Mend.io

WHITEPAPER

The Complete Guide for Open Source Licenses 2024

Table of contents

01

Introduction

The present state of most technology rests on a vast number of free and freely available open source programs and libraries, the products of untold hours of hard work and dedication by software engineers. It's truly a beautiful thing.

But standing on the shoulders of these giants isn't always so easy or straightforward. Open source licenses can be more than a little confusing for those that just want to write some code. But with open source components playing such a big part in the products that we create, open source licenses and compliance simply can't be ignored.

We've compiled this one-stop resource guide for working compliantly with open source components, including answers to FAQs about the most popular licenses in 2023.

02

Open Source Licensing 101

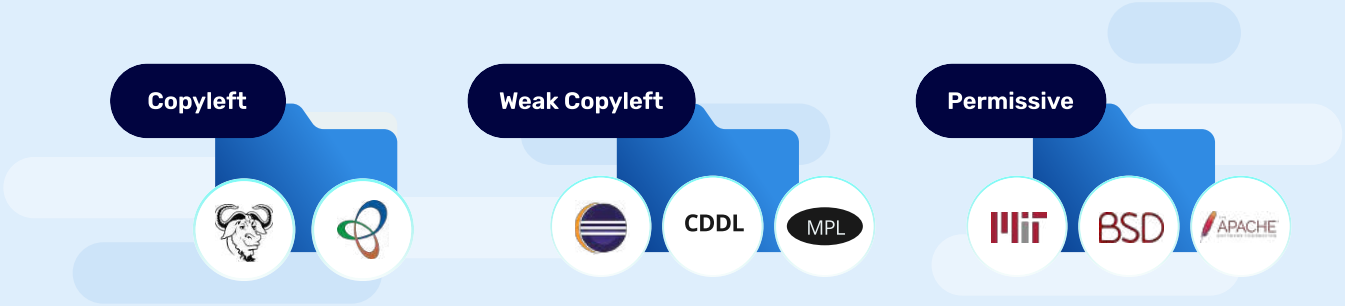
Similar to a painting, a song recording, or a movie script, software is considered a work of art in the eyes of the law. As such, sharing software with the rest of the world does not mean anyone can download it or use it however they please. Anyone who wishes to use software has a legal obligation to receive approval from the holder of the software's copyrights.

An open source license is a binding legal contract between author and user that declares the certain conditions in which a piece of software can be used in commercial applications. This license is what turns software components into open source components, allowing developers to use that software so long as they uphold the specific terms and conditions laid out in the license.

Each open source license lays out in its terms and conditions what users are permitted to do with the software components, what they are not permitted to do, and what their obligations are when using said components. While this sounds straightforward, there are over 200 open source licenses, each one varying in complexity and requirements. Staying compliant is up to each developer or organization who must choose which components are most compatible with their policies and existing software.

03

Types of Open Source Licenses: Copyleft vs. Permissive



Based on the requirements and restrictions they place on users, open source licenses can be divided into two major categories: copyleft and permissive.

In law, copyrights restrict the right to use, modify, and share creative works without the permission of the copyright holder. Under a copyleft license, an author makes a claim on the copyright of the work and declares that other people have the right to use, modify, and share the work as long as the reciprocity of the obligation is maintained. In other words, if someone uses a component with this kind of open source license, then they too must offer a copyleft license on their final product and make their code open for use by others, which is why some refer to copyleft licenses as "viral".

On the other side of the open source license coin are permissive licenses. A permissive open source license is a non-copyleft open source license that guarantees the freedom to use, modify, and redistribute while also permitting derivative works that are proprietary. These licenses place few if any restrictions on how others can use open source components meaning a user is granted varying degrees of freedom to use, modify, and redistribute open source code plus permission to create proprietary derivative works – all while requiring nearly nothing in return in terms of obligations moving forward.

No one license is better than another and there are no “good” or “bad” licenses. Anyone can create an open source license that suits their fancy, which is why there are so many of them out there. Luckily, out of the 200+ licenses that exist, only a handful are commonly used.

Top Licenses at Glance

License	Type	Must distribute source code	Distribute modified code under same license	Distribute entire project that uses licensed code under same license	Other notable items	Patent grant
LICENSE	Permissive	✗	✗	✗	Requires notice of significant modifications	✓
BERKELEY SOFTWARE DISTRIBUTION (BSD) 3-CLAUSE	Permissive	✗	✗	✗		✗*
COMMON DEVELOPMENT & DISTRIBUTION LICENSE (CDDL)	Weak copyleft	✓	✓	✗	Requires notice identifying author of modifications	Yes, if code is used unmodified
ECLIPSE PUBLIC LICENSE (EPL) V2.0	Weak copyleft	✓	✓	✗		✓
GNU AFFERO PUBLIC LICENSE (AGPL) V3.0	Strong copyleft	✓	✓	✓	Interaction with software over a network counts as distribution	✓
GNU GENERAL PUBLIC LICENSE (GPL) V2.0	Strong copyleft	✓	✓	✓		✓
GNU GENERAL PUBLIC LICENSE (GPL) V3.0	Strong copyleft	✓	✓	✓		✓
GNU GENERAL PUBLIC LICENSE WITH CLASSPATH EXCEPTION	Weak copyleft	✓	✓	✗		Yes, if based on GPL v3
GNU LESSER GENERAL PUBLIC LICENSE (LGPL)	Weak copyleft	✓	✓	✗		Yes, if based on GPL v3
MICROSOFT PUBLIC LICENSE (MS-PL)	Weak copyleft	✗	Yes, if source code is distirbuted	✗		✓
MIT LICENSE	Permissive	✗	✗	✗		✗*

* Patents are not mentioned in the original licenses; however, versions of these licenses with patent grants added exist.

