



# Open Source License Obligations: Attribution and Copyleft

Dirk Riehle , Friedrich-Alexander-Universität Erlangen-Nürnberg

*License compliance is the process of complying with the obligations posed by licenses upon the users of code with that license. Here, we are typically talking about open source license compliance, that is complying with the obligations required by open source licenses.*

In this column, I previously discussed open source licenses, though in somewhat general terms. All open source licenses

1. provide defined rights to you
2. have obligations you must fulfill to receive those rights
3. are disallowing you to do certain things

4. typically, also disclaim all warranties and liabilities.

The Open Source Initiative (OSI) has accepted more than 100 licenses as open source licenses, conforming to their open source definition. Of these, about half of them are in actual use, and about 20 licenses are in frequent use.

That said, while some licenses can be used as-is and without modifications, software developers love to fiddle with open source license texts. For example, the popular and quite fundamental JavaScript Object Notation (JSON) project added the obligation:

“The Software shall be used for Good, not Evil”

to the MIT license to create the JSON license. This makes the JSON license a nonopen source license because it restricts the use of any so-licensed code to not do evil. In the case of the JSON license, this is a no-op, that is, has no further consequences, as people will not agree on what is good or evil.

In other cases, license text modifications are less harmless. Developers like to take popular open source licenses

## FROM THE EDITOR

Complying with open source licenses remains a basic activity of any product vendor who sells projects or products that contain open source code. After discussing open source licenses in past instances of this column, this column discusses how to comply with the two most common obligations resulting from the open source licenses. Ignore at your own peril!—Dirk Riehle

and subvert some of the wording to make the licensed software commercial software, for example, by restricting fields of use or by requiring financial compensation.

As a consequence, a developer using open source components needs to read each and every license carefully to understand their implications.

At the time of writing, the ScanCode LicenseDB counted more than 2,000 practically relevant license texts, commonly derived from any of the OSI-approved licenses. Most modifications are harmless and don't change the meaning of the underlying open source license, but as noted, some aren't.

Each license is its own legal text. Developers need to comply specifically with the obligations spelled out by that text. Still, as noted early on, some of the obligations keep recurring and can be handled in a consistent way across all valid and well-intentioned licenses.

Effectively, there are two main obligations that you may have to fulfill:

1. *Provision of legal notices (attribution, license texts, and other notices):* The open source legal notices are a best practice in which you compile all texts you are required to provide to recipients into one file, the legal notices file, which you then pass on to recipients.
2. *Provision of corresponding source code (copyleft):* The provision of

corresponding source code is how you comply with a copyleft obligation: By truthfully collecting all needed source code, tooling, keys, and so on, so that recipients can rebuild the software and fix any bugs.

These obligations only fall upon you if you are distributing open source code. If you are an end user, you typically don't have to do any of this.

Beyond these two main obligations, there are further important but less frequent obligations (like indemnification) that you may be required to fulfill. The details will be explained in the specific licenses, and you need to read and comply with them.

License compliance can be a tedious process, so you will want to automate as much as you can and standardize most of the human involvement. We will discuss all this in turn.

## THE OPEN SOURCE LEGAL NOTICES

The first of two main requirements put upon distributors of open source code is to provide the open source legal notices to recipients.

Legal notices are (typically textual) notices that are given by a distributor of an artifact to the recipient of the artifact. They spell out any legal information the recipient needs to hear. The common example is a vendor selling a product to a customer. Legal notices are not limited to software, and they existed long before open source software. Legal notices serve many different purposes, for example:

- › *Declaration of ownership:* The distributor may want to or have to declare who owns what in the artifact, including components sourced from suppliers. Such declarations may include copyright statements, patent use permissions, and so on.
- › *Declaration of information required by law:* Often the law puts

requirements on the distributors of artifacts, for example, to warn them about radiation from devices like mobile phones or to inform them about compliance with required standards.

- › *Declaration of limitation of liabilities:* The distributor may want to try to limit any liabilities resulting from receiving and using the artifact. Whether such disclaimers or limitations will hold in court is often unknown until tried.

Any software artifact distributed to third parties should have an open source legal notices section as part of a more general legal notices. The open source legal notices are a result of the following common obligations found in open source licenses:

1. *Provide copyright notices (also known as attribution):* The distributor needs to provide all copyright notices found in the open source code they are distributing.
2. *Provide license texts:* The distributor needs to provide all license texts found in the open source code they are distributing.
3. *Provide disclaimers:* The distributor needs to provide all disclaimers and limitation statements to recipients. These are often already included in the license texts.
4. *Provide change notices:* The distributor needs to create and provide change notices (descriptions of modifications made to the original code).
5. *Provide other notices:* The distributor needs to provide all other relevant notices found in the open source code. This is a catch-all to not forget anything.

A given license may not require you to do all of these. However, you are rarely only ever distributing open source code of one license. Most likely, all obligations will be present.

Also, no license requires the compilation of a legal notices file. However, it is a best practice to create a single presentation of all text snippets rather than inundate the recipient with often tens of thousands of small snippets as their own files.

Compiling open source legal notices requires sifting through the open source code and collecting all copyright notices and license texts into one document, the open source legal notices. An explicit request for providing disclaimers is typically already satisfied through providing the license text, where they are typically located. Change notices are only required if you changed the open source code rather than just incorporating it as a library, and other notices often don't exist.

The creation of legal notices may sound harmless, but in practice can turn into a significant amount of work. The Linux kernel by now had more than 25,000 contributors, all with their own copyright notice that you need to find and include in the legal notices (or else not fulfill the obligations and hence lose the usage rights grant).

