
  - It covers software, books, and similar works
- Patents: useful inventions that are novel and non-obvious
  - Limited monopoly to incentivize innovation
- Trade secrets: protects valuable confidential information
- Trademarks: protects marks (word, logos, slogans, color, etc.) that identify the source of the product
  - Consumer and brand protection; avoid consumer confusion and brand dilution

This chapter will focus on copyright and patents,
the areas most relevant to Open Source compliance.

Copyright Concepts in Software

- Basic rule: copyright protects creative works
- Copyright generally applies to literary works, such as books, movies, pictures, music, maps
- Software is protected by copyright
    - Not the functionality (that’s protected by patents) but the expression (creativity in implementation details)
    - Includes Binary Code and Source Code
- The copyright owner only has control over the work that he or she created, not someone else’s independent creation
- Infringement may occur if copying without the permission of the author

Copyright Rights Most Relevant to Software

- The right to ***reproduce*** the software – making copies
- The right to create "***derivative works***" – making modifications
  - The term derivative work comes from the US Copyright Act
  - It is a "term of art" meaning that it has a particular meaning based on the statute and not the dictionary definition
  - In general it refers to a new work based upon an original work to which enough original creative work has been added so that the new work represents an original work of authorship rather than a copy
- The right to ***distribute***
  - Distribution is generally viewed as the provision of a copy of a piece of software, in binary or source code form, to another entity (an individual or organization outside your company or organization)

Note: The interpretation of what constitutes a "derivative work" or a "distribution" is subject to debate in the Open Source community and within Open Source legal circles

Patent Concepts in Software

- Patents protect functionality – this can include a method of operation, such as a computer program
  - Does not protect abstract ideas, laws of nature
- A patent application must be made in a specific jurisdiction in order to obtain a patent in that country. If a patent is awarded, the owner has the right to stop anybody from exercising its functionality, regardless of independent creation
- Other parties who want to use the technology may seek a patent license (which may grant rights to use, make, have made, sell, offer for sale, and import the technology)
- Infringement may occur even if other parties independently create the same invention

Licenses

- A “license” is the way a copyright or patent holder gives permission or rights to someone else
- The license can be limited to:
  - Types of use allowed (commercial / non-commercial, distribution, derivative works / to make, have made, manufacture)
  - Exclusive or non-exclusive terms
  - Geographical scope
  - Perpetual or time limited duration
- The license can have conditions on the grants, meaning you only get the license if you comply with certain obligations
  - E.g, provide attribution, or give a reciprocal license
- May also include contractual terms regarding warranties, indemnification, support, upgrade, maintenance

Check Your Understanding

- 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?

CHAPTER 2

Introduction to Open Source Licenses

Open Source Licenses

- Open Source licenses by definition make source code available under terms that allow for modification and redistribution
- Open Source licenses may have conditions related to providing attributions, copyright statement preservation, or a written offer to make the source code available
- One popular set of licenses are those approved by the Open Source Initiative (OSI) based on their Open Source Definition (OSD). A complete list of OSI-approved licenses is available at http://www.opensource.org/licenses/

Permissive Open Source Licenses

- Permissive Open Source license: a term used often to describe minimally restrictive Open Source licenses
- Example: BSD-3-Clause
  - The BSD license is an example of a permissive license that allows unlimited redistribution for any purpose in source or object code form as long as its copyright notices and the license's disclaimers of warranty are maintained
  - The license contains a clause restricting use of the names of contributors for endorsement of a derived work without specific permission
- Other examples: MIT, Apache-2.0

License Reciprocity & Copyleft Licenses

- Some licenses require that if derivative works (or software in the same file, same program or other boundary) are distributed, the distribution is under the same terms as the original work
- This is referred to as a “copyleft” or “reciprocal” effect
- Example of license reciprocity from the GPL version 2.0:

You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed [...] under the terms of this License.

- Licenses that include reciprocity or Copyleft clauses include all versions of the GPL, LGPL, AGPL, MPL and CDDL

Proprietary License or Closed Source

- A proprietary software license (or commercial license or EULA) has restrictions on the usage, modification and/or distribution of the software
- Proprietary licenses are unique to each vendor – there are as many variations of proprietary licenses as there are vendors and each must be evaluated individually
- Open Source developers often use the term “proprietary” to describe a commercial non-Open Source license, even though both Open Source and proprietary licenses are based on intellectual property and provide a license grant to that property

Other Non-Open Source Licensing Situations

- Freeware – software distributed under a proprietary license at no or very low cost
  - The source code may or may not be available, and creation of derivative works is usually restricted
  - Freeware software is usually fully functional (no locked features) and available for unlimited use (no locking on days of usage)
  - Freeware software licenses usually impose restrictions in relation to copying, distributing, and making derivative works of the software, as well as restrictions on the type of usage (personal, commercial, academic, etc.)
- Shareware – proprietary software provided to users on a trial basis, for a limited time, free of charge and with limited functionalities or features
  - The goal of shareware is to give potential buyers the opportunity to use the program and judge its usefulness before purchasing a license for the full version of the software
  - Most companies are very leery of Shareware, because Shareware vendors often approach companies for large license payments after the software has freely propagated within their organizations.

Other Non-Open Source Licensing Situations

- “Non-commercial” – some licenses have most of the characteristics of a Open Source license, but are limited to non-commercial use (e.g. CC-BY-NC).
  - Open Source by definition cannot limit the field of use of the software
  - Commercial use is a field of use so any restriction prevents the license from being Open Source

Public Domain

- The term **public domain** refers to software not protected by law and therefore usable by the public without requiring a license
- Developers may include a ***public domain declaration*** with their software
  - E.g., "All of the code and documentation in this software has been dedicated to the public domain by the authors."
  - The public domain declaration is not the same as a Open Source license

- A public domain declaration attempts to waive or eliminate any intellectual property rights that the developers may have in the software to make it clear that it can be used without restriction, but the enforceability of these declarations is subject to dispute within the Open Source community and its effectiveness at law varies from jurisdiction to jurisdiction
- Often the public domain declaration is accompanied by other terms, such as warranty disclaimers; in such cases, the software may be viewed as being under a license rather than being in the public domain

License Compatibility

- License compatibility is the process of ensuring that license terms do not conflict.
- If one license requires you to do something and another prohibits doing that, the licenses conflict and are not compatible if the combination of the two software modules trigger the obligations under a license.
  - GPL-2.0 and EPL-1.0 each extend their obligations to "derivative works" which are distributed.
  - If a GPL-2.0 module is combined with an EPL-1.0 module and the merged module is distributed, that module must
    - (according to GPL-2.0) be distributed under GPL-2.0 only, and
    - (according to EPL-1.0) under EPL-1.0 only.
  - The distributor cannot satisfy both conditions at once so the module may not be distributed.
  - This is an example of *license incompatibility.*

The definition of "derivative work" is subject to different views in the Open Source community and
its interpretation in law is likely to vary from jurisdiction to jurisdiction.

Notices

Notices, such as text in comments in file headers, often provide authorship and licensing information. Open Source licenses may also require the placement of notices in or alongside source code or documentation to give credit to the author (an attribution) or to make it clear the software includes modifications.

- **Copyright notice** – an identifier placed on copies of the work to inform the world of copyright ownership. Example: Copyright © A. Person (2016)
- **License notice** – a notice that specifies and acknowledges the license terms and conditions of the Open Source included in the product.
- **Attribution notice** – a notice included in the product release that acknowledges the identity of the original authors and / or sponsors of the Open Source included in the product.
- **Modification notice** – a notice that you have made modifications to the source code of a file, such as adding your copyright notice to the top of the file.

Multi-Licensing

- Multi-licensing refers to the practice of distributing software under two or more different sets of terms and conditions simultaneously
  - E.g., when software is “dual licensed,” the copyright owner gives each recipient the choice of two licenses
- Note: This should not be confused for situations in which a licensor imposes more than one license, and you must comply with *all* of them

Check Your Understanding

- What is a 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?

CHAPTER 3

Introduction to Open Source Compliance

Open Source Compliance Goals

- **Know your obligations.** You should have a process for identifying and tracking Open Source components that are present in your software

- **Satisfy license obligations.** Your process should be capable of handling Open Source license obligations that arise from your organization’s business practices

What Compliance Obligations Must Be Satisfied?

Depending on the Open Source license(s) involved, your compliance obligations may consist of:

- **Attribution and Notices.** You may need to provide or retain copyright and license text in the source code and/or product documentation or user interface, so that downstream users know the origin of the software and their rights under the licenses. You may also need to provide notices regarding modifications, or full copies of the license.
- **Source code availability.** You may need to provide source code for the Open Source software, for modifications you make, for combined or linked software, and scripts that control the build process.
- **Reciprocity.** You may need to maintain modified versions or derivative works under the same license that governs the Open Source component.
- **Other terms.** The Open Source license may restrict use of the copyright holder name or trademark, may require modified versions to use a different name to avoid confusion, or may terminate upon any breach.

