

11

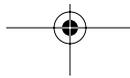
Shared Source, Eventual Source, and Other Licensing Models

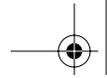
Alternatives to Open Source

There are many ways to license software. None is legally privileged. Contract law allows parties to license software under almost any terms and conditions that people can dream up. Copyright and patent law acts as a backstop, preventing anyone from copying, modifying, distributing, making, using, or selling protected software without the licensor's permission—but otherwise leaving to the parties themselves the terms and conditions of their licenses.

Open source software distribution is a young but maturing business model. Enormously successful software has been created and is available worldwide, usually for free. Despite this success, companies often refuse to “go all the way” with open source, afraid that giving software away for free is contrary to their profit motive.

At one extreme, of course, there is fully proprietary software that cannot be copied, modified, or distributed. Source code is not available, reverse engineering is forbidden, and none of the copyright rights are given away. (Remember, though, that you don't need a separate license to install a copy of software you own, and to make backup copies; see 17 U.S.C. 117.) This model remains quite successful in the market, as anyone can





plainly see. Proprietary software will doubtless continue to thrive.

In between fully proprietary and fully open source models there are other software distribution alternatives as well. This chapter describes some of those variations that pay homage to open source but don't quite go all the way. These licenses are, one might say, partly proprietary and partly open source.

Each of the licenses described in this chapter provides source code to licensees. You will remember from Chapter 1 that source code is but the means to an end; it is not an end itself. (Open Source Principle #4.) The real goal is software freedom, as reflected in the right to use for any purpose, to copy without payment of royalties, and to freely create and distribute derivative works. (Open Source Principles # 1, 2, and 3.)

The problem with all of the licenses described in this chapter is that they fail to fully promote software freedom. Their terms are far more reasonable than typical proprietary software licenses, but the software they license is not truly free.

That doesn't mean that you shouldn't accept software under them. Some of them are good licenses, just not good enough to be open source.

Shared Source

In response to the demands of its customers for access to source code, Microsoft created its shared source licensing program. This program allows Microsoft customers to read and examine certain of the company's source code.

The Microsoft Shared Source License is a dramatic leap forward for the world's largest proprietary software vendor, a company that has traditionally kept its source code secret for





competitive reasons. At long last, Microsoft's customers may examine some of that company's source code and learn from it. Of course, from the perspective of open source licensing, the shared source concept is a weak alternative that doesn't go nearly far enough to provide software freedom.

The Microsoft Shared Source License has limited purposes:

You may use this Software for any non-commercial purpose, subject to the restrictions in this license. (Microsoft Shared Source CLI, C#, and JSCRIPT License.)

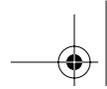
By itself, the "non-commercial purpose" restriction of this license makes it incompatible with Open Source Principle # 1. But this license goes even further, making it also incompatible with Open Source Principles # 2 and 3. Open source software must be available to anyone for any purpose, to create derivative works, and to sell the software. The Microsoft software isn't so available:

You may not use or distribute this Software or any derivative works in any form for commercial purposes. Examples of commercial purposes would be running business operations, licensing, leasing, or selling the Software, or distributing the Software for use with commercial products. (Microsoft Shared Source CLI, C#, and JSCRIPT License.)

In a more fundamental way, this is what the license says you may do—and what you are forbidden from doing—when you see Microsoft's shared source code:

You may use any information in intangible form that you remember after accessing the Software. However, this right does not grant you a license to any of Microsoft's copyrights or patents for anything you might create using such information. (Microsoft Shared Source CLI, C#, and JSCRIPT License.)



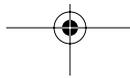


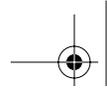
It is fascinating to consider whether an engineer with a photographic memory is allowed, without infringing Microsoft's copyrights, to re-create Microsoft's software from *intangible* information that he or she *remembers*. But that's not the legally interesting question for most engineers. Instead, the effect of this license provision is that engineers/licensees can use the information in some of Microsoft's source code to do practical things but they do not thereby obtain rights under copyright or patent. Source code can help licensees to design interfaces to Microsoft's products and to create programs that read and write Microsoft's data formats. It can be used to validate the security or reliability of Microsoft's products. For some of Microsoft's customers, this availability of source code for limited purposes is sufficient for their needs; they don't really need the software freedom provided by open source licenses.

