Software Freedom Law Center Guide to GPL Compliance
2nd Edition

Eben Moglen & Mishi Choudhary

October 31, 2014

Contents

SFLC first published its guide to compliance with the GNU GPL and related licenses in 2008. We think that after six years of further adoption of GPL'd software, including the immense success of Android and other systems relying upon GPL'd software embedded in devices, it is time for a major revision of our advice on the subject. The changes that have occurred in the industrial use of copylefted software over the last decade make different emphasis necessary. This version of the Guide contains a clause by clause analysis of the compliance obligations created by each of the GNU GPL family of licenses,[1] to provide clear legal guidance on the licenses' requirements. Introducing that analysis, however, we begin by describing the underlying concepts of copyleft, which may be particularly helpful to newcomers to the subject. We also want to emphasize *software governance*, the general business processes for taking in, using, modifying and emitting third-party software, which have become far more relevant to businesses large and small. In the final chapter of this Guide, therefore, we discuss the relation of governance to compliance, and provide direct practical advice about responding to inquiries or compliance complaints from copyright holders.

The What, Why and How of GNU GPL

There are several major types of free or open source software licenses. Many of those licenses, which are known conventionally as "permissive," or "weak copyleft" licenses are concerned with preserving the freedoms of programmers. The "copyleft" licenses of the Free Software Foundation, which are the subject of this document, are concerned with protecting the freedoms of programmers, but more importantly they protect the freedoms of all *users*. The goal of the copyleft licenses is to ensure that all users of a program, or any work based on the program, have four fundamental freedoms:

0. The freedom to run the program for any purpose, without any additional permission;

[1]This document is about compliance with the GNU family of licenses, namely, GNU General Public License (GPL) versions 2 & 3, Lesser General Public License (LGPL) versions 2.1 & 3, Affero General Public License (AGPL)version 1 & 3. Many of our comments address all of these licenses and we refer to them as "the licenses" or "GNU licenses". When we are speaking of a particular license or a particular version, we refer to it in its short form along with the version number. In accord with longstanding free software community usage, we use "GPL'd" to mean "licensed under the GPL".

1. The freedom to read, study, understand and use any know-how or techniques taught or contained by the source code of the program;
2. The freedom to modify, adapt, improve, or reuse any or all of the program code; and
3. The freedom to share with anyone, or no one, both modified and unmodified versions of the program.

The easiest way to understand license terms is to begin with *why*, rather than *what*. The GNU licenses have been written by developers and their lawyers for application by developers without any aid or assistance from lawyers, for the purpose of assuring these freedoms to all their users, and all users of modified versions or new programs containing portions of their programs. The essence of these freedoms is the prevention of proprietary enhancements to copylefted programs. If a GPL'd program can be enhanced with proprietary code, however packaged or however written, then the intentions of the developers to protect users' rights will be frustrated. The *why* of copyleft, then, is to protect all downstream users' freedoms by prohibiting proprietary enhancement. As the Preamble of GPLv3 puts it,

> To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

This document explains what those responsibilities are, and how to discharge them most productively for users, at least cost to intermediate users, modifiers, packagers and resellers.

Copyright and Copyleft

The primary legal regime that applies to software is copyright law. Copyleft, which uses functional parts of copyright law to achieve an unusual result (legal protection for free sharing) forms the core legal principle of these licenses. It modifies, or "hacks" copyright law, which is usually employed to strengthen the rights of authors or publishers, to strengthen instead the rights of users.

Any work that is based on a copylefted program must also be licensed under the same copyleft license. This is sometimes referred to as the "hereditary effect" of copyleft or "share and share alike" principle. This "reciprocity" or "share and share alike" rule protects both developers, who avoid facing a "proprietized" competitor of their project, and users, who can be sure that they will have all four basic freedoms not only in the present version of the program they use, but in all its future improved versions.

In order to implement users' freedoms to study, modify and share, all users of GPL'd programs who receive binary or executable copies are entitled to receive complete and corresponding source code (C&CS). If C&CS is not provided along with the binaries, a written offer to provide C&CS must accompany the binary copy, irrespective of the medium of software distribution. This requirement to provide or offer C&CS is basic to the protection of users' rights.[2] Satisfying this requirement is also the most important component of license compliance. The licenses' requirement for the provision of C&CS is exacting: the supplier of binaries must supply or offer all the source code, makefiles, and other building materials required to produce the binary executable received by the user.

Unless every individual user is free to fix bugs, enhance features, and reuse code, users are deprived of the full value of the software that authors intended for them to have. If the license terms are not enforced, then proprietary enhancements will eventually come to represent the state of the art, as the copyleft program is submerged under the subsequent layers of proprietary enhancement. The source code requirement therefore applies whether users receive software in the form of digital transmissions over a network, as bits fixated in computer-readable media, or as software embedded in physical devices. The form of software has no effect on the extent or quality of the rights of the users.

Copyright law grants exclusive rights to authors. The "preclusive" use of copyleft to protect users' rights still leaves the authors, as copyright holders, or their agents in the sole position of enforcers or protectors of their users' rights. Actions for copyright infringement are the ultimate legal mechanism for enforcement, and copyright law allows only a copyright holder or her agent to bring an action for infringement. There also exist community efforts at compliance that help copyright holders in enforcement of their rights, but

[2] Parties who criticize copyleft or the GPL for making this requirement "viral" may well believe that programmers should be allowed to do whatever they want with code. They may thus see licenses without the source code requirement as "more free." But if the goal of the license is to protect *users* rights, then the requirement to provide complete source code is unavoidable: without the source code, no freedoms of *users* can be effectively protected.

only the copyright holders or their legal representatives can actually initiate enforcement proceedings in the legal system.

Concepts and License Mechanics of Copyleft

In our experience working with communities and companies, we have encountered numerous misconceptions about the GNU licenses' terms and core concepts. In this section we show how the working parts of the licenses implement the basic concepts upon which copyleft is based, followed by an explanation of specific terms of each license and how those terms impose specific compliance obligations. What may at first seem arbitrary or counter-intuitive becomes easier to grasp if your constant reference point is the intention of the licenses and the core concepts that they are based upon.

"The Work" and Copyleft

The unit of copyright law is "the work." In that sense, the "work" referenced by the licenses is anything that can be copyrighted or will be subject to the terms of copyright law. The GNU licenses exercise their scope fully. Anything which is "a work" or a "work based on a work" licensed under GPL is subject to its requirements, including the requirement of complete and corresponding source code, unless it is specifically excepted. This principle often causes theoretical or speculative dispute among lawyers, because "the work" is not a unit of computer programming. In order to determine whether a "routine" an "object," a "function," a "library" or any other unit of software is part of one "work" when combined with other GPL'd code, we must ask a question that copyright law will not directly answer in the same technical terms. A lawyer's conclusion on that subject will be based on the application of rules made in the context of literary or artistic copyright to the different context of computer programming. The key here is that the GNU GPL licenses mean to "plow fence row to fence row," covering every version of "work based on the work" recognized by local copyright law, but no further.

The GNU GPL licenses express this scope by saying that their terms apply to the work as licensed, and to all works "based on the Program." GPLv2, a license made by US lawyers for US readers, says that

> "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language.

This form of explanation was unfortunately unhelpful. It led to years of fruitless discussion about the role of "derivative works" doctrine (a US concept) in software (where US courts have largely failed to provide any guidance). So in GPLv3, we and our clients at the Free Software Foundation decided to drop all illustrative reference to US "derivative works," returning to the base concept only: GPL covers the licensed work and all works based on the work, where "based on the work" is defined as any modification or combination with the licensed work that requires copyright permission to make.

But the GNU GPL licenses recognize that what is outside their scope is as important to the protection of programmers' rights as their inclusive scope is important to the protection of users' rights. Where a programmer's work is "separate and independent" from any GPL'd program code with which it could be combined, then the obligations of copyleft do not extend to the work separately distributed. Far from attempting to extend copyleft beyond the scope of copyright, the licenses explicitly recognize, in the words of GPLv2 §2, that

> If identifiable sections of [other program code] are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License.