Open Source Compliance Issues: Distribution

- Dissemination of material to an outside entity
  - Applications downloaded to a user’s machine or mobile device
  - JavaScript, web client, or other code that is downloaded to the user’s machine
- For some Open Source licenses, access via a computer network can be a “trigger” event
  - Some licenses define the trigger event to include permitting access to software running on a server (e.g., all versions of the Affero GPL if the software is modified) or in the case of “users interacting with it remotely through a computer network”

Open Source Compliance Issues: Modification

- Changes to the existing program (e.g., additions, deletions of code in a file, combining components together)
- Under some Open Source licenses, modifications may cause additional obligations upon distribution, such as:
  - Providing notice of modification
  - Providing accompanying source code
  - Licensing modifications under the same license that governs the Open Source component

Open Source Compliance Program

Organizations that have been successful at Open Source compliance have created their own ***Open Source Compliance Programs*** (consisting of policies, processes, training and tools) to:

1. Facilitate effective usage of Open Source in their products (commercial or otherwise)
2. Respect Open Source developer/owner rights and comply with license obligations
3. Contribute to and participate in Open Source communities

Implementing Compliance Practices

Prepare business processes and sufficient staff to handle:

- Identification of the origin and license of all internal and external software
- Tracking Open Source software within the development process
- Performing Open Source review and identifying license obligations
- Fulfillment of license obligations when product ships
- Oversight for Open Source Compliance Program, creation of policy, and compliance decisions
- Training

Compliance Benefits

Benefits of a robust Open Source Compliance program include:

- Increased understanding of the benefits of Open Source and how it impacts your organization
- Increased understanding of the costs and risks associated with using Open Source
- Increased knowledge of available Open Source solutions
- Reduction and management of infringement risk, increased respect of Open Source developers/owners’ licensing choices
- Fostering relationships with the Open Source community and Open Source organizations

Check Your Understanding

- 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?

CHAPTER 4

Key Software Concepts for Open Source Review

How do you want to use a Open Source component?

Common scenarios include:

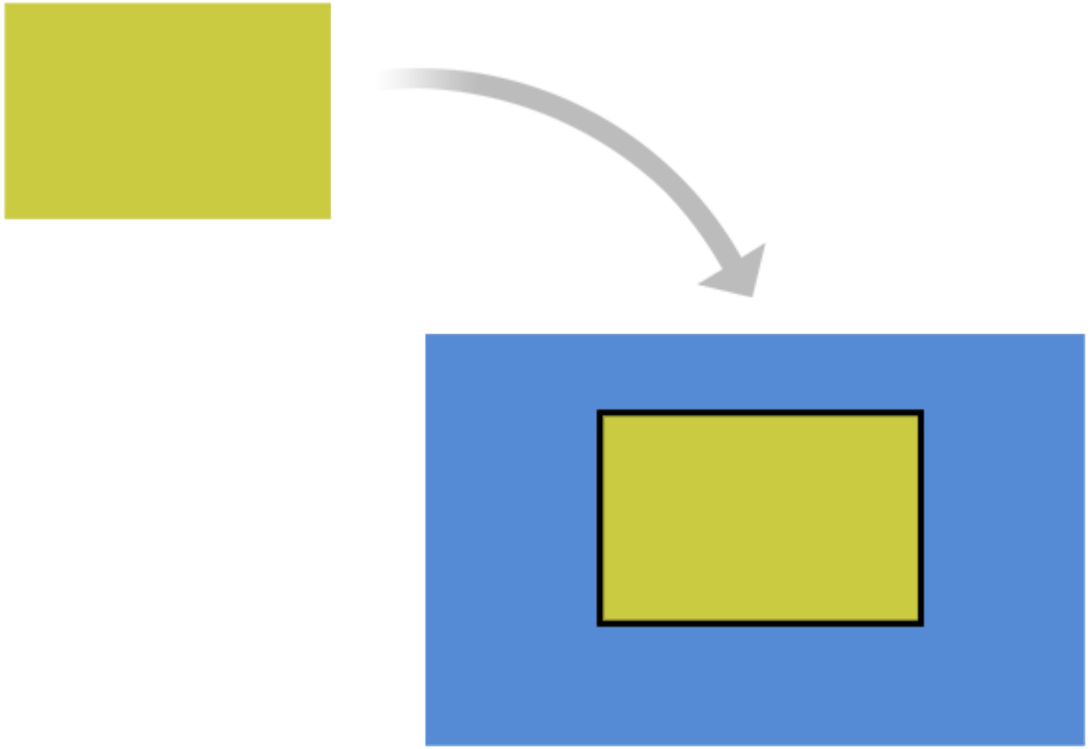
- Incorporation
- Linking
- Modification
- Translation

Incorporation

A developer may copy portions of a Open Source component into your software product.

Relevant terms include:

- Integrating
- Merging
- Pasting
- Adapting
- Inserting



Linking

A developer may link or join a Open Source component with your software product.

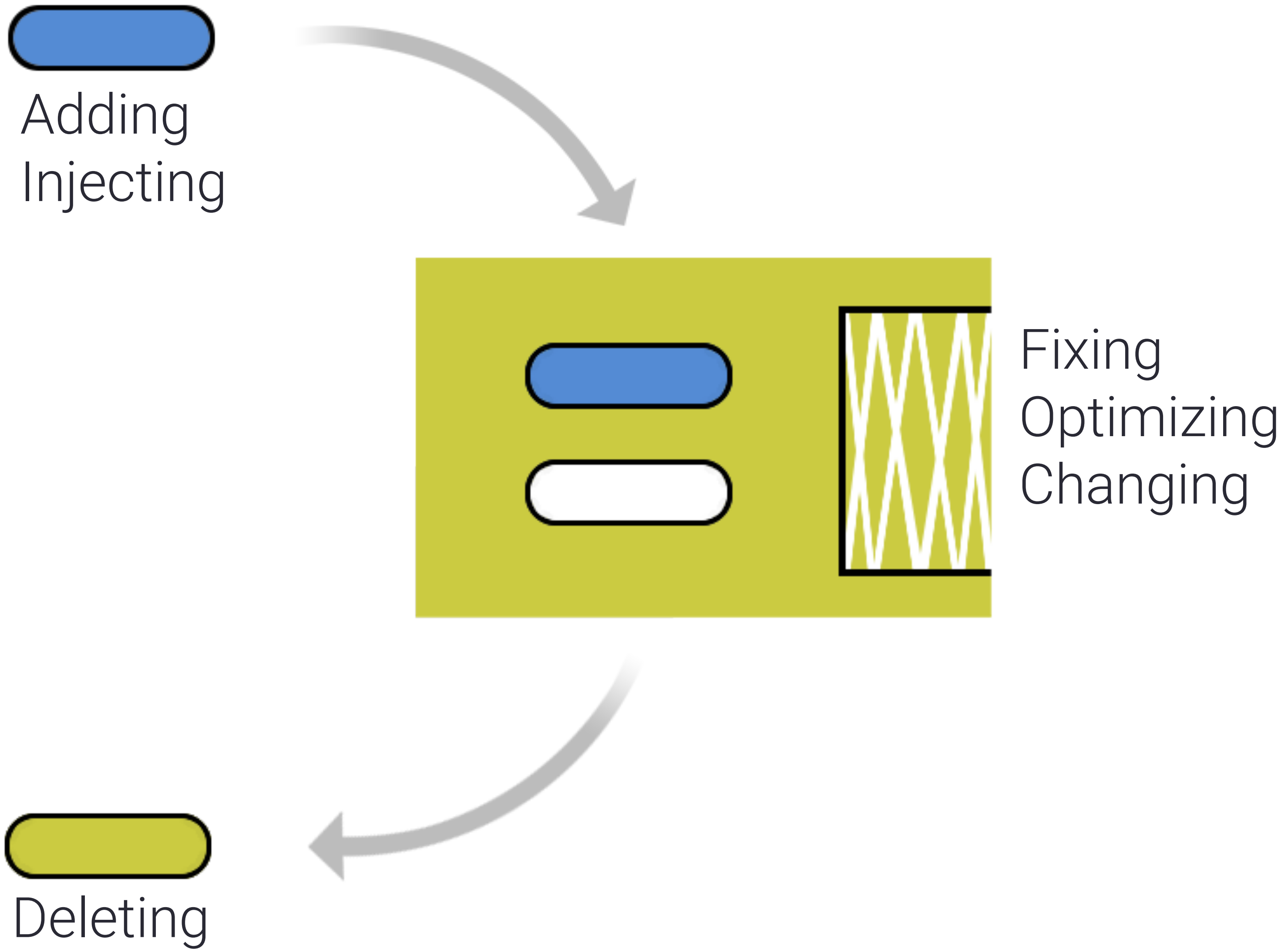
Relevant terms include:

