Open Source Software Licences User Guide

If you are working on a piece of software and you want to make it open source (i.e. freely available for others to use), there are questions you may need to consider, for example, whether you need a licence and why; and if so, how to choose a licence. This guide will help with those questions so that you have a better idea of what you need for your particular project.

What is Open Source Software? 1

Why do I need a licence for Open Source software? 1

How do I choose a license? 2

What if I have dependencies or my work is a derivative? 5

Combined Works and Compatibility 5

What if I want to change licenses? 6

What if my software is protected by a patent? 6

Can I commercialise Open Source Software? 6

Who should I speak to if I need help? 7

Additional information 7

Summary table 9

What is Open Source Software?

Generally, Open Source Software is software that can be freely accessed, run, used, changed, and shared (in modified or unmodified form) by anyone. To be useful, open source software *should* be distributed under licences that comply with the Open Source Definition[1].

Open Source software is sometimes known as Free Software[2]. The two terms are essentially interchangeable, some people use both terms, choosing according to context and audience. Some people also prefer to use the term Free and Open Source Software (FOSS) or Free/Libre and Open Source Software (FLOSS)[3].

Open source licences do not restrict anyone from using the software, and do not restrict how people use them. All Open Source software can be used for commercial purposes, though note, depending on the licence, it may affect the business model.

Why do I need a licence for Open Source software?

If you intend to make your software open source, the first question you may ask is whether you need a licence. A short answer is yes, it is recommended that you publish your work under a licence otherwise it is unclear to others whether they are allowed to use it. Not attaching a licence to your work does not mean you grant permission for others to use your work. By default, when you create a

[1] Open Source Definition https://opensource.org/osd

[2] Free Software Definition https://www.gnu.org/philosophy/free-sw.html; https://www.fsf.org/

[3] https://www.gnu.org/philosophy/floss-and-foss.en.html

code or other creative works, only you have exclusive copyright of the work. No one can use, copy, distribute or modify your work without a licence. Attaching a licence to your work also enables further collaboration within the community.

There are many open source licences available and it can be difficult to determine which one you need. The overview below will help you to make decisions when you need to choose a licence that is appropriate for you.

How do I choose a license?

Although there are many open source licences[4], six widely used licenses represent over 85% of Open Source Projects. Below we discuss theses most widely used licenses. These licenses can be categorised into permissive ("anything goes"), copyleft and weak copyleft. A more detailed comparison of these licenses can be found in the summary table.

Permissive ("Anything Goes") Licences – these place very few restrictions on what can be done with the code, including using the code in proprietary derivative works. They require only attribution in a specified manner. The most widely used licences of this type are:

MIT license[5]

- The most permissive, simple, and popular open source license
- Users can use, copy, modify, distribute, sublicense, and sell copies of the software as long as they include the license and copyright notice
- Compatible with GPL licenses

3-Clause BSD License[6]

- Users can freely modify and distribute your software's code as long as a copy of the copyright notice, list of conditions, and the disclaimer are retained
- Differs from MIT in that it prohibits others from using the name of the project or contributors to promote or endorse derived products
- Compatible with GPL licenses unlike the original 4-clause BSD license which had a problematic advertising clause

Apache 2.0[7]

- Similar to the MIT License but with some additional restrictions and clarifications
- Explicit patent grant: Gives licensees a license to any patents necessary to run the software
- Due to patent grant clause, only compatible with GPLv3 and not GPLv2

Copyleft ("viral") Licences – these also allow open distribution, modification, and re-use of the code (with attribution), but insist that any derivative works be distributed under the same terms. Thus, proprietary derivatives by third parties are not possible (unless the copyright holder gives permission).

GNU General Public License v2 (GPLv2)[8]

[4] List of open source software licences please see https://opensource.org/licences/alphabetical
Licences in different categories please see https://opensource.org/licences/category

[5] https://opensource.org/licenses/MIT

[6] https://opensource.org/licenses/BSD-3-Clause

[7] https://opensource.org/licenses/Apache-2.0

[8] https://opensource.org/licenses/GPL-2.0

- Derivative works must also be licensed under GPLv2, which requires that complete source code be distributed to licensees. For example, a derivative work comprising MIT License, GPL 2.0, and your own code would have to be released under GPL 2.0

GNU *General Public License v3 (GPLv3)*[9]

- Same spirit as GPLv2, but updated to include technical and legal developments and address ambiguities
- Explicit patent grant, whereas GPLv2 gave implicit patent grant
- Allows Apache 2.0 work to be combined with GPLv3 and released under GPLv3 only

GNU Affero GPL v3 (AGPLv3)[10]

- Strongest copyleft license
- Effectively the GPLv3 with additional clause in which network use is considered distribution ensuring modifications intended for network use are also freely available.