GPLv3 §5 goes further to protect the rights of programmers in such works from overextension of copyleft when it says that

> A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individ-

> ual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

The GPL licenses, then, are explicit about limiting the scope of copyleft to the scope of copyright. They do not, however, as is sometimes suggested, do so in a way that distinguishes "dynamic" from "static" linking of program code in "early-binding" programming languages. It is occasionally suggested that a subroutine "dynamically" linked to GPL'd code is, by virtue of the linking alone, inherently outside the scope of copyleft on the main work. This is a misunderstanding. When two software components are joined together to make one work (whether a main and some library subroutines, two objects with their respective methods, or a program and a "plugin") the combination infringes the copyright on the components if the combination required copyright permission from the component copyright holders, and such permission was either not available or was available on terms that were not observed. Where a combination is made with GPL'd or AGPL'd components, the only available permission is copyleft, and its terms must be observed on the combination as a whole if the GPL'd component is to be used at all. Whether the combination is made with a linker before distribution of the executable, is made by the OS kernel in order to share libraries for execution efficiency at runtime, or results from "late-binding" of references in the language at runtime (as in Java programs) is irrelevant.[3]

Automatic Downstream Licensing

Each time you redistribute a GPL'd program, the recipient automatically receives a license, under the terms of the GPL involved, from every upstream licensor whose copyrighted material is present in the work you redistribute. You can think of this as creating a three-dimensional rather than linear flow of license rights. Every recipient of the work is "in privity," or is directly receiving a license from every licensor.

This mechanism of automatic downstream licensing is central to the working of copyleft. Every licensor independently grants licenses, and every licensor independently terminates the license on violation. In the case of GPLv2, this termination is automatic, while under GPLv3 the party breaching the

[3] It is important to note that LGPL may differ in its requirements for statically or dynamically linking modules with a covered work. This is explained in detail at page 37, below.

license's terms may be able to cure before termination. Parties further downstream from the infringing party remain licensed, so long as they don't themselves commit infringing actions. Their licenses come directly from all the upstream holders, and are not dependent on the license of the breaching party who distributed to them. For the same reason, an infringer who acquires another copy of the program has not thereby acquired any new license rights: once any upstream licensor of that program has terminated the license for breach of its terms, no new automatic license will issue to the recipient just by acquiring another copy.

Protection Against Additional Restrictions

Users' freedoms cannot be protected if parties can add restrictive terms to the copyleft. The "no additional restrictions" principle is therefore unwaivable if the GPL licenses are to achieve their primary objective. GPLv2 therefore requires that the *only* license terms available for works based on GPLv2 works are the terms of GPLv2. GPLv3, in §7, enumerates a few classes of permissible additional terms, to allow very limited license variations in particular circumstances. But with these exceptions, the "no further restrictions" principle applies strictly. For these reasons, acceptance requirements or ceremonies, including "click to accept" installation routines, violate the terms of GPL.

Imposition of Repugnant Terms

The principle of "no additional restrictions" deals with securing the copyleft against deliberate modification by one party in the sequence of sharing. But what if a party is under restrictions invisible to the license? If A has no legal right to pass to B all the rights in a program that he got from C, B will wind up with fewer rights than GPL seems to afford her, because the law will always conclude that A could not pass more rights than he had. The GNU GPL licenses resolve this difficulty in the interest of users' rights: if legal conditions imposed on you (by court order, or by your acceptance of legal restrictions in another instrument) prevent you from passing along all the rights contained in the GPL, you cannot distribute at all. The copyleft licenses do not allow for external legal conditions as an excuse for non-compliance with their terms.

This principle, against the "imposition of repugnant terms" is of importance with respect, for example, to patent licensing. If you accept a patent license that contains more restrictive terms than those of the relevant GNU GPL license, the repugnant terms in the patent license will prevent you from distributing the program altogether. It isn't the existence, or the assertion, of the patent claims themselves, or even the offering of a GPL-incompatible patent license that triggers the provision: only your acceptance of those terms, or a court order imposing them on you, can do so.

License Provisions Analyzed

From our discussion of *why* the GNU family of licenses requires some users and distributors of programs to take measures to protect users' freedoms downstream, we can now turn our attention to *what* those requirements are, and *how* to satisfy them. The following section describes, for each of the family of licenses, which provisions in the license text give rise to compliance obligations, with some specific suggestions for the most efficient means of discharging them. In the following section, we discuss more broadly how to achieve compliance through better software governance, and how to respond to compliance inquiries and complaints.[4] In practice, we have found that presenting this information in license-terms order yields a useful reference for lawyers.

GPLv2

Promulgated in 1991, GPLv2 is still after 23 years in service, the most widely used free software license of any type.[5] The Preamble of GPLv2, which primarily restates the social intentions of the licensor, also, along with Section 1 is the source of the requirement that the full license text must accompany every distribution of a source or binary version of each licensed work, to ensure that users have actual notice of their rights. This requirement is responsible for a surprisingly significant fraction of compliance complaints, primarily because users are not provided with required information about

[4] Most of the material presented here can be found in the authoritative and extremely useful "Frequently Asked Questions about the GNU Licenses" by the Free Software Foundation, available at `https://www.gnu.org/licenses/gpl-faq.html`.

[5] `http://www.blackducksoftware.com/resources/data/top-20-open-source-licenses`.

the presence of GPL'd programs and the applicable license terms in physical products that they have purchased. The most effective mode of compliance engineering is to treat the required license texts as a "make target" in the compiling, packaging and distribution of the software, so that license texts and other "collateral" for the software in a product stack are produced and verified at the same stages and in the same fashion that the binaries themselves are generated, tested and packaged. Businesses that SFLC has worked with around the world, including large multiform firms with many semi-independent business units making and distributing products containing GPL'd software have been able to achieve perfect compliance on these and similar points with fixes of this type.

The other compliance obligation arising from the frontmatter of GPLv2 is the obligation not to modify the license text itself. Occasionally compliance issues arise solely because a distributor has elected to remove words from GPL, or to add remedy or other "ancillary" provisions. Like the rest of the GNU family of licenses, GPLv2 is the copyrighted work of the Free Software Foundation, and only verbatim copying is allowed.

Section 1: Copying and Distributing Unmodified Versions of the Program

This section addresses the right to copy and distribute verbatim copies of the Program as you receive it and mandates the following:

- conspicuous publication of appropriate copyright notices;
- retention or addition of disclaimer of warranty, unless you are offering a warranty, which you may do;
- retention of all notices referring to GPLv2; and
- provision of a copy of the license.

Contrary to popular belief in the United States—where copyright notice was required for copyright protection until 1978—notice and registration are not required under the Berne Convention to secure copyright. In the US and most of the rest of the world, each developer holds copyright in his or her code the moment it is written. Notices, and—in the US—registration, do have some remaining legal effect, however. Addition and preservation of notices are required by almost every free software license. For detailed discussion of best practices for maintaining copyright notices, see SFLC's "Managing Copyright Information within a Free Software Project."[6]

Section 2: Modifying the Program and Distributing the Modified Form

Section 2 is the heart of the copyleft in GPLv2. It directly implements the requirement that modified versions be distributed on the same license terms. The right of modification is one of the four basic rights, and GPLv2 therefore protects rather than restricting its exercise. But as section 2 states, any modified version of a GPL'd program is *ipso facto* a "work based on the Program," and can only be distributed to others on the terms contained in GPLv2 and on no other terms.

Section 2(a) requires that all modified versions be so marked, with basic indication of the modifications made, the date of modification, and some identification of the modifier. Compliance is achieved by any markings in source code that contain this information in a reasonable form. Not all the information available from a source code version control system need be provided, nor is the requirement a substitute of the project-level ChangeLog or similar file. Appropriate compliance assists a programmer using or reviewing any particular source file to know from what version of a project or program that source file comes, and to trace the history of recent substantive modification.

