Introduction to Open Source License Compliance Management (LFC193)

Course Introduction / Welcome

This course provides a reference example of how an open source compliance program should be structured. It is designed to be used in the context of OpenChain ISO/IEC 5230:2020 but can be used for any open source compliance program. We have seen previous versions of this course used by consultancies, law firms and other parties for education, services and assessment purposes.

The course provides knowledge from the basics of intellectual property through to key concepts of an open source review. It is based on real-world experience and focuses on outcomes that are directly applicable to product and service deployment. The outcome of this course will be a clear understanding of how to use compliance as business optimization, reducing resource use and increasing efficiency.

It is important to remember that everyone started somewhere, and no matter what your current level of experience in this field, you are on the right track to becoming an expert in the strategy and actions needed. We look forward to supporting your journey.

CHAPTER 1: Rights and Licensing Around Software

# Introduction

## Chapter Overview

This chapter is about license compliance in software. In practice, this means a focus on copyright, which both restricts and grants rights to users of the software. The "owner (individual or legal entity)" of software possesses the copyright and the users have a license to use it. It is important to understand how this works to ensure compliance with terms around the use of the software. This chapter does not require a legal background. While it discusses some concepts from the law, it does this in a non-technical way.

## Learning Objectives

By the end of this chapter, you should be able to understand what rights apply to software and what this means for open source software.

# Fundamental Concepts Around Software Licensing

Intellectual Property laws apply to software as well as many other products and services. Intellectual Property comes in various forms. These include [Copyright](https://en.wikipedia.org/wiki/Copyright), [Patents](https://en.wikipedia.org/wiki/Patent) and [Trademarks](https://en.wikipedia.org/wiki/Trademark). We will be focusing on copyright, which protects original works of authorship. In one sense, the software is protected with roughly the same approach as a written book. These are both creative items authored by one or more people.

### How Copyright Applies To Software

Copyright around software applies to the expression or implementation of the software. In other words, it applies to the source code or binary code. The original author or authors own the copyright and therefore have legal control over the original software and any derivative works.

The author can choose to provide a right to use the software in any way a user might want, or they might choose more restrictive licensing terms. Authors usually choose licensing terms with some conditions around use. These terms are usually expressed in a (software) license agreement.

### The Specific Rights Related To Software

When we talk about copyright and software we are usually referring to the right to reproduce the software (make a copy), the right to create derivative works (making modifications) and the right to distribute the software (sharing the software or copies of it with others). The details of how distribution in a legal sense is understood may differ from country to country, but the basic legal concepts are shared across the world.

### A Brief Overview of Patents And Software

Copyright is an important concept in software. It helps to protect authors and provide grants or restrictions to users. However, it is not the only part of Intellectual Property that applies to software. In addition to copyright, the software may also be affected by patents.

While copyright applies to written software code, just like it applies to a written novel or screenplay, patents protect functionality implemented by that written code. In other words, a patent applies to a technical method of operation, rather than the abstract idea. Patents cover how a software program accomplishes something, or how an engine works, but not the idea of word processors or the idea of an engine. Copyright arises automatically as soon as the software is written. Patents, in contrast, have to be applied for through a lengthy process and only apply to new inventions.

If you are a supplier or customer and want to use the specific functionality implemented in a product, and it is covered by a patent, then you may need to get permission through a patent license (either a standalone license or as a patent license section in another license). These licenses may describe how you can use, share, modify, sell or import the technology, or have other requirements and limitations. Patent licenses are usually specific to the functionality being licensed, and therefore have a significant variation in the market.

You may be subject to a legal case even if you independently implement specific functionality. In other words, patents apply regardless of whether you are using the original software or a derivative of the original software, as long as the specific functionality is implemented.

### What is a License and how do they work?

A “license” is the way a copyright or patent holder grants permission or rights to someone else to use the item(s) of interest under certain legal terms and conditions. These must be followed to use the software.

License conditions can include:

1.Types of use allowed and how it can be shared or used for business i.e.,

1. Commercial or non-commercial, educational use only,
2. How the artifact can be distributed, i.e., medium of distribution or format of distribution.
3. How others can make use of the artifact, whether used wholly or as derivative works / to make have made, manufacture
4. The geographical scope of where the artifact can be used or sold can be specified and limited too; generally this is in case of commercial software or non open-source software licenses.
5. The time limits on the right of use of the software

All of the above terms and conditions often apply to non open source software. In contrast, open source licenses typically have fewer conditions than proprietary software licenses, but the conditions they do have tend to focus on the conditions for redistributing the software to others (something which proprietary software licenses tend to try to prohibit entirely).

A license may also include contractual terms regarding warranties, indemnification, support, upgrade, maintenance.

# Knowledge Check

## Knowledge Check 1.1

* What does copyright law protect?
  + Copyright protects the brand recognition.
  + Copyright protects original creative works --> Correct answer -- Reason: Software, like books, music and film, is a creative work. These types of work are protected by copyright to ensure the original authors are rewarded for their creations.
  + Copyright protects an invention in the same manner as a patent.
* What rights does a patent give to the patent owner?
  + Patent holders can exclude others from using an invention regardless of whether the others have independently created a version of the same invention. --> Correct answer -- Reason: A patent gives the owner the right to exclude others from making, using, or selling an invention for a limited period of time.
  + Patent owners must prove that an infringer was aware of, and copied, their invention in order to claim for patent infringement.
  + Patent owners cannot exercise any rights over granted patents except claim royalty over the commercialised invention.
* If you develop your own software will you need a copyright license from a third-party?
  + Independently developed software needs explicit proprietary copyright and license notice on each source file to claim ownership.
  + The independently developed software will not need a copyright license from a third-party. --> Correct answer -- Reason: If you develop the software without the help of third-party code the copyright exclusively belongs to you.
  + The independently developed software will need a copyright license from a third-party if it later turns out to be similar to code that the third party has already written.