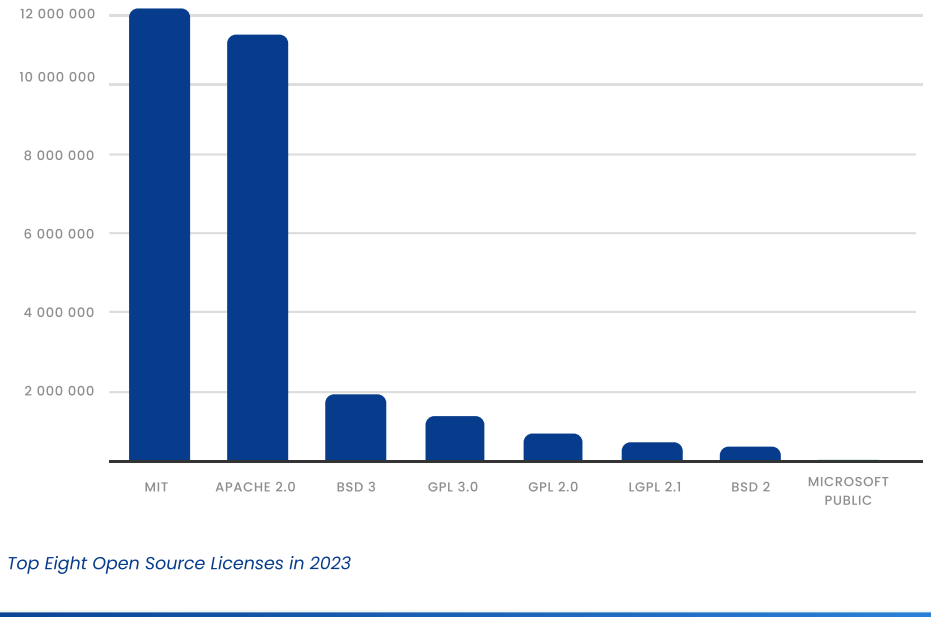
** All licenses in this chart permit commercial use and all require a copy of the license in distribution.

04

Top Open Source Licenses in 2023

We took a close look at the trends of open source license usage in 2023 and compared them to previous years.

Our research team has collected information from the Mend.io database, over 37 million open source packages in 2023 alone, covering over 200 programming languages, to learn which were the most popular open source licenses in 2023, compared to previous years. Results show that use of permissive open source licenses continues to rise, while usage of copyleft licenses, and the GPL-family in particular, continues to decrease.



Top Eight Open Source Licenses in 2023

Permissive Open Source Licenses Continue to Dominate

Together, the MIT, Apache 2.0, and BSD licenses were used for about 70% of all open source projects with a license in 2023. Permissive open source licenses place minimal restrictions on how others can use open source components so this is great news for legal departments as it greatly minimizes the challenges of open source licensing.

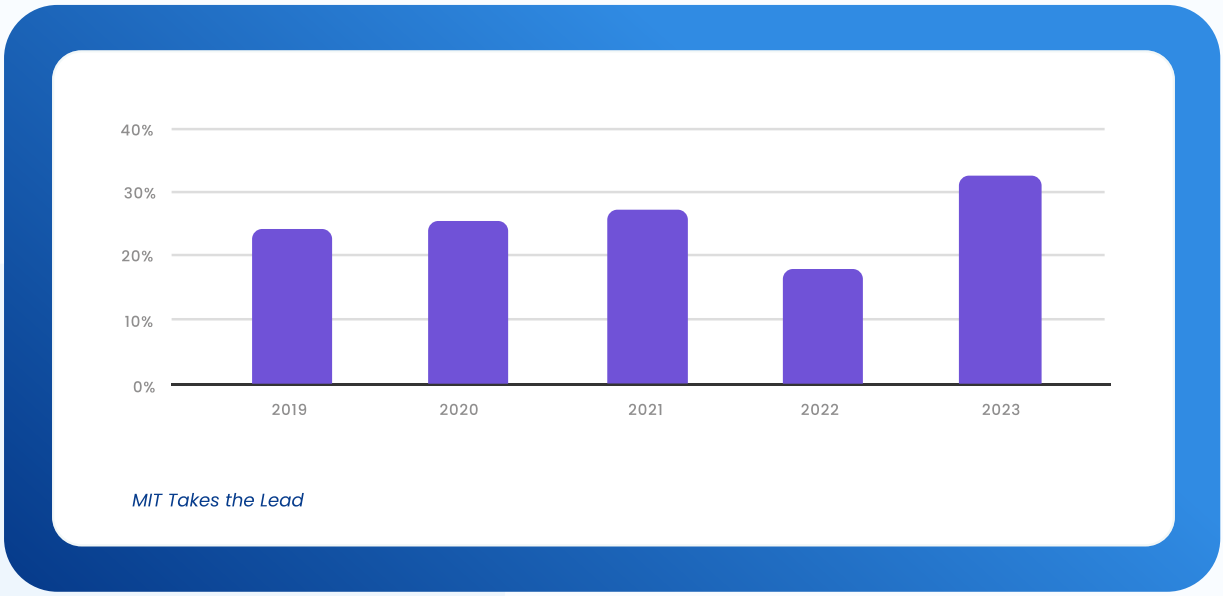
The sizable increase in permissive license use over the last decade comes as no surprise. The software development industry has gone from being wary of open source to enthusiastically supporting it. Today, industry giants sponsor some of the most popular open source projects, and the skepticism and rivalry that once ruled the tense relationship between the commercial and open source communities have turned. Microsoft and many others have wholeheartedly chosen the path of 'if you can't beat them, join them' – or acquire them, as the case may be. In the interest of this widespread cooperation and encouraging open source usage, permissive licenses are winning.