In addition, as a further protection for others' legal notices, section 2(c) provides that if the program before modification "normally reads commands interactively when run" and displays or prints legal information, all copyright notices, warranty disclaimer, modification indications and a pointer to the license text must be displayed or printed in interactive use. For example, the GNU Debugger, gdb, or GNU zile.

GPLv2 §2 is as much about what copyleft does *not* require, as about the requirements themselves. The remaining three paragraphs of §2 limit the scope of copyleft in both intention and operation. Section 2 makes clear that GPLv2 does *not* purport to extend copyleft to code written entirely by non-licensing authors who have made separate and independent works that *could* be usefully combined with GPL'd software. So long as those separate,

[6] `http://softwarefreedom.org/resources/2012/ManagingCopyrightInformation.html`.

independent works are delivered separately, or even are only "aggregated" with GPL'd programs on a medium of software distribution like a DVD or a USB stick, the license specifically disclaims any copyleft obligations on that code. Only when a new "work based on the Program" is created by combining this previously-separate work with GPL'd code, or when the work is not really independent because it cannot execute unless combined with GPL'd code, does the copyleft obligation attach to the new work so created.[7] In the aggregation case, the license makes clear, the aggregate cannot be released under an umbrella license that prohibits users from exercising rights that each program's individual license grants them.

Section 3: Copying and Distributing a Program in Object Code Form, Modified or Unmodified

GPLv2 §3 sets the terms for distribution of executable or binary versions of the work. Though §2 is the heart of the copyleft, §3 is the source of the most important compliance obligations. Users cannot study, understand , modify and share programs without buildable source code, which is therefore what §3 requires they be offered. There are three primary forms of compliance with §3:

1. When executable copies of GPL'd programs are available for copying over the Net, complete and corresponding source code may be made available for copying from the "same" place;
2. When executable copies of GPL'd programs are distributed on physical media, they can be accompanied by source code on the same media, or on other units in a collections of, e.g. DVDs or USB memory sticks;
3. Executables distributed in these and other ways may be accompanied by a written offer, valid for at least three years, for complete and corresponding source code. If you are a non-profit redistributor casually redistributing someone else's packaging of GPL'd code, which was accompanied by written offer rather than source code, you may pass

[7]This covers the full scope of primary infringement liability for non-compliance. Secondary copyright liability may attach in other ways. If, for example, I create a proprietary enhancement to a GPL'd work and deliver it separately, advertising its availability and encouraging downstream users to make and redistribute the infringing combination of my proprietary enhancement and the original GPL'd work, once such an act of primary infringement has occurred, the GPL-licensing copyright holders can and will succeed in claims for inducing and contributing to copyright infringement against me. See MGM Studios, Inc. v. Grokster, Ltd., 545 U.S. 913 (2005).

> along the written offer you received. For an example of a compliant written offer for source code, see Appendix I.

What constitutes "complete and corresponding source code"? In the first place, source code is "the preferred form for making modifications to the work." For interpreted computer languages, from shell script to Perl to Javascript to APL, the preferred form for making modifications may well be the only form of the program. (Attention should be paid, however, to compression, minimization, obfuscation, and other modifications to interpreted code that may result in the creation of a "non-source form.")

Complete, and as §3 says, "corresponding" source means the source code, build scripts, makefiles, configuration files and other materials necessary to build a version of the program precisely equivalent to the executable delivered, to modify that source code, and to build the modified version. Lacking anything necessary to build the precise version delivered means the source distributed is incomplete and not "corresponding." "Corresponding" source code unsuitable for modification and rebuilding into modified versions, as for example because too heavily obfuscated to be practicably modifiable, would not be "complete."

The completeness requirement contains an exception for system libraries, defined by §3 as:

> anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

The goal of this exception is to relieve distributors of the requirement to deliver source code that the downstream user does not require in order to exercise her rights, because that code is available as part of the operating environment the user can be presumed to have already available to her.

Failure to provide or offer complete and corresponding source code is the single largest failure mode leading to compliance disputes. All the litigated cases of community complaint against commercial redistributors involved failure to provide complete and corresponding source on valid request. Verification that source code provided in response to compliance request is in fact complete and corresponding is the single most burdensome and difficult part of compliance enforcement. All these problems result from inadequate governance. Parties do not retain the source code of binaries embedded in their products. They lose the human resources that knew how to build their software components. When, at some later time, copyright holders whose works have been modified and redistributed seek the source code, organizations don't know how to provide it, or provide radically incomplete or inadequate code, and reflect helplessness and ignorance as though either constituted legal excuse.

None of this should or need happen. If the tarball of complete and corresponding source code is also a "build target" in the software production process, so that no binary build is successful unless complete and corresponding source code has also been produced, no source code can be lost. So long as the component can be built, the engineering knowledge to build it cannot leave the firm. Provisioning of public-facing source code on the Web can be an automated part of product shipment or release to manufacturing. In our work with firms around the world, we have repeatedly found that relatively simple and completely inexpensive changes to software development practices could entirely eliminate modes of compliance failure that elude the scrutiny of "compliance tools" costing orders of magnitude more.

Use of a written offer for source code may time-shift your obligation to provide source code but it also increases your compliance costs in the long run. Your offer must be good for at least three years even if you have stopped distributing your product. This also makes it possible for "any party" to demand a copy of complete and corresponding source code, even if the requester has not received a binary.

If you are a commercial distributor of GPL'd program, even if you merely redistribute software produced by someone else, you are bound by the terms of the GPL. In such cases it is never sufficient to merely pass along the same written offer for source that you received from your upstream licensor or supplier and expect your users to get the source code from your supplier. In such instances it is advisable that you exercise your own rights as a user to request C&CS for all the GPL programs that your suppliers provided to you, preferably in an automated process. Once you receive such C&CS, passing it along with your product will ensure your compliance with the license.

The equivalent provision in GPLv3 is Section 6 that deals with conveyance of source code in non-source forms.

Section 4

Section 4 expresses the "no additional restrictions" concept directly. It also prohibits sublicensing on non-GPLv2 terms. This prohibition bears notice.

It is conventional, but incorrect, to state that copyleft obligations apply only when GPL'd code is distributed. Sublicensing is a separate activity that also gives rise to copyleft obligations.

Until the second decade of this century, the possibility of noncompliance with GPLv2 through sublicensing was a rare, near theoretical occurrence. No longer. "Cloud" offerings of "software as a service," in which virtualized server stacks are sold on a utility-computing basis have brought about significant noncompliance. Firms offering such virtualized server instances often enter contracts with their users purporting to "license" the software in their virtualized server stacks, which will almost always include GPL'd OS components being thereby sublicensed on incompatible terms. Efforts to avoid this outcome require no technical or engineering measures.

GPLv3 prohibits sublicensing altogether, see page 20 below. "Cloud" or "software as a service" deployments of GPLv3 software can therefore also be technically fully compliant but infringing as a result of mistaken contracting.

Section 4 of GPLv2 provides that license termination is automatic in the event of violation of terms. This rule was changed in GPLv3 §8, see page 26 below. Because license termination automatically occurs on infringement, resumption of distribution (which can be automatic under GPLv3) requires an affirmative regrant of the right to distribute from the copyright holder. Although community copyright holders have been uniformly cooperative with requests by former infringers to resume distribution upon rectification of rights, it is the inevitable consequence of more concentration of GPLv2 copyrights in for-profit organizations engaged in monetizing their copyrights that there will be future cases in which automatic termination results in substantial monetary demands for reinstitution of rights. Because automatic termination leads immediately to injunction a non-community monetizing GPLv2 copyright holder gains potentially unfair leverage from automatic termination. This is among the most important reasons why commercial redistributors should now prefer GPLv3 to GPLv2 terms from their upstream technology providers.