CHAPTER 2: Introduction to Open Source Licenses

# Introduction

## Chapter Overview

This chapter explains how open source licenses work. It builds on the general topic of intellectual property by guiding participants into the specific area of Open Source licensing. It is important to respect third-party Open Source licensing grants to ensure your organization can continue using the relevant Open Source software. This chapter does not require a legal background. While it discusses some concepts from the law, it does this in a non-technical way.

## Learning Objectives

By the end of this chapter, you should be able to understand the basic rights and obligations provided by Open Source licenses.

## License Use Cases Related to Open Source Software

### Open Source Licenses

Open source licenses vary in specific terms but they all allow users to use, study, share and improve the software they receive. By necessity, this means that source code for the software is made available. These licenses often also have conditions related to providing a copy of the license, a copyright statement, and a written offer to make the source code available.

To ensure continuity and clarity around open source software licenses there are various definitions to contextualize the field. A frequently used definition is the Open Source Definition (OSD) hosted by the Open Source Initiative (OSI). This lists ten criteria that determine whether a license qualifies as open source or not. There can be licenses that are very similar to open source licenses but do not meet all ten of these criteria.

It is worth remembering that Open Source by definition cannot limit the field of use of the software. On occasion you may see some normal open source license text with an added "only for non-commercial" clause, rendering it a non-open source license. Licenses such as [Creative Commons Attribution-NonCommercial (CC-BY-NC)](https://creativecommons.org/licenses/by-nc/2.0/) are inherently non-commercial and therefore not open source.

### Proprietary License or Closed Source

A proprietary software license is similar to an open source license in that they both apply to the software. However, proprietary software licenses generally have more restrictions on how the licensed software can be used. For example, it is pretty common for proprietary software licenses to disallow sharing the software with other users, or to disallow making any derivative versions of the software. Proprietary licenses are usually vendor-specific and therefore there are as many proprietary licenses as there are vendors selling proprietary software.

There is nothing wrong with proprietary software licensing. It is simply a business decision taken by the author or authors of the software. However, it is important to be aware that this is a different approach to open source software licensing, and therefore it is usually incompatible.

## Other Non-Open Source Licensing Situations

There are various types of software licenses that may provide software without cost but under terms different to an open source license. This highlights two things:

1. Open Source software is not a synonym for free of charge software.
2. Software that is free is not a synonym for open source software.

One common example is Freeware, a type of proprietary software that is free of charge. This is usually fully functional and available for unlimited use. However, Freeware usually imposes 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.). The latter point is particularly important. Open Source by definition cannot limit the field of use of the software. You may even see a typical open source license text with an added "only for non-commercial" clause, rendering it a non-open source license and effectively Freeware. Licenses such as [Creative Commons Attribution-NonCommercial (CC-BY-NC)](https://creativecommons.org/licenses/by-nc/2.0/) are inherently non-commercial and therefore not open source.

Another common example is public domain software. This refers to software not protected by copyright and therefore usable by the public without requiring a license. For example, “All of the code and documentation in this software has been dedicated to the public domain by the authors.” A public domain declaration attempts to waive or eliminate any intellectual property rights. It may be accompanied by other terms like warranty disclaimers. Many countries (e.g. the UK) do not allow the software to be dedicated to the public domain, so the legal effect of such dedication is questionable. The [Creative Commons CC0](https://creativecommons.org/share-your-work/public-domain/cc0/) declaration contains a very broad license designed to avoid this issue in countries where public domain dedication is not possible. Having said that, it seems very unlikely that any copyright holder who has declared that they have abandoned their rights in the software will later try to assert those rights. In practice, public domain software is generally treated as open source software with an extremely permissive license.

## Notices

Notices, such as text in comments in file headers, often provide authorship and licensing information. They are generally regarded as best practices for ensuring the clarity and sustainability of codebases. 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.

Some examples of notices are copyright notices, license notices, attribution notices and modification notices. An example of a copyright notice is something that says “Copyright, Year, Author Name.” An example of a license notice is a statement identifying the license being used. An example of an attribution notice is a statement identifying the authors of the software. An example of a modification notice is a statement explaining what has changed.

All of these statements are usually made visible in the software documentation. In the current digital world, they are often accessed via a menu under “licensing terms” or “legal notices” or similar. In the past, such notices may have been supplied in physical user manuals provided with consumer electronics and so on.

## Permissive Open Source Licenses

What is generally termed permissive open source licenses have minimal conditions that users must follow. For example, a permissive license might allow unlimited use and redistribution of software for any purpose as long as the copyright notices and the license terms are made visible to existing and future users.

There are quite a few permissive licenses with slightly different terms. The most common include the BSD three-clause license, the MIT license and the Apache 2.0 License.

## License Reciprocity & Copyleft Licenses

Some software licenses contain more terms than others. For example, you will often hear about “Copyleft” licenses when people talk about open source software. Copyleft is a domain-specific term that can also be referred to as “license reciprocity.” These licenses require any derivative works to be distributed under the same license as the original software which may include making the source code available under that license.

In the context of open source software, the term “derivative work” often refers to software based directly on the source code, software in the same source code file as the original, or in the same program. It is important to understand how a specific license defines derivative work to ensure compliance with that license.

Examples of Copyleft open source licenses include the GPL, the LGPL, the AGPL, the MPL, and CDDL. The Creative Commons CC-BY-SA (Attribution-Share-Alike) has a similar effect: “sharealike” is the Creative Commons term for copyleft. Each has subtly different terms, but all include the concept of derivative works needing to remain under the same licensing terms as the original software.

## Software License Compatibility

Software license compatibility allows two or more different pieces of software under different licenses to be combined into one new derivative work. This is not possible with all types of software licenses or even all types of open source software licenses. It is therefore important to have a basic understanding of what allows or disallows software license compatibility.

