NGI Zero PET and Discovery Legal To-Dos

Free and Open Source Licensing
Frequently Asked Questions

Table of Contents

CHOOSING A NAME FOR YOUR PROJECT

Select a good name for your project

When a user is searching for software to solve a problem they have, the first thing that they will encounter when they come across your project is its name. It is therefore important that you come up with a good name for your project.

An unfortunately named project can slow down its adoption if people do not take the name seriously, or if users have trouble remembering it. Some things to consider could be to give a name that would give users an idea of the project, or is related to what the project is used for, as well as one that would be easy to remember.

Make sure your project name is not already trademarked

Most importantly, it is absolutely essential that the name you choose is not the same as the name of some other project, and does not infringe upon any trademarks. The unauthorized use of a trademark (or a substantially similar mark) may result in a trademark infringement lawsuit being brought against you.

Finally, you might wish to consider registering your project's name as a trademark with the European Union Intellectual Property Office (the "EUIPO"). While it can be costly to do so, it can ensure that your project name cannot be used by another person or entity for their project or business. Another option would also be to register the name of the project as a business name in your local companies register.

Summary

The goal here is to choose a project name that will not infringe on the existing rights of another. In other words, a name that will not be confused with another entity's name for their software or related goods and services.

TRADEMARKS

What is a trademark?

A trademark is a symbol used by a trader to distinguish their goods or services from those of their competitors. By recognizing the trademark, users or customers can identify which goods or services offered on the market originate from a particular trader.

Names and logos are the most common symbols used as trademarks.

Why can't I use a name that is already trademarked?

Using a name or logo that is the same or substantially similar to an existing trademark is known as trademark infringement. Trademark infringement causes consumers to think a product or service is provided by a particular company, when it actually is not.

Laws against trademark infringement are put in place to protect both consumers and trademark owners against this type of confusing use of a trademark. They also serve to protect the trademark holder from potential damage to their reputation that can arise due to a difference in quality of the goods or services provided by the infringer.

What are the consequences of trademark infringement?

Facing a lawsuit arising from trademark infringement would be costly, both in terms of payment of monetary damages to the original trademark holder, as well as the possible payment of court costs and attorney fees. Additionally, you would still need to come up with a new name at the end of the process.

It is therefore vital that you check if the name you intend to use for your project is not already taken, or substantially similar to, a registered trademark.

How do I check if my potential project name is already trademarked?

You can run a search for trademarks that could conflict with the name you have chosen with the EUIPO here[1]. It does not cost anything to do a search on this website.

Additionally, you can conduct online search engine searches, look in public source code repositories, and check domain names to see if there are any others who are using the same name as your project.

---

1   https://euipo.europa.eu/ohimportal/en/search-availability

One useful tool that you can also use is the **"whohas" tool**[2].

What do I get out of registering my project name as a trademark?

A European Union trademark will grant you the exclusive rights in all current and future Member States of the European Union. You can obtain these rights through a single registration. The trademark is valid for 10 years, and can be renewed indefinitely, for 10 years at a time.

Who will own the trademark if I register my project name?

Trademark rights can only be owned by a person or entity who is capable of owning property. This is different from ownership of copyright in the code, which can be co-owned by community members who have contributed to the code. Because only the owner of the project name may file for trademark registration, you must decide on who (whether an individual or an incorporated entity) will own the trademark rights.

Here are some approaches you can take to decide who assumes ownership of your project's trademark rights:
   a) The person who created the name remains the owner of the trademark, or transfers ownership to another individual;
   b) A group of people, acting as partners or an unincorporated association owns the trademark; or
   c) The person who created the name transfers ownership to an organization, society, or non-profit corporation.

It is best to outline an approach to the project name and any trademark rights, to avoid any potential conflict in the future. For example, the owner of a trademark can revoke the right of the project to use that trademark.

How do I register my project name as a trademark?

You can apply online for a European Union trademark online here[3]. The EUIPO's online application form will guide you through the application process. Registering a trademark will cost between €850 and €1500.

Does the software license cover my trademark?

The idea of a trademark is to inform users/customers about the origin of your software. Therefore, it does not make really sense to license a trademark logo under a software license.

---

2   Read more: http://www.philippwesche.org/200811/whohas/intro.html
3   Read more: https://euipo.europa.eu/ohimportal/en/apply-now

In order to tell users/customers what they can do with your trademark, the best way to do it is through a trademark policy.

With a trademark policy you can explain which icons or logo are your trademarks. Besides, you can state that they are not covered by the project license, and icon or logo can be used only in compliance with the trademark policy. Don't forget to make clear how you can compile the software without using the trademark!

What is a trademark policy? How I make one?

Since your software license is not the best way to protect a trademark, with a trademark policy you can explain in simple words what is permitted and not to do with your trademark.

If your project has trademarks consider having a trademark policy with the following guidelines:

1. Inform which icon or logo is your trademark and state that they are not covered by the project licenses. The use of trademark should be accordingly to this policy only.
2. Explain in simple words what is permitted to do with your trademark.
3. Make clear when users need specific permission to use your trademark.
4. When allowed, explain how should users display your trademark.
5. **Make sure how users can compile the software without using the trademark!**

Summary

Trademarks are important legal instruments for FOSS projects. A trademark represents the source of origin of a product, it also represents the quality of goods associated with the source. However, trademark should not be mistaken with project's name. Many FOSS projects set written guidelines for trademark use.

FREE SOFTWARE AND LICENSES

The Basics

What is Free Software?

The term "Free Software" refers to software that enables its users to maintain their control and freedom over how to use such software. More specifically, the following four essential freedoms define Free Software:

- **Freedom to Use**

  Placing restrictions on the use of Free Software, such as time ("30 days trial period", "license expires January 1st, 2004"), purpose ("permission granted for research and non-commercial use", "may not be used for benchmarking"), or geographic area ("must not be used in country X") makes a program non-free.

- **Freedom to Study**

  Placing legal or practical restrictions on the comprehension or modification of a program, such as mandatory purchase of special licenses, signing of a Non-Disclosure-Agreement (NDA) or - for programming languages that have multiple forms or representation - making the preferred human way of comprehending and editing a program ("source code") inaccessible also makes it proprietary (non-free). Without the freedom to modify a program, people will remain at the mercy of a single vendor.

- **Freedom to Share**

  Software can be copied/distributed at virtually no cost. If you are not allowed to give a program to a person in need, that makes a program non-free. This can be done for a charge, if you so choose.

- **Freedom to Improve**

  Not everyone is an equally good programmer in all fields. Some people don't know how to program at all. This freedom allows those who do not have the time or skills to solve a problem to indirectly access the freedom to modify. This can be done for a charge.

These freedoms are rights, not obligations, although respecting those freedoms for society may at times oblige the individual. Any person can choose to not make use of them but may also choose to make use of all of them.

Free Software does not exclude commercial use. If a program fails to allow commercial use and commercial distribution, it is not Free Software. A growing number of companies base their business model completely, or at least partially on Free Software, including some of the largest proprietary software vendors. Free Software makes it legal to provide help and assistance; it does not make it mandatory to do so.

Free software is therefore a matter of liberty, not of price; the word "free" in Free Software means "free" as in "free speech", not as in "free beer".

What is proprietary software?

Proprietary software is the opposite of Free Software. This term is used to refer to software that is distributed under restrictions that prevent users from enjoying the four essential freedoms. Even software that is distributed gratis can be considered proprietary, if its restricts users from any of the four essential freedoms.

The term "commercial software" is sometimes used carelessly to refer to proprietary software. This would be incorrect: as explained below, Free Software does not exclude commercial use.

How can Free Software be used commercially?

Free Software can be sold, as long as the buyers are not restricted from sharing copies of the software after their purchase. Free Software can also be commercialized in other ways, such as by selling support, services, and certification. Free Software remains free as long as the four essential freedoms outlined above are afforded to users.

Why is it important that your project uses/produces Free Software?

Software is the most important cultural technology of the 21st Century; it is almost impossible to imagine daily life today without it. When others control a tool as important to society as software, they can exert great influence over our lives and actions.