Section 5: Acceptance of License Not Required

The English common law tradition has defined "license" for more than 500 years as a unilateral permission to use property rights granted by the owner to non-possessors. In the traditional law school property course, the example of a license is an invitation to dinner. If I invite you to my house for dinner, and sue you for trespass when you cross the threshold, your defense is "license," a unilateral permission to be there. Confusion about the status of licenses as contracts is endemic in global legal analysis, because in many other legal systems, including those claiming descent from Roman law, the idea of non-contractual unilateral obligation is unrecognized. (Though you still can't sue someone for going where invited.) The GNU licenses are licenses in the common law sense: unilateral copyright permissions within limits. GPLv2 §5 states this frequently-controverted but unassailable legal distinction. Acceptance of the license is not required, nor is it sought. A requirement of acceptance, through clickwrap or other ceremonies, is not only not required, it is an imposition of additional conditions forbidden by the terms of the permission granted in the license. Compliance with §§4 and 5 therefore requires the explicit avoidance of contractual forms, acceptance ceremonies, monitored assent, or any other activity designed to turn unilateral permission into bilateral obligation.

Section 6: Automatic Downstream Licensing

This is GPLv2's "automatic downstream licensing" provision. Each time you redistribute a GPL'd program, the recipient automatically receives a license from each original licensor to copy, distribute or modify the program subject to the conditions of the license. There is no requirement to take any action to ensure the downstream recipient's acceptance of the license terms, see above. This places every copyright holder in the chain of descent of the code in legal privity, or direct relationship, with every downstream redistributor. Two legal effects follow. First, as §6 says, parties themselves remaining in compliance have valid permissions for all actions including modification and redistribution even if their immediate upstream supplier of the software has been terminated for license violation. Their licensed rights are not dependent on compliance of their upstream, because their licenses issue directly from the copyright holder. Second, automatic termination cannot be cured by obtaining additional copies from an alternate supplier: the license permissions emanate only from the original licensors, and if they have automatically terminated permission, no act by any intermediate license holder can restore those terminated rights.

It also follows, as §6 makes clear, that licensors are in no way responsible for enforcing compliance by third party recipients or distributors. Every licensee gains or loses permissions from each original licensor solely on the basis of its own conduct.

Section 7

This provision deals with the problem of the licensor who is prevented from passing on the freedoms guaranteed by the license due to interfering obligations. Compliance with §7 means if conditions are imposed upon you that conflict with the requirements of the license, you cannot and must not use the license to redistribute others' code. Unless the copyright holders have provided you with additional permissions, therefore, if you are under exogenous legal obligations you may not redistribute the program or works based on the program at all.

The term "conditions imposed on you" is crucial to the interpretation of GPLv2 section 7 in this regard. Conditions are "imposed" if a court or another settlement agreement you may have entered into has imposed them. So a compulsory license or a court-ordered settlement which contains provisions requiring, e.g. per-unit royalties or bars downstream resale would trigger §7 obligations. Accepting a patent license, or entering into any other contract restrictive of rights has the same effect.

Compliance with GPLv2 §7 is therefore a matter of legal review rather than technical or engineering practice. To our knowledge, no litigation or adversary proceedings have been brought by community organizations for breach of §7 in the past, but in our practice we have more than occasionally discussed with firms whether particular arrangements they had entered into should be nullified or modified to avoid §7 problems.

The other crucial function performed by GPLv2 section 7 is the function of discouraging separate settlements in patent disputes. A settlement allowing only the settling party to distribute GPL'd software with patent safety cannot be an effective limit unless it prohibits redistribution by downstream parties. If it does so, however, it constrains the rights being transferred by GPL, and falls foul of §7.

GPLv3

Promulgated in 2007 after eighteen months of public discussion, GPLv3 adds some additional compliance obligations under certain circumstances, but also makes central compliance obligations easier to fulfill, and substantially removes the risk of automatic termination for unintentional infringement.

The drafting principles adopted in GPLv3 took as their principal aspiration the avoidance of US-specific or system-specific vocabulary, in order to avoid importation of system-specific legal assumptions. The words "distribution" and "derivative work" in GPLv2, for example, caused analytical confusion for little overall benefit. With the immense expansion of use of GPL'd software around world since 1991, FSF and its lawyers here decided to avoid system specific language to remedy that risk of confusion.

Two terms are used in GPLv3 deliberately exotic to the language of international copyright. To "propagate" a work covered by the license means any activity that requires permission of copyright holders in the local legal system where the activities are carried on. Personal use or modification for personal use are activities explicitly excluded from "propagation" regardless of domestic copyright law, in order to prevent domestic copyright law from trenching on freedoms 0 to 2. As a further benefit, because "propagation" includes all exclusive rights granted under any particular copyright regime, regimes that require a valid license to account for all exclusive rights are automatically complied with.

Any propagation that enables other parties to receive or make copies of the work, is called "conveying." In general, conveying is the activity that triggers copyleft obligations. Explicitly exempted from "conveying" is "interaction with a user through a computer network, with no transfer of a copy," which distinguishes GPLv3 from AGPLv3, see page 33 below.

Use of these terms allows GPLv3 to be interpreted authoritatively based solely on local legal knowledge and the technical facts. Parties who may have questions about compliance with GPLv3 under domestic copyright provisions need only to ask two questions, one legal and the other factual:

1. Do I need a license under local copyright law to carry out this activity?
2. Does my activity enable any other party to make or receive a copy of the program?

If question 1 is answered by a local copyright lawyer in the negative, then the activity involved is not propagation and is permitted by GPLv3 without more. If the answers to both questions are positive, then the activity involved is "conveying," and copyleft obligations imposed by §§4-6, below, apply.

GPLv3 requires the conveyors of the work to provide all the source necessary to build the program, including supporting libraries, compilation scripts, etc. "Corresponding Source" in GPLv3 §1 is defined as:

> [A]ll the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

Both versions of GPL except from the requirement to produce complete and corresponding source code the source code for "System Libraries", commonly called "the system library exception." As can be seen in the quotation above, the intention of the "System Library exception" is to avoid the need for production of source code which the user can expect to receive with the copy of the operating system, to the extent source code for the operating system is available at all, or of the generally-available programming tools used to produce the program.

Section 2: Basic Permissions

The license explicitly affirms your unlimited permission to run the unmodified program. You can only make modifications, or otherwise propagate a covered work, even one that you do not convey, so long as you are in compliance with the license terms. If your license is terminated, then, for example, you cannot continue to service your copies by making new modified versions.

Section 2 clearly states that sub-licensing under the terms of the license is not allowed. As we have previously noted, incautious contract drafting in "software as a service" or other cloudy contexts can fall, and has fallen, foul of this prohibition.

Section 4: Conveying Verbatim Copies

Under §4 anyone may convey without limit verbatim copies of program source code. The conveyor's compliance obligation is limited to the curation of copyright notices and the provision of the license text along with the source code. If copyright notices are missing, they must be added. All notices of attribution and additional permission pursuant to §7 below must be preserved (or added in the case of new attributions and permissions), and all upstream disclaimers of warranty must be preserved. Automated software governance processes are sufficient to meet all §4 obligations simply and cheaply.

Section 5: Conveying Modified Source Versions

Notice obligations under §5 are identical to those under §4, with the additional requirement to provide notice of modification, date and some identification of modifier as under GPLv2 §2(a), above.

§5 also restates the aggregation provision of GPLv2 §2, as follows:

> A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

Section 6: Conveying Non-Source Forms

Section 6 states the compliance obligations for distributing "non-source forms" of a program, which means any form other than the form preferred for making modifications. In addition to binaries or executables, non-source forms therefore include obfuscated, minimized, compressed or otherwise non-preferred forms for modification. Non-compliance with GPLv3 in the distribution of Javascript on the Web is becoming more frequent, and although no disputes have so far resulted, in the absence of more careful compliance activity in this area they are eminently foreseeable.

The requirement to provide complete and corresponding source code under §6 closely parallels the provisions of GPLv2 §3, above, but with changes designed to make compliant provisioning easier under contemporary technological conditions. Source code may be provided by online distribution as under GPLv2, but the "same place" requirement of GPLv2 §3 is loosened slightly:

> If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

This provision allows, for the first time, for third-party provision of complete and corresponding source code in commercial distribution situations. The obligation remains on the party distributing the non-source form to point prominently ("next to" the non-source download) to the third-party source code provisioning server, and to ensure that this third-party server remains in operation for required period.