Software license compatibility is determined by whether the relevant software licenses have conflicting requirements. For example, if one license says “you can share this code” and the other license says “you cannot share this code,” these are conflicting license requirements, and you cannot combine these two pieces of software into one new derivative work, and then distribute them (Generally, since open source license restrictions tend to apply only when the software is distributed, compatibility only becomes a major problem when the software is intended to be distributed. If the software is intended for internal use only, and never distributed, this is likely to be less of a problem).

In a more practical way, the GPL-2.0 license and the EPL-1.0 open source licenses both allow derivative works. However, they both require that their respective derivative works, if distributed, are licensed only under the original license. This makes it impossible to distribute a derivative work of GPL-2.0 software licensed under the EPL-1.0, and vice versa. These two licenses are therefore similar but incompatible.

## Multi-Licensing of software artifacts

The concept of Multi-licensing refers to the practice of distributing software under two or more different sets of terms and conditions simultaneously, where the recipient can choose which license to apply for. This adds flexibility in choosing a convenient license for the user based on their purpose.

Example:., when software is “dual licensed,” the copyright owner gives each recipient the choice of two licenses i.e., A piece of software can be released under MIT license as well as GPL v2.0 simultaneously, and here we have a choice of complying with only one of the licenses from the choice given based on the interest and purpose of the individual user or organisation.

# Knowledge Check

## Knowledge Check 2.1

* What is an Open Source license?
  + It makes software available under terms that allow for use, modification and redistribution. --> Correct answer -- Reason: An Open Source license gives everyone the ability to modify and redistribute open source software.
  + It is a template to apply conditions on proprietary products.
  + Open source license is a freely available license.
* What are the typical obligations of a permissive Open Source license?
  + An Open Source license typically contains no obligations
  + An open source license that does not ask you to oblige to fulfill any terms and conditions.
  + An Open Source license typically requires a copyright notice and warranty disclaimer to be included with software --> Correct answer -- Reason: An Open Source license acknowledges ownership of the software but also typically disclaims warranty as a consequence of unrestricted distribution.
* Name some permissive Open Source licenses.
  + The General Public License (GPL) and the Affero General Public License (AGPL).
  + Eclipse public license Version 1.0
  + Unknown licenses
  + The MIT and BSD licenses. --> Correct answer -- Reason: These licenses - unlike the GPL or AGPL, do not require that the freedoms they provide to use, modify, share and improve are transferred to other parties on further distribution.

CHAPTER 3: Introduction to Open Source Compliance

# Introduction

## Chapter Overview

This chapter explains how compliance with open source licenses works. It is intended to add the “how” to the “what” discussed in the last chapter. It is important to understand how this works to ensure your organization avoids unnecessary mistakes in the adoption, use and deployment of software made from or containing Open Source. This chapter does not require a legal background. While it discusses some concepts from the law, it does this in a non-technical way.

## Learning Objectives

By the end of this chapter, you should be able to understand the objective of compliance, the basics of Open Source compliance and common compliance issues.

# Open Source Compliance Basics

## Open Source Compliance Goals

**Know what you are using.** You should have a process for identifying and tracking Open Source components that are present in your software. For example, your policy may require that no software component is incorporated into your software unless you have identified where it has come from, identified the license applicable to it, determined whether that license is acceptable to use, and recorded all of this information.

**Know how you are using it.** You should have a process that identifies how you are using each Open Source component and/or a policy covering how you are allowed to use it. The conditions within each open source license may vary depending on the use case, with an example being that using code licensed under GPL-2.0 on the server-side of a SaaS (software as service) application is not distributed and there is no requirement to make the source code available to people making use of the service. However, if you take the same code and distribute it to customers so they can run their own SaaS instance then you would need to make the source code available to them. Typical use cases which could be recorded are

* Internal use only, with no distribution to or use by third parties
* Sharing the application source code with third parties (such as partners) who build the application themselves
* Distributing the application as a package or binary with third parties (such as customers)
* Making the functionality of the application available to customers on a SaaS basis (while running the application itself on servers under your own control

There may be other use cases that are applicable to your circumstances and which are impacted by some of the licenses to software which you are using (especially if those licenses are not open source, but are proprietary licenses with some open source characteristics).

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

## Open Source Compliance Program

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

1. Facilitate effective usage of Open Source in their products and projects (commercial or otherwise). There is a huge range of high-quality, well supported Open Source code available. A well-considered Open Source Compliance Program will make it as easy as possible for developers to select the most appropriate of these components, ensure that they are available under licenses that are suitable for the organisation's chosen use case, understand clearly the basis on which they may be used, and encourage involvement with relevant open source projects and the project’s community.
2. Contribute to and participate in Open Source communities. This involvement will facilitate access to support and assistance, lead to further development of and bug fixes to the code which can benefit all users, and ultimately, give the participants an opportunity to influence the future development of the project.

## Implementing Compliance Practices

Compliance practices consist of more than preparing a policy and processes. It is also necessary to ensure that the policy and processes are appropriate for the organisation (or business unit), and, crucially, have management buy-in, and are supported by sufficient staff and resources to handle:

•Identification of the origin and applicable license of all software sourced from outside the business, and any constraints and obligations applicable to software that has been developed internally.

•**Tracking Open Source software within the development process.** For example, components used in a specific project may be test suites, or tools, code from which does not ultimately form part of a binary that is shipped. Depending on the use case, you may need to provide this code (possibly in the source) depending on the licenses of other components which are incorporated into the final shipping binary. It is important to understand which components are contained in a product or service that gets shipped, and this involves various decisions about monitoring the development cycle and for what purposes. This includes recording whether any open source code is modified and whether any code snippets are incorporated in custom code, and from where. It is also important to record which version of a component is used, as sometimes the license varies from version to version of the same component.

•**Performing Open Source practices review and identifying license obligations.** Regular review of your practices and procedures, ideally with tests of the processes, to confirm they are effective and allow you to refine them.