Weak Copyleft Licenses - A weak copyleft license is similar to a strong copyleft license, but is not considered "viral" and is therefore often used for libraries. Modifications to the code itself must retain the original license but develops can use the weak copyleft protected library in proprietary software. There are fewer examples of these but the most noteworthy is:

GNU Lesser Public License (LGPL)[11]

- Sometimes still referred to by its original name, GNU "Library" General Public License, since it is most used in shared libraries
- Non-LGPL works may link to LGPL libraries and be distributed under non-LGPL terms as long as the new work is not a derivative work and that the library is prominently attributed
- Like with GPL, the source code of the LGPL library must always remain open

Mozilla Public License 2.0 (MPL2.0)[12]

- Similar to LGPL but does not have any attribution requirements; you may redistribute MPL projects, and link to functions within an MPL project, without any need to announce that you're doing so.
- MPL 2.0 is compatible for use with GPL-licensed projects; prior versions were not.

Choosing a license

The choice of the licence depends on what effect you want. If you are comfortable with users coming along and using the code in potentially proprietary products, then use the 3-clause BSD licence from the "anything goes" category. It is short and simple, compatible with all the other licences, and avoids controversy.

On the other hand, if you don't want proprietary derivatives from other parties appearing, or if you want the software to remain available to the research community, then use a copyleft licence: the GPLv3 or the AGPLv3. Alternatively, use the LGPL for libraries.

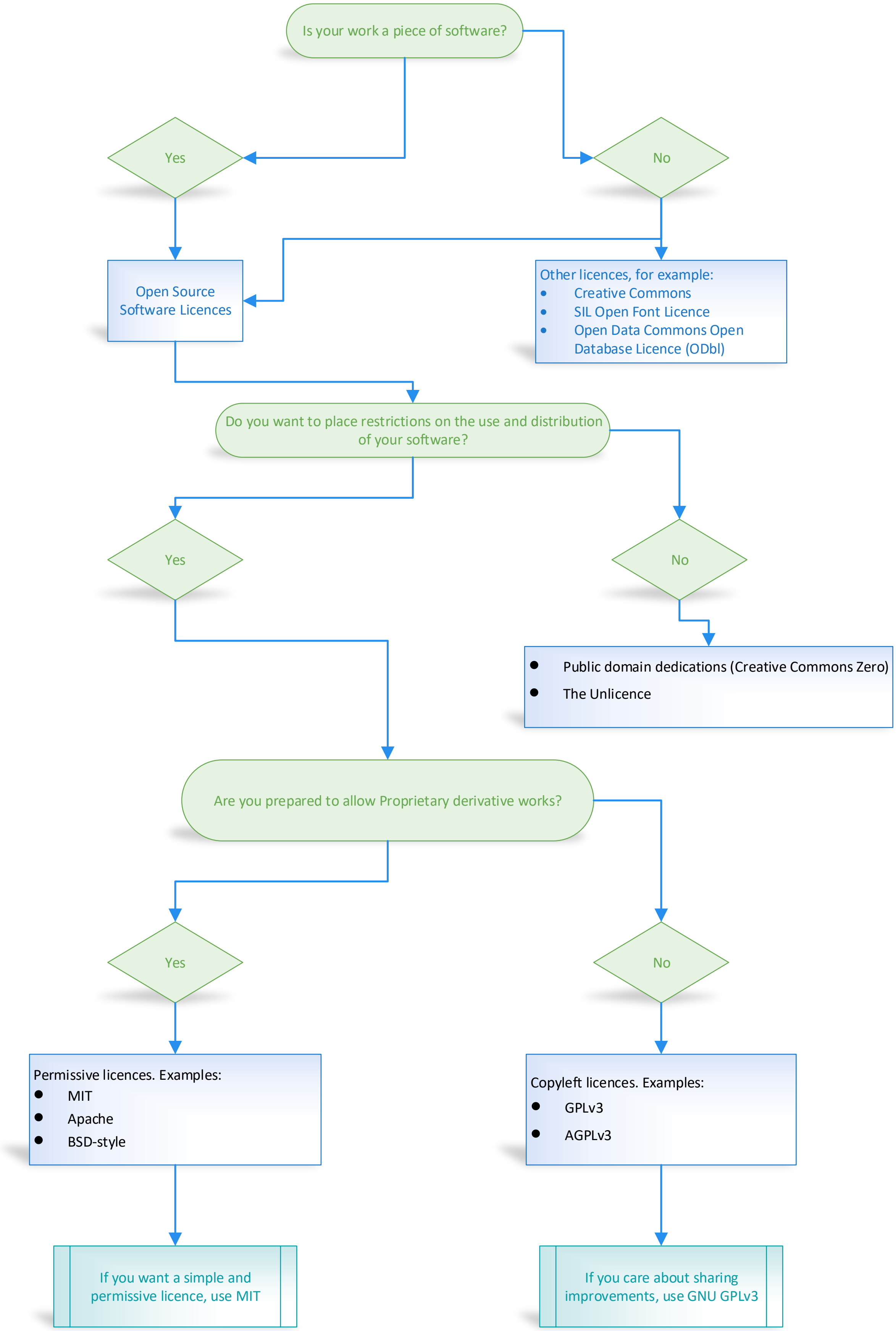
[9] https://opensource.org/licenses/GPL-3.0

[10] https://opensource.org/licenses/AGPL-3.0

[11] https://opensource.org/licenses/LGPL-2.1; https://opensource.org/licenses/LGPL-3.0

[12] https://opensource.org/licenses/MPL-2.0

Here is a decision tree that helps you to choose a licence.