- Static/Dynamic Linking
- Pairing
- Combining
- Utilizing
- Packaging
- Creating interdependency

Modification

A developer may make changes to a Open Source component, including:

- Adding/injecting new code into the Open Source component
- Fixing, optimizing or making changes to the Open Source component
- Deleting or removing code

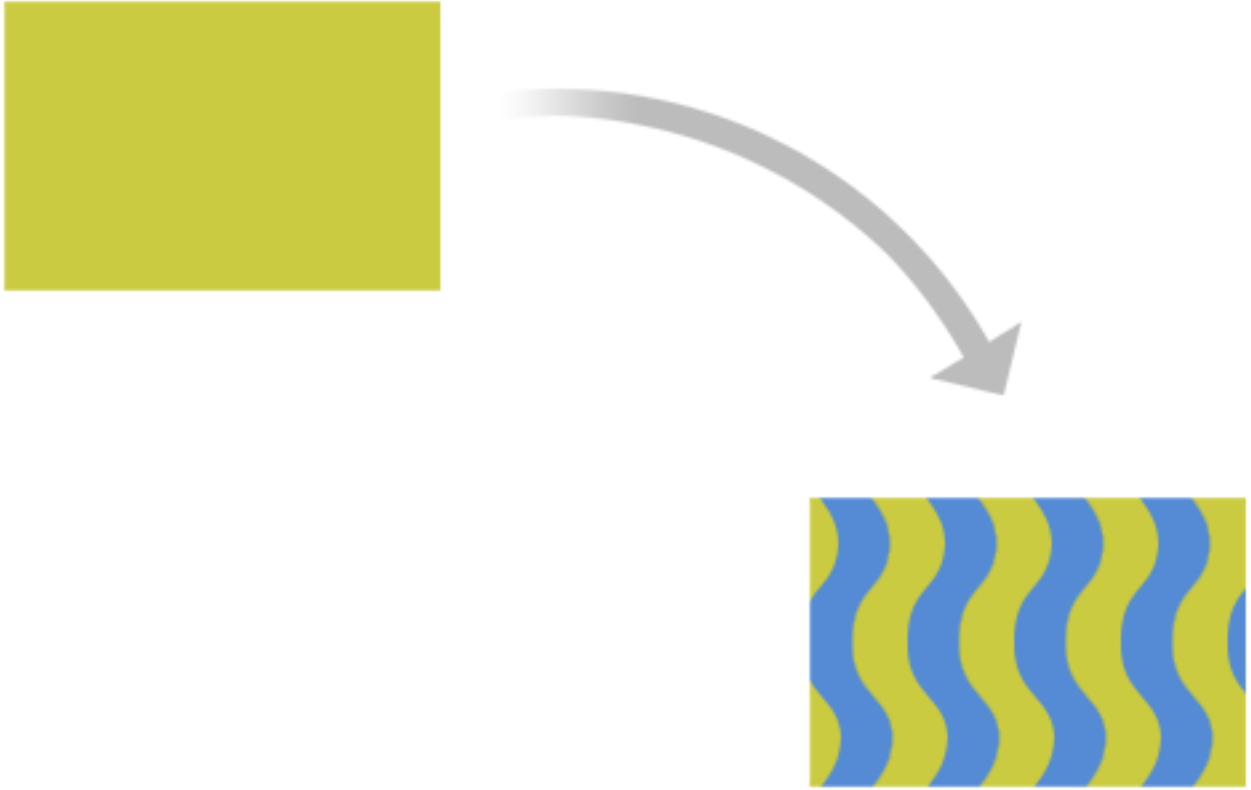


Translation

A developer may transform the code from one state to another.

Examples include:

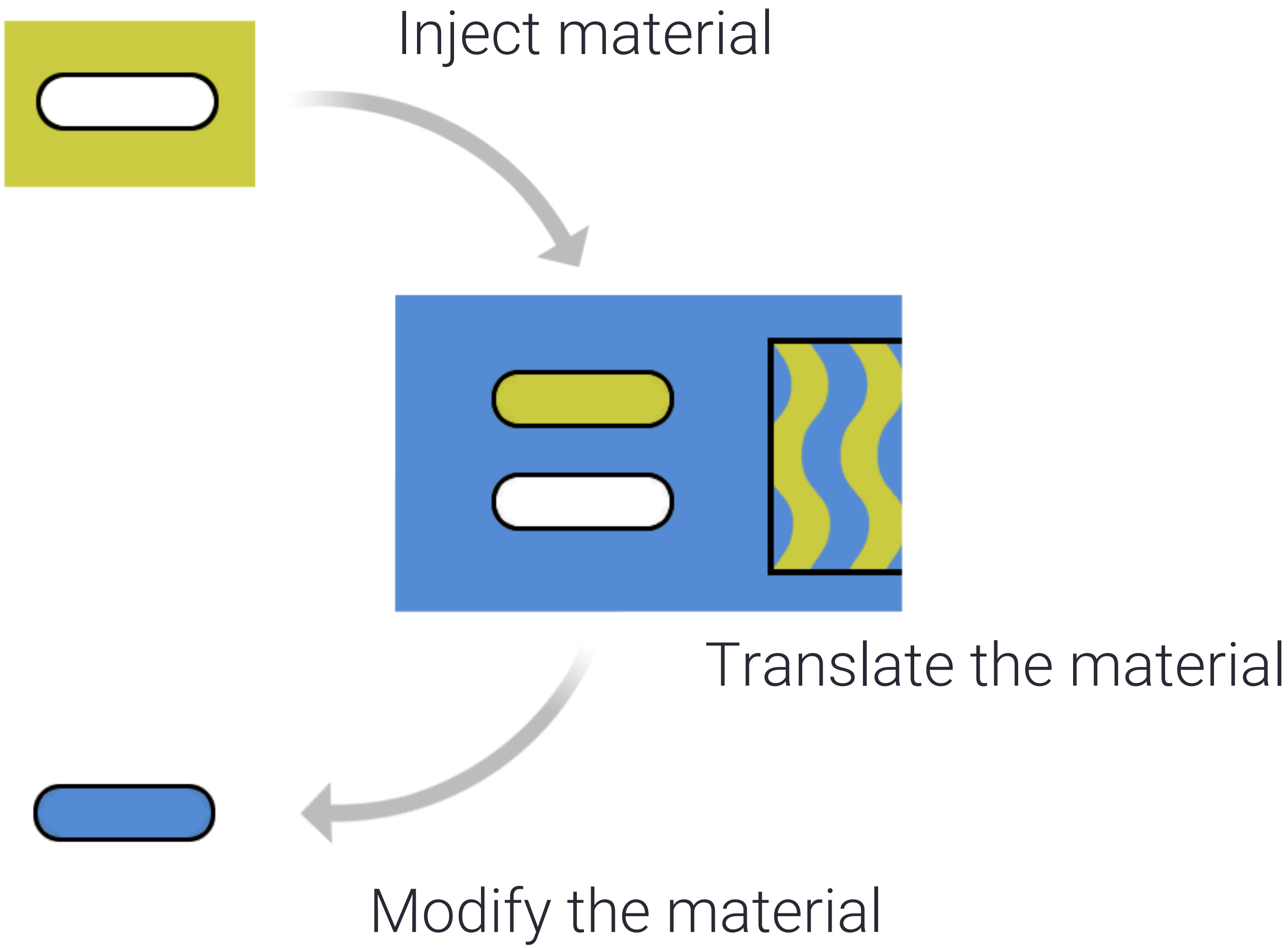
- Translating Chinese to English
- Converting C++ to Java
- Compiling into binary



Development Tools

Development tools may perform some of these operations behind the scenes.

For example, a tool may inject portions of its own code into output of the tool.



Inject material

Translate the material

Modify the material

How is the Open Source component distributed?

- Who receives the software?
  - Customer/Partner
  - Community project
  - Another legal entity within the business group (this may count as distribution)

- What format for delivery?
  - Source code delivery
  - Binary delivery
  - Pre-loaded onto hardware

Check Your Understanding

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

CHAPTER 5

Running a Open Source Review

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 a Open Source Compliance Program is a ***Open Source Review*** process. This process is where a company can analyze the Open Source software it uses and understand its rights and obligations
- The Open Source Review process includes the following steps:
  - Gather relevant information
  - Analyze and understand license obligations
  - Provide guidance compatible with company policy and business objectives

Initiating a Open Source Review

Initiate a Open Source Review
Program Manager
Product Manager
Engineer

Anyone working with Open Source in the company should be able to initiate a Open Source Review, including Program or Product Managers, Engineers, and Legal.