Permissive vs. Copy-left Licenses Over Time

MIT Takes the Lead

Covering just shy of a third of all open source software, the MIT license was the champion in 2023 – though it's still neck and neck with Apache 2.0, a similarly permissive license and last year's winner. Ben Balter, attorney, open source developer, and senior product manager at GitHub, said that developers choose the MIT license because "It's short and to the point. It tells downstream users what they can't do, it includes a copyright (authorship) notice, and it disclaims implied warranties (buyer beware). It's clearly a license optimized for developers. You don't need a law degree to understand it, and implementation is simple."

Popular open source projects licensed under MIT include Python, Ruby on Rails, TensorFlow, React, and Node.js.

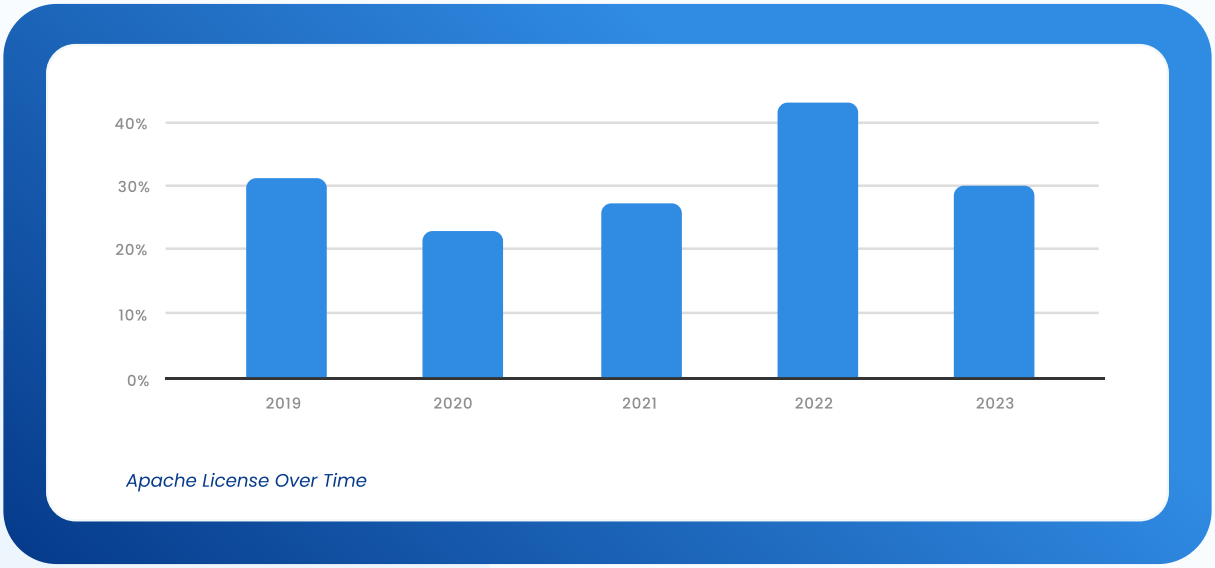


MIT Takes the Lead

Apache 2.0: Still Strong

Second place by just a few percentage points, the Apache 2.0 license is still a top choice for open source software creators. Those looking for wide adoption of their open source software often choose permissive licenses like the MIT and BSD licenses for their brevity and ease of understanding, but the Apache 2.0 license is usually the preferred permissive license by legal experts because of its explicit grant of patent rights and spelled out terms that avoid misinterpretation.

A number of popular open source projects use the Apache 2.0 license, including Kubernetes, OpenSSL, Swift, and Rust.