So if you merely use intangibles that you remember, and if you base your software on those intangibles, you are allowed to do so. Microsoft's source code cannot be used, however, to write software that infringes Microsoft's copyrights or patents.

If you are a software developer who intends to write software that might potentially compete with Microsoft's copyrights or patents, there is great risk in looking at Microsoft's source code. Under the copyright law in the United States, if Microsoft proves that there is "substantial similarity" between your commercial software and theirs, you may be an infringer. You may have to prove that you saw and read Microsoft's source code but that you relied only on intangibles and only on your memory when you wrote your own software.

That's a difficult evidentiary burden. I'm not sure how even an experienced programmer can walk that fine line. Perhaps the best way is simply not to look at Microsoft's source code at all. At the very least, if you are directing corporate projects





relating to products competing with Microsoft's shared source software, build a sturdy wall separating those who look at Microsoft's source code and those programmers who might otherwise—even inadvertently—create derivative works or any commercial products from that source code.

This risk is not unique to shared source software. Employees can be contaminated by proprietary source code they saw or wrote while working for previous employers. Even open source software contains intangibles that can contaminate the memory of a programmer.

The solution obviously is not to avoid source code entirely, but to build sturdy walls around those in your company who will create proprietary software. Make sure those engineers don't inadvertently create derivative works of any source code they read, because you must honor the conditions and limitations of those licenses.

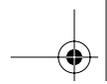
As for those who create open source software, don't create derivative works of Microsoft's shared source software. The Microsoft Shared Source License—unlike open source licenses—doesn't give you software freedom.

Public Source

Many companies are willing to go much farther than Microsoft, allowing their source code to be used for more than just examination and interfacing. Licensees can make copies, create derivative works, and distribute their works.

They draw the line, though, at commercial uses of the resulting software. They argue that the free use of open source software for commercial purposes exacerbates the free-rider problem I described in Chapter 10. It reduces the incentives for contributors because profits from the software will go to





large companies rather than to contributors. While not always prohibiting commercial uses, as the Microsoft Shared Source license does, public source licenses typically require the payment of royalties for commercial uses.

This form of license is referred to as *public source*, to indicate that the source code is published but that the software is not distributed under an open source license.

There can be many varieties of public source licenses, depending on the characteristics of the software being distributed and the business model of the licensor. For example, Ping Identity Corporation (see *www.pingid.com*) distributes some of its software under the following terms:

a. Without payment of royalty for unlimited Personal Use or Non-Commercial Distribution (as those terms are defined below);

b. Without payment of royalty for other than Personal Use and Non-Commercial Distribution as long as Licensed Software will run on fewer than 100 processors (as that term is defined below); and

c. Subject to the payment of one-time paid-up Royalty Fees for other than Personal Use and Non-Commercial Distribution on 100 or more processors. Licenses to run the Software on additional processors are subject to the Royalty Fees and payment terms as obtained at <http://www.pingidentity.com> and in effect on the date such additional licenses are obtained from Licensor. Royalty Fees to run the Software on additional processors are due and payable to Licensor prior to first use on those processors. (SourceID Public Source License section 1.)





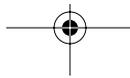
Unlike the CPL license, which leaves the term *commercial distribution* undefined, the SourceID Public Source License defines its terms precisely:

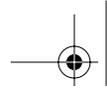
As used in this License, the term "Personal Use" means the functional use of software by an individual solely for his or her personal, private and non-commercial purposes. An individual's use of software in his or her capacity as an officer, employee, member, independent contractor or agent of a corporation, business or organization (commercial or non-commercial) does not qualify as Personal Use. (SourceID Public Source License section 3.)

As used in this License, the term "Non-Commercial Distribution" means the distribution of software to any third party for which no payment is made in connection with such distribution, whether directly (including, without limitation, payment for a copy of the software) or indirectly (including, without limitation, payment for a service related to the software, or payment for a product or service that includes a copy of the software "without charge"). (SourceID Public Source License section 3.)

As used in section 1 of this License, the term "processors" refers to a single processor running a single instance of Licensed Software. Each additional processor or instance of Licensed Software counts as an additional processor. (SourceID Public Source License section 3.)