Note: The process often starts when new Open Source-based software is selected by engineering or outside vendors.

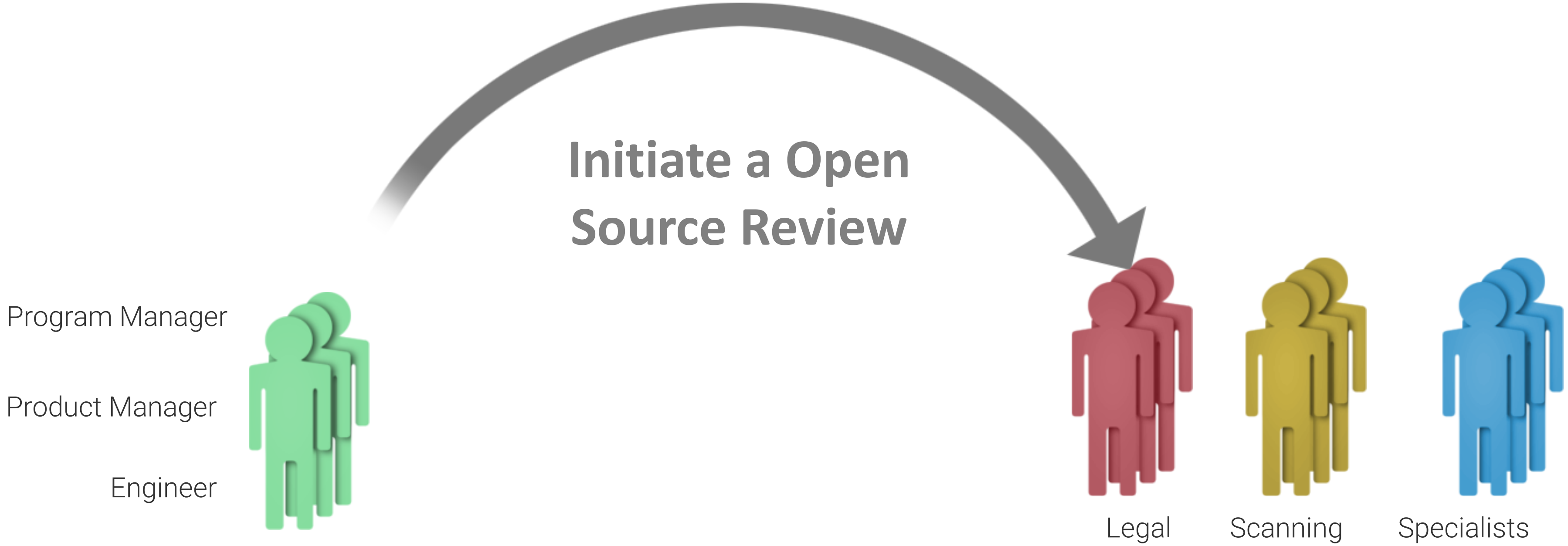
What information do you need to gather?

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

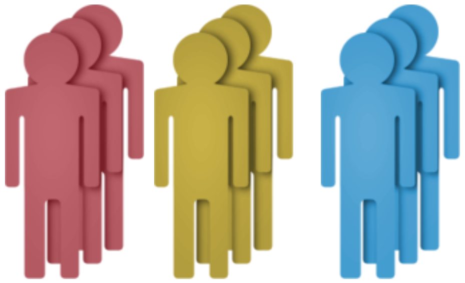
Open Source Review Team



A Open Source Review team includes the company representatives that support, guide, coordinate and review the use of Open Source. These representatives may include:

- Legal to identify and evaluate license obligations
- Source code scanning and tooling support to help identify and track Open Source usage
- Engineering Specialists working with business interests, commercial licensing, export compliance, etc., who may be impacted by Open Source usage

Analyzing Proposed Open Source Usage



The Open Source Review team should assess the information it has gathered before providing guidance for issues. This may include scanning the code to confirm the accuracy of the information.

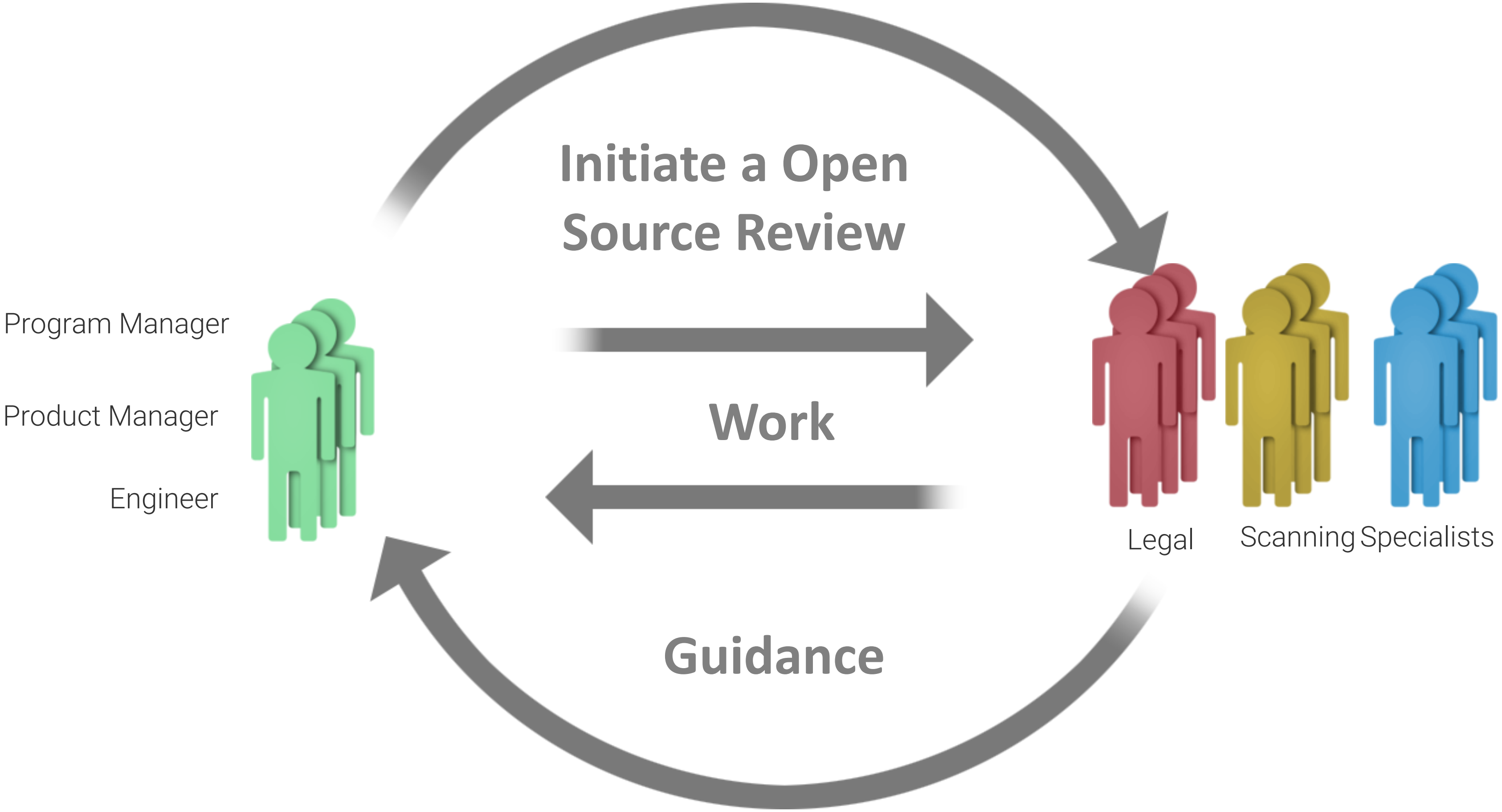
The Open Source Review team should consider:

- 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?