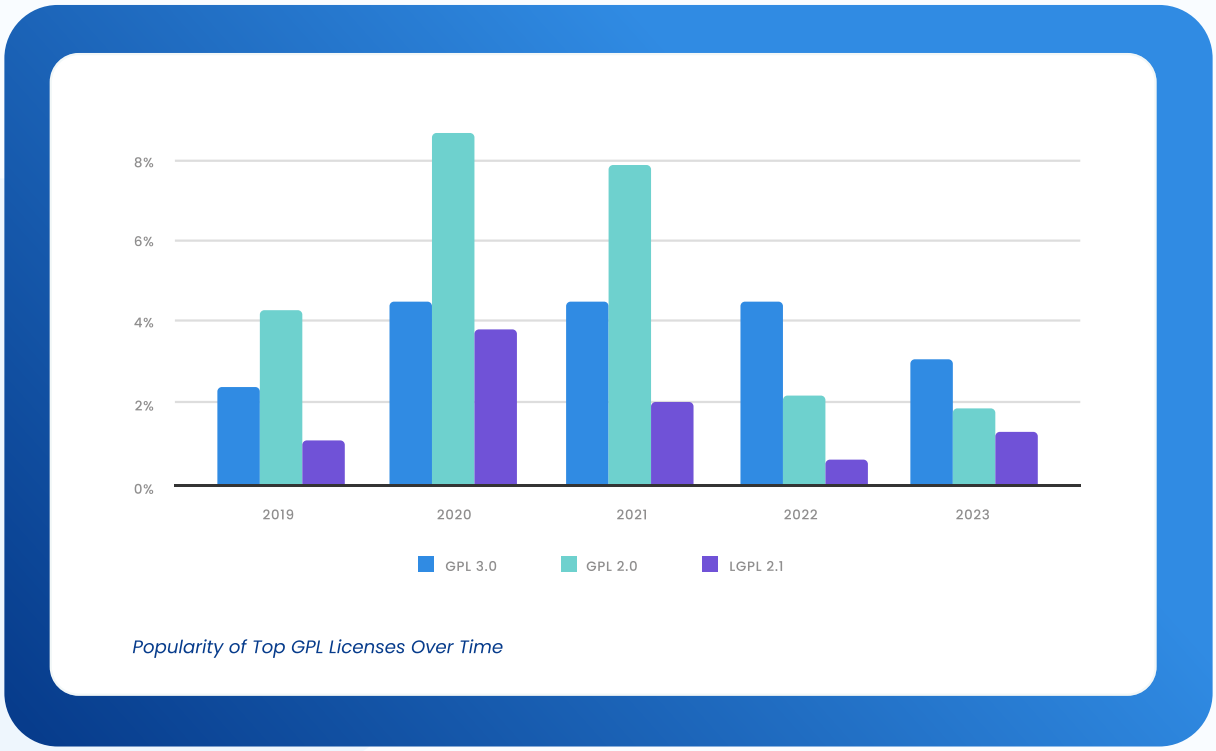
Apache License Over Time

The GNU GPL Family and Other Copyleft Licences In Major Decline

Just three years ago, GNU GPL family licenses covered nearly 18% of all open source projects with a license. Today, they are down to just over 6%. Add in the Microsoft Public license, the 8th most popular open source license, and the top copyleft licenses still don't reach 7%.

The GPL was a trailblazer at the start of the open source revolution and is the OG of the copyleft or viral license. It uses U.S. and international IP law to enforce a philosophy of sharing and freedom in software, which can be a headache for businesses.

There will always be GPL users. It's the Linux kernel license of choice, supported by a huge open source community. However, it's clear at this point that business-wise, the preference is for licenses with fewer restrictions and limitations.



Popularity of Top GPL Licenses Over Time

05

Open Source Licensing in 2024: What's Next?

Over the years, the numbers have consistently shown that copyleft license use is shrinking and permissive license use is expanding. We don't expect that to reverse in 2024.

The tension between creating a viable business model and maintaining a robust and successful open source project continues to grow. We will continue to see open source projects struggling to find the balance between making a profit and being supportive members of the open source community.

Big companies and startups alike contribute open source projects and as their interests change, so may their licenses change as they seek to increase revenues and block their competition from using their technology. Of course, we will also see the community engaged in lively debates whenever larger enterprises update their open source offerings claiming they can't afford to give away their work.

In a similar vein, one trend that is on the rise (but may eventually reverse course due to community backlash) is the use of "open source" for projects that are not, in fact, open source. This marketing ploy has been used by a few companies to drive use of and contribution to their projects while restricting those contributors in ways that no license accepted by OSI or the open source community at large ever would.

In past years we have lamented the likelihood that some number of hard-working unpaid creators and maintainers of small and critical projects will abandon their projects due to burn-out or update their license for a better business model. Though that's still a possibility, this year there are reasons to be a little more optimistic. Extra support for the open source community is not just coming from major companies; governments around the world are opening their eyes to the importance of open source software and are stating their intentions to assist the open source community in keeping open source code secure.

While open source license trends may rise and fall, one thing is certain: open source software isn't going anywhere any time soon.

06

Top Open Source Licenses – Cheat Sheet

GNU General Public License (GPL)

Once the most popular open source license, the GNU's General Public License is a specific implementation of the "copyleft" concept created by Richard Stallman to prevent GNU software from becoming proprietary.

Because GPL is a copyleft license, any software that is written based on any GPL component must be itself released as open source. Projects that use these components, no matter the percentage of total code, are legally required to release their complete source code plus all of the rights to modify and distribute that code. GPL-based licenses with some exceptions exist, one of which will be covered in the next section.

Here are some answers to common questions about GPL:

1. What are the GPL terms and conditions?

Upon using a GPL'ed component in your software, that entire software is now considered a 'work based on' that GPL'ed component and therefore you are not allowed to claim patents or copyright on the software. Additionally, you are obligated to display a copyright notice, disclaimer of warranty, intact GPL notices, and a copy of the GPL notice in your distribution. Your software is also now under the reciprocity obligation meaning you must release the source code and all of the rights to modify and distribute that code. Changing the license or introducing additional terms and conditions is not allowed.

For all but one version of the GPL, the reciprocity obligation is triggered by distribution. If you use or modify GPL'ed code in a larger work, you are required to release the entire source code, license, and notices to all who receive the software. In the case of internal applications that are never available to the public, this may be simply the employees who use it. The exception to this rule is the Affero License or AGPL. The AGPL was created to prevent the use of GPL'ed code in proprietary SaaS and other network-available code that is never technically "distributed" to the customer and therefore does not trigger the reciprocity obligation of the other GPLs which were not written with this technology in mind.