You may wonder why the creation of legal notices is not performed centrally, as part of the open source project, given that they are independent of the distributor. With the exception of a few notable examples, including the Linux kernel, this has not happened. Open source programmers often don't care because the source code already includes all notices. Companies usually worry that any mistake in a contribution to such legal notices may be used to sue them.

The smartest way to compile the legal notices is during software composition analysis, as discussed before. At that time, a person is looking closely at the code, trying to understand its license conditions and third party rights. During this process, all copyright statements, license texts, and further notices should be marked and saved. Most tools support this process.

What is not sufficient, as also explained, is to rely on the information provided by package managers: Such

information is often incomplete, out of date, or simply incorrect. A SCA Tool<sup>a</sup> will help you, but the creation of legal notices will still remain a laborious task for the foreseeable future.

The compilation of disparate copyright statements, license texts, and other notices into one legal notices file is only the first of two steps of open source proper license compliance. The second step is to bring the legal notices in front of the recipient of the open source code. I will discuss this in an upcoming article.

## CORRESPONDING SOURCE CODE

The second of two main requirements put upon distributors of open source code is to provide the corresponding source code of any copyleft-licensed code you distribute to recipients. If you are an end user only, this does not apply to you.

To recap: The copyleft obligation requires that any incoming open source code (of a license with a copyleft obligation) be distributed to third parties (like customers) only under the same license as the incoming license: The incoming license must be the outgoing license. If you don't comply, you don't receive the usage rights to the open source code.

The copyleft obligation only applies to so-called covered work and if the distribution conditions are triggered.

- ▶ **Covered work:** What is covered work, that is, affected code, is defined in the license using copyright law. Any code you write that taps into copyleft-licensed code by incorporating some of the copyrighted text (and be it interface symbols) is typically covered code. Then, any further code that taps into your code may also become covered code. This propagation is sometimes called a “viral effect” and is intentional.

- ▶ **Obligation trigger:** Distribution of the covered code to third parties triggers the copyleft obligation. Before the advent of the web, this was easy: Distribution meant passing on binary code to recipients. Since the web and cloud services, new licenses have tried to frame the provision of software as a service as distribution, with mixed success. The most well-known cloud copyleft license is the AGPL-3.0 license family.

Your system's architecture, how your code is coupled, and the specifics of the licenses therefore determine which of the code in a distribution is affected by the copyleft obligation of any copyleft-licensed code may be using. A simple example is a device you might be selling, running Linux, with a user-space application. Linux is mostly covered by GPL-2.0, and hence you have to provide corresponding source code for the Linux-related code on your device. Your application is shielded from the copyleft obligation because of the Linux Syscall Note, and hence you do not have to provide the corresponding source code to your customers.

Many open source licenses contain a copyleft obligation, sometimes weakened by exceptions like the Linux Syscall Note, sometimes unmitigated. Details vary, and you may sometimes believe you found a loophole in a license so that you could comply with the license by letter, but not in spirit. I generally advise not to go against the grain of a license, but to comply with its intentions.

The intention of a copyleft obligation is to empower recipients to enhance the software or to fix any bugs. To modify the software or a component, users typically need the source code for this component, a copyleft licenses were invented to ensure this. If a component is covered by a copyleft license, its distributor needs to make the source code available to recipients and everything that is needed to put a replacement in place, including

<sup>a</sup>See <https://scatool.com>.

but not limited to custom tools, cryptographic keys, and necessary instructions. Only “standard software” that is, off-the-shelf tools can be omitted.

The comprehensiveness (tools, keys, and so on) was added only in more recent license generations, most notably the GPL-3.0 family. It was caused by vendors’ active attempts to work around the copyleft obligation, dubbed tivoization based on the Tivo digital video recorder, for which these circumvention techniques were invented.

Recipients of corresponding source code generally expect one file, an archive, with all relevant files and information, prepared in a way that allows the recreation, modification, and deployment of the covered work.


The provision of corresponding source code does not necessarily require you to ship the source code with the binary code. Some licenses only require

that you make an offer and maintain this offer for a defined period of time. However, if recipients ask for the source code, you have to comply. For this reason, you should create the corresponding source code together with the binary code and archive it, even if you don’t ship it.

Creating corresponding source code for old software versions is a difficult task and highly error-prone. The creation and archiving of corresponding source code should be part of your build processes. Then, if a recipient comes asking, you simply pull the code from the archive and provide it to the recipient.

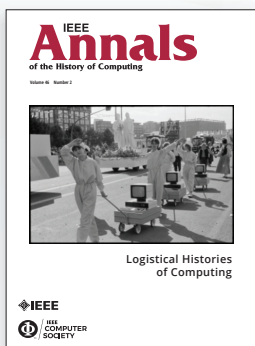
Also, not everyone has a right to ask and receive, only recipients of the binary distribution do. However, because of the copyleft license, recipients have a right to put the source code they receive from you onto the web under the open source license.

Hence, you might as well provide your corresponding source code on the web as well. To protect yourself from copyright trolls, you may want to password-protect access, though.

**P**ast relationships between vendors and copyleft enthusiasts were often adversarial. These fights have calmed down, including a more amicable approach to remediation and license compliance that I will discuss in an upcoming section. 

**DIRK RIEHLE** is the professor for open-source software at Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany. Contact him at [dirk@riehle.org](mailto:dirk@riehle.org).

# IEEE Annals of the History of Computing



*IEEE Annals of the History of Computing* publishes work covering the broad history of computer technology, including technical, economic, political, social, cultural, institutional, and material aspects of computing. Featuring scholarly articles by historians, computer scientists, and interdisciplinary scholars in fields such as media studies and science and technology studies, as well as firsthand accounts, *Annals* is the primary scholarly publication for recording, analyzing, and debating the history of computing.



[www.computer.org/annals](http://www.computer.org/annals)