Section 6 also allows the provision of source via such a server when the binary or other non-source form is distributed by peer-to-peer protocols such as BitTorrent. Here the requirement is only that each peer be effectively informed of the location of the source code on a server as above.

Section 6 slightly modifies the rules pertaining to written offers. Under GPLv2 §3(b) above, written offers must be valid for a period of three years. GPLv3 §6 requires that the period of validity of the offer be the longer of three years and valid for the period for which other spare parts and/or customer service are available for any physical product embedding the program.

The definition of "system libraries" in the exception from corresponding source code in §1 substantially varies from the parallel definition in GPLv2 already discussed. The new definition says:

> The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal

> form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

This definition clarifies that language processors (compilers and interpreters) included in the distribution of the program's target operating system are not part of the corresponding source. As in GPLv2, libraries or other enabling or ancillary code accompanying major components of the operating system are excepted as well, as is any code providing an implementation of a publicly documented "Standard Interface" for which a source code version of an implementation is publicly available.

The most widely-discussed new requirement in GPLv3 is also found in §6: the requirement to provide "installation information" for GPLv3 code in "user products." The intention of the "installation information" provision of §6 is to protect the rights of users to modify and run modified versions of GPLv3 code embedded in products that they own and personally use. Without such a provision, "lockdown" of devices to prevent modification of software by legal owners could vitiate the freedoms GPL-licensing authors intend their users to have. If all embedded devices implemented such lockdown, the freedoms guaranteed by GPL would become, as the US Supreme Court once said, "a promise to the ear to be broken to the hope . . . ." *Edwards* v. *California,* 314 U.S. 160, 186 (1941) (Jackson, J. concurring).

A "user product" is

> either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling.

When non-source forms of a GPL'd program are conveyed to a user embedded in a "user product" by permanent transfer of ownership or control of the device, or by lending or rental for a fixed term, the corresponding source must be supplemented by sufficient technical information

> methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made

Unless technical measures have been employed in the device to prevent the installation and execution of modified versions, this requirement is trivial. Experience since 2007 has tended to show that manufacturers who intend to lock down their consumer products therefore avoid GPLv3 software. No adversary actions over non-compliance with the installation information requirement have occurred to our knowledge since the promulgation of GPLv3.

Section 7: Additional Terms

While GPLv2 does not allow for any additional restrictive terms, GPLv3 allows for some specific limited variations, thus varying the strict copyleft of GPLv2 in the interest of broader compatibility with other licenses. Section 7 consists of additional permissive and non-permissive terms which can be used to supplement the terms of the License. Additional permissions beyond those of §7 can be placed by any copyright holder on her contributions to a licensed work. GPLv3 states, as GPLv2 does not, that all additional permissions must be in writing. In the drafters' view, the granting of informal or unwritten additional permissions under GPLv2 created unnecessary doubt. Written additional permissions previously granted by upstream authors can be removed before conveyance of a copy of the Program if so desired by the conveying party.

Section 7(a) – Disclaimer of warranty: Copyright holders can disclaim warranty or limit liability differently from the terms as provided under §§15 & 16. Internationalization of GPL required the drafters to permit the variation in such terms necessary to deal with the absence of any international harmonization of the laws of warranty and disclaimer.

Section 7(b) – Preservation of appropriate legal notices. Section 7 permits additional terms requiring preservation of legal notices, including on output from execution of covered works.

Section 7(c) – This provision allows for prohibition of misrepresentation of original material and makes GPLv3 compatible with permissive licenses that require modified versions be marked in "reasonable" ways which are different from the precise marking requirements of GPL itself.

Section 7(d) – Limiting the use of names of licensor for publicity purposes. This provision was added to provide compatibility with licenses which prohibit the use of the licensor's name on unmodified versions and other prohibitions on advertising rights. The long-standing and occasionally troublesome incompatibility with the "BSD advertising clause" is an example of the friction removed by this provision.

Section 7(e) – No grant of rights under Trademark Law. This provision serves a similar function with respect to "no trademark grant" clauses in permissive licenses.

Section 7(f) – allows for requiring of Indemnification of authors and licensors, removing one of two primary sources of incompatibility with the Apache Software License version 2.0.

Any other non-permissive additional terms apart from those stated above are considered "further" restrictions and are prohibited by §10. If you add any kind of additional terms in accordance with §7, you must ensure that the terms are placed in the relevant source files or provide a conspicuous notice about where to find the additional terms. Compliance obligations as presented by §7 are entirely a matter of legal review, without technical or engineering consequences.

Example: GNU Compiler Collection (GCC) Runtime Library Exception This GCC Runtime Library Exception ("Exception") is an additional permission as provided by Section 7 of GPLv3. The purpose of this Exception is to allow compilation of non-GPL (including proprietary) programs making use of the header files and runtime libraries covered by this Exception and containing code from the copyleft toolchain embedded by the compiler in the object code of the program as part of the compilation process. The GCC Runtime Library Exception covers any file that has a notice in its license headers stating that the exception applies. This includes libgcc, libstdc++, libfortran, libgomp, libdecnumber, libgcov, and other libraries distributed with GCC.

Section 8: Termination

The change from automatic termination on violation to a period for self-cure and automatic reinstatement of rights for first-time violators is one of the most important modifications of the basic rules of compliance enforcement from GPLv2 to GPLv3.

> However, if [after your discovery of a breach of the license terms] you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
>
> Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

The realignment of incentives to find and fix problems is a major advantage of GPLv3 over GPLv2. Automatic termination requiring positive reinstatement by copyright holders to avoid loss of distribution rights created incentive to hide problems rather than fixing them. Cases of "whole distribution infringement," that is, failure to provide corresponding source code for all the copylefted programs in an entire OS distribution embedded in a device, result in the infringement of hundreds or thousands of copyrights at once. This situation poses immense obstacles to successful remediation by violators, who may without restoration of rights be unable to distribute their products. In one recent matter, we were retained to audit a medium-sized business relatively inexperienced in compliance engineering. We found that a "whole distribution infringement" situation in past products required them to contact all the copyright holders in more than 100 packages licensed under GPLv2, in order to assure their right to continued distribution of new models of products built compliantly, because their previous non-compliant distribution of the same programs in earlier products led to undiscovered automatic termination.

Section 10: Automatic Licensing of Downstream Recipients

This Section manifests the no-additional restrictions and automatic downstream licensing principles as discussed in other parts of this document. Enforcing your patent claims against those to whom you distribute a program constitutes an additional restriction and is prohibited by Section 10

Mergers and Acquisitions Section 10 also clarifies that in business acquisitions, whether by sale of assets or transfers of control, the acquiring party is downstream from the party acquired. This results in new automatic downstream licenses from upstream copyright holders, licenses to all modifications made by the acquired business, and rights to source code provisioning for the now-downstream purchaser.

In our experience, the process whereby these matters are adjusted in most M&A situations are ludicrously expensive and inefficient. A simple waiver and release of all claims to GPL compliance against the purchased entity by the purchaser, issued before closure, removes the problem. If the purchasing entity has adequate software governance systems in place, all software acquired in the course of the entity transaction is input to the standard governance processes for acquired software, and downstream compliance by the new merged entity is automatically handled.

Section 11: Patents

GPLv3 provides for two classes of patent commitments:

1. Prohibition of Enforcement of patent claims against those to whom you distribute: GPLv3 §10 makes explicit that demands for acceptance of patent licenses or payment of patent royalties by those to whom a licensee directly distributes are additional conditions that may not imposed. This provision establishes a uniform rule of patent exhaustion with respect to GPL'd programs regardless of the domestic patent law in any particular system or locale.