For example, whoever controls the search engines that we use has the power to determine what we find online; whoever controls the software on which our online transactions are run can have access to our personal data. It would be dangerous to democracy if the critical social instrument that software constitutes were controlled by only a small group.

The freedoms in Free Software to use, study, share, and improve can therefore be applied to hand control of the future of software back to the people. In line with the Next Generation Internet mission of creating an internet that respects human and societal values, having your project use and be Free Software would help to foster an open and transparent internet; one where users are able to share and improve the tools that it provides.

Software Licensing

What is a software license?

Generally, a license is an authorization to use, release, or distribute someone else's property. This includes intellectual property such as a piece of text, a song, or software.

Software licenses tell people how the rights holder (usually the author of the software) wants the software to be used and what freedoms, or restrictions, the software has. Without a license, many uses of the software may be prohibited.

The absence of a license does not make a software free to use. Software is copyrighted by default, so unless it has been explicitly and validly placed in the public domain, using code without a license may be considered copyright infringement.

Are there different types of software licenses?

Yes. Software licenses can be classified according to the rights granted or retained by the copyright owner. When the owner of a software waives all the rights, the software is in the Public Domain. If a software is under a proprietary license, only a few rights are granted; for example, the right to use program but not to distribute copies of it. If the license grants the four essential freedoms to use, study, share and improve the software it is a Free Software license[4].

The following diagram illustrates the type of licenses and where they would fall on how much rights they would grant to the user or retain for the copyright owner.

Rights granted or retained in software licensing

Public Domain	Permissive Licenses (non-reciprocal)	Copyleft Licenses (reciprocal)	Proprietary Licenses	Trade Secret
All rights waived	**<<< More rights granted to user**		**>>> More rights retained for owner**	**All rights retained**

Can I use a software license for other types of works?

Although Free and Open Source Software licenses can be used for non-software works, there are licenses like the Creative Commons CC0-1.0 (Public Domain), CC BY 3.0 (Attribution), CC-BY-4.0 (Share Alike) that suit better other purposes. However if you use different licenses for your software and its documentation, please note that the source code examples in the documentation are also licensed under the software license[5].

---

4   To see the compatible licenses with the notion of Free Software, please visit: https://www.gnu.org/licenses/license-list.en.html

5   For more information, please check: https://choosealicense.com/non-software/

What is a Free Software license?

In order for a program to be considered Free Software[6], its copyright holders must explicitly grant users the four essential freedoms to use, study, share, and improve, covered above. Any license that grants users these four essential freedoms can be considered a *Free Software license*[7].

Free Software licenses are enforced by "copyleft", a legal system that uses the power of copyright to grant everyone the freedom to use and study the program, conditioned on the requirement to allow sharing and modification of it. Therefore, most Free Software is not in the public domain, and the copyright holders have legally given permission for everyone to use accordingly the terms of the Free Software license.

What is an Open Source license?

Open Source licenses are software licenses which comply with the Open Source Definition, promulgated by the Open Source Initiative (the OSI)[8]. Accordingly to the OSI, the distribution terms of Open Source Software must comply with the following criteria:

a) Free Redistribution: The license shall not require a royalty or other fee for such sale.
b) Source Code must be included
c) Derived Works must be permitted
d) Integrity of The Author's Source Code: The license may require derived works to carry a different name or version number from the original software.
e) No Discrimination Against Persons or Groups
f) No Discrimination Against Fields of Endeavor
g) Distribution of License: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
h) License Must Not Be Specific to a Product
i) License Must Not Restrict Other Software
j) License Must Be Technology-Neutral

The differences between Free Software and Open Source licensing models are small. Both require the release of source code[9]. However, the major terminological difference occurs between the "copyleft licenses" and "permissive licenses", since there are some Open Source licenses that allow redistribution of the source code not under the same license terms. For copyleft licenses, the source code must be available under the same copyleft terms.

---

6   To see the compatible licenses with the notion of Free Software, please visit: https://www.gnu.org/licenses/license-list.en.html
7   For more information, please check: https://www.gnu.org/philosophy/free-sw.html
8   Read more: https://opensource.org/osd
9   Read more: https://www.gnu.org/philosophy/categories.html

What is copyleft?

Copyleft is a form of licensing that uses the power of copyright law to enforce conditions for the distribution of works, including software. When related to computer programs, the Free Software licensing model is enabled by copyleft[10]. A Free Software license gives each person possessing a copy of the software the same freedoms as the copyright holder, including the rights to use, study, share, and improve the software.

Copyleft licenses allow the essential freedoms of using, reproducing, adapting, or distributing under the condition that those freedoms remain intact in further distribution of the software or its derivative works. That's why copyleft licenses are also called "reciprocal licenses"[11], in contrast to "permissive or non-reciprocal" licenses that allow creating derived versions of the software with diverse license than the original, even with proprietary licensing.

The most common Free Software license is the GNU General Public License[12]. As a copyleft license, derivative work can only be distributed under the same license terms. This is in distinction to permissive (non-reciprocal) Open Source licenses, of which the MIT License[13] and Apache License 2.0[14] are widely-used examples.

For more information, see the table "Basic Conditions in Software Licensing" in the summary at the end of this section.

What is weak and strong copyleft?

Copyleft is a form of licensing that gives each person possessing a copy of the software the same freedoms as the copyright holder, including the rights to use, study, share, and improve the software. Depending on how the licensing conditions are applied to combination of code compiled together to make a program to run, there is a common reference to weak and strong copyleft. However, these expressions can lead to misunderstanding[15], since depending on how and when the license terms can be interpreted, they can introduce *weak* or *strong* copyleft terms*.*

Free and Open Source licenses embody the norms of the communities that use them, in a way that the reciprocity of these norms is to be expected. Reciprocity means that consistent licensing must be maintained.

In its strongest form, copyleft can place conditions on the licensing of all the other code compiled together to make the eventual binary executable program. Licenses like the GPL and EUPL set the scope of the expected reciprocity to submit under copyleft any code needed to create the resulting project binary. In this case the license is "project-scoped reciprocal".

---

10   Read more: https://www.gnu.org/copyleft/
11   Read more: *Permissive and Copyleft Are Not Antonyms, 2017.* https://opensource.org/node/875
12   See the license text here: https://www.gnu.org/licenses/gpl-3.0.html
13   See the license text here: https://opensource.org/licenses/MIT
14   See the license text here: https://www.apache.org/licenses/LICENSE-2.0
15   Read more: *Permissive and Copyleft Are Not Antonyms,* 2017. https://opensource.org/node/875

Weaker copyleft licenses set the scope of the expected reciprocity to the individual source files within the project, not the whole project collectively. If you change a file that is part of the project, or reuse code from a file in the project in a new codebase, the resulting file must be licensed the same way as the original file, but there are no requirements placed on other files combined together to create the program. Therefore, these licenses can be called "file-scoped reciprocal licenses."

For more information, see the table "Basic Conditions in Software Licensing" in the summary at the end of this section.

What is network copyleft?

Copyleft licenses allow the essential freedoms of using, reproducing, adapting, or distributing under the condition that those freedoms remain intact in further distribution of the software or its derivative works.

In general terms, the trigger of the copyleft conditions is the distributing of the software. However, for programs offered as services over a network, the traditional distribution concept becomes blurry.

Therefore, network copyleft licenses (AGPL for example) ensure that the source code is available to users of network, triggering copyleft not only by distributing but also using it..

What is a permissive license?

The Free Software licensing model is enabled by copyleft, a condition that anyone creating derived versions has to agree they will license the new version the same way as the original. Free and Open Source licenses embody the norms of the communities that use them. They are in many ways the "constitution of the community," so the embodiment of norms of reciprocity is to be expected. That is why copyleft licenses can be called reciprocal licenses.

The "permissive" licensing model reflects the minimal requirements about how the software can be redistributed. Licenses like MIT License[16] and Apache License 2.0[17] do not impose a condition that any derivative works must be distributed under the same copyleft license terms. Therefore, they are also know as non-reciprocal licenses[18].

In contrast with copyleft, permissive licenses permit the code to be used in proprietary derivative works.