Source Code Scanning Tools

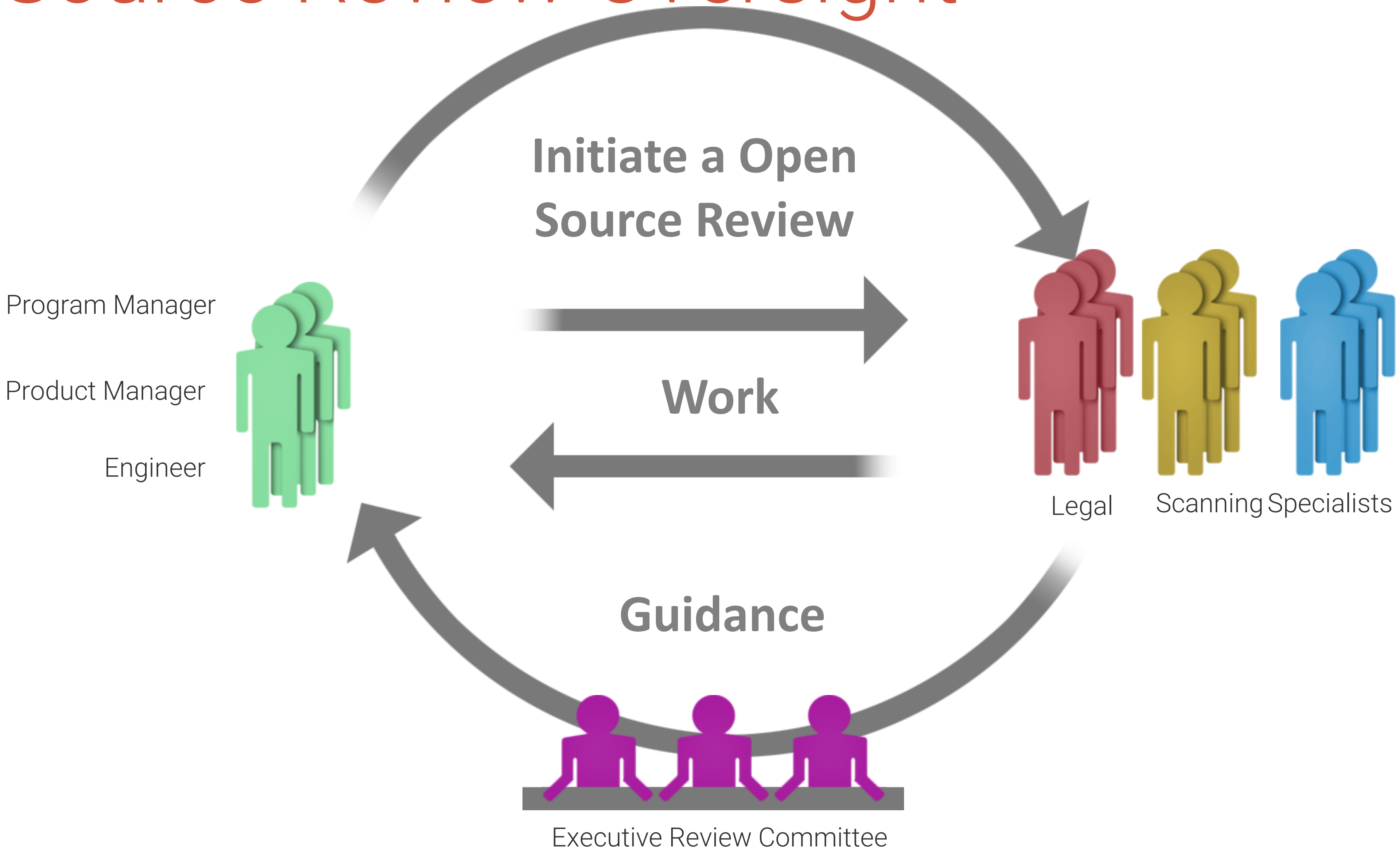
- There are many different automated source code scanning tools.
- All of the solutions address specific needs and - for that reason - none will solve all possible challenges
- Companies pick the solution most suited to their specific market area and product
- Many companies use both an automated tool and manual review
- A good example of freely available source code scanning tool is FOSSology, a project hosted by the Linux Foundation: https://www.FOSSology.org

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.

Open Source Review Oversight



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

Check Your Understanding

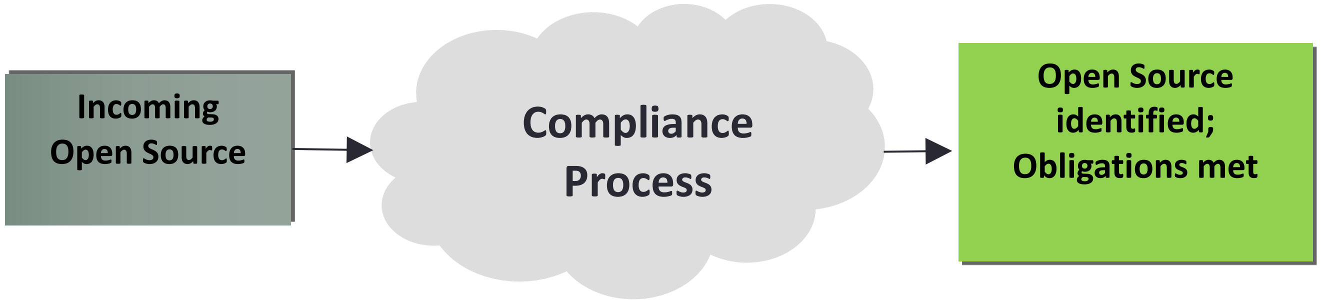
- What is the purpose of a Open Source Review?
- What is the first action you should take if you want to use Open Source components?
- What should you do if you have a question about using Open Source?
- What kinds of information might you collect for a Open Source review?
- What information helps identify who is licensing the software?
- What additional information is important when reviewing a Open Source component from an outside vendor?
- What steps can be taken to assess the quality of information collected in a Open Source Review?

CHAPTER 6

End to End Compliance Management (Example Processes)

Introduction

- 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.



Example Small to Medium Company Checklist

OPENCHAIN

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

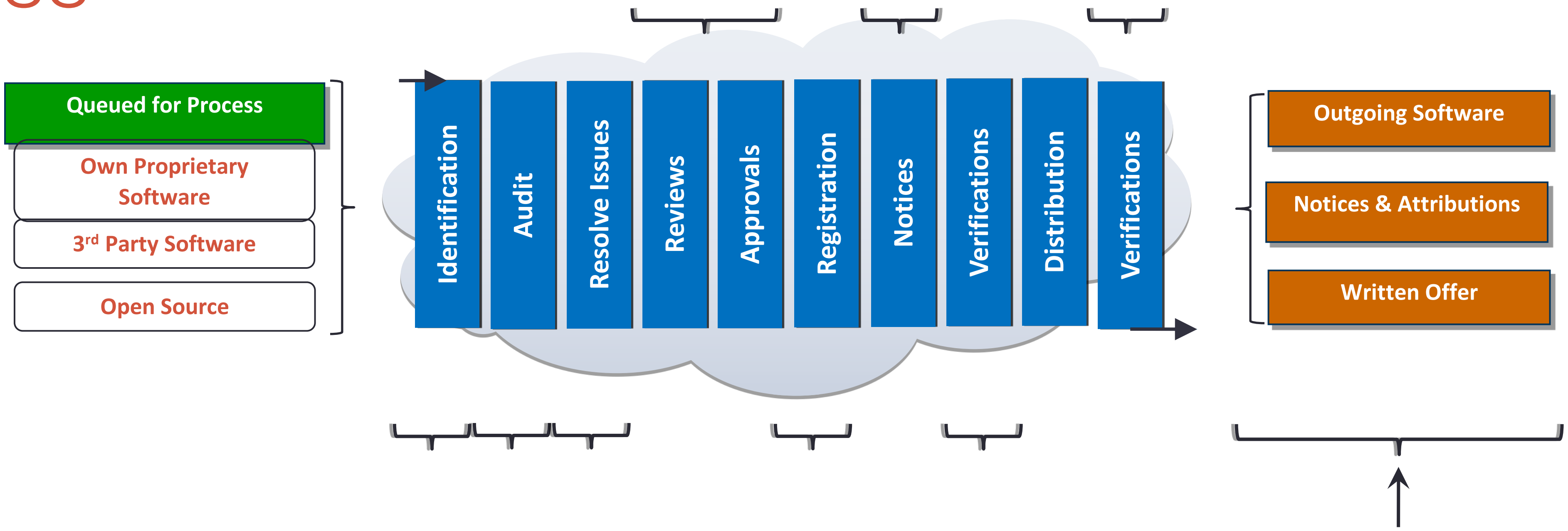
You can get detailed checklists for these items here: https://www.linuxfoundation.org/projects/opencompliance/self-assessment-compliance-checklist