2. Grant of license to claims in contributor versions: Section 11 introduces an affirmative grant of rights to patent claims by those who contribute code to GPL'd programs. The intent is to prevent parties within the commons from aggressively asserting patents against

> users of code they have themselves modified, preventing a form of commons betrayal by "insiders" to the community. A contributor's patent claims necessarily infringed by the version of the program created by the incorporation of its modifications are licensed to all subsequent users and modifiers of the program, or programs based on the program. No patent claims only infringed by subsequent modifications by other parties are thus licensed. Patent claims acquired after the making of the "contributor version" necessarily infringed by that version are also licensed by this provision at the time of their acquisition or perfection. When a company with a large number of such claims acquires the program's modifier, all claims held or thereafter acquired by the purchaser are automatically licensed under this provision. The acquisition of Nokia by Microsoft, for example, resulted in the automatic licensing of all Microsoft claims now or hereafter acquired which read on any contributor version of any GPLv3 program ever modified by Nokia. The wholesale decimation of Microsoft patent claims on GPLv3 programs arising from the purchase of Nokia has so far been unremarked in the industry.

Section 11 also deals with the possible efforts to undermine common rights in GPL'd code by parties colluding to enforce discriminatory patent licenses. If A takes a patent license from B that benefits A only, rather than A's customers or their distributees, A imposes risk from B's patents on others that it does not suffer itself. Under many circumstances, this is an acceptable outcome. If, however, A is the only possible source of the program, by taking such a license and distributing in reliance on it, A is in effect helping B to "take the program private." Section 11 poses several possible solutions A could undertake, including disclaiming the license or securing its extension to all parties. But the most likely compliant action, and the least costly, is simply to ensure that the program source code is available permanently through some third party, so that those who wish to develop or commercially deploy or redistribute the code at their own risk can do so.

In addition to this "downstream shielding" provision, §11 takes two other steps to avoid exploitive manipulation of discriminatory patent licensing tactics:

> If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate,

> modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

If, for example, company B offers purchases from company A a number of coupons entitling third parties to the GPL'd programs distributed by A, and provides access to its patent claims reading on those programs only to those acquiring A's product through purchase or gift of the coupons from B, this provision deprives B of its value in the transaction by automatically extending the patent license to all other users. Experience suggests that B will retroactively assert that the coupons thus issued did not apply to any GPLv3 programs in the collection of software covered by the coupons. This won't work.

> A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

This provision was adopted in order to prevent the spread of a particular transaction type entered into between two firms during the public discussion process conducted by the Free Software Foundation leading to the adoption of GPLv3. There are no current compliance issues related to this provision, so far as we are aware.

Patent Assertion & Retaliation: GPLv3 implements a narrow scheme of patent retaliation against those who undertake patent aggression against licensed works which they distribute or contribute to. Read together, §§8, 10 and 11 establish defensive patent suspension and termination conditions.

The termination provision of section 8 states that if a licensee violates the terms of the GPL, a contributor may terminate any patent licenses that it granted under §11 to that licensee, in addition to any copyright permissions the contributor granted to the licensee. Therefore, a contributor may terminate the patent licenses it granted to a downstream licensee who brings patent infringement litigation in violation of §10.

Section 12: No Surrender of Others' Freedom

GPLv3 §12 is functionally identical to GPLv2 §7. An illustration in the GPLv2 provision caused more confusion than it was worth, and has been omitted. In all other respects, including in their compliance consequences, the provisions have the same effect. See page 18 above.

Section 13: Use with the GNU Affero General Public License

This section specifically gives permission to link or combine any work that is licensed under GPLv3 with a work licensed under AGPLv3. The compliance consequences of this permission are discussed with respect to the AGPLv3, below.

The resultant work will be governed by the terms of GPLv3 but if the work is used over a network then the terms of AGPLv3 shall apply to the entire work.

You cannot take code released under the GNU AGPL and convey or modify it however you like under the terms of GPLv3, or vice versa. However, you are allowed to combine separate modules or source files released under both of those licenses in a single project.

When a combined work is made by linking GPLv3-covered code with AGPL-covered code, the copyleft on one part will not extend to the other part. That is to say, in such combinations, the Affero requirement will apply only to the part that was brought into the combination under the Affero license. Those who receive such a combination and do not wish to use code under the Affero requirement may remove the Affero-covered portion of the combination.

GPL Special Exception Licenses

GNU Classpath & GPL Linking Exception

GNU Classpath is the GPL-licensed set of essential libraries for the Java programming language. It is licensed under the GPL plus a special exception:

> Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.
>
> As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

AGPL

Whether to extend the copyleft concept to the delivery of services by free software over a network is a complex issue long discussed in the free software community. Freedom zero requires that any user be allowed to run any program for any purpose, which of course includes the provision of computing services to others. Freedom two requires respect for the right of private modification. Their combination requires that anyone be able to run privately-modified copies of GPL'd programs for the purpose of providing computing services to others.

But there are good reasons why programmers developing software for use in network services provision, particularly the infrastructure of Web applications, would want to require that parties offering services using modified versions of free software allow their users the same rights offered to those modifiers by the original developers. Federated or "free" services platforms benefit from this licensing model.

Rather than resolving the conflict within the scope of one license, the Free Software Foundation experimented over an extended period with licenses for such "copylefted services" software that could be made partially compatible with GNU GPL. Two such licenses, AGPLv1 and AGPLv3, different in their architecture but similar in their intention, have been published and used.[8] The "A" in AGPL standards for "Affero," the name of a software project pioneered by Henry Poole, a director of FSF. AGPLv1 was written by FSF specifically for use by Affero, and was published in 2003. AGPLv3 was designed and drafted in parallel with GPLv3, to unify more closely the relation among these licenses, and to permit use of GPLv3-licensed code in AGPLv3-licensed projects.

The common architectural feature of the AGPL licenses is a separate "trigger" condition, in addition to the triggering conditions for copyleft obligations ("distribution" and sublicensing established by GPLv2) and "conveying" established by GPLv3. The *scope* of copyleft under the AGPL licenses is the same as the scope of copyleft under the respective version of GPL. Only the condition that gives rise to the obligations to provide corresponding source code and license texts are changed.

AGPLv1

AGPLv1 is GPLv2 with a single additional requirement, found in §2(d):

> If the Program as you received it is intended to interact with users through a computer network and if, in the version you received, any user interacting with the Program was given the opportunity to request transmission to that user of the Program's complete source code, you must not remove that facility from your modified version of the Program or work based on the Program, and must offer an equivalent opportunity for all users interacting with your Program through a computer network to request immediate transmission by HTTP of the complete source code of your modified version or other derivative work.

[8] AGPLv2, published in 2007, is a transitional license "stub" permitting any licensee to distribute works based on AGPLv1 programs under the terms of AGPLv3.

This requirement, parallel to the "preservation of notices in interactive use" requirement of GPLv2 §2(c) was invented by Bradley Kuhn when he was Executive Director of the Free Software Foundation. Compliance with its requirement was straightforward: if the feature to request server-side source code was present in a program "intended to interact with users through a computer network," compliance consisted of not removing it.

AGPLv3

AGPLv3 is GPLv3 with one additional requirement, contained in AGPLv3 §13. Like AGPLv1, AGPLv3 makes no modification whatever to the scope of copyleft. In deciding what source code is the "corresponding source" to be provided or offered under AGPLv3 §6, it is sufficient to ask what source code would have to be provided if the same program were delivered to its user in a non-source version on a medium of physical distribution.

The additional triggering condition of AGPLv3 is as follows:

> if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Compliance requires that any version of the program that you yourself modify, and which retains the architecture of interacting with its users remotely over the network, must afford the user an opportunity to receive corresponding server-side source code over the network. If a corresponding source archive is itself a "make target" in the process of building the program, as we suggest with respect to all copylefted works, then the implementation of the compliant feature for early-binding languages is as simple as outputting that source archive (and possibly other source archives from other running programs that are part of the same system) into the communication stream as constant data when the running program receives the relevant request. This technical approach to compliance also resolves "scope of copyleft" issues for programs written in early-binding compiled languages: if components of the program as run are determined at make time, then what it took to make the program, minus the system libraries and any other excepted code, is the corresponding source.