For more information, see the table "Basic Conditions in Software Licensing" in the summary at the end of this section.

---

16 See the license text here: https://opensource.org/licenses/MIT
17 See the license text here: https://www.apache.org/licenses/LICENSE-2.0
18 Read more: *Permissive and Copyleft Are Not Antonyms,* 2017. https://opensource.org/node/875

Summary

A Free Software license is an important instrument used to hand control of your project's software back to the people, allowing them to use, study, share, and improve the code. In this section you will find, therefore, information about the basic conditions of FOSS licenses.

Basic conditions in software licensing[19]		
Permissive licenses (Non-reciprocal)	**Weak copyleft licenses (file-scoped reciprocal)**	**Strong copyleft licenses (project-scoped reciprocal)**
Examples: MIT, Apache 2.0. etc	Examples: MPL, EUPL, etc	Examples: GPL, AGPL, etc
. No restriction to code's use . No liability . Copyright notice requirement	. No restriction to code's use . No liability . Copyright notice requirement . Along the binaries, source code must be avaiable . Source code must be avaiable under the same copyleft terms . No restrictions on the exercise of the license . Combined code may be not submitted to copyleft	. No restriction to code's use . No liability . Copyright notice requirement . Along the binaries, source code must be avaiable . Source code must be avaiable under the same copyleft terms . No restrictions on the exercise of the license . Combined code may be submitted to copyleft

---

19   Meeker, Heather. *Open Source for Business.* 2.ed. Fleming, 2017. p. 33, 34.

CHOOSING A FREE SOFTWARE LICENSE

There are many Free Software licenses out there for you to choose from for your project. A large number have been written to satisfy the particular legal needs of a corporation or person, and in all likelihood would not be appropriate for your project. In most cases, it would be best for you to take a look at the most commonly used Free Software licenses, and to choose one from there that most suits your needs.

In this section, we will help you understand the importance of ensuring that the licenses in your project are compatible, and provide guidance on how to choose a license for your project's needs.

Once you have decided on which Free Software license to use for your project, you will then need to apply it to the software. The final part of this section will provide you guidance on how to do so.

How to choose a Free Software license

Can I write my own Free Software license for my work?

Anyone can write their own Free Software license to dictate the terms and conditions of use of their work. However, we strongly urge you to use an existing well-known license, instead of making up your own.

Licenses should be written to apply in complex legal issues, and people are more likely to use works under established and existing licenses whose legal effects are well known and have been clearly documented. The legal effects of a new license would be ambiguous, and this would have the effect of discouraging users from adopting your software.

Another reason to not write your own license is to prevent license proliferation. There are already numerous licenses available with very similar terms, which are often incompatible with each other. The standardization of license terms can make things easier for everyone; conversely, inventing a new license would make it harder for the community to keep track of all available licenses and would be a hindrance to developers who simply want to combine code under different licenses.

What should I consider when choosing a license?

Having a Free Software license is the best way to protect your project's contributors and users. A license can determine the way people contribute to the project and how the software will be used and shared.

If you are taking part in a project that has already determined which licenses to use, the best way to contribute is to continue using that project's license. Some communities have strong preferences for particular licenses. If you want to participate in one of these, it will be easier to use the preferred license even if you're starting a brand new project with no existing dependencies[20]. However, if you don't see your project as part of a community, you can choose a FOSS license.

There are two objectives to pursue with a FOSS license:

a) Make the code as simple and permissive as possible. If you plan to let people and organisations to use your code in the most simple way, permissive licenses let people do almost anything they want with your project, including to make and distribute closed source versions.

b) Protecting the code's freedom. Copyleft protects software freedom, permitting people do almost anything they want with your project, *except* to distribute closed source versions.

What are some of the most common licenses that I can use for my project?

The following table[21] lists some of the most common Free Software licenses, in order of the most permissive to the least.

---

20 See: https://choosealicense.com/community/

21 Table based on the following works: https://choosealicense.com/licenses/ and Wheeler, David. *The Free-Libre / Open Source Software (FLOSS) License Slide*, 2017. Avaiable at: https://dwheeler.com/essays/floss-license-slide.html

Permissive (non-reciprocal)	Weak copyleft (file-scoped reciprocal)	Strong copyleft licenses (project-scoped reciprocal)	Network Copyleft
Allow any kind of usage of code as long as attribution is provided with no liability. Permissive licenses permit software to become proprietary.	*Prevent the software component (often a software library) from becoming proprietary, yet permit it to be part of a larger proprietary program.*	*Prevent the software from becoming proprietary.*	*Prevent the software from becoming proprietary on network enviroment, triggerring copyleft not only by distributing but also using it.*
Most popular liceses			
MIT (SPDX code MIT)	**LGPL** (SPDX code LGPL-3.0-only)	**GPL v3** (SPDX code GPL-3.0-only)	**Affero GPL v3** (SPDX code AGPL-3.0-only)
A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.	The license allows developers and companies to use and integrate a software component released under the LGPL into their own (even proprietary) software without being required by the terms of a strong copyleft license to release the source code of their own components. The use of this license is discouraged by the FSF.	The most popular Free Software license. Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.	Based on GPL v3, is dedicated to software used over a network (SaaS for instance). This provision requires that the full source code be made available to any network user.
Apache 2.0 (SPDX code Apache-2.0)	**European Union Public License, version 1.2** (SPDX code EUPL-1.2)		
A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.	The European Commission FOSS license. Designed to be foster eGovernment initiatives. Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved.		

What license should I use for my software project?

For the purposes of the Next Generation Initiative, we recommend that you use the **GNU GPL v3.0** for your software project.

However, this is not a one-size-fits-all recommendation. You should consider the nature of your project, and decide which of the licenses above would fit your needs.

What is an MIT-style license? When should I use it?

The MIT License[22] is a permissive license, that only requires the preservation of copyright and license notices. It permits users to use, copy, modify, merge, publish, distribute, sublicense, and/ or sell copies of the licensed software without any restrictions. This means that any derivative works can be distributed under different terms and without access to the source code, and can be made into proprietary software.

If you are comfortable with your project's code potentially being used in proprietary programs, you can choose to use an MIT-style license.

What is the GPL? When should I use it?

The GNU General Public License version 3[23] (commonly referred to as the "GPL") is a copyleft license that is probably the most widely recognized Free Software license available. This is in itself a big advantage, since many potential users and contributors will be familiar with it, and therefore won't have to spend extra time to read and understand your license.

The GPL is the also the most popular Free Software license. If you would like your code to be able to be merged freely with code that is covered by the GPL, then you should choose the GPL, or a GPL compatible license.

The Free Software Foundation in the United States of America maintains a list showing which licenses are compatible with the GPL (as well as those that are not) **here**.

Compatibility of Free Software licenses

What does it mean for Free Software licenses to be compatible or incompatible?

When your project combines two pieces of Free Software into one, or merges code from one into another, it is important to pay attention to whether the licenses of each software or code allow this combination, or prohibit it. If the licenses allow it, the licenses can be said to be "compatible" with one another. If they prohibit it, the licenses can be said to be "incompatible" with one another.

In other words, we say that several licenses are compatible with each other if there is a way to merge code under these various licenses while still complying with all of them[24].

---

22   For full text, see: https://opensource.org/licenses/MIT
23   For full text, see: https://www.gnu.org/licenses/gpl-3.0.en.html
24   For more information, see "License Compatibility and Relicensing", by Richard Stallman (https://www.gnu.org/licenses/license-compatibility.en.html)

What kind of licenses are compatible with copyleft licenses?

License compatibility is particularly important for copyleft licenses. Copyleft licenses such as the GPL provide that derived works may only be used under the same license conditions (that is, those of the GPL).

Copyleft licenses are therefore only compatible with other Free Software licenses when:

a) the other open-source license does not contain any license requirements that are not provided by the compatible copyleft license[25];

b) the other open-source license contains a special compatibility or opening clause[26].

For more information about License Compatibility, you can refer to our documentation entitled Third Party Code and License Compatibility.

Summary

Choosing a FOSS license for your project is one of the first steps to be made. We listed several licenses that fit a broad spectrum of purposes for you to choose for your project. It is fundamental to understand how pieces of software under different licenses interact in the final product. In this section you can find the description of the most common licenses and some pointers about their compatibility.