These distinctions among users are not permitted in open source licenses under Open Source Principle #1. Nor can there be conditions like these that require open source licensees to count processors or similar metrics of software use. Public source licenses like this one do not guarantee software freedom—they are not open source.





Dual and Multiple Licensing

The owner of a copyright can license his or her work any number of times. Distributors of proprietary software do that when they grant discounts to favored customers, issue blanket licenses for unlimited copies to large corporations, and apply shrink-wrap licenses to copies sold in stores.

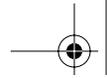
The MPL license described in Chapter 7 offered one example of dual licensing. Under the MPL, the Initial Developer may designate portions of the Covered Code as Multiple-Licensed. This allows any licensee to choose to accept those portions under the MPL or a second license specified in “Exhibit A.” Where that option is used, Initial Developers often choose the GPL.

More sophisticated examples than this of dual and multiple licensing are now widely used for important software. The owners of copyrights in open source software may simultaneously license that same software under non-open source licenses. This is particularly attractive for licensees who are reluctant to accept certain conditions of the available open source licenses and who are willing to pay extra license fees to relieve themselves of those conditions.

Such software, as originally licensed, *is* open source. It is available under an open source license. But it is also available under other licenses.

Consider the MySQL data base, which is distributed under the GPL and also under a separate commercial license. MySQL software is often incorporated into larger packages. Depending upon how the GPL is interpreted, such larger packages may become subject to the reciprocity condition of the GPL. This is unacceptable to some potential customers of MySQL who want to keep their derivative works proprietary.





The distributor of MySQL is also the owner of the copyrights in the software. It is thus free to license MySQL simultaneously under as many different licenses as it wants. In addition to the GPL, MySQL offers commercial licenses without reciprocity obligations—for a fee.

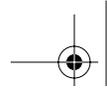
Mårten Mickos, the CEO of MySQL, describes his company's dual licensing commercial model this way:

Our paying customers get what they pay for: a commercially supported product with a level of assurance from the vendor and without any typical open source requirement that linked software must be open sourced as well.... Dual licensing allows companies to build viable long-term businesses while at the same time accommodating the needs of the open source/free software community. (See www.mysql.com.)

Mickos explains the *quid pro quo* of this dual licensing bargain. He points out that their commercial customers benefit from the open source customers because open source software is inherently more reliable and effective. (He calls it “rigorous ‘battle-testing.’”) Meanwhile, their open source customers benefit from the commercial customers because the MySQL company “can afford to develop and improve the product at a fast pace.”

One problem with this model is that contributions made by third parties to MySQL's GPL version must themselves be licensed under the GPL. (See GPL section 2[b].) The owners of the copyrights in the improvements *may* authorize dual licensing of their contributions under MySQL's commercial licenses, but nothing in the GPL requires them to do so. MySQL can try to avoid this problem by requesting that contributors assign their copyrights to the company, or by expressly accepting contributions under a license that permits MySQL to use the contributions as it sees fit.





Such dual licensing alternatives may have uses other than to avoid reciprocity obligations. Other conditions in an open source license may be unacceptable to prospective licensees. Some companies object to patent termination clauses (e.g., MPL section 8.2, CPL section 7, OSL/AFL section 10.) Some companies seek more elaborate warranties or forms of indemnification than are usually available under open source licenses. Licenses containing special waivers or additional benefits can sometimes be negotiated.

Any prospective licensee dealing with an unacceptable open source license should contact the licensor for other available licensing alternatives. Any licensor of open source software should consider dual licensing options as a way of attracting new customers.

Eventual Source and Scheduled Licensing

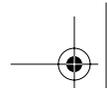
In business, timing is everything. A few months' lead developing and commercializing a product can mean the difference between commercial success and failure. For some commercial licensees, obtaining access to the source code now rather than eventually may justify paying for those license rights.

This business reality has encouraged companies to create licensing strategies that generate revenue from customers willing to pay extra for additional lead time to develop their products.

Artifex Software, the distributor of Ghostscript, uses such a scheduled licensing model. Initially new versions of Ghostscript are not fully open source, but at a later time they become open source under the GPL.

Ghostscript is intended to be embedded into printers to support industry-standard page description languages like PostScript and PDF. The lead time to introduce enhanced printers is short and the competition among printer vendors is





fierce. Some of Artifex Software's customers seek a marketing advantage by getting new versions of Ghostscript early.