Taken together, GPLv3 §13 and AGPLv3 §13 ¶2 establish the compatibility structure for combining GPLv3- and AGPLv3-licensed code. GPLv3 gives permission for code under its terms to be combined with code under AGPLv3's terms, each part to be treated as under the terms of its respective license. AGPLv3 code cannot be relicensed under GPLv3, or the reverse, without additional permission from the respective copyright holder. Source code licensed under GPLv3 is part of the corresponding source of any work combining code under the two licenses.

LGPL

The Lesser General Public License, whose terms apply to the GNU C Library, among other works, is sometimes described as a "weak copyleft" license, because code licensed under LGPL's terms can be combined with code under non-free licenses, and is sometimes used in that fashion. But, although the two versions of LGPL currently in use differ very substantially in their content and architecture, they are instead both strong copyleft licenses with broad linking exceptions. Understanding how to comply with the licenses hinges on this often-misunderstood distinction. LGPL'd code is never placed under new or alternative terms when it is combined with code under non-copyleft licenses. The combination is permitted, but the LGPL'd components of the program remain under the terms chosen for them by their copyright holders. This is not closely analogous to the behavior of licenses like the Eclipse license that, at the file level, assures source code is distributed under copyleft terms, while binaries may be distributed under any terms at the redistributor's discretion. LGPL's versions are intended to assure downstream users' rights in that potion of the permitted combination which consists of LGPL'd code, in both source and non-source forms.

LGPLv2.1

LGPLv2.1 is a copyleft license in its own right, to which §6 is an exception permitting non-copyleft combinations. The analytical peculiarity is that the copyleft license from which LGPLv2.1 §6 is an exception is not GPL. It is

LGPLv2.1 §§1-5, which does not behave in all respects like any version of GPL.

Section 0: Definitions and Scope of License Section 0 of LGPLv2.1 states that translation "straightforwardly" into another programming language of an LGPL'd program or library of routines is the creation of a modified version of the licensed program. GNU GPLv2 and GNU GPLv3 do not contain this definition.

Section 2 Section 2(a) states that if a licensed work is a software library (defined in §0 as "a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables") permission is given to distribute modified versions only if those versions are themselves libraries. LGPLv2.1 code can therefore *not* be compliantly taken from its context in a library and placed in a non-library modified version or work based on the work. Section 6 does not provide an exception for this rule: a combination may be made of a modified version of an LGPL'd library with other code, but the LGPL'd code must continue to be structured as a library, and to that library the terms of the license continue to apply.

Section 2(d) states that modified versions of library routines must not require non-argument global data unrelated to library function in order for library functionality to work. This means that libraries may not be modified to require keys, tokens, tables or other non-functional restrictive data to be present in order for applications to call and receive services from routines in the library. Section 6 does not except from these requirements. Compliance requires avoidance of such schemes, which in the years since LGPLv2.1 was promulgated in 1999 have become relatively common.

These requirements of §2 are stringent, and standing alone they would trench too closely on users' freedoms. But LGPLv2.1 §3 allows all works under its terms, copy by copy, to be used instead under the terms of GPLv2 or any later version. This provides a pathway for those who do not want to use code under the requirements of LGPLv2.1 to do so under GPLv2 or GPLv3 at their discretion.

Section 6 of the license contains a functional anti-lockdown provision, requiring that any LGPL'd code provided to the user as part of such a proprietary or non-copyleft work must permit relinking or re-installation of a modified version of the library.

Section 2 repeats, as in GPLv2, that

> These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Given the context of §2, it is apparent that this clause is not to be read as limiting in any way the exception for combinations under §6. But with respect to modified versions of the LGPL'd work itself, this language in §2 reaffirms that §6 is inapplicable.

Section 4 Section 4 covers distribution of non-source forms of unmodified versions of covered works, or works modified as permitted by §2. Compliance with its provisions is as for compliance with the parallel provisions of GPLv2, above.

Section 5 Section 5 addresses directly the ambiguities arising from the application of copyright law to software:

> A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.
>
> However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

> When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

Accordingly, with the intention of avoiding situations in which such ambiguity may arise, §5 goes on to state:

> If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)
>
> Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

Here, the clear purpose of the provision is to waive copyleft in ambiguous situations to the extent of permitting combinations under the exception provided in §6. To our knowledge, no public action of enforcement of LGPLv2.1 in the area delineated by Section 5 has ever occurred.

Section 6 §6 provides the combination exception that is often misunderstood as a "weak copyleft" license, which it is not. It permits combinations of the licensed work or works based on it

> to produce a work containing portions of the Library, and [to] distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

These requirements on the licensed combination require that the license chosen not prohibit the customer's modification and reverse engineering for debugging these modifications in the work as a whole. In practice, enforcement history suggests, it means that the license terms chosen may not prohibit modification and reverse engineering for debugging of modification in the LGPL'd code included in the combination.

§6 also requires that:

> You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License.

This is not identical to the roughly parallel requirements of GPLv2 and GPLv3. Compliance requires slightly different measures with respect to the "credits" or "licenses" or "about" screens in interactive programs.

In addition, §6 requires either that source code be provided or offered for the LGPL'd portion of the combined work, or that a "shared library" mechanism be used that allows dynamic replacement of the licensed work by modified versions.

If source code is provided, §6(a) states requirements that vary from those of GPLv2 and GPLv3:

> [You must] Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

Not only must the source code be complete and corresponding, but it must be provided in such a way that a modified version of the entire combined work can be regenerated with a modified version of the LGPL'd work replacing the version originally provided to the user. When LGPL'd code is statically linked to a non-copyleft executable, for example, the required source code must also include sufficient material to split the distributed executable and relink with a modified version of the library.

Under §6(c), this source code may be offered in writing rather than provided, or it may be distributed by network under the terms of §6(d). In addition, under §6(e) the distributor may "verify" that the user has already received, or at least that the distributor has already sent to this particular user, the relevant source. This is evidently intended to prevent requiring duplicate deliveries in "whole distribution" situations.

If the distributor of the combined work intends not to distribute or offer the source code of the LGPL'd components, the LGPL'd work must be separately distributed (subject to source code delivery requirements as part of that separate distribution) and packaged in a "shared library" mechanism, which means

> that [it] (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

Taken together, these provisions mean:

1. If you create a program that links through a shared library mechanism to a work that is separately distributed under LGPLv2.1, then you can distribute the resultant program under a license of your choice and you need not convey the LGPL'd work's source code. If you distribute the library along with your program, or are the separate distributor of the work in another context or as another product, you must distribute its corresponding source under the terms of LGPLv2.1 or GPLv2+, at your option.

2. If you choose to statically link or otherwise combine your program with an LGPL'd work, you may choose your own license for the work provided the license terms limitations for user modification, reverse engineering and debugging are met, and given that the LGPL'd components are still governed by LGPL's terms. You must offer or provide complete and corresponding source code for the LGPL'd components. The source code material provided must be sufficient to regenerate the combined work with a user-modified version of the LGPL'd components.

Non-compliance with LGPLv2.1 is relatively widespread, probably because misimpressions about its terms easily dispelled by reading them are equally common. In our practice we represent some licensors who enforce their rights on behalf of their users. Public disputes are infrequent, because parties who are asked to square their behavior with the language of the license have always chosen to do so when politely but firmly requested.

LGPLv3

LGPLv3 was designed to rectify the architectural plan of the GNU family of licenses, by making the copyleft license from which LGPLv3 is a combination exception GPLv3. LGPLv3 is therefore an additional permission in the form provided for in GPLv3 §7, above.

Section 0: Additional Definitions Section 0 defines the "Library" it covers as a work that presents one or more interfaces at which a "use" can be made by an "Application." Class inheritance is "deemed" a use of an interface. An "Application," which is other program code using one or more "Library" interfaces can be combined with the code on the other side of the interfaces it uses to form a "Combined Work."