---

25 For example, this is the case with the BSD license without an advertising clause, whereas the BSD license with an advertising clause contains an information requirement that the GPL and other copyleft licenses do not provide
26 This is the case for example in sec. 3 of the LGPL, Version 2.1 that permits the use of LGPL code under the GPL. GPLv3 contains a compatibility clause for the Affero GPL and opening clauses for Apache license 2.0 and other licenses. The European Public License (EUPL) and German free software license (d-fsl) contain compatibility clauses for the GPL

APPLYING A FREE SOFTWARE LICENSE

Once you have decided on which Free Software license to use for your project, you will then need to apply it to your software. This section will provide you guidance on how to do so, as well as best practices that you are encouraged to adopt.

Applying a Free Software license to your software

What steps should I take to ensure the Free Software license is applied to my software?

To apply the license to your software, you will need to do 2 things: first to effectively inform the public what license you intend the software to be released under, and second to ensure that the software itself includes the license.

Step 1: Informing the public of your intended license

You should first state which license you are using clearly on the project's front page. You do not have to include the actual text of the license there; what is important is that the name of the license, as well as a link to the full license text is available.

Step 2: Including the license in your software

You can include the license in your software by putting the full license text in a file called COPYING (or LICENSE) included with the source code. You should also place a short notice in a comment at the top of each source file, naming the copyright date, holder, and license. This notice should also include a note on where to find the full text of the license.

What would a notice in my source files look like?

In general, the notices that are placed in source files do not have to look exactly like each other, and can take many different forms, as long as the standard requirements are met.

It is therefore important to ensure that the notice begins with the same notice of copyright holder and date, states the name of the license, and makes clear where to view the full license terms.

An example of such a notice can be found in the text of the GNU General Public License version 3.

> *<one line to give the program's name and a brief idea of what it does.>*
> *Copyright (C) <year> <name of author>*

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <https://www.gnu.org/licenses/>.

If your software does terminal interaction, the GPL also suggests making it output a short notice when it starts in an interactive mode, using the following template:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

Software Package Data Exchange (SPDX)

What is SPDX?

SPDX is a project by the Linux Foundation and stands for Software Package Data Exchange[27]. It is an open standard for communicating components, licenses, and copyrights associated with a particular version of software packages. The need to identify the license for open source software is critical for both reporting purposes and license compliance.

SPDX reduces redundant work by providing a common format for companies and communities to share important data about software licenses, copyrights, and security references. SPDX provides common language and vocabulary to form a precise and unambiguous software bill of material (SBOM). SPDX creates a simple format for licenses that is machine-readable to help automation and short to help humans quickly handle common cases. In order to use SPDX it is necessary to understand three basic concepts[28]:

- SPDX license identifiers
- SPDX license expressions
- SPDX files

---

27   More information about SPDX can be found here: https://spdx.org/
28   This material is based on David Wheeler's tutorial on SPDX: https://github.com/david-a-wheeler/spdx-tutorial

What is an SPDX license identifier?

An SPDX License Identifier is a human readable short text string that uniquely identifies a license, or in other words, a standardized shortened version of the license name.

The SPDX License List can give you the commonly found licenses and exceptions used in Free and Open Source Software, and other collaborative software or documentation. You can also refer to the REUSE section of these documents to learn how to apply the SPDX License Identifiers into projects.

What are SPDX License Expressions?

In some situations where there isn't just one license, for example where software might be offered with a choice between two or more licenses, SPDX allows you to clearly express choices when you need to. You can use the following different operators to combine license expressions to create other license expressions:

- **"OR": Recipients can choose between two or more licenses.**
- **"+": This means "this license or any later version".**
- **"AND": Recipients are required to simultaneously comply with two or more licenses.**
- "WITH": Add the following named exception. For example, "(GPL-3.0-or-later WITH Classpath-exception-2.0)" means recipients must comply with the GPL version 3.0 or later with the Classpath 2.0 license exception.

Most programming languages have a language library package manager, and most systems have a system-level package manager. Each package manager has a package format. In almost all of them there is a "license" field, and in most cases you can simply use a SPDX license expression in that field.

What are SPDX files?

SPDX files are a way to capture and exchange license information in a way that is independent of the programming languages and package managers used.

The SPDX specification actually supports the *tag:value* format. A *tag:value* file is normally a sequence of lines, where each line is a tag name, a colon, a space, and its value. There are many tags defined in the SPDX specification[29].

There are some examples of tags?

Here are some especially important or useful tags:

---

29   See more at https://spdx.org/specifications

- SPDXVersion: The version of the spec used, normally "SPDX-2.1".
- DataLicense: The license for the license data itself.
- Creator: Who or what created this SPDX file (not the package creator). This is in one of 3 formats:
  - For a person: person name, optionally followed by email in parentheses.
  - For an organization: organization name, optionally followed by email in parentheses.
  - For a tool: toolidentifier-version
- PackageName: The full name of the package as given by Package Originator.
- PackageOriginator: The person or organization from whom the package originally came.
- PackageVersion: The version number of this particular version of the package (optional).
- PackageHomePage: The package's home page URL.
- PackageLicenseDeclared: The license identified in text in one or more files (for example a COPYING or LICENSE file) in the source code package. This field is not intended to capture license information obtained from an external source, such as the package website.

An example of a package expressed as a *tag:value* is:

```
SPDXVersion: SPDX-2.1
DataLicense: CC0-1.0
PackageName: Foo
PackageOriginator: David A. Wheeler
PackageHomePage: https://github.com/david-a-wheeler/spdx-tutorial/
PackageLicenseDeclared: MIT
```

We recommend that you use ".spdx" for an SPDX file in the *tag:value* format (i.e. "LICENSE.spdx").

If you are developing software or other copyrightable content, you will still need to select a license and express it in a manner that humans can understand. For software, create a file named LICENSE or COPYING (possibly with a .md or .txt extension) to provide human readable text of the license.

Also create a SPDX file, as described above, so programs can automatically process exactly what the license is, and humans can have a short and precise way of representing the license of the software.

The REUSE Initiative

What is the REUSE Initiative?

The REUSE Initiative (or simply, "REUSE"), is a project of the Free Software Foundation Europe to develop awareness of the best practices for expressing license and copyright

information in free and open source projects. It intends to facilitate management of source code by making licensing and copyright information more consistent in how it is added to source code in ways which allow for automating many of the processes involved.

How do I make my project REUSE compliant?

Making your project REUSE compliant can be done in three steps:

- Choose and provide licenses
- Add copyright and license information to each file
- Confirm REUSE compliance

For a tutorial on how to do so, please refer to the later section in this FAQs on REUSE.

_____

Summary

In order to apply a Free and Open Source license to your project, you need to follow some steps. SPDX and REUSE represent the best procedures to achieve that. You can learn more about them in this section.

REUSE FAQ

The REUSE Initiative (or simply, "REUSE"), is a project of the Free Software Foundation Europe to develop awareness of the best practices for expressing license and copyright information in free and open source projects. It intends to facilitate management of source code by making licensing and copyright information more consistent in how it is added to source code in ways which allow for automating many of the processes involved.

This section will show you the basic methods to make a software project REUSE compliance. By the end of this document, all of your files should clearly have their copyright and licensing marked, and you will be able to verify this using a linter tool.

Making your project REUSE compliant can be done in three simple steps:

- Choose and provide licenses
- Add copyright and license information to each file
- Confirm REUSE compliance

You can use the following tutorial to do so. For the purposes of this tutorial, we will assume that you have chosen to use the GNU General Public License v3.0 or any later version (the "GPL"), and that the directory of your project looks like this:

```
project/
├── img/
│       ├── cat.jpg
│       └── dog.jpg
├── src/
│       └── main.c
├── .gitignore
├── Makefile
└── README.md
```

Choosing and Providing Licenses

How do I provide licenses?

After choosing a license, you will need to put the license in your project directory. You can find your license in the SPDX License List. SPDX is an open standard for communicating license and copyright information. Each license is uniquely identified by a shortform SPDX License Identifier. The SPDX License Identifier for your chosen license is GPL-3.0-or-later.