•**Fulfillment of license obligations when the product ships.** The conditions in most open source licenses apply at the point of distribution. Accordingly, it is necessary to ensure that those obligations are fulfilled (such as the obligation to retain attribution notices or the offer to provide source). Sometimes, these obligations will apply in other cases (for example, some licenses apply obligations when the functionality of the software is made available to third parties over the internet, even if the software itself is not distributed).

•**Oversight for Open Source Compliance Program, creation of policy, and compliance decisions.** Alongside the necessary for regular review we mention above, it is equally necessary to ensure that there is effective oversight of the operation of the policy, and a mechanism to enable queries about code usage and incorporation (both from inside and outside the company) to be handled. Any decisions should be recorded, both so as to streamline the resolution of similar queries in the future, but also so that the overall effectiveness of the compliance program can be checked and improved over time.

•**Training** for a program to be effective, participants should be appropriately trained. The training should be recorded and assessed, and there is no necessity for everyone to have the same training: training should be tailored to each relevant role.

## Compliance Benefits

Benefits of a robust Open Source Compliance program include:

1. Increased understanding of the benefits of Open Source and how it impacts your organization
2. Increased understanding of the costs and risks associated with using Open Source
3. Increased knowledge of available Open Source solutions
4. Reduction and management of legal risk
5. Increased respect of Open Source developers/owners’ licensing choices
6. Fostering relationships with the Open Source community and Open Source organizations

## What Compliance Obligations Must Be Satisfied?

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

1. **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.
2. **Notice of Amendment**. You may need to record and notify recipients of the code if it has been modified from the original version you received. The extent to which the changes have to be notified varies from license to license, from a simple requirement to note that changes have been made, to ensuring that the changelog of the software has to be maintained in the software package.
3. **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. These obligations may continue even after the production and sale of the product or service is complete.
4. **Medium of source code availability.** You may also need to provide source code for the Open Source software and related modifications. i.e., either by upstreaming (uploading) the code to the original open source software repository (complying under terms and conditions of the repository for upload or contribution). or if distributing to other users, by providing a channel (e.g. downloadable URL or attached file of the code) from where the code can be obtained.
5. **Reciprocity.** You may need to ensure that modified versions or derivative works are made available and released under the same license that governs the Open Source component.
6. **Other terms.** The Open Source license may have other obligations or conditions. and these can be terms that may touch upon topics like patent grants to you or from you andthe Open Source license may restrict the 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 that may occur during the distribution of software.

For some Open Source licenses, access via a computer network can   
“trigger” license conditions, e.g. in the case of applications downloaded to a user’s machine or mobile device and JavaScript, web client, or other code that is downloaded to the user’s machine or browser. For example, all versions of the Affero General Public License require providing modified source code if “users interacting with it remotely through a computer network.”

## Open Source Compliance Issues may also occur with untracked modification

An example is that changes to the existing program (e.g., additions, deletions of code in a file, combining components together) may be subject to additional obligations upon distribution such as providing a notice of modification or providing the accompanying source code.

# Knowledge Check

## Knowledge Check 3.1

* What does Open Source compliance mean?
  + Open Source compliance means following the terms of Open Source licenses. --> Correct answer -- Reason: The concept of compliance means adhering to something, and in the case of Open Source this means adhering to the terms of the applicable licenses.
  + Open Source compliance means using a license approved by a third-party.
  + Open source compliance means development of open source software in community approved steps.
* The important practices of an Open Source Compliance Program include
  + - Identification of Open Source software
    - Tracking Open Source software within the development process
    - Identifying license obligations
    - Fulfilling license obligations when a product ships
    - Creation of a policy to manage processes
    - Having a training program
  + Only some of the above are applicable
  + All of the above are applicable --> Correct answer -- Reason: Managing Open Source license compliance is an end-to-end activity in the same manner as managing proprietary software licenses. It requires processes, policy and training.
* What are some benefits of an Open Source Compliance Program?
  + It helps prevent potential errors around the use of third-party intellectual property. --> Correct answer -- Reason: Open Source license compliance ensures that you know how, when and under what terms you are relying on third-party code, and under what conditions you are doing so.
  + It guarantees that all code in a company's project is free of bugs and security issues.
  + It yields us monetary profit on the software developed.

CHAPTER 4: Key Concepts for Code Building and Distribution

# Introduction

## Chapter Overview

This chapter explains specific software development activities that occur when using open source (or any other software). It is important to understand these general software development activities to plan, prepare and deploy processes to effectively manage open source licensing because the way you use the open source may impact which obligations you have to meet. This chapter does not require a software engineering background. While it discusses some concepts from software engineering, it does this in a non-technical way.

## Learning Objectives

By the end of this chapter, you should be able to understand different possible ways to integrate Open Source within a product and what that means for the distribution of the product.

# Key Concepts for Code Building and Distribution

## How do you want to use an Open Source component?

When creating software out of small software components the team needs to make decisions about their approach. Sometimes they need to decide what parts of the software will be proprietary and what parts will be open source. They also need to make decisions about specific actions undertaken in the development process. These decisions can impact what licenses can be used or which open source components you would consider for different parts of the software, different product offerings and so on.

Incorporating open source with proprietary software can be done in several ways. Depending upon the license of the open source, the means of incorporation can have an impact on the obligations required to be satisfied in order to comply with the license.

## The common scenarios are:

* Incorporation of software
* Linking of software
* Modification of software

## Incorporation

Incorporation is a situation where portions of one software component are copied into another software component or product. In our illustration below we are visualizing a situation when a developer copies portions of an open source component (green) into your software product (blue).

There are various terms that describe different forms or activities around incorporation. It is useful to be aware of these terms to ensure you understand briefings and decisions about this activity.