Example Enterprise Process

Review and approve compliance record of Open Source software components

Compile notices for publication

Post publication verifications



Identify Open Source components for review

Scan or audit source code
– and –
Confirm origin and license of source code

Resolve any audit issues in line with company Open Source policies

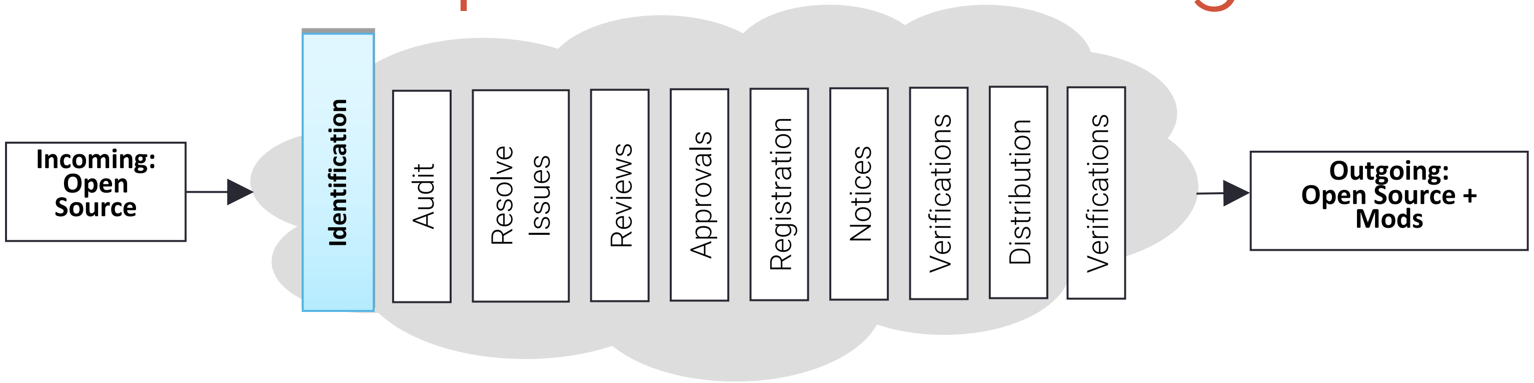
Record approved software/version
in inventory per product and per release

Verify source code packages for distribution
– and –
Verify appropriate notices are provided

Publish source code, notices and provide written offer

Example of Compliance Management End-to-End Process

Identify and Track Open Source Usage

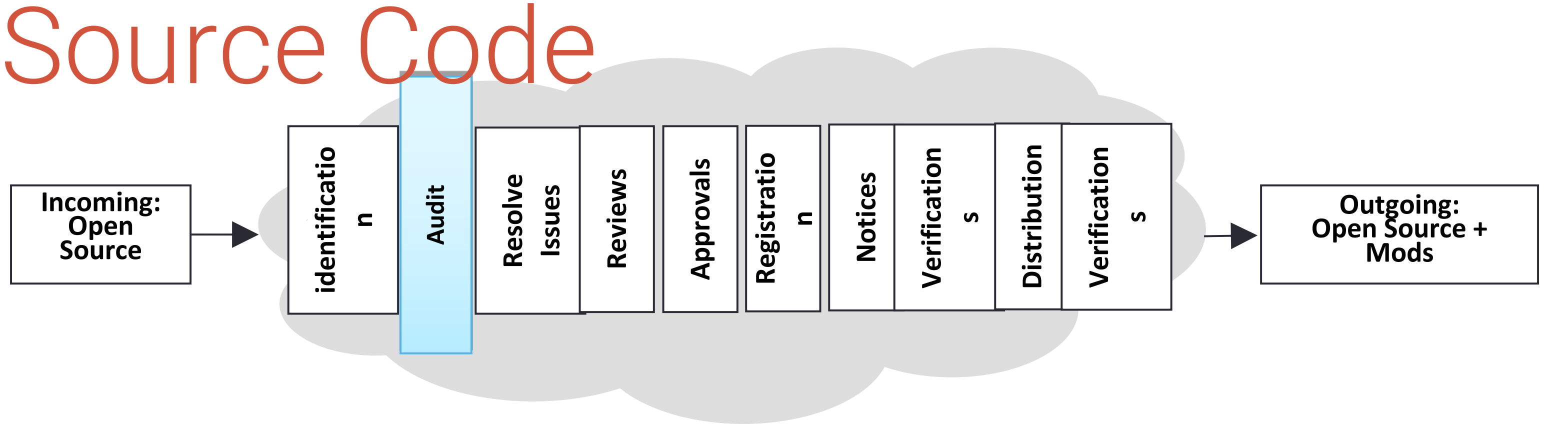


Identify Open Source components

- Steps:
  - 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.

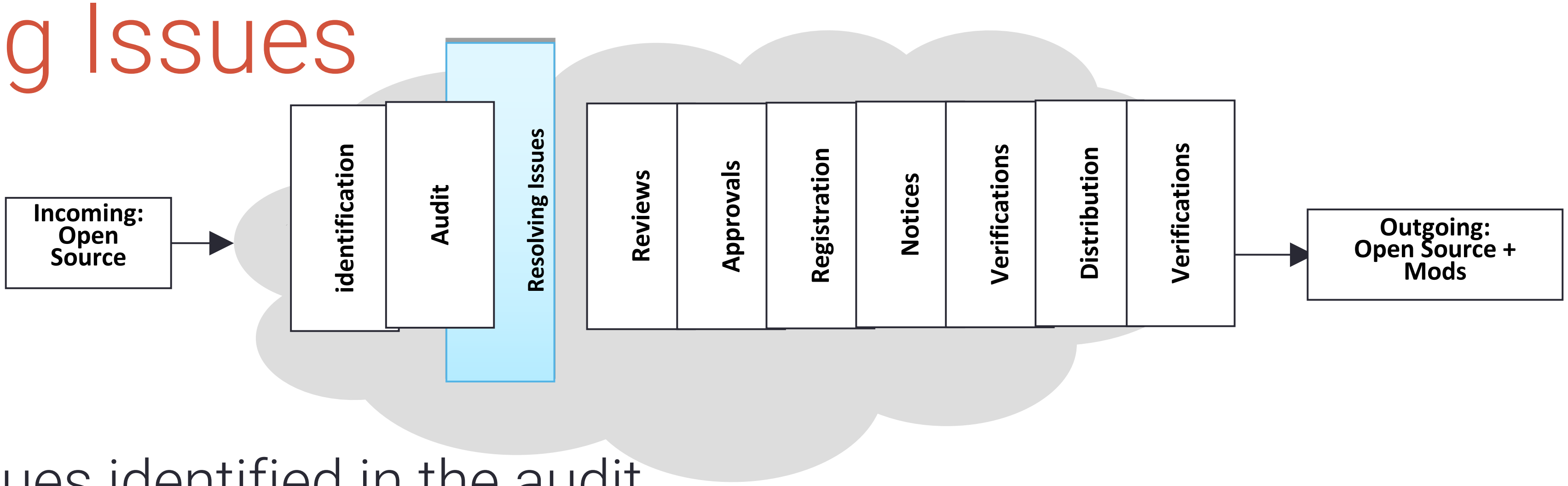
Auditing Source Code



Identify and audit Open Source licenses

- Steps:
  - 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



Resolve all issues identified in the audit

- Steps:
  - 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 slide for template)

- Outcome:

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

Architecture Review (Example Template)

Legend

🟩 Proprietary

🟧 3rd Party Commercial

🟥 GPL

🟨 LGPL

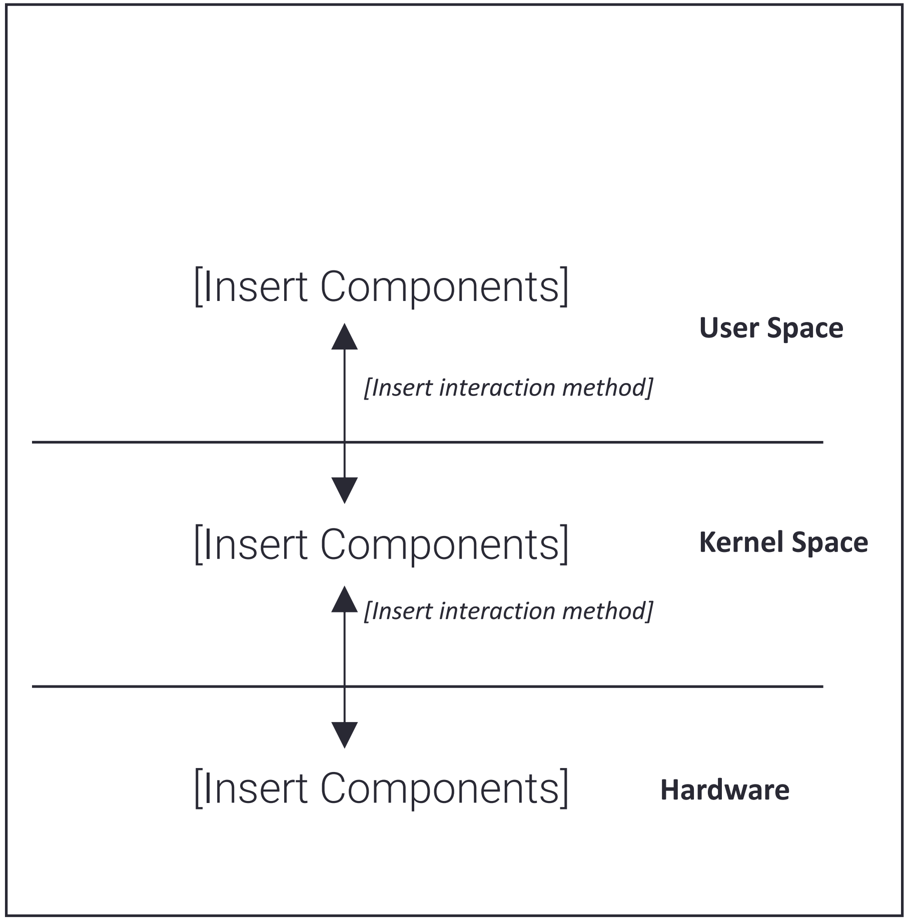
🟦 Open Source Permissive

(fc) ↔ Function call

(si) ↔ Socket interface

(sc) → System call

(sh) ↔ Shared headers



[Insert Components]