You create a LICENSES directory in your project root which will contain all the licenses that you use in your project. You then download your license from the *license-list-data* repository and put it in the *LICENSES* directory.

Adding Copyright and Licensing Information

How do I add copyright and license information to each file?

Now that you have a license, you need to indicate in the relevant files that these files fall under that license. You edit the comment header of *src/main.c* as such:

```
/*
 * SPDX-FileCopyrightText: 2019 Jane Doe <jane@example.com>
 *
 * SPDX-License-Identifier: GPL-3.0-or-later
 */
```

The *SPDX-FileCopyrightText* tag records the publication years and the copyright holder of the contents of the file.

The *SPDX-License-Identifier* tag is followed by a valid SPDX License Expression, typically just the SPDX License Identifier of the license.

You can be flexible with the format, just make sure that the line starts with *SPDX-FileCopyrightText:*.

Each file must always contain these two tags in the header. You are allowed to use the tags multiple times if you have multiple copyright holders or licenses.

In the example project, you also edit *Makefile* and *README.md* using this header information, but of course with corresponding comment syntax.

What do I do with binary and uncommentable files?

You will also want to license your image files under GPL-3.0-or-later. Unfortunately, images and other binary files do not have comment headers that one can easily edit. Other examples include automatically generated files and certain data and configuration files for which comments are non-trivial.

There is a simple trick to circumvent this. Create the files *cat.jpg.license* and *dog.jpg.license*, each containing the same information about license and copyright holder as above.

How do I change licensing information?

It may arise that the photos of the cat and the dog were not licensed under the GPL at all, but under the Creative Commons Attribution 4.0 International, and owned by John Doe. If this happens, then you should check the SPDX License Identifier of the license. In this case, it is *CC-BY-4.0.*

You should then create the file *LICENSES/CC-BY-4.0.txt*, following the same steps you used for GPL-3.0-or-later.

You can then edit *cat.jpg.license* and *dog.jpg.license* to say:

> *SPDX-FileCopyrightText: 2019 John Doe <john@doe.com>*
>
> *SPDX-License-Identifier: CC-BY-4.0*

Do I need to provide licensing information for build artifacts?

When you compile your program, you will generate some build artifacts, such as *src/main.o*. You do not need to provide any licensing information for those files. Just use your *.gitignore* file to ignore these build artifacts. The REUSE tool will respect the contents of *.gitignore*.

What do I do with insignificant files?

You probably will have files in your project that you do not find particularly copyrightable, for example configuration files such as *.gitignore*. Intuitively you may not want to license these files, but the fundamental idea of REUSE is that all your files will clearly have their copyright and licensing marked.

One way to indicate that you do not exercise any copyright over this file is by marking it as being in the public domain. The CC0 license is a good way to do this. Edit the file to contain:

> *# SPDX-FileCopyrightText: 2019 Jane Doe <jane@example.com>*
> *#*
> *# SPDX-License-Identifier: CC0-1.0*

Resulting Project Tree. Your project tree will now look like this:

```
project/
├── img/
│   ├── cat.jpg
│   ├── cat.jpg.license
│   ├── dog.jpg
│   └── dog.jpg.license
├── LICENSES/
│   ├── CC0-1.0.txt
│   ├── CC-BY-4.0.txt
│   └── GPL-3.0-or-later.txt
├── src/
│   └── main.c
├── .gitignore
├── Makefile
└── README.md
```

Confirming REUSE Compliance

How can I confirm if my project is in compliance with REUSE guidelines?

Now that you have marked all files with their copyright and licensing, it is time to check whether you did not miss anything. To do this, we provide a linter tool for you to use. You can read the [full documentation here](#), or read the quick steps below.

> *$ # Install the dependencies for the tool.*
> *$ sudo apt install python3 python3-pip*
> *$*
> *$ # Install the tool*
> *$ pip3 install --user fsfe-reuse*

The executable is now in *$HOME/.local/bin/*. Make sure that this is in your *$PATH*. Now go to the project directory and run the linter.

> *$ cd path/to/project/*
> *$ reuse lint*
> *SUMMARY*
>
> *Bad licenses: 0*
> *Missing licenses: 0*
> *Unused licenses: 0*

Used licenses: CC-BY-4.0, CC0-1.0, GPL-3.0-or-later
Read errors: 0
Files with copyright information: 6 / 6
Files with license information: 6 / 6

Congratulations! Your project is REUSE compliant :-)

As you can see in the last line, the tool confirms that your project is compliant with REUSE now! To learn what the different numbers mean, please have a look at the full documentation of the reuse tool.

Getting More Help

We hope that this tutorial has helped you to understand how to adopt REUSE, and the three basic steps to properly license your software project. Although we have covered a few edge cases, you might run into more questions soon. Here are some more resources for your reference if you need more help:

- Our Frequently Asked Questions covers common questions as well as extraordinary cases and will constantly be updated.
- The full REUSE specification formally describes REUSE and the methods to become compliant.
- The REUSE tool documentation describes installation and usage of the REUSE tool.

If none of the links above were able to answer your question, please contact us by:

- opening an issue on reuse-docs for questions on the tutorial, FAQ or specification;
- opening an issue on reuse-tool for questions on the REUSE tool;
- or sending an email to the FSFE. Please note that we would prefer issues because they are publicly searchable for other people.

Summary

REUSE is part of the best practices to comply with FOSS licenses copyright and license notices requirements. This section will show you the basic methods to make a software project REUSE compliant. You will learn how to have your code properly display your copyright and licensing, and verify this using a linter tool.

DISPLAYING PROPER NOTICES

Displaying proper license and copyright information is of great importance for any Free and Open Source Software developer, user, or distributor. Incorrect license and copyright information can lead to inconvenient or even unlawful situations.

For example, incorrectly displaying such notices can lead to software developers not being credited for the work that they have done, or for the license or copyright information being misunderstood. Complications can also arise where using a particular software project becomes a burdensome process, because users are unable to use automated tools due to the non-machine readable manner in which the notices are being displayed.

Quick Steps on How to Display Proper Notices

We recommend that you take the following steps to properly display your license and copyright information.

Provide Licenses
After you have chosen a Free Software license for your project, find the SPDX identifier of your license in the **SPDX License List**. Download the license text for your license from the **license-list-data repository** and put it in the *Licenses/* directory.

Add copyright and licensing information to each file
For all files edit a header to contain the following:

> # SPDX-FileCopyrightText: [year] [copyright holder] <[email address]>
> #
> # SPDX-License-Identifier: [identifier]

Provide an inventory for included software (Optional)
Aside from the license files included in the project, and the file level copyright information, you may also include a bill of materials for your project, but you can choose to omit doing this if you cannot generate it automatically.

The bill of materials should conform to the SPDX specification and included in a file in the top level directory of your repository called *LICENSE.spdx.*

Confirm REUSE Compliance
Use the *REUSE Tool* to automate some of these steps, and to check whether you did everything correctly.

Some Basic Information

What is copyright?

Copyright is a legal construct that grants someone exclusive rights over a creative work. The most important exclusive right is in the name: The right to produce copies. Only the copyright holder is allowed to give new copies of their work to people.

Usually, the author is the copyright owner, but copyright is often transferred to the author's employer. In most jurisdictions, you do not need to do anything to gain copyright. As soon as you make a creative work, you (or your employer) instantly gain copyright over it.

What is a license?

One problem with copyright as it relates to software is that it makes software unshareable by default. A license changes that. A license defines the terms under which the copyright holder allows the recipient of the license to use the software. If the license allows the recipient to use, study, share, and improve the software, then that software is Free Software.

How do I copy someone else's work?

If someone else has made their work available for you to use and copy, you can incorporate their work into your project.

When you put the work in one of the files in your project, you should add an *SPDX-FileCopyrightText* tag for the copyright holder(s) and an *SPDX-License-Identifier* tag for the license(s) under which the work was made available.

If the work was licensed differently from your project, you should verify whether the licenses are compatible, and add the new licenses to your project.

Can I copy a work that has no copyright notice or license?