•Integrating (putting software into other software)

•Merging (combining two pieces of software)

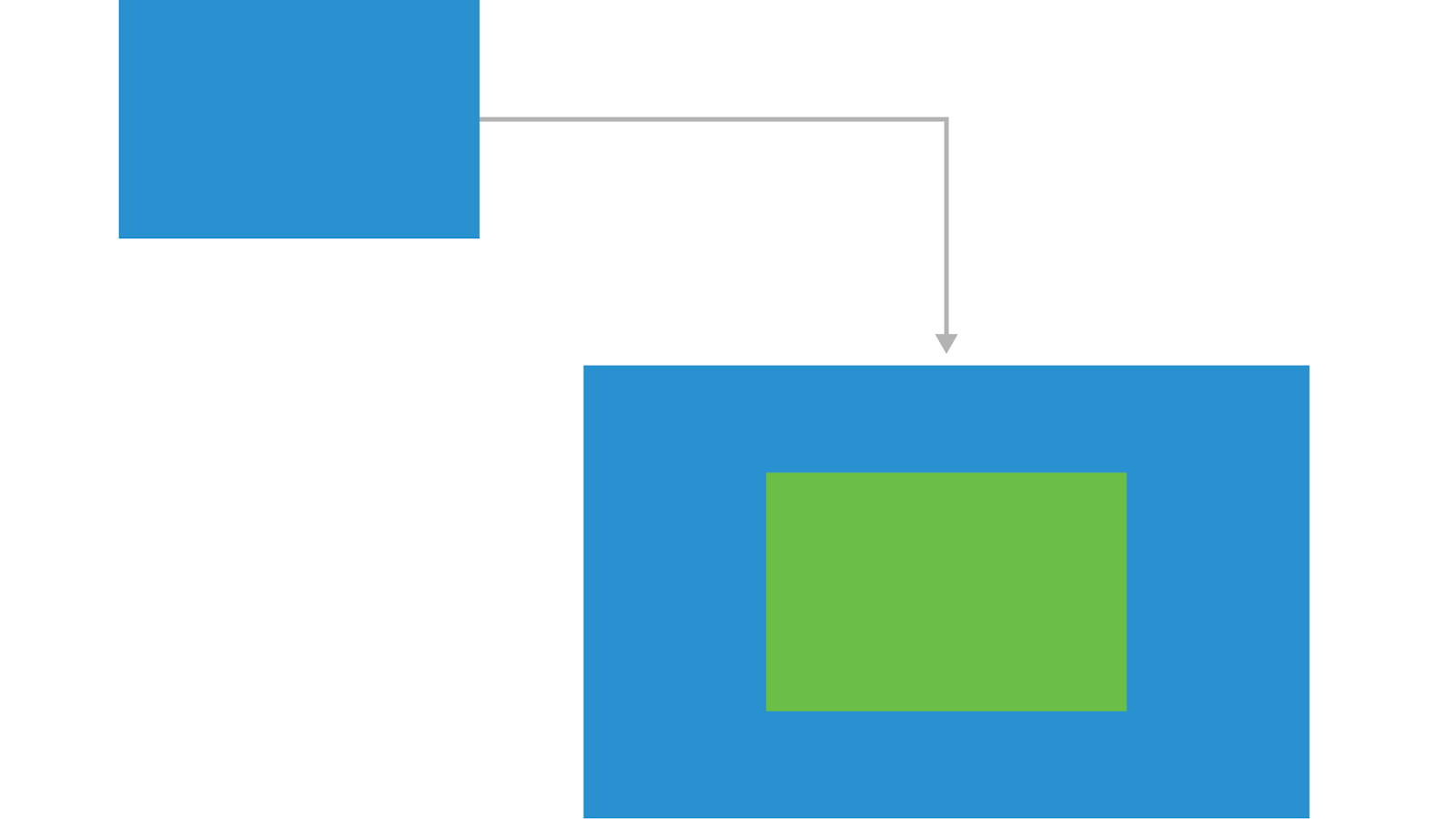
•Pasting (copying and pasting parts of one software into another)

•Adapting (adjusting part or the whole of one piece of software with or without external code)

•Inserting (this is another way of talking about integrating)

•Embedding (this is another way of talking about integrating)

**Incorporation:**



E.g. for if the code incorporated is under a GPL license, under that license the entire resulting code (green and blue) would have to be made available under the GPL license as a “derivative works” when the code is distributed.

## Linking

Linking is a situation where two different software components are connected in two specific manners. In our illustration below we are visualizing a situation where a developer links one open source component (green) and one proprietary component (blue).

There are various terms that describe different forms or activities around linking. It is useful to be aware of these terms to ensure you understand briefings and decisions about this activity. Linking is particularly important to consider in the context of copyleft open source licenses.

* Static Linking (incorporated at build time)
* Dynamic Linking (sought out at runtime)
* Pairing (this is another way of talking about either static or dynamic linking)
* Combining (this is another way of talking about either static or dynamic linking)
* Utilizing (this is another way of talking about either static or dynamic linking)
* Packaging (this is another way of talking about either static or dynamic linking)
* Creating interdependency (this is another way of talking about either static or dynamic linking)

**Linking:**



## Modification

Modification is a situation where a software component is changed, improved or otherwise edited. In our illustration below we are visualizing a situation where a developer adds a software component (blue) to an existing codebase. The outcome is a new software component influenced by both the original software component and the developer's contribution (green).

There are various terms that describe different forms of modification. It is useful to be aware of these terms to ensure you understand briefings and decisions about this activity.