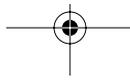
New versions of Ghostscript are distributed initially under the Aladdin Free Public License. They are also distributed under Artifex Software's commercial licenses.

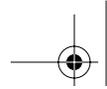
Despite its confusing name, the Aladdin Free Public License is not an open source license. It prohibits commercial distribution of Ghostscript or of products containing Ghostscript. Commercial distribution of Ghostscript requires an Artifex commercial license—for which there is a royalty.

Peter Deutsch, the author of Ghostscript and the first practitioner of this scheduled licensing model by which commercial time-advantages can be paid for, describes the Aladdin Free Public License this way:

This License is not an Open Source license: among other things, it places restrictions on distribution of the Program, specifically including sale of the Program. While Aladdin Enterprises respects and supports the philosophy of the Open Source Definition, and shares the desire of the GNU project to keep licensed software freely redistributable in both source and object form, we feel that Open Source licenses unfairly prevent developers of useful software from being compensated proportionately when others profit financially from their work. This License attempts to ensure that those who receive, redistribute, and contribute to the licensed Program according to the Open Source and Free Software philosophies have the right to do so, while retaining for the developer(s) of the Program the power to make those who use the Program to enhance the value of commercial products pay for the privilege of doing so. (Aladdin Free Public License.)

The Aladdin Free Public License imposes certain specific restrictions on distribution. Among other things, it prohibits





the commercial distribution of Ghostscript software if any payment is made. The license describes (in section 2) some types of distribution that are not allowed:

- When payment is made directly for a copy of the Program.
- When payment is indirect, as for a service related to the Program.
- When payment is made for a product or service that includes a copy of the Program “without charge.”

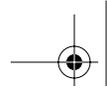
In many other respects, the Aladdin Free Public License reads like the GPL. Like the GPL it allows examination of the source code and the creation and distribution of derivative works. It even contains a reciprocity condition:

*You must cause the Work to be licensed as a whole and at no charge to all third parties under the terms of this License.
(Aladdin Free Public License section 2[c][ii].)*

Artifex Software, the commercial distributor of Ghostscript, simultaneously sells licenses to new versions of the program under commercial licenses. Those licenses allow customers to embed the most recent versions of Ghostscript into their printers. They also allow commercial licensees to use the source code in any way they wish, and they do not impose reciprocity obligations for derivative works.

Approximately one year after a version of Ghostscript is made available under the Aladdin Free Public License and its commercial licenses, Artifex Software re-releases that version under the GPL, at which point the software becomes truly open source.





The incentives for Artifex customers to buy commercial licenses are obvious. They can use the very latest versions of the software, with all the latest features. They can contract for the support of Artifex Software engineers to help them create their own products and derivative works. They can purchase warranties.

That extra time and those added-value services make scheduled licensing succeed as an open source business model. But such software isn't initially open source, and its licensors promise only that it eventually will be.

Combining Licensing Models

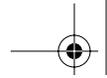
Nothing obliges a licensor to release all of its software under an open source license. Even companies that are friendly to open source may decide that some of their software should be kept proprietary.

A good example of this is Jabber, Inc., which creates and distributes instant messaging software. Jabber comes in both client and server versions. The client versions of Jabber's software are open source and the server versions of Jabber's software are not.

Jabber on user desktops is fostered by the easy availability of open source client software, available for free download from the *www.jabber.org* website. Meanwhile, companies can build proprietary instant messaging applications tailored to their needs on top of Jabber's commercial server software available from *www.jabber.com*. This convenient division into *.org* and *.com* distributors of related software highlights the distinction between open source and proprietary software, but it also demonstrates that the two worlds can actually support and encourage each other.

Just as they may choose to license different components of their software separately, software distributors may also offer





advanced versions of their open source products only under commercial licenses. Red Hat sells its Red Hat Applications, a collective work optimized for Linux, in that way. (See www.redhat.com.) These products are supported by a range of Red Hat Services, offered for a fee.

Open source licensing is a successful model, but it is not a religion. Alternatives are possible, and some of those alternatives are not entirely unfriendly to open source. Licensing creativity can allow contributors and distributors to make money while still encouraging, creating, and sharing open source software.