Is your work a piece of software?

Yes

No

Open Source Software Licences

Other licences, for example:

- Creative Commons
- SIL Open Font Licence
- Open Data Commons Open Database Licence (ODbl)

Do you want to place restrictions on the use and distribution of your software?

Yes

No

- Public domain dedications (Creative Commons Zero)
- The Unlicence

Are you prepared to allow Proprietary derivative works?

Yes

No

Permissive licences. Examples:

- MIT
- Apache
- BSD-style

Copyleft licences. Examples:

- GPLv3
- AGPLv3

If you want a simple and permissive licence, use MIT

If you care about sharing improvements, use GNU GPLv3

IMPORTANT: Another important factor in choosing a license is do you have any dependencies and/or is your project a derivative work? If the answer is yes to either or both questions, you must consider what licenses have been used in these works to ensure the one you choose for your work abides by their conditions. The following section discusses this subject in more detail.

What if I have dependencies or my work is a derivative?

Unless you have started completely from scratch, your project very likely has dependencies. Each of those libraries you depend on will have its own open source license. If each of their licenses is “permissive”, you can use any license you want. Common permissive licenses include MIT, Apache 2.0, and BSD.

On the other hand, if any of your dependencies’ licenses are “strong copyleft”, then your project will have to use the same license. Common strong copyleft licenses include GPLv2, GPLv3, and AGPLv3.

If you have derived your work from a GPL licensed software, you are bound by the GPL license and therefore must use the same license. So, what is a derivative work? This is a matter that is still up for debate, both within the Open Source community and within the courts. The following are the most generally held viewpoints:

Derivative work	Not derivative work
Modification of the source code of the GPL-licensed software. OR Embedding your code into the GPL code, or vice versa, such that in order to run the work, both codes must be distributed together.	A work that may be distributed separately from an unmodified GPL work or library and still function.

Due to the ambiguity surrounding GPL and derivative works, if you do not wish to release your software under GPL, you should create as much separation as possible between your non-GPL work and the GPL work. For example, when distributing your non-GPL software, you can inform users that they may download the unmodified GPL-licensed modules from third party locations, and your software will compile the modules when the user runs the program.

What if an Open Source license is not specified for a project? This is a common scenario, as found in a GitHub survey in which many developers have either specified a different license or has not specified one at all. If the work is under a non-Open Source license, you must follow that license. If no license has been specified, then the work is still under copyright and you would need to reach out to the author regarding permissions.

Combined Works and Compatibility

What do we mean when we say that two licenses are compatible? For example, Apache 2.0 is compatible with GPLv3 but not GPLv2, and 3-Clause BSD is compatible with all GPL whereas the original 4-clause BSD is not.

Compatibility means that if two code components with different licenses are combined to form a new work, conditions from both licenses can be satisfied. Conversely, if there is no way to satisfy both licenses at once then they are incompatible and the two components with their current licenses cannot be linked.

For example, the GPL says that a combined program including a GPL licensed component must be released under the GPL. Even though the *entirety* of the combined program must be licensed as GPL, you may still license your component with a more permissive, but GLP-compatible, license to give additional permissions for your specific component.

Permissive licenses are generally compatible with each other whilst copyleft licenses are more restrictive. The original 4-clause BSD license included a clause for advertising which was not compatible with conditions set out in GPL licenses. The 3-clause BSD, which does not contain this incompatible clause, is therefore recommended for its increased compatibility. MIT is compatible with GPL licenses whilst Apache 2.0 is only compatible with GPLv3 and not GPLv2 due to the restriction that terminates the grant of patent rights if the licensee sues over patent infringement.

Compatibility issues can be avoided by separating components and distributing only your code, relying on the user to separately license the other component. By not distributing the two components together, you are allowed to use a different license including proprietary licenses.

Example: Open University VLE and Moodle

The Open University uses Moodle as the basis of its VLE, and Moodle is licenced under the GPLv3. The Open University is allowed to develop proprietary extensions to Moodle that are hosted by the OU and used for paying courses. However, it is not allowed under the GPL to sell a bundled version of Moodle with those proprietary extensions or distribute that software bundle under any licence except a GPL one.