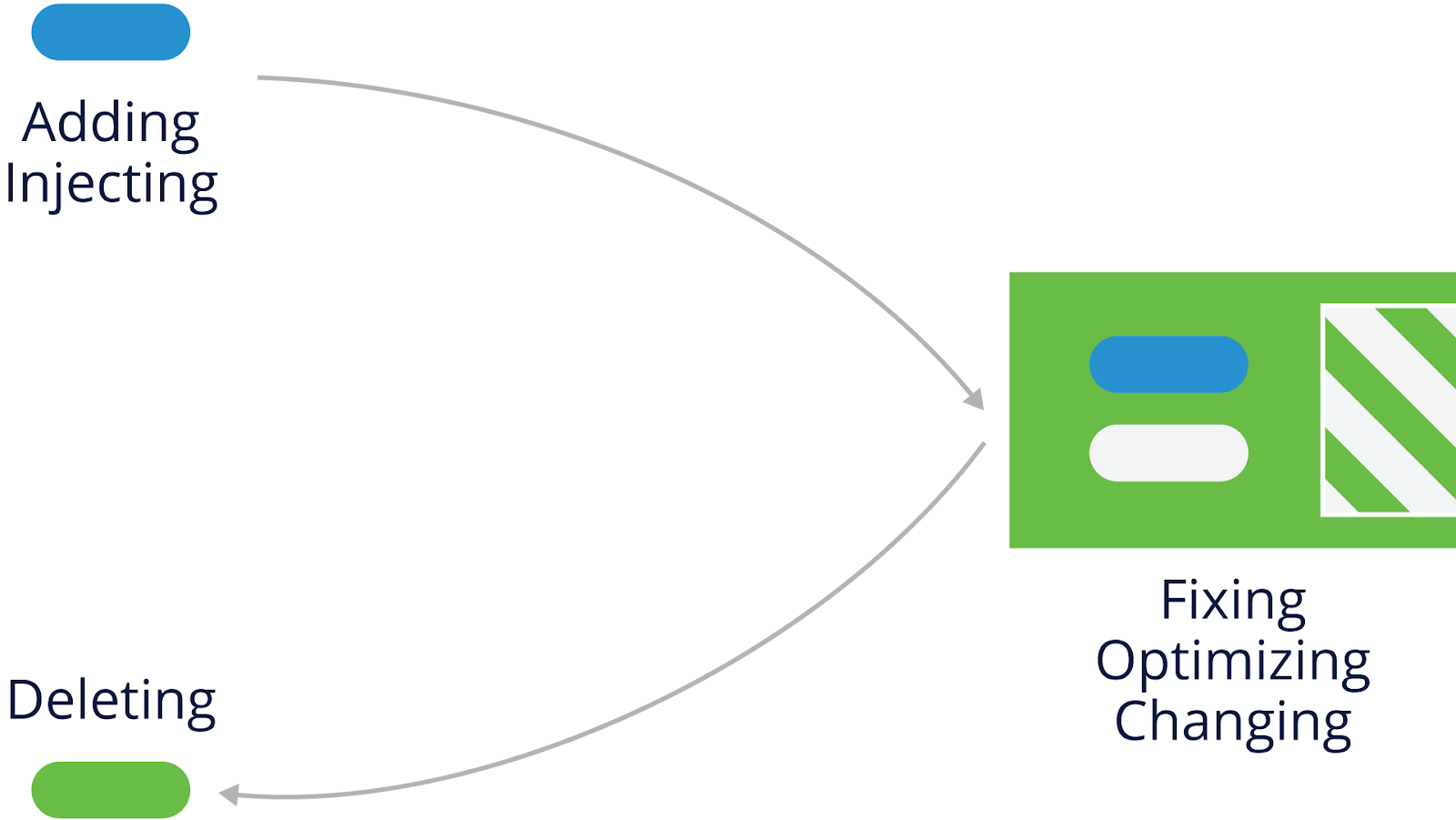
•Improving (an alteration is made to enhance a software component)

•Bug fixing (this is another way of talking about improving)

•Removing technical debt (this is another way of talking about improving)

•Deletion (this is another way of talking about optimization)

**Modification:**



E.g. Many licenses require that modifications are documented (which is good software engineering practice to document in comments anyway!). Some licenses have more burdensome obligations such as the MPL where a modified code source needs to be made available.