User Space

[Insert interaction method]

[Insert Components]

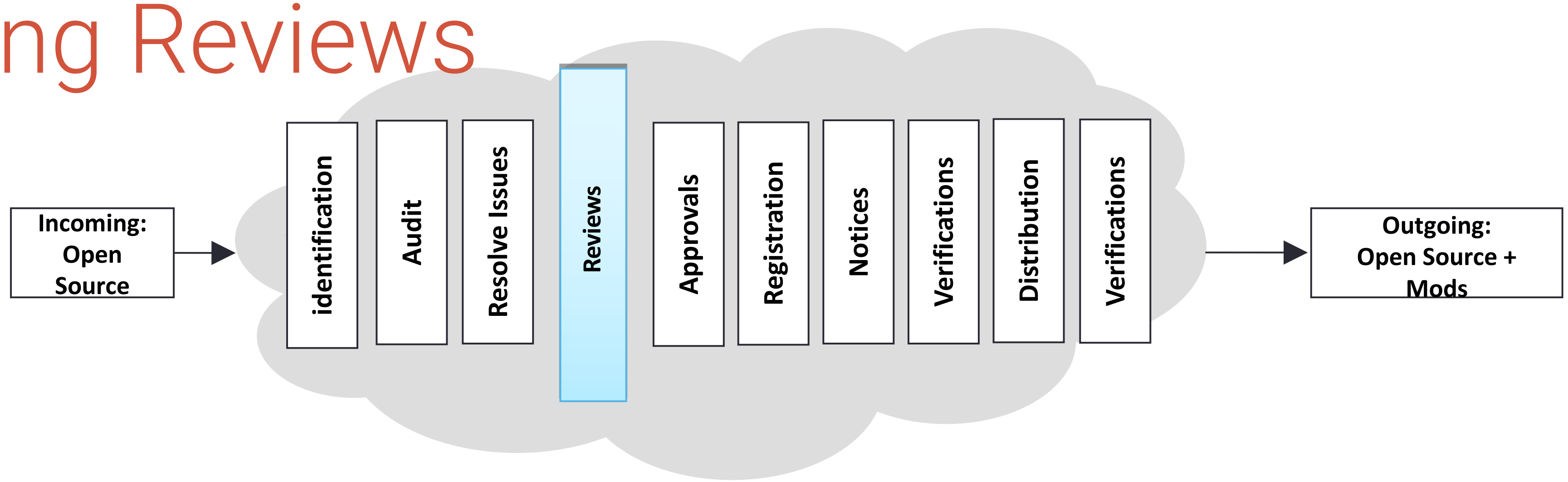
Kernel Space

[Insert interaction method]

[Insert Components]

Hardware

Performing Reviews



Review the resolved issues to confirm it matches your Open Source policy

Steps:

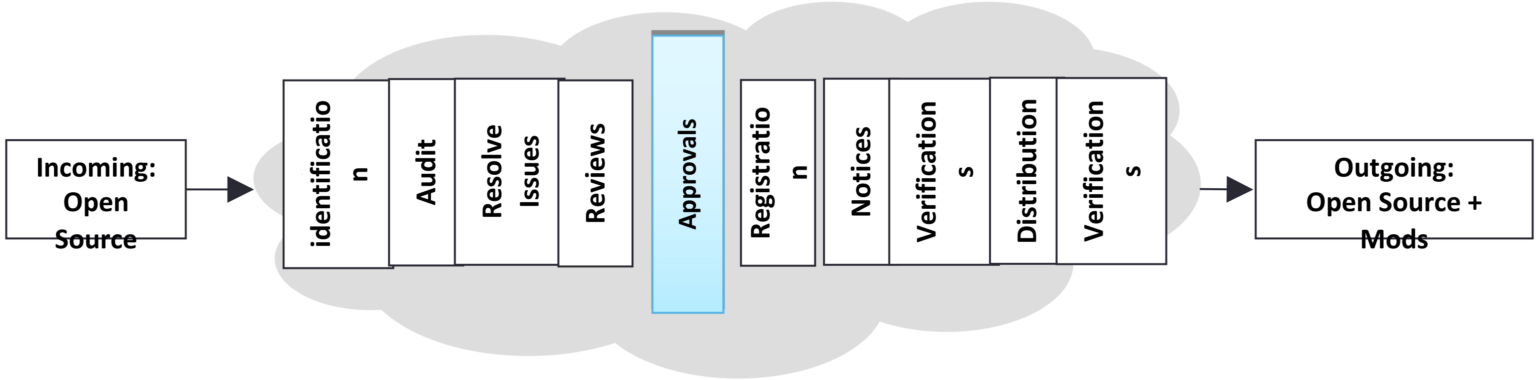
- Include appropriate authority levels in review staff
- Conduct review with reference to your Open Source policy

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)

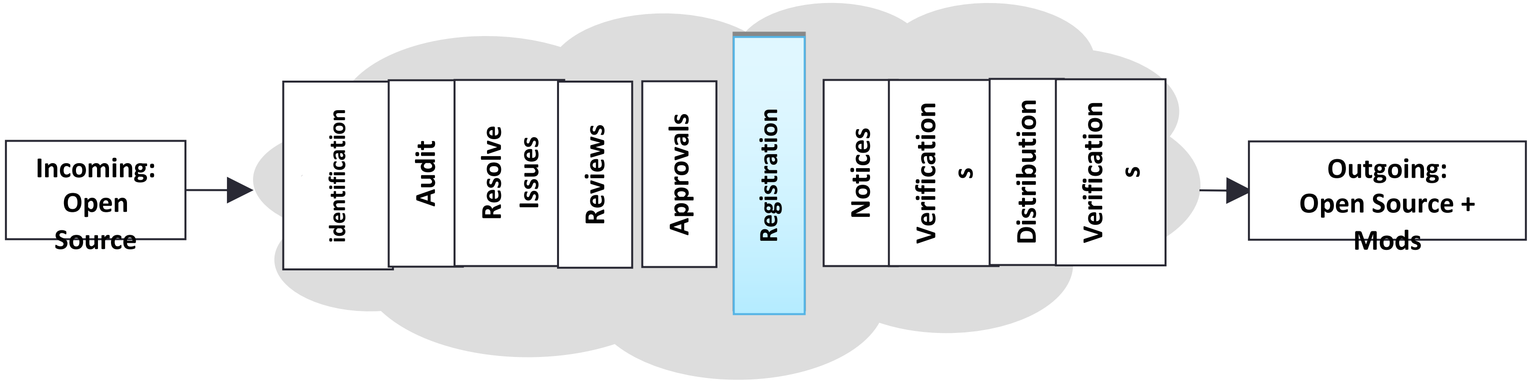
Approvals

- 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

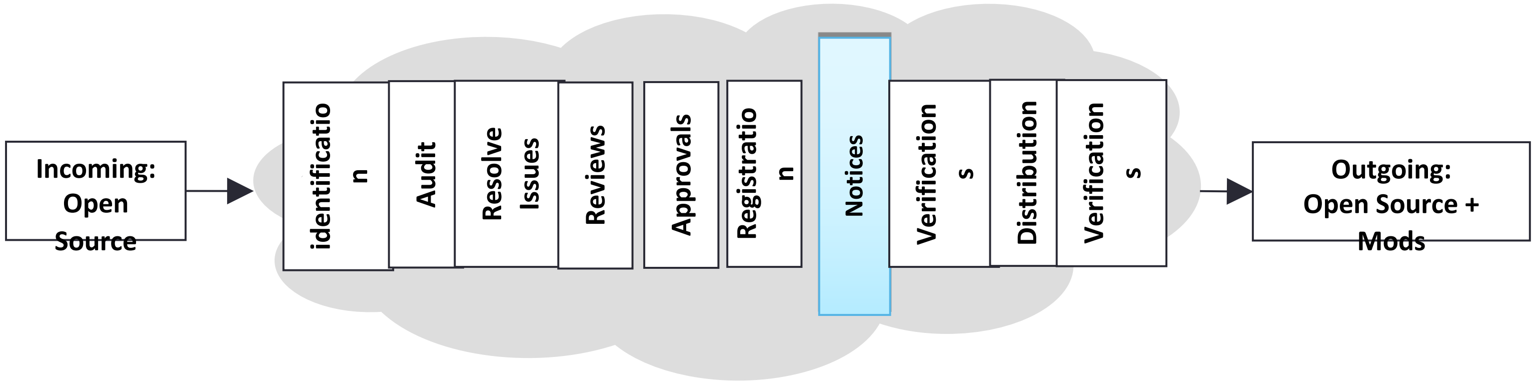


Registration / Approval Tracking

- Once a 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 a Open Source component or if a new usage model is proposed