2. Is the GPL enforceable?

Yes, as a valid copyright license, the GPL is enforceable. The copyright holders of the GPL'ed software can enforce the GPL on distributed or derivative works of the software if they choose to. As an example, The Free Software Foundation (FSF) is the copyright holder on many pieces of the GNU system including the GNU Compiler Collection. As such, the FSF has the authority to enforce the copyleft requirements of the GPL on any copyright infringement that occurs on that software.

3. What is the difference between the GPLv2 and the GPLv3?

The GPLv3 is nearly twice as long as the GPLv2 and most of that added length is dedicated to clarifying the same terms and conditions that are in the GPLv2, specifically what constitutes a 'work based on' another work that triggers the GPL reciprocity obligation. The GPLv3 also includes language changes to make the FSF's intentions more likely to be accurately interpreted by international courts.

Two updates in the GPLv3 are true changes to the functionality of the license. The first is that the GPLv3 discusses patent rights in detail thus introducing compatibility with the Apache 2.0 License. The second is that the GPLv3 prevents the use of hardware to circumvent the GPL's core intentions. That means that if GPL'ed code is put on a personal device it must include not simply the source code but instructions on how that code can be modified and reinstalled onto the device.

4. Can you mix a GPL with other licenses?

While it is often believed that code covered by the GPL cannot be mixed with code covered by other open source software licenses, it is sometimes possible under both GPLv2 and GPLv3. The new language in the GPLv3 establishes more clearly when this is permissible and the FSF has explicitly stated that the GPLv3 is compatible with the Apache 2.0 license.

GNU GPL with Classpath Exception

There are cases where an author can choose to release their code under the GNU GPL with a classpath exception. Here are the answers for the most common questions regarding this exception:

1. What is the GNU classpath exception?

Every work based on a program with a GNU GPL – that is, every derivative of the original program or any modifications one writes to it – is subject itself to the GPL. As such, your original code may be covered by the GPL if you combined it with a GPL module.

A GPL license with a classpath exception permits linking a GPL library with an independent module "which is not derived from or based on the library," without the resultant program becoming subject to the GPL. The classpath exception enables certain uses of GPL licensed components without risking the integrity of your proprietary program and intellectual property.

The resulting executable can thus be copied and redistributed under any license you choose so long as you meet the terms and conditions that govern the existing modules you're using.

Summed up: the classpath exception exempts you from the obligation of releasing your project under the GNU license if you link to a GPL with classpath exception library, saving you from having to publicly share your entire source code and licensing it under a GPL.

2. How should I link a GPL with classpath exception components to my software?

A GNU GPL with a classpath exception permits both statically and dynamically linked modules.

3. Do I have to extend the classpath exception downstream?

If you use the GPL library with a classpath exception as it is, then it must stay under its original license which includes a classpath exception. If you do make changes to said library, then you must release the modified version of the library under the same GPL, but it is up to you if you want to keep the classpath exception. Of course, if you do not include the classpath exception, using that library in a larger program would create an obligation for that program to be licensed under the GPL.

4. Who can apply the classpath exception to a library?

The copyright owner – usually the developer of the library – chooses whether or not the program is released under a GPL with a classpath exception.

5. What's the difference between the GPL with a classpath exception and the GNU GPL as such?

The classpath exception is language allowing for developers to link to libraries without subjecting the "derived" result to the terms and conditions of a GPL under certain circumstances. This exception is added to, not a replacement of, a GNU GPL.

GNU Lesser General Public License (LGPL)

The LGPL is considered a “weak copyleft” license that typically applies to libraries. In fact, in the very first version released in 1991, the L in LGPL stood for “libraries”. This was changed in subsequent versions of the license.

1. What is the difference between the LGPL and the GPL?

Similar to the GPL with classpath exception and usually used in a similar way, the LGPL is a compromise between the strong copyleft GPL and the more permissive licenses like the Apache and MIT Licenses.

Under a strong copyleft license like the GPLv3, if any of the code in your software, including libraries and other components, is under a GPL, you must release your entire program under that GPL. This is not the case with the LGPL.

If you modify LGPL code and distribute it then you must release that derived work under the LGPL. If you simply use unmodified LGPL code, however, you may keep the rest of your code under any license you’d like or even make it proprietary.

2. What is the difference between the LGPL and the GPL with classpath exception?

While they were both created with similar intentions, there are some differences between the LGPL and the GPL with classpath exception in both form and practice. The GPL with classpath exception is simply an additional permission added to a GPL whereas the LGPL is a standalone license. The GPL with classpath exception is also easier to comply with when employing static linking and the LGPL is best used only with dynamic linking.

3. Does LGPL require dynamic linking?

The LGPL does not forbid static linking but it is generally discouraged. Dynamic linking is considered the best way to use LGPL’ed libraries and applications, as staying compliant with the license is a much more complicated process when linking the code statically.

The Apache License

The Apache License is a popular and widely deployed open source software license released by the Apache Software Foundation (ASF).

Here are the answers to the most common questions regarding the Apache license:

1. What are the Apache License terms and conditions?

One of the most permissive open source software licenses, the Apache license allows you to release your modified version of an Apache-licensed product under any license you may choose. You are permitted to freely use, modify, distribute, and sell software under an Apache License whether your use case is personal, internal, or commercial.