Before you proceed, always first make sure that you can find the copyright and licensing information elsewhere. Some projects only include this information in the root directory or in their README file.

If you can find no copyright notice, then that is no problem. You can add the copyright notice yourself.

If the work has no license, then that means that you do not have the right to copy it. If you believe that this is a mistake and the author clearly meant for you to be able to copy this work, you should contact the author and ask them to license their work. Feel free to refer them to https://reuse.software.

Which files are copyrightable?

All files that are original works of authorship are copyrightable. In essence, if someone sat down typing their own original thoughts on a keyboard, then that person holds copyright over the output. Common examples are source code, documentation, audio, and video.

However, there are some edge cases. For example, the program

    print("Hello, NGI!")

probably does not meet the threshold of originality. Similarly, data files and configuration files may not meet that threshold either.

What is a copyright holder, and what is an author?

In these resources, we maintain a distinction between the copyright holder and the author. The author (also known as creator) is the person who sat down and created a work. Think of the author as a programmer, writer, or artist.

The copyright holder is the person who has the exclusive rights over that work. Often the author and the copyright holder are the same. However, if the author is being paid by their employer to create a work, the employer is often the copyright holder.

Keep in mind that in some jurisdictions, the word "author" is often used as a synonym for "copyright holder". In other jurisdictions, authors maintain some rights over their work even if they are not the copyright holder.

Displaying Proper Copyright and License Notices

How do I copy someone else's work?

If someone else has made their work available for you to use and copy, you can incorporate their work into your project. When you put the work in one of the files in your project, you should add an *SPDX-FileCopyrightText* tag for the copyright holder(s) and an *SPDX-License-Identifier* tag for the license(s) under which the work was made available.

If the work was licensed differently from your project, you should verify whether the licenses are compatible, and add the new licenses to your project.

Is there a standard format for declaring copyright?

Generally, we recommend that you use the following format:

SPDX-FileCopyrightText: [year] [copyright holder] <[contact address]>

You may choose to drop items except the copyright holder, which must always be included. We recommend that you include all items, however.

Do I use SPDX-FileCopyrightText, Copyright, or ©?

The following copyright notices are valid examples that you can use for your copyright notices:

SPDX-FileCopyrightText: 2019 Jane Doe <jane@example.com>
SPDX-FileCopyrightText: © 2019 John Doe <joe@example.com>
© Example Corporation <https://corp.example.com>
Copyright 2016, 2018-2019 Joe Anybody
Copyright (c) Alice

Out of the above, we highly recommend using the first two. The others exist primarily to be compatible with existing conventions.

What years do I include in the copyright statement?

Generally, there are four options for you to choose:

- The year of initial publication
- The year of the latest publication
- All years of publications, either as range (e.g. 2017 – 2019) or as separate entries (e.g. 2017, 2018, 2019).
- Do not include any year

Which option you choose is ultimately up to you.

Where else do I put my license information?

Marking all individual files with *SPDX-License-Identifier* tags goes a long way towards unambiguously communicating the license information of your project, but it helps to communicate the license information in natural language as well.

In the README of your project, feel free to provide a summary of the licensing information, or simply redirect the reader to your *LICENSES/* directory.

Addtionally, many package hosting sites expect that you declare the licensing information of your package.

Can I copy a work that has no copyright notice or license?

Before you proceed, always first make sure that you can find the copyright and licensing information elsewhere. Some projects only include this information in the root directory or in their README file.

If you can find no copyright notice, then that is no problem. You can add the copyright notice yourself.

If the work has no license, then that means that you do not have the right to copy it. If you believe that this is a mistake and the author clearly meant for you to be able to copy this work, you should contact the author and ask them to license their work. Feel free to refer them to https://reuse.software.

What should I do with uncopyrightable files?

There are two things that you can do with such a file to make sure that its copyright and licensing is recorded. The first option is to simply use your regular copyright and license header for this file. There is nothing that stops you from claiming copyright over your own works. However, a court would still likely find such files uncopyrightable.

The alternative is to waive your copyright by using the CC0-1.0[30] license.

It is important to note that you can only do this for your own works. If the file was authored by someone else, you must declare their copyright and license in the header.

How do I deal with a file that has been edited by many people?

Some files are edited by many people and would have an extremely long list of copyright holders in the header. This may be aesthetically unpleasing, but is not incorrect.

If you would rather not deal with having so many copyright notices, some projects such as Chromium circumvent this problem by using "Copyright (c) 2013 The Chromium Authors" as their copyright tag. You may consider doing this, but then you should keep a list of copyright holders and authors in a separate file in your project.

Why can't I use version control to record copyright?

As previously explained, there is difference between the copyright holder and the author, which do not always refer to the same person. Version control typically only records authorship, which makes it unsuitable for the task of recording copyright.

---

30   More information can be found here: https://creativecommons.org/publicdomain/zero/1.0/

Another obstacle is that version control history may contain errors, and fixing such an error would require rewriting the history, causing all contributors to have to re-download the new trunk.

A further issue with version control is that the *blame* command that is typically used to find authorship line-by-line shows only the author of the last commit in that line, even if it was a trivial commit such as fixing a typo.

Can I edit copyright notices and license disclaimers?

If you find out that some information is incorrect, you are free to adjust it. Otherwise, it is usually a good idea to leave copyright notices and license disclaimers intact. But there is no one-size-fits-all answer here.

What are license exceptions and what do I do with them?

License exceptions are additions or alterations to a license that often work to permit a certain use of the code that wouldn't be allowed under the original license. It is often used by compilers, where a portion of compiler code may end up in the resulting binary. The exception may waive rights over portions of code that end up in binaries.

Exceptions are treated almost identically to licenses. You can combine a license with an exception by marking a file with the following tag:

SPDX-License-Identifier: GPL-3.0-or-later WITH GCC-exception-3.1

Summary

Copyright notices is a central element on FOSS licenses compliance. In this section you will learn the basic elements of introducing copyright notices to your code.

LICENSE COMPATIBILITY AND THIRD PARTY CODE

If you intend to incorporate or merge existing code into your software project, it is important that you ensure that the licenses that cover those existing code are compatible with each other, and with the license that you intend to issue your software project under.

Compatibility of Free Software Licenses

What does it mean to say that licenses are compatible?

When your project combines two pieces of Free Software into one, or merges code from one into another, it is important to pay attention to whether the licenses of each software or code allow this combination, or prohibit it. If the licenses allow it, the licenses can be said to be "compatible" with one another. If they prohibit it, the licenses can be said to be "incompatible" with one another.

In other words, we say that several licenses are compatible with each other if there is a way to merge code under these various licenses while still complying with all of them[31]. It is therefore important that, when you incorporate existing code from a third party, you ensure that the license of that third party code is compatible with the license that you intend to issue your software project under.

Being compatible also means that code under one Free Software license can be combined with code under another, and the resulting software can be distributed under either Free Software license without violating the terms of the other.

Can code covered by Free Software licenses be used in proprietary software?

Generally, Free Software licenses allow anyone to modify the code covered, and to redistribute it in both its original and modified form.

Permissive Free Software licenses (i.e. non-copyleft licenses) allow their code to be used in proprietary derivative works. In such a case, the derivative work will still be proprietary, not withstanding that it contains code from a source that is covered by a Free Software license. Examples of permissive licenses that allow covered code to be used in proprietary derivative works include the Apache license, the MIT license, and BSD licenses. Conversely, copyleft licenses like the General Public License (GPL) requires that any derivative work must be distributed under the same terms, and be Free Software as well.

---

31 For more information, see "License Compatibility and Relicensing", by Richard Stallman (https://www.gnu.org/licenses/license-compatibility.en.html)

Are all Free Software licenses compatible with each other?

No. While most of the commonly-used permissive Free Software licenses are compatible with each other, copyleft licenses (that do not allow their covered code to be used in proprietary software) can generally be said to be incompatible with permissive Free Software licenses (that allow their covered code to be used in proprietary software).

Can copyleft licenses be compatible with other Free Software licenses?