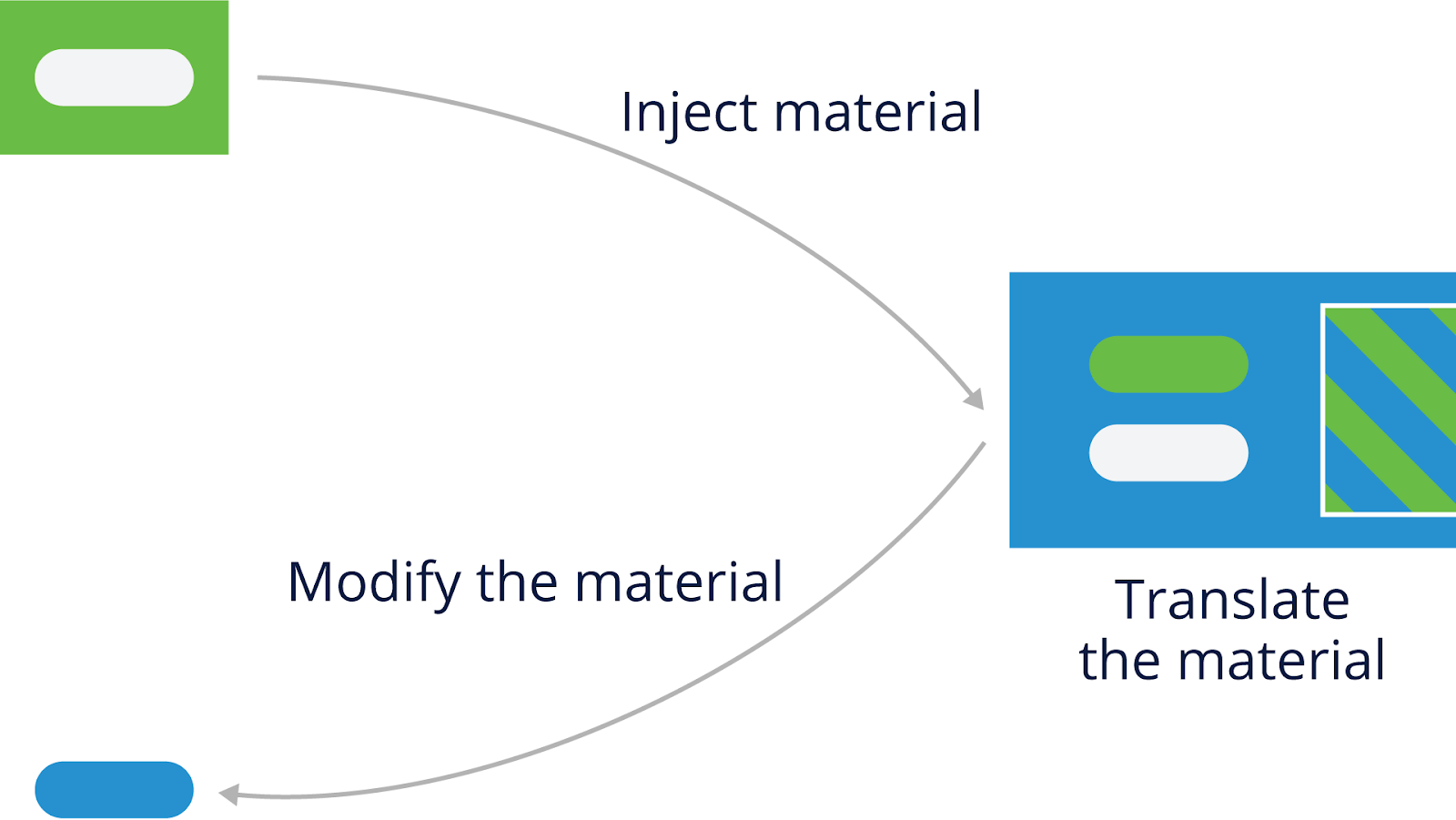
## Development Tools

All of the activities we have discussed above are undertaken by developers. However, it is also important to understand that software development is a process facilitated by tools. These are referred to as "development tools" or "development environments" and provide a level of automation to ensure efficiency. The consequences of using development tools can include more than manipulation of software components. For example, in some cases, a development tool may inject material into a software component to provide an outcome. It is important to be aware of this and ask questions around this to ensure developers and managers know what the final software component contains. Two common examples are:

1. A tool may silently pull in dependencies with license requirements that you need to know about. Usually, the tool will have an option for outputting dependencies used.

2. There may be a silent injection of code from the tool itself under an open source license. This is often under the same license or a more permissive license but should be tracked regardless.

**Development Tools Facilitating A Software Development Activity:**



## How is the Open Source component distributed?

Finally, an important consideration for an open source compliance review relates to the end product, the software component deliverable, and who will receive it. Different parties receiving the deliverable will have different rights and requirements according to the open source licenses.

This can be broken broadly into two categories: people inside your legal entity and people outside your legal entity. Open source licenses use copyright as their fundamental building block, and copyright is applicable at the moment of distribution.

If you are giving the final software component to people inside your legal entity you are not regarded as distributing the software. You are effectively giving it to yourself. This means that the terms applied by the open source license will not be activated.

If you are giving the final software component to people outside your legal entity (including other group companies with a different legal entity), you are distributing the software. This means that the terms applied by the open source license will be activated.

The format of distribution may also be impacted by the open source licenses. It is important to ask:

1. Who is the software component distributed to?
2. How is the software component distributed?
3. Is source code included or offered?
4. Have we documented this?
5. Have we archived the software for future reference?

The concepts discussed above will be important while deriving license obligations for the open source components we use. It helps us understand how an open source license allows us to use the respective component and be bundled with our proprietary project.

# Knowledge Check

## Knowledge Check 4.1

* What is incorporation?
  + Incorporation is making copies of documentation explaining Open Source functionality.
  + Incorporation is acknowledgement of using open source software in proprietary products.
  + Incorporation is copying portions of Open Source components into your software product. --> Correct answer -- Reason: Incorporation is about including something, and in the case of Open Source compliance it is about including open source software in your internally developed software.
* What is linking?
  + Linking is another term for the compilation of software.
  + Linking is CI/CD execution during build.
  + Linking is when you join an Open Source component with your software product. --> Correct answer -- Reason: Linking is about connecting two things, and in the case of Open Source compliance, it is about linking open source software with your internally developed software.
* What are development tools?
  + In the context of Open Source compliance, development tools are the computers and other devices used by engineers to create software.
  + IDE, version control repositories are the build tools in the context of this chapter.
  + In the context of Open Source compliance, development tools are software programs that provide an environment to support the writing of software. --> Correct answer -- Reason: Development tools range from code editors to automation systems that help support the creation or modification of software.

Chapter 5: Bringing Things Together

# Introduction

## Chapter Overview

This chapter explains how knowledge of intellectual property, Open Source licensing and the practical use of Open Source comes together as Open Source compliance. It is intended to complete our brief look at Intellectual Property, software licensing, Open Source license and how it applies to your organization. It is important to understand how this works to ensure your organization gets maximum benefit from the use of Open Source whilst avoiding risks of non-compliance.

## Learning Objectives

By the end of this chapter, you should be able to understand the overall meaning and benefits of having an Open Source compliance program and/or policy, and some of the methods to do this in practice.

# What are the key takeaways and challenges learnt from the previous chapters?

Key takeaways:

* We learnt the basics of copyright and the rights and obligations it confers to software.
* We had an introduction to open source licenses and how they are structured.
* We gained an understanding of the key items needed for open source compliance.

Challenges:

* To understand how licenses work and how this impacts organizational decisions.
* To understand what situations invoke licenses and decisions around licenses.
* To ensure open source compliance in an organisation or within a project.

Why do we need Open Source compliance?