Unlike other permissive licenses that are applicable only to copyrights and not patents, the Apache License explicitly grants rights to users that can be applied to both copyrights and patents. The rights given are perpetual, worldwide, and irrevocable, but also non-exclusive, meaning you can use the licensed work and so can anyone else.

To redistribute software with any Apache licensed components, you must include a copy of the license, provide a clear Apache License attribution, and add modification notices to any files you modify.

You can choose to release the modified or derived products under different licenses, but the unmodified parts of the software must retain the Apache License. Another rule is that you cannot name your modified version in any way that suggests that the final product is either endorsed or created by the ASF.

Lastly, if you want to add a copyright statement about all of the modifications you've done to any Apache-licensed software, you are free to do so. The Apache License doesn't require you to release modified code under the same license so you can choose to add specific license terms and conditions that govern how others use, reproduce, or distribute your modified code.

2. What is the difference between the different versions?

It's rare to come across components that still carry the first version of the Apache License, but the main difference is that the Apache License version 2.0 grants patent rights and provides solid definitions of the concepts it uses to make the license clearer. The first version of the Apache License also had an advertising clause requiring that all promotional materials of derivative works of licensed products needed an Apache License attribution. In versions 1.1 and 2.0, simply including the attribution in the documentation alone is sufficient.

3. What is the difference between the Apache License 2.0 and the GNU GPL?

The Apache License 2.0 is a permissive license and the GNU GPLs are copyleft licenses. Software that uses any GPL-licensed component has to release its full source code under the same GPL. The Apache License 2.0 does not impose any such terms. You are not required to release your modified version but if you do choose to release it, you may release it under a different license. You are only required to retain the Apache License for the unmodified parts of the code.

4. Is the Apache License compatible with the GNU GPL?

Apache License 2.0 is compatible with GPLv3 but the resulting software must be released under GPLv3. The Apache License 2.0 is not, however, compatible with earlier versions of the GPL because of patent termination requirements that the FSF did not adopt until GPLv3. All previous versions of the Apache License are considered by the FSF to be incompatible with any version of the GPL.

5. What is the difference between Apache License 2.0 and BSD?

Both the BSD license and the Apache License 2.0 are highly permissible licenses that allow you to modify and redistribute software as you like. Earlier versions of the Apache License were nearly identical to the original BSD licenses, but the Apache License 2.0 is different from BSD licenses in the following key ways:

- Explicit grant of patent rights: Apache License 2.0 explicitly lays down the grant of patent rights while using, modifying, or distributing Apache licensed software as well as the circumstances under which such grant are withdrawn.
- Clear definitions of concepts: Apache License 2.0 explicitly defines all of the terms and concepts that it uses leaving little scope for ambiguity.
- Reusable without rewording: Apache License 2.0 is available for use by other projects without any need for rewording the license document itself.

Massachusetts Institute of Technology License (MIT License)

The MIT License is one of the most permissive and popular licenses out there. There are actually two variants of the same license that are frequently called “The MIT License” — the Expat License and the X11 License. Being the shorter and first variation, and the one the Software Package Data Exchange (SPDX) identifies as “MIT”, we will refer to the Expat License as the MIT License and the X11 License as the X11 License.

1. What are the terms and conditions of the MIT License?

The MIT License includes language expressing that the software is presented as-is and without warranty. As a permissive license, there is no requirement that modifications to MIT Licensed code be also released under the MIT License making it a great choice for proprietary and commercial uses. The license’s only requirement of a user is that the license be included in “all copies or substantial portions of the Software”. At less than 200 words, that’s just a few lines of comments in the code. There’s a reason the MIT License accounts for nearly a third of all uses of open source licenses.

2. Are there any other versions of the MIT License?

There is only one version of the MIT License, also known as the Expat License, but there are some variations on it. The X11 License adds a clause restricting the use of the copyright holders’ names for advertisement, the FPA License limits some commercial use, and the MIT No Attribution License is a “zero clause” license and as free as it gets – it removes even the requirement to include a copy of the license.

3. Is the MIT License compatible with GNU GPL?

Yes, the MIT License and variants listed above are compatible with all versions of the GNU GPL, however the FSF recommends larger projects be licensed under Apache 2.0 as the MIT licenses provide no protection from what they call “patent treachery”.

Microsoft Public Licenses (Ms-PL)

The Microsoft Public License (Ms-PL) is a free and open source software license created by Microsoft for its own open source projects.
Here are the answers to the most common questions about the Ms-PL:

1. What are the Microsoft Public License (Ms-PL) terms and conditions?

The Ms-PL is a short, concise, and straightforwardly written license that allows you to freely reproduce and distribute original or derivative works of any software that it governs. In doing so, however, you may not use any contributors' names, logos, or trademarks. The Ms-PL also protects software authors by explicitly offering no express warranties or guarantees, meaning authors are not liable if their code doesn't work well in some cases.

When distributing software under the Ms-PL, modified or unmodified, in whole or in part, you are not obligated to distribute the source code. You are, however, required to retain all copyright, patent, trademark, and attribution notices that were originally present. If you do choose to distribute any portion of the software, modified or unmodified, in its source code form, you must do so only under the Ms-PL by including a complete copy of the license in the distribution. If you instead choose to release the code in a compiled or object code form, you may release it under any compatible license.

2. Is Microsoft Public License (Ms-PL) considered copyleft?

When Microsoft first created the Ms-PL, they titled it the "Microsoft Permissive License", which gives us a hint at their intentions. The name changed to "Microsoft Public License" in the process of getting approval by the Open Source Initiative (OSI) but the text of the license itself stayed the same. A likely reason for the name change was that it is not strictly a permissive license. But neither is it a true copyleft license. The Ms-PL is both permissive and weakly copyleft, depending on distribution.