Some Free Software licenses are also compatible with some copyleft licenses, meaning that code covered under a permissive Free Software license and code covered under copyleft license can be distributed as a combined work under the copyleft license, with the original code in each case remaining under its original license. In such a situation, the resulting work should be covered under the copyleft license, as it is the license that places more conditions.

License Compatibility and the GPL

Why is the GPL incompatible with certain Free Software licenses?

The primary goal of the GPL is the promotion and furtherance of Free Software. Accordingly, the GPL was crafted specifically to make it impossible to merge GPL covered code into proprietary derivative software works. This can be seen in the two most important of the GPL's requirements:

- Any derivative work from GPL covered code must itself be distributed under the GPL;
- No additional restrictions may be placed on the redistribution of either the original work or a derivative work.

With these conditions, the GPL succeeds in making the four freedoms of Free Software "contagious". Once a program is covered under the GPL, these freedoms are passed on to all other works that the code gets incorporated into, thereby making it practically impossible to use GPLed code in proprietary or closed source programs.

These same conditions however also means that the GPL is incompatible with certain other Free Software licenses. This usually happens when the other Free Software license imposes a requirement that is not present in the GPL, which makes it incompatible with the GPL's condition not to add any additional restrictions on a derivative work.

Can using the GPL/a GPL-compatible license make my project easily license compatible?

Yes. The GPL is by far the most popular Free Software license. There are approximately twice as many software projects licensed and released under the GPL than under the next most popular licenses. If you want your code to be freely mixed with GPLed code, then we strongly recommend that you pick a GPL-compatible license, or use the GPL itself.

What licenses are compatible with the GPL?

The Free Software Foundation maintains a list of licenses compatible with the GPL. You can find the list online here: https://www.gnu.org/licenses/license-list.en.html

Summary

One challenging aspect of FOSS licensing is the necessity to maintain compatibility between various licenses in code that you have incorporated into your project. If your project has software under diverse licenses, it is important to understand the basic compatibility workflow. In this section you will learn more about that.

CONTRIBUTIONS FROM EXTERNAL DEVELOPERS

If your software project has many external contributors (i.e. developers from outside your team who contribute to the code), this presents some issues with copyright ownership. If not properly managed, external developers may in the future claim copyright on the sections of the project that they contributed to. This issue has come up various times in the past in Free Software projects.

Accordingly, you may wish to consider whether steps should be taken to ensure that ownership of the entirety of the code belongs to you. Typically, it would be prudent to refrain from accepting external contributions into your repository, until the contributor agrees in some capacity to either transfer their copyright interest to you, or to explicitly waive their rights to enforce any copyrights they may hold within your software project.

Contributor Agreements

What are contributor agreements?

Contributor agreements are agreements between a FOSS project and contributors to that project that set out what the project can do with the copyright of the contributions made; whether it be the code, translations, documentation, artwork, etc.

The purpose of such agreements is to make the terms under which contributions are made explicit, and thereby protect the project, the users of the project's code or content, and often also the contributors themselves. They provide confidence that the guardian of a project's output has the necessary rights over all contributions to allow for distribution of the product under either any license or any license that is compliant with specific principles.

What are the different types of contributor agreements available?

There are two categories of contributor agreements: Copyright License Agreements (CLAs) and Copyright Assignment Agreements (CAAs).

Whereas the CLA require an irrevocable license to allow the project owner to use the contribution, the CAA requires assignment and therefore a transfer of copyright to the project owner.

Contributor License Agreements (CLAs)

What are Contributor License Agreements (CLAs)?

A Contributor License Agreement (CLA) is a legal document that defines the terms under which intellectual property, including software code, is contributed to a larger project. CLAs typically state that once the contributor has provided a project with a contribution, they cannot try to withhold permission from the project for use of the contribution at a later date. For your purposes, this would allow your software project to be confident that any external developers contributing to your code cannot attempt to stop you from using pieces of the code at a later date.

Nowadays, CLAs can come in an electronic form that a developer fills out and sends in to the project, or even a web-based checkbox that the developer checks before completing their first contribution to the project.

What kind of terms should be included in a CLA?

A CLA should grant copyright over the contribution to the project's owner (in this case, to you or your project) and enable the release of the software in question. Accordingly, the contributor needs to at the very least grant the rights that will be granted to the project's owner in the license to be used for the distribution of the overall project.

When granting rights it is common to grant a very broad range of rights. This is in order to avoid the need to return to the contributor for authorization to take a desired action with their contribution, such as releasing under a different license.

What kind of contributions require a CLA?

It is very important to record the assignment of copyright or grant of rights for each contribution and from each contributor. This means that your software project must track and record any CLAs submitted by contributors. Maintaining CLAs can require a fair amount of effort, and therefore make it more onerous on your part to accept contributions from external developers.

As some contributions may be very small, for example, simple bug fixes or spelling corrections, you may be tempted to bypass requirements for CLAs for small contributions. For small contributions, the risk of any loss suffered from any potential legal complications can be small. Conversely, the larger the contribution, potential losses from legal complications not covered by a CLA can be much higher. Ultimately, it is up to the project owners to decide which contributions they choose to accept with and without a CLA.

Contributor Assignment Agreements (CAAs)

What are Contributor Assignment Agreements (CAAs)?

A Contributor Assignment Agreement (CAA) is a legal document that effectively transfers ownership of the copyright over contributed material from the contributor to the project. This differs from CLAs, which give the licensee a license to use the copyrighted material, while the contributor still retains copyright ownership.

Developer Certificates of Origin (DCOs)

What are Developer Certificates of Origin (DCOs)?

Developer Certificates of Origin (DCOs) is a legal mechanism that can be said to be a simplified version of a CLA. DCOs are often used as alternatives to CLAs. DCOs are in essence an affirmation that the contributor intends to contribute the contributed code under the project's license, and that the contributor has the right to do so. It was introduced in 2004 by the Linux Foundation to enhance the submission process for software contributions used in the Linux kernel.

In using a DCO, external developers certify that they adhere to these requirements by signing off on their commits. Typically, this would mean that the external developer offers their changes under the same license as the software project they are contributing to, that they had the right to do that, and that they did not contribute someone else's work to the software project other than their own.

What should I use to handle external contributors for my software project?

It depends.

DCOs set the incoming license of the external contribution to be the same as the outgoing license of your software project. On the other hand, using a CLA reserves the right for you to decide whether to relicense the entire software project (and accordingly all past contributions to it) under a different license in the future, if the current license used for your project is found to be inadequate down the road. This can be a benefit as you would not be required to contact everyone who has contributed to the project to obtain their permission to do so. However, this presents a social risk, that the project takes a direction that contributors find unacceptable, and find themselves unable to act upon it.

Using CLAs for your external contributors also add an additional layer of bureaucracy for your project, and would require resources to draft and record, but they do provide more concrete

protections for your project. Conversely, DCOs reduce the barriers for external developers to contribute, while still requiring them to certify that they are submitting their own work.

CAAs are a more heavy-handed manner of ensuring that the software project always has the rights to use the code that has been contributed.

Accordingly, you should consider the needs of your specific software project and the community that it serves, and decide accordingly.

Where can I find a suitable contributor agreement for the purposes of my software project?

You can find templates for contributor-friendly, multi-purpose contributor agreements on the following website: https://contributoragreements.org.

In particular, you can use the Agreement Chooser here to select a template agreement: http://contributoragreements.org/agreement-chooser.html.

_____

Summary

How does the project get the legal right to redistribute some contributor's code changes? How does the project know the contributor won't sue later for copyright infringement? In this section you will have a handy overview on how the contributor *gives* the project that right formally, by signing an agreement saying that the project can redistribute the contributed code.

FREE SOFTWARE AND SOFTWARE PATENTS

Software patents are a highly controversial topic, with negative consequences for Free Software. However, if your project has software patents involved, here are some pointers.

Patents

What are patents?

A patent is a type of intellectual property separate from copyright and trademarks. It grants an exclusive property right for an invention that enables the patent holder to exclude other people from practicing the invention or idea claimed in the patent.

How are patents different from copyright licenses?

While copyright protects the *expression* of ideas, patents work to protect the *ideas* themselves, not just the expressions of them. The main effect of a patent is therefore to give patent holders the right to challenge any use of the invention or the idea by another person.