A good mental model to apply to this topic is the concept of supply chains. Every organization has external and internal supply chains. Software (and everything else) comes from a third-party supplier. It is ingested into an organization and moves through the organization in various internal or external products and services. Finally, when software is brought into an organization and when it is distributed outside the organization there are considerations regarding intellectual property. Tracking the supply chain is key.

Open Source license compliance is about addressing these requirements from the perspective of open source license obligations. Failure to do this can have negative implications for M&A due diligence, legal action, injunctions and bad publicity.

How do we address Open Source compliance in practice?

Open Source compliance is addressed by introducing processes, policy and training that captures the key points around Open Source obligations and grants. These can be as simple or as sophisticated as required by an organization. The starting point is to have an Open Source review process.

The Open Source review process is where a company can analyze the Open Source software it uses, or plans to use, and understand its rights and obligation s. Part of it is a review for usefulness and quality, and part of it is a review of the rights and obligations associated with the use of the selected components. The outcome should be guidance compatible with company policy and business objectives to achieve compliance with license obligations for the open source components used.

It is beyond the scope of this course to delve into the details of Open Source reviews and the organizational structures that support them. However, there are a couple of simple, important points to remember when addressing open source compliance management:

1. Many companies create an [Open Source Program Office (OSPO)](https://github.com/todogroup/ospodefinition.org) to centralize their management of Open Source as a whole, including Open Source compliance. This may have assigned personnel or it may be a virtual office. In smaller organisations, it might be a single person with compliance as a part-time responsibility. However, the OSPO usually includes representatives from both the engineering and legal departments.
2. There are plenty of community resources available to support the management of Open Source compliance. Two important examples are the [OpenChain Project](https://www.openchainproject.org/) (steward of ISO/IEC 5230:2020, the international standard for Open Source license compliance) and the [SPDX Project](https://spdx.dev/) (steward of ISO/IEC 5962:2021, an international standard for [Software Bill of Materials](https://en.wikipedia.org/wiki/Software_bill_of_materials)).

# Knowledge Check

## Knowledge Check 5.1

* What are some challenges associated with Open Source license compliance?
  + Understanding how licenses work and what situations invoke the licenses. --> Correct answer -- Reason: Open Source licensing provides great flexibility around the use of software but it does come with certain terms that must be followed.
  + Understanding how software functionality can be improved.
  + Understanding how to validate the quality of software.
* What is the purpose of an Open Source license review?
  + An Open Source license review focuses on determining the quality of third-party software under an open source license.
  + Open Source license review helps us identify the right copyright applicable on the open source package in use.
  + An Open Source license review focuses on the rights and obligations provided by third-party software under an open source license. --> Correct answer -- Reason: Open Source licenses provide rights and obligations that determine how you can use the software.
* How can companies approach an Open Source review?
  + By engaging with community resources such as projects associated with Open Source compliance. --> Correct answer -- Reason: Open Source license compliance is a mature field and processes, policies and training material are freely shared among companies to ensure effective supply chains.
  + By attempting to read and interpret Open Source licenses without reference to third-parties.
  + By raising a join request in the github repositories.

Final Exam

Copyright protection for software applies at which point in time? (TRUE/FALSE - PLEASE ADD FALSE QUESTIONS IN RED)

* Automatically on creation.
* When a © Copyright statement is added by the Author.
* When a © Copyright statement is added by a Company.

A blog post that includes some example code is automatically public domain? (TRUE/FALSE - PLEASE ADD FALSE QUESTIONS IN RED)

* FALSE
* TRUE

What are software licenses? (TRUE/FALSE - PLEASE ADD FALSE QUESTIONS IN RED)

* They are documents describing rights and obligations associated with the distributed software.
* Documents that give you the right to use the software part of your proprietary product
* Documents that explain how to sell the software to others.

What do open source licenses allow? (TRUE/FALSE - PLEASE ADD FALSE QUESTIONS IN RED)

* They allow users to use, study, share and improve the software.
* Incorporate copyleft open source code into proprietary code and keep a proprietary license.
* Use of the software royalty-free regardless of any patents the software implements

Identify Open Source compliance goals?   
(MULTIPLE CHOICE - PLEASE ADD FALSE QUESTIONS IN RED)

* Know what you are using.
* Know how you are using it.
* Satisfy license obligations.
* Make sure that there are no bugs in the code.
* Ensuring that all known vulnerabilities are fixed.

When should an Open Source compliance review be done? (TRUE/FALSE - PLEASE ADD FALSE QUESTIONS IN RED)

* Prior to distributing a product.
* Or During software development process.
* When a customer requests.
* If a license troll sues the company for license infringement.

## What are common scenarios for Code Building and Distribution? (MULTIPLE CHOICE - PLEASE ADD FALSE QUESTIONS IN RED)

* + Incorporation of software.
  + Linking of software.
  + Modification of software.
  + Selling software on a USB stick.
  + Making software available to download.

What is static linking? (TRUE/FALSE - PLEASE ADD FALSE QUESTIONS IN RED)

* Static linking is when a software component is incorporated at build time.
* Adding a software library to the same read-only DVD for distribution.
* Linking of Operating system libraries during execution of the application.

What questions are important to ask around the distribution of software? (MULTIPLE CHOICE - PLEASE ADD FALSE QUESTIONS IN RED)

* Who is the software component distributed to?
* How is the software component distributed?
* Is source code included or offered?
* Have we documented all the compliance activities?
* Have we archived the software for future reference?
* To which countries the software is distributed?
* How big is the distributed executable?

What are some of the key challenges organizations face with software license compliance? (MULTIPLE CHOICE - PLEASE ADD FALSE QUESTIONS IN RED)

* To understand how licenses work and how this impacts organizational decisions.
* To understand what situations invoke licenses and decisions around licenses.
* To ensure open source compliance in an organisation or within a project.
* To understand who outside the organization is responsible for license compliance?