In a closed-source distribution, the Ms-PL is permissive and compatible with any other license that allows a close-source distribution. In an open-source distribution, the Ms-PL is a weak copyleft license that requires modifications to be also licensed under the Ms-PL, but also allows for other parts of a larger project to fall under different licenses.

3. What is the difference between Microsoft Public License (Ms-PL) and the Microsoft Reciprocal License (Ms-RL)?

Both licenses are very similar, with the Ms-RL having additional requirements on source code sharing. Any file that contains any amount of code under the Ms-RL must be itself released under the Ms-RL and the source code and a copy of the Ms-RL must both be provided in the distribution. Your other files in the larger project can be under any license you choose. Under the Ms-PL, releasing the source code or not is up to you. If you choose to release source code that contains code licensed under the Ms-PL, you must release it all under the Ms-PL. While the Ms-PL is less explicit about this than the Ms-RL, it is generally interpreted that the Ms-PL, written to be less restrictive than the Ms-RL, does not automatically apply to other files in a larger work, specifically those files that are your own work and do not include Ms-PL'ed code.

4. Is Microsoft Public License compatible with GNU GPL?

No. Neither the Ms-PL nor the Ms-RL are compatible with any version of the GNU GPL. There are many reasons these licenses are incompatible but one is that the GPL requires the distribution of source code of the combined work be released and that it be released under the GPL, whereas the Ms-PL does not require release of the source code but if source code containing any Ms-PL'ed code is in fact released, it must be released under the Ms-PL. Since neither the GNU licenses nor the Microsoft licenses in question allow for sublicensing, they are strictly incompatible.

5. How can you use a component licensed under the Microsoft Public Licenses in your commercial project?

If you are using Ms-PL'ed components and decide to release the source code of your product, then you will be able to distribute your software only under the Ms-PL. If you choose to release the compiled or object code, you can release it under any compatible license.
If you are using Ms-RL'ed components, you will need to distribute the modified source files under the Ms-RL, adding an extra step not required of you under the Ms-PL, a step which may or may not be an issue for a particular commercial product. However, you may license other files that are entirely your own work under any terms you choose.

Berkeley Software Distribution (BSD)

Berkeley Software Distribution licenses are a family of permissive free software licenses and include the original BSD License, the 3-clause Modified BSD License, and the 2-clause Simplified BSD License/Free BSD License.

Here are the answers for the most common questions about BSD licenses:

1. What are the terms and conditions of the BSD Licenses?

The BSD License allows you to freely modify and distribute your software's code in the source or binary format so long as you retain a copy of the copyright notice, list of conditions, and the disclaimer.

The original 4-clause BSD License also contains an advertising clause and a non-endorsement clause which will be explained in the following answers. The 3-clause Modified BSD License removes the advertising clause and the 2-clause FreeBSD License removes both the advertising and non-endorsement clauses.

2. What is the difference between the original 4-clause BSD License and the Modified 3-clause BSD License?

In theory, the advertising clause from the original BSD License was meant to ensure that users acknowledge the original authors of any BSD-licensed components in all advertising materials mentioning features or use of their software. In practice, this was frequently misapplied due to both misunderstanding and malice leading to situations where developers were required to list too many attributions, each corresponding with a BSD-licensed component used in their software, to be feasible. Additionally, this advertising clause made the original BSD License incompatible with the GNU GPL. Following feedback from developers, the advertising clause that appears in the original BSD License was removed to create the 3-clause Modified BSD License.

3. What is the difference between the Modified 3-clause BSD License and Simplified 2-clause BSD License?

The 2-clause Simplified or FreeBSD License does not include the non-endorsement clause which ensured that users could not make it sound like their software was endorsed by any of the acknowledged developers or organizations. It instead includes a disclaimer about views and opinions expressed in the software being those of the authors and not that of the FreeBSD Project.

4. Are the BSD Licenses compatible with GPL?

The original BSD License is not compatible with any version of the GNU GPL, as mentioned previously. The 2 newer versions of the BSD License, the Modified and Simplified BSD Licenses, are compatible with any GPL.

5. What are the differences between the Modified BSD License and the MIT License?

The MIT license can be used as-is, without any requirements other than including a copy of the original license, unchanged. The Modified BSD License requires you to modify the license to suit the project at hand. Additionally, the Modified BSD License, thanks to its non-endorsement clause, protects you from having your name involved in a project unless you wish to do so.

Common Development and Distribution License (CDDL)

The Common Development and Distribution License (CDDL) is an open source license published by Sun Microsystems (now Oracle) to replace the Sun Public License (SPL) and is considered to be SPL version 2. It is inspired by the Mozilla Public License (MPL). The CDDL is often considered a cleaned-up version of the MPL and was made to facilitate reusability.

Here are some answers to common questions about the CDDL:

1. What are the Common Development and Distribution License (CDDL) terms and conditions?

You are free to reproduce and distribute any original or derivative works of any software licensed under the CDDL but you must not remove or make any changes to any copyright, patent, or trademark notices present in the software. You must also retain any notices of licensing or any descriptive text that gives attribution to any contributor or the initial author.