Software Patents

What are software patents?

A software patent is a patent on a piece of software.

An example of a software patent would be the patent over Amazon One-Click, a software that lets internet shoppers purchase something on their web platform with a single click, if a user has their payment, billing, and shipping information saved.

Patent law varies fundamentally among countries. In Europe, the patent system is run by the European Patent Organisation (EPO), a supranational organ governed by the European Patent Convention of 1970. Although many countries place limits on the patenting of inventions involving software, including the European Patent Convention, the EPO has already granted thousand patents related to computer programs[32].

Yet, there is no consensus about the definition of a software patent. What the European legislation agrees is, in contrast to copyright, which is exercised over the code itself, software patents are exercised over the **functionality** in which the software is intended to work. It is independent of the code. The EPO argues that as soon as a computer program has a "technical

---

32   Read more: http://en.swpat.org/wiki/Software_patents_wiki:_home_page

effect" – making a hard drive spin, lighting up pixels on a screen – it is a physical machine, and therefore patentable.

How can software patents be problematic?

Under copyright law applicable to software, if part of your code infringes on someone else's copyright, you can solve this issue by rewriting the offending section of code, while continuing to ensure that the underlying function that the code aims to achieve remains the same.

However, in the case of a software patent, it does not matter how the code is written, or what programming language is used, as the patent acts as a blanket restriction against anyone but the patent holder from implementing a certain idea. Accordingly, once a patent holder accuses a software project of infringing a patent, the project must either stop implementing the offending feature, or expose the project and its users to legal risk.

This attitude place threats to FOSS projects. For example[33]:

- Software patents add legal risks, and therefore costs, to software development.
- Software patents specifically inhibit the development of useful software by blocking compatibility and interoperability.
- Patents are incompatible with software because software is so complex – too many ideas are used for it to be practical for a developer to check each individual idea against existing patents, to prevent infringement.

One of the major downsides of the patent system is, differently from copyright, enforcing a patent does not require the patent owner to engage in any business he/she is trying to protect. This is why patent "trolls" can exist – companies that do not engage in any business except suing other for patent infringement[34].

Nevertheless, Free Software projects has a tendency to be less vulnerable to patent claims. Among the factors is fact that open source code is freely available to review, avoiding legal discovery procedures, as well as due to the cooperation on engineering among FOSS community to find solutions around patent claims. Besides, FOSS licenses contain liability limitation for the developer, what can restrict in some cases the liability for patent infringement.

My project has software patents. How a FOSS license affects it?

FOSS licenses are primarily copyright licenses, but many of them have expressed and implied patent licensing built into them. So, if you foresee handling software patents in your project, here are some aspects that you should be aware of.

---

33   Read more: http://en.swpat.org/wiki/Software_patents_wiki:_home_page
34   Read more: Meeker, Heather. *Open Source for Business.* 2.ed. Fleming, 2017. p. 153-178.

While traditional patent licenses usually seek to slice the patent rights granted into narrow pieces, FOSS licenses are purposely broad and vague, mainly because if someone distributes Free Software and holds software patents, a absurd situation may arise where parties may use, study improve and share the software, but simultaneously be prohibited from using the software by the software patent.

Therefore, in general terms, FOSS licenses can:

- **Grant patent rights**: they require that distributors of a software give recipients a licence to use any necessary patents. Some licenses contain expressed grant but other grant patent rights by implication. Even in the absence of an express patent grant, all FOSS licenses grant patent rights in some extension. An FOSS patent license grant has only one field limitation, which is that the right is granted only in connection with the exercise of the copyright granted for the software. Any other field limitation, such as territory, commercial or technology fields is against Free Software definition and therefore not included.
- **Retaliate in case of patent aggression**: the license makes patent aggression less attractive by revoking patent rights that any aggressor received through the license. A **defensive termination** revokes the patent grants in case if someone exercising the license brings a claim accusing the licensor of patent infringement.

How do I know if my software is already patented?

Patent is an monopoly on using an idea. Software patents are exercised over the **functionality** in which the software is intended to work. It is independent of the code. Therefore it is very hard to determine if a software that you are developing has patented parts on it. That's why GPL v.3 states in its preamble that every program is threatened constantly by software patents.

A detailed search for patented software can be costly and involve lawyers and patent experts. It is quite common for developers to do a patent search and find nothing even when there are things that could be found by a professional searcher.

If you are going to do your own preliminary patent searching, you can start with the website of the European Patent Office[35], where useful information can be found.

Nevertheless, there are also initiatives to minimize patent aggression towards the Free Software community, for instance the Open Invention Network[36], which enforces a license agreement among organisations in support of patent non-aggression and for free access to OIN's patents.

---

35 European Patent Office: https://www.epo.org/searching-for-patents/helpful-resources/first-time-here.html
36 Open Invention Network: https://www.openinventionnetwork.com/

Does my FOSS license provide patent grants and/or retaliation clause?

The following table[37] on the next page lists the patent grants terms and defensive termination provisions of the recommended FOSS licenses. Please note that some license text excerpts were edited for better reading.

License/ SPDX code	Grant Patent Rights	Termination Trigger	Rights Terminated
GNU GPLv3 GPL-3.0-only	**Section 11** **Express grant** *Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.*	**Section 10** *You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may (…) initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim in infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.*	**Section 8** *All rights can be terminated, including any patent licenses granted.*
European Union Public License v1.2 EUPL-1.2	**Clause 2** **Express grant** *The Licensor grants to the Licensee royalty-free, non-exclusive usage rights to any patents held by the Licensor, to the extent necessary to make use of the rights granted on the Work under this Licence.*	**Clause 5** *The Licensee (becoming Licensor) cannot offer or impose any additional terms or conditions on the Work or Derivative Work that alter or restrict the terms of the Licence.*	**Clause 12** *The Licence and the rights granted will terminate automatically upon any breach by the Licensee of the terms of the Licence.*
GNU AGPLv3 AGPL-3.0-only	**Section 11** **Express grant** *Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.*	**Section 10** *You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may (…) initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim in infringed by making, using, selling, offering for sale, or importing the Program or any portion of it*	**Section 8** *All rights can be terminated, including any patent licenses granted.*
GNU LGPLv3 LGPL-3.0-only **Note:** This license incorporates the terms and provisions of **GNU GPLv3.**	**Section 11 (GNU GPLv3)** **Express grant** *Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.*	**Section 10 (GNU GPLv3)** *You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may (…) initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim in infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.*	**Section 8 (GNU GPLv3)** *All rights can be terminated, including any patent licenses granted.*
Mozilla Public License 2.0 MPL-2.0	**Clause 2.1** **Express grant** *Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license: under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.*	**Clause 5.2** *If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent.*	**Clause 5.2** *The rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.*
Apache License 2.0 Apache-2.0	**Clause 3** **Express grant** *Each Contributor grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,*	**Clause 3** *If You institute patent litigation against any entity (including a cross-claim or counterclaim in a law suit) alleging that the*	**Clause 3** *Any patent licenses granted to You under this Licese for that*

---

37   Table built according to Meeker, Heather. *Open Source for Business.* 2.ed. Fleming, 2017. p. 167-169.

	irrevocable patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work,where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) (...)	*Work ir a Contribution incorporated within the Work constitutes direct or contributory patent infringement.*	*Work shal terminate as of the date such litigation is filed.*
MIT License MIT **Note:** The MIT license does not include an express patent license. However, there are opinions that supports the interpretation of implied patent grant provision.[38]	**Implied grant** *The permission is here granted to deal in the Software without restriction, and to permit persons to whom the Software is furnished to do so (…)*	**General termination conditions** No copyright notice and permission notice included in all copies or substantial portions of the Software.	**General termination of rights** The grants are subject to the license's conditions.

Summary

Although highly controversial, software patents are a reality that many projects should face. In this section you will learn how software patents are covered by the FOSS licensing terms.

---

38   Read more: Peterson, Scott. *Why so little love for the patent grant in the MIT License?,* 2017. https://opensource.com/article/18/3/patent-grant-mit-license