What if I want to change licenses?

Once a licence is granted, you cannot revoke the licence from the previously distributed version. Therefore, you should choose carefully which licence you want to use. However, it is possible to change the licence for new versions.

Additionally, it is possible to release several versions with different licenses. For example, Mozilla chose to dual license Rust with Apache 2.0 and MIT which would give people the option to use the MIT licensed version if they wanted to combine Rust with GPLv2 code.

What if my software is protected by a patent?

Several licenses such as GPLv2 and MIT predate the rising popularity of software patents and therefore do not explicitly grant or prevent users from rights to use associated patents. It is still unclear if these licenses give an implicit right to use patents. To overcome this ambiguity, newer licenses have been given additional clauses explicitly granting or denying rights to use patents.

If there is patent protection on your software, you should consider if you want to use a license with explicit grant of patent rights. If you envisage large businesses using your software, then they will likely want express patent license. In this case, Apache 2.0, GPLv3, AGPLv3, MPL2.0, and LGPLv3 have you (and them) covered.

Can I commercialise Open Source Software?

Open Source Software can be used for commercial purposes and can also be sold. However, depending on the license, when you distribute your software you must make the source code available and therefore you probably cannot stop customers from redistributing or selling it in the same manner as you.

Despite this, there are examples of companies that have built up a successful business from an open source product. Commonly used methods include selling proprietary modules, releasing several licensed versions, providing professional services and software as a service offering.

Who should I speak to if I need help?

If you need further information or have enquiries about IP, please contact RES-Intellectual-Property@open.ac.uk.

Additional information

Software in the "public domain" vs. Open Source[13]

"Public domain" is a technical term in copyright law that refers to works not under copyright. Not all legal jurisdictions have a public domain, and it doesn't always mean exactly the same thing in the jurisdictions that do have it. It is recommended that you always apply an Open Source licence to software you are releasing, rather than try to waive copyright altogether. Using a clear, recognized Open Source licence actually makes it easier for others to know that your software meets the Open Source Definition. It also enables the protection of attribution, and various other non-restrictive rights, that cannot be reliably enforced when there is no licence.

There are certain circumstances where it is not easy to apply a licence, and the software must be released into the public domain. For example, works authored by U.S. government employees, on government time and as part of their job, are automatically in the public domain. Such software is effectively open source, or open source for most practical purposes, even though it is not officially released under an open source licence.

Notes

Open source software licences can be also used for non-software works, for example, when the works can be edited and versioned as source such as open source hardware designs. If your project contains a mix of software and other material, you can include multiple licences and explicitly note which licence applies to each part of the project.

Take notice of Terms of Services where you publish your work. E.g. for public repositories on GitHub, by accepting their terms of services you allow others to view and fork[14] your repository.

Open Government Licence[15]

The National Archives in the UK has developed the Open Government Licence as a tool to enable information providers in the public sector to licence the use and re-use of their Information under a common open licence. The National Archives invites public sector bodies owning their own copyright and database rights to permit the use of their information under this licence. The Licence covers information protected by copyright or by database right (for example, literary and artistic works, content, data and source code).

[13] For a detailed discussion of the complexities of the public domain and open source, search for the words "public domain" and "PD" in the subject headers of the January 2012, February 2012, and March 2012 archives of the OSI Licence Review mailing list.

[14] Fork https://guides.github.com/activities/forking/

[15] https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/

Creative Commons[16]

Creative Commons licences give everyone from individual creators to large institutions a standardized way to grant the public permission to use their creative work under copyright law. The licence is **not** applicable to the copyright of software and source codes, however you can release other related documents (e.g. reports) under this licence.

Additional Resources

To find out more details on open source licences please see: https://choosealicense.com/licenses/ https://choosealicense.com/appendix/

To find out information on license-usage statistics please see:
https://github.blog/2015-03-09-open-source-license-usage-on-github-com/ (2015)
https://resources.whitesourcesoftware.com/blog-whitesource/open-source-licenses-trends-and-predictions (2019/20)

[16] https://creativecommons.org/about/cclicences/

Summary table

A comparison of commonly used licenses arranged in order of least to most permissive

	License	Clause explicitly granting patent use	Required to disclose source code upon distribution	Network use counts as distribution	Viral copyleft condition	Changes to licensed material must be stated
Copyleft	AGPLv3	Yes	Yes	Yes	Yes	Yes
	GPLv3	Yes	Yes	No	Yes	Yes
	GPLv2	No	Yes	No	Yes	Yes
Weak copyleft	LGPLv3	Yes	Yes	No	Yes, unless used as library	Yes
	MPL2.0	Yes	Yes	No	Yes, unless used as part of larger work	No
Permissive	Apache 2.0	Yes	No	No	No	Yes
	3-clause BSD	No	No	No	No	No
	MIT	No	No	No	No	No