Similar to the Ms-RL, when you distribute your software in any form other than source code, you are required to make the source code of any file containing CDDL'ed code, whether modified or in its original form, available under the CDDL as well and include a copy of the license in the distribution. The executable form and any project files not containing CDDL'ed code may be released under the CDDL or any compatible licenses.

Additionally, for each modification that you make you must identify yourself as a modifier by including a notice in your modified files.

2. Is the CDDL considered copyleft?

The CDDL is considered a weak copyleft license as its use does not grant downstream users of the program the same rights you received by requiring your entire code become open source like with strong copyleft licenses such as the GNU GPL, MPL, or Eclipse License. Only files including CDDL'ed code or modifications to CDDL'ed code are under any copyleft obligations. Other files and the executable of the whole program can be under any license that is compatible with the CDDL.

3. Does the CDDL grant patent rights?

Yes, the CDDL takes a very clear stance on patents and explicitly grants patent rights. Any contributor grants you the right to use the patents that their contribution embodies. You may use, modify, and redistribute CDDL'ed components without any concern about patents that the code contributors might hold on the contributed technology. The CDDL also discourages patent litigation against developers by terminating the usage rights of anyone who initiates a patent claim against a developer about the code they have contributed.

4. What is the difference between CDDL version 1.0 and CDDL version 1.1?

CDDL version 1.1 was submitted just a year after version 1.0 in early January 2005. Version 1.1 includes some corrections that prevent the CDDL from being in conflict with European copyright law and that allow single developers to use the CDDL for their work.

5. Is the CDDL compatible with the GNU GPL?

No, the CDDL is not compatible with any version of the GNU GPL. The GNU GPLs require all source code in a larger piece of software using GPL'ed code be released under the same GPL or later whereas the CDDL's reciprocity only applies to the files that contain CDDL'ed code. In general, both strong and weak copyleft licenses without explicit compatibility or sub licensing clauses are not compatible with each other. Similar to the situation of Ms-PL and Ms-RL compatibility with the GPL, combining GPL'ed and CDDL'ed code puts the CDDL'ed code under the requirements that it be licensed under both the GPL and the CDDL, which neither license allows for.

Eclipse Public License (EPL)

The Eclipse Public License (EPL) is an open source license developed by the Eclipse Foundation. It's derived from the Common Public License (CPL). The Eclipse codebase is now available under the EPL and was formerly licensed under the CPL.

Here are some questions and answers regarding the EPL:

1. What are the terms and conditions of the Eclipse Public License?

The EPL is a weak copyleft license. If you modify code under the EPL and distribute the source code as part of your program, you're required to license the modified code under the EPL. If you distribute such a program in its object code form, you're required to state that the EPL'ed source code is available to the recipient upon request and to share the method for making that request.

The Eclipse Foundation makes it clear that "merely interfacing or interoperating" with an Eclipse plugin does not make your code a derivative work of the plugin. As with all weak copyleft licenses, if you wish to use a different license for the rest of your code, keeping the EPL'ed code in a separate file is the best practice, if not a requirement. The EPL version 1.0 is less clear about this but version 2.0 is explicitly a per-file copyleft license.

The EPL protects the author from possible lawsuits or damages caused if a company used their component in a commercial product. Any warranties made by a commercial contributor to EPL'ed code is theirs alone and not that of previous contributors. Like many other open source licenses, the EPL also includes an explicit patent grant.

2. What is the difference between the different versions of the EPL?

There are a few major differences between the original EPL version 1.0 and the current EPL version 2.0.

Two of the changes aim to bring clarity to the license and its terms. EPL version 2.0 defines "source code" explicitly, making it suitable for scripted languages. Additionally, it better defines what is or is not considered a derivative work by replacing the term of art "modules" in version 1.0 with the more precise term "files".

EPL version 2.0 also aims to be more usable. US-centric legal language from version 1.0 is omitted, making EPL version 2.0 more suited to world-wide software projects and a clause is added allowing for secondary licensing. This secondary licensing allowance is important for GNU GPL compatibility which we will address in a subsequent question.

3. What is the difference between the Eclipse Public License and IBM's Common Public License (CPL)?

The first version of the EPL is largely identical to IBM's CPL, however the EPL revises the CPL by deleting the first sentence in the 7th section of the original CPL that was believed to be overly broad and non-conducive to the growth of the Eclipse ecosystem. The removed content explained how the CPL handled patent retaliation.

4. What is the difference between the Eclipse Public License and the GNU GPL?

The GNU GPLs are strong copyleft licenses that require users to release their software's full source code irrespective of the percentage of GPL'ed code included. The EPL and other weak copyleft licenses only require that the modified file under the weak copyleft license be released. Complying with the EPL specifically only requires that if you release the source code of your project, you release the modified EPL'ed components under the EPL and if you distribute the program as object code, you make the EPL'ed component's source code available by request.

5. Is the Eclipse Public License compatible with the GNU GPL?

The first version of the EPL is not compatible with any GNU GPL. The latest version, EPL 2.0, is not strictly compatible with any GPL as-is, however, through an optional clause, it allows for contributors to specify that their code may be additionally licensed under the GPL. By using this secondary license clause, the original EPL'ed code stays under the EPL but the entire project, including any modifications made to the EPL'ed code, can be combined with GPL'ed code and the resultant code released under version 2 or later of the GPL.

*The author of this white paper is not a lawyer, and you should not interpret this as legal advice of any kind.

Information is provided on an as-is basis. For a legal consultation, please contact your legal advisor.

Visit our website