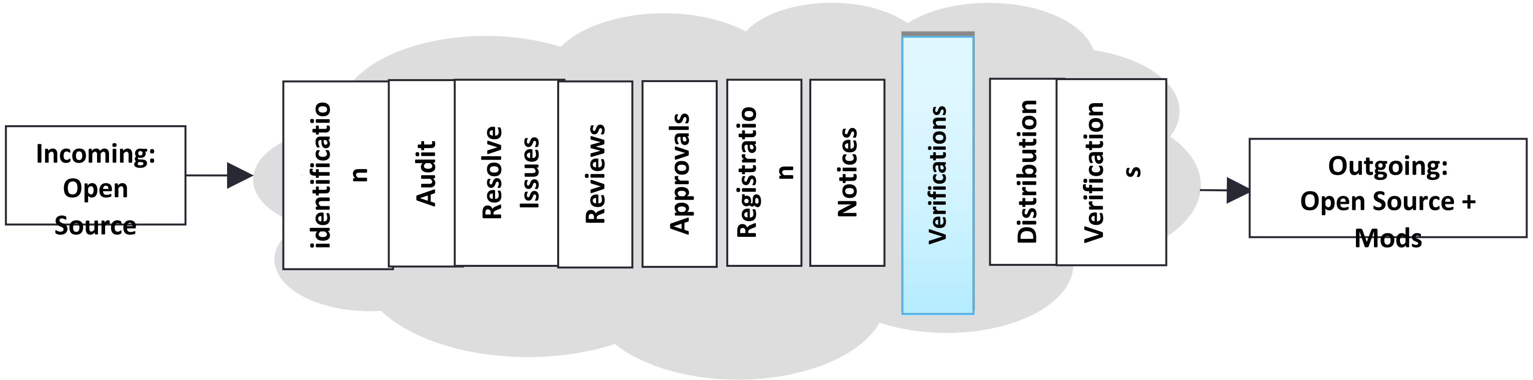


Notices



- 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

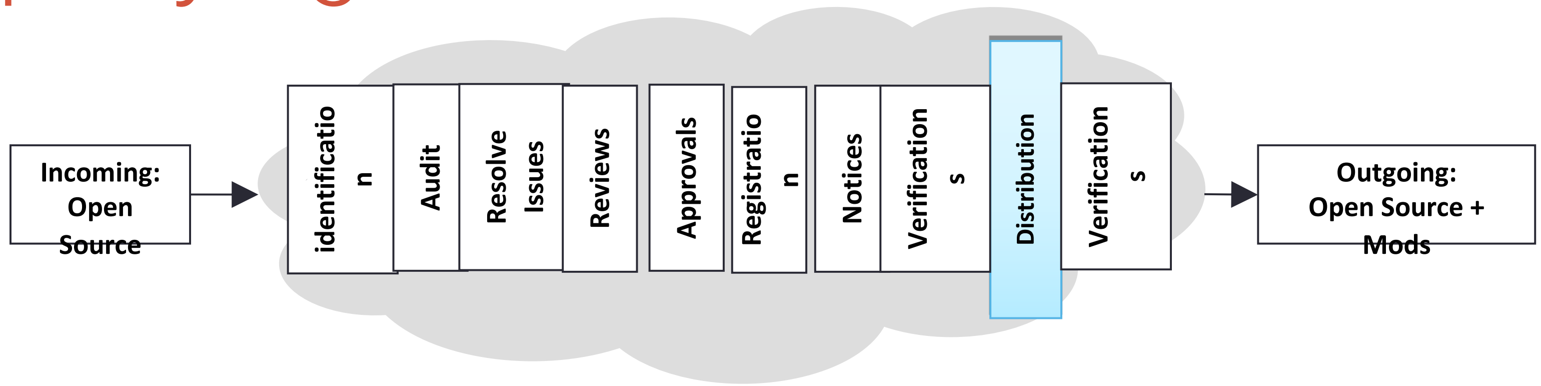
Pre-Distribution Verifications



Verify that distributed software has been reviewed and approved

- Steps:
  - 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

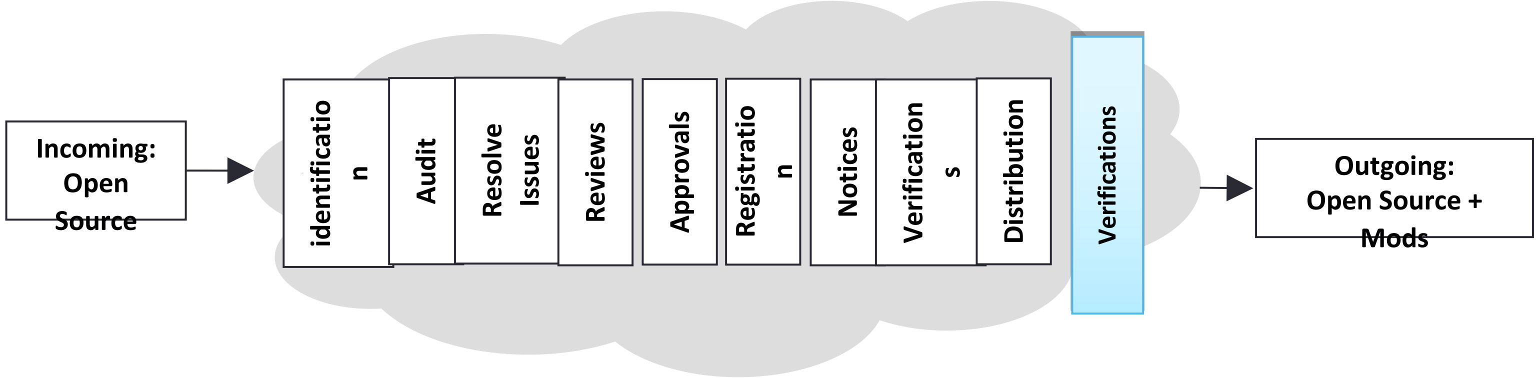
Accompanying Source Code Distribution



Provide accompanying source code as required

- Steps:
  - 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



Validate compliance with license obligations

- Steps:
  - 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

Check Your Understanding

- What is involved in compliance due diligence (for our example process, describe the steps at a high level)?
  - Identification
  - Audit source code
  - Resolving issues
  - Performing reviews
  - Approvals
  - Registration/approval tracking
  - Notices
  - Pre-distribution verifications
  - Accompanying source code distribution
  - Verification
- What does an architecture review look for?

CHAPTER 7

Avoiding Compliance Pitfalls

Compliance Pitfalls

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

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.

Intellectual Property Pitfalls

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 you 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

License Compliance Pitfalls

Type & Description

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)

Avoidance

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

Compliance Process Failures

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 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.

Check Your Understanding

- What types of pitfalls can occur in Open Source compliance?
- Give an example of an intellectual property failure.
- Give an example of a license compliance failure.
- Give an example of a compliance process failure.
- What are the benefits of prioritizing compliance?
- What are the benefits of maintaining a good community relationship?

CHAPTER 8

Developer Guidelines

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 a 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

Check Your Understanding

- Name some general guidelines developers can practice when working with Open Source.
- Should you remove or alter Open Source license header information?
- Name some important steps in a compliance process.
- How can a new version of a previously-reviewed Open Source component create new compliance issues?
- What risks should you address with in-bound software?

Learn more through the free Compliance Basics for Developers hosted by the Linux Foundation at:
https://training.linuxfoundation.org/linux-courses/open-source-compliance-courses/ compliance-basics-for-developers