Some Common Issues in Open Source Licensing

The ways in which all open source[1] licenses are the same are greater than the ways in which they differ – but their differences can still be significant. This document gives an overview of some common issues in open source licensing and license compatibility. We emphasize "overview": the issues described below usually require case-specific analysis, and in order to provide a readable overview, we have chosen to omit a great deal. This document does not constitute legal advice.

Background – Copyleft vs Non-Copyleft:

"Copyleft" licenses are licenses that require redistributed copies and derivative works to be under the same license. In other words, you're free to use the code for any purpose, and to share and modify it, but any resultant works you distribute must be distributed under the same license. The best-known copyleft licenses are the GNU General Public License (GPL)[2] and its more recent cousin the GNU Affero General Public License (AGPL)[3], but there are others as well.

A "non-copyleft" license is simply one that permits the code to be incorporated in a program that is, overall, distributed under some other license – even a proprietary one. Non-copyleft licenses are thus sometimes called "permissive" licenses. Some examples of popular non-copyleft licenses are the MIT license[4], the BSD license (or license family)[5], and the Apache License 2.0[6] from the Apache Software Foundation.

Neither style is dominant in open source. Much widely-used code is released under copyleft licenses, especially the GPL, but much is also released under non-copyleft licenses.

License Compatibility (Use vs Distribution):

1 The terms "free software" and "open source" may be treated as synonymous for licensing purposes. This document uses "open source" for consistency, but could have used "free software" just as easily. There is a common misconception (see http://developers.slashdot.org/story/12/04/22/0211251/open-source-project-licenses-trending-toward-open-rather-than-free for example) that "free" refers only to copyleft licenses and "open source" refers only to non-copyleft (so-called "permissive") licenses. However, this interpretation is rejected by all the major standards-making and license-reviewing bodies in the field, including the Free Software Foundation and the Open Source Initiative.

2 http://www.gnu.org/licenses/gpl.html

3 http://www.gnu.org/licenses/agpl-3.0.html

4 http://opensource.org/licenses/MIT

5 http://en.wikipedia.org/wiki/BSD_licenses

6 http://www.apache.org/licenses/LICENSE-2.0.html

Just because two licenses are open source does not always mean code under them can be mixed together and redistributed. This is one of the most frequently neglected considerations when organizations first approach open source licensing, especially because the issue can remain hidden during in-house development – only to come to light as release nears and the licenses' distribution-specific clauses are about to go into effect[7]. (This is one reason why starting development in the open, or moving it there as early as possible, is an open source best practice[8].)

The most common such incompatibility is between the GPL (or AGPL) and certain permissive licenses. That may sound counterintuitive, but remember that in this context "permissive" just means "non-copyleft" – a permissive license may still have provisions that make it incompatible with the GPL for outbound distribution. Because a great deal of open source software is distributed under the GPL, incompatibility with the GPL should be avoided where possible.

The Free Software Foundation maintains a list of GPL-compatible licenses at [9] and of GPL-incompatible licenses at [10].

Patent Reciprocality (or: The Tradeoff Between Simple and Comprehensive):

As the software industry has grown in complexity, open source licenses have evolved to address various new concerns. Sometimes this is seen in updated versions of existing licenses (for example, the GPL is now on version 3) and sometimes in *de novo* open source licenses being created.

But opposing that growth in complexity is an understandable desire to release open source software under licenses that are widely-recognized and/or easy to comprehend – in other words, licenses that are familiar, or short, or both. Naturally, a license that is familiar to many is one that has been around for a long time, so by definition will not have evolved much in response to industry changes. And a license that is short cannot also be comprehensive, so even if it is relatively recently updated, it must still leave many issues unaddressed.

To illustrate this tension, we can compare two examples from the permissive licensing spectrum: the so-called "BSD" licenses, and the Apache Software Foundation's "Apache License, Version 2.0".

The BSD licenses are extremely popular and are by now familiar to many developers and tech industry lawyers. The original BSD license was already short and simple, and the only modifications made to it since the 1980s have been further simplifications, so that modern BSD remains both familiar and *extremely* simple – it is a mere 185 words long, and basically says you can do whatever you want with the code as long as you abide by some minimal and easily-met attribution requirements.

---

7 Note that under the GNU Affero General Public License, making the software's functionality accessible via a network connection is considered a form of distribution, for licensing purposes. The AGPL is unique in this, and in fact that clause is its reason for being, as it is otherwise the same as the GPL.

8 http://blog.civiccommons.org/2011/01/be-open-from-day-one/

9 http://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses

10 http://www.gnu.org/licenses/license-list.html#GPLIncompatibleLicenses

The Apache License 2.0 is much longer (nearly ten times so) and although fairly well-known at this point, it still does not have the mindshare BSD has. On the other hand, Apache 2.0 addresses some issues that BSD does not. For example, it has a clause that deters contributors to a work from later claiming patent infringement based on their code contributions – if they bring such a claim, they lose their own right to the implicit patent grants of all the other contributors. This clause may be seen as an immune response to the climate of increased patent litigation in the software field in general; older licenses do not have such clauses, because patent litigation was less common when they were written. The Apache License 2.0 also gives clearer and more specific instructions for exactly how attribution is to be performed than older licenses tend to. Thus, although Apache 2.0 gives lawyers more analysis work to do up front, it can also leave them with fewer questions afterward, and with more options should a downstream redistributor fail to abide by the license's terms.

Inbound vs Outbound:

As the example above indicates, the question of *what license to release code under* ends up being closely connected to the question of *what existing code might need to be incorporated* into a product. For example, if outbound code is to be under BSD, then code licensed under Apache 2.0 cannot be incorporated into the outbound product, because the result would need to be licensed under Apache 2.0 – Apache imposes requirements that BSD does not, whereas the reverse is not true.

Yet the situation is not symmetrical: if outbound code is to be licensed under Apache 2.0, then code licensed under BSD *can* be incorporated into a product that remains, overall, released under Apache 2.0.

We emphasize that none of this is a recommendation for or against any particular license. Similar comparisons could be made with other license pairs, and there are many aspects of open source licensing we have not touched on here. The purpose of the above example, and of this document as a whole, is simply to show that there can be unexpected considerations in choosing an open source license, that it often requires thinking prospectively about what existing code might be incorporated and by whom, and that the choice will sometimes involve tradeoffs.