Section 1: Exception to Section 3 of the GNU GPL Section 1 excepts away the interference with use of LGPLv3 code as part of "effective technological measures" of access limitation for other copyrighted works provided otherwise by GPLv3 §3.

Section 2 Conveying Modified Versions Section 2 continues to require, as LGPLv2.1 §2(d) required, that the Library not be modified to require keys, tokens, tables, or other global non-argument data unrelated to function. This is again stated as a "good faith effort" requirement, but failure to cure on notice is strong evidence of the absence of good faith. Use of GPLv3 terms by removal of the additional permission, as provided for by GPLv3 §7, is the alternate path to compliance.

Section 3: Object Code Incorporating Material from Library Header Files Section 3 disposes entirely of the area of ambiguity in the use of header files and other such forms of Library material covered by LGPLv2.1 §5 by stating a rule applicable at the user's discretion to all such uses within copyright scope: giving notice that the library is used in the program and providing copies of GPLv3 and LGPLv3 along with the work.

Section 4: Combined Works Section 4 is the combination permission at the heart of LGPLv3. It restates the license limitation provision of LGPLv2.1 §2 to clarify that the terms on the Combined Work may not prohibit user modification of the Library code, or the debugging of such modifications to the Library code by means of whatever reverse engineering is necessary.

Section 4(d)(0) contains the source provision requirement, for the Minimal Corresponding Source, which "means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version [of the Library]." The alternative to the provision of source code is distribution by way of the "shared library" mechanism under §4(d)(1), described with respect to LGPLv2.1 §6, above.

In addition, §4(e) requires the delivery of "installation information" required to install the modified version of the Library in "user products" under GPLv3 §6. Where Library Minimal Corresponding Source is not made available under §4(d)(1), §4(e) reaffirms that "installation information" must still be compliantly delivered under the terms of GPLv3 §6.

All other provisions of GPLv3 are in force as previously described, and are not excepted by the additional permission granted in LGPLv3.

Understanding Your Compliance Responsibilities

In the preceding sections, we have explained why developers of free software use the GNU family of copyleft licenses, and we have presented a detailed review of the provisions of these licenses in relation to the licensors' intentions and objectives. In the following sections, we review the compliance responsibilities of parties "downstream" from copylefted projects, and we offer some general advice about responding to source code provisioning inquiries from users and compliance complaints from copyright holders.

Who Has Compliance Obligations?

The fundamental focus of the licenses, as we have previously explained, is the preservation of users' rights to run programs for any purpose, to copy, modify and share without restriction. Compliance obligations are parsimoniously imposed by the licenses, only to the extent that the drafters of the licenses and the licensors' using them consider to be minimally necessary to protect downstream users' rights. Copyleft is the central pillar of that system: the requirement to "share and share alike" all works based on the copylefted code freely offered from upstream. The licenses' ancillary provisions are intended to prevent the dilution or flouting of the "share alike" principle. The GNU licenses create a "commons" in traditional economic vocabulary; the restrictions imposed by the license are "governance" or "management" rules for the commons, like fisheries quotas, or groundwater preservation regulations in land use. Parties' obligations to the commons depend on their role in relation to its resources. We can delineate those roles and their respective responsibilities as follows:

1. Users of code under the licenses have no obligations, only rights. They are entitled to read, study, understand, run, modify, reuse, and transform all code so licensed, to the extent the particular license on the work, and any applicable exceptions from its restrictions, dictate. In the exercise of these rights, they are entitled to possess and make use of the complete and corresponding source code of the work, which means each computer program in the form in which it is most convenient to be modified, along with the ancillary scripts, makefiles, etc. required to build modified versions of the program and any applicable installation information. Because such rights are useless unless users know of them, they must also receive actual notice of the terms covering the program.

2. Providers of services over networks—when they are using computer programs specifically designed to provide such services over networks and licensed under GNU AGPL—have the obligations described above to the users of those services.

3. Distributors of licensed works—whether they are distributing modified or unmodified versions of the works, whether they have embedded executable copies of licensed works in a device, or are selling or otherwise transferring only a digital copy—have obligations to at least the users to whom they or intermediary parties distributed those copies. Whether those obligations run also to third parties not directly receiving their distribution of the works depends on the precise license involved, and their chosen mode of either distributing or offering to distribute source code. In addition, they have obligations to upstream

parties, to preserve reasonable legal notices embedded in the code, and to mark modified versions appropriately.

4. Both service providers and distributors have the obligation, in order to protect users' rights, to refrain from imposing any additional restrictions on downstream parties. They must refrain from terms in "umbrella licenses," EULAs, or sublicenses that restrict downstream users' rights as described above. Under the terms of LGPL, they must also refrain from license terms on works based on the licensed work that prohibit replacement of the licensed components of the larger non-LGPL'd work, or prohibit decompilation or reverse engineering in order to enhance or fix bugs in the LGPL'd components.

5. Patent holders having claims reading on works they distribute have an obligation to refrain from enforcing those claims against parties to whom they distribute. Patent holders modifying and distributing works under the version 3 family of licenses have an obligation to refrain from enforcing any claims reading on the version they distributed, not only against that version as distributed, but also against any subsequent version or work based thereon that also practices those claims.

6. All parties have an obligation to refrain from acting as a provider of services or distributor of licensed works if they have accepted, or had imposed on them by judicial action, binding legal conditions that would prevent them from meeting obligations to users as described. If a party is under such conflicting obligations, it has a duty to refrain from playing the role in which it is no longer free to meet its license obligations.

How to Meet Compliance Obligations

In our experience of working with commercial parties building GPL compliance programs—as well as in our role as lawyers representing GPL licensors coping with the consequences of compliance failures—we have observed that there is a significant mismatch between the assumptions businesses make about compliance and the realities of what goes wrong, what causes disputes, and how those disputes are resolved. Often, we have found companies preparing at great expense to avoid unlikely risks that have low historical incidence of occurrence and low cost of remediation, while leaving unmanaged the risks that have historically resulted in all the litigation and other adverse outcomes. In this section, we describe in broad terms the activities that help businesses prepare to meet their compliance obligations with minimal effort at minimal cost, dealing preventively with the compliance risks they really face.

The mismatch between actual compliance risk and compliance risk management, in our experience, results from a misunderstanding of licensor intentions. Commercial parties often expect copyleft project communities to approach compliance as a form of copyright monetization, or else as an ideological effort to force proprietary software to be relicensed under copyleft terms. Under the assumption that the intention of the licensors is to take advantage of non-compliance to extract royalties, or to force the business's proprietary products to be distributed under copyleft, businesses manage the risk that they will "accidentally"—or as the result of unsupervised activity by individual programmers—copy infringing "snippets" of copylefted code into their own proprietary computer program. Risk management involves the purchase of expensive proprietary "code scanning" services that purport to detect such accidental inclusions. Effort is concentrated on how proprietary computer programs are made, to prevent "infection" by free software.

In fact, however, development communities that use copyleft regard compliance failures as an opportunity to improve compliance. Every compliance failure downstream from their project represents a loss of rights by their users. The project, as copyright holder, is the guardian of its users' rights. Their activity is designed to restore those rights, and to protect the project's contributors' intentions in the making of their software. Projects' goals in seeking compliance are more often frustrated by the way software is delivered to users than by the way combinations of proprietary and free software are made. In particular,

- Users aren't provided with required information about the presence of copylefted programs and their applicable license terms in the product they have purchased; or

- Users can't reliably get complete and corresponding source code to copylefted programs the distributor knew it was using and intended to use pursuant to the license terms; or

- Users get no response when they communicate with published addresses requesting fulfillment of businesses' obligations.

In these and similar situations, the project's goal is compliance with obligations intentionally incurred by intentional use of copylefted programs, through observance of fulfillment obligations to downstream users. Failures of this type, which are uncaught by scanning programs or other similar services, have resulted in all the litigation ever brought by copyleft communities around the world.

Inclusions of free software in commercial proprietary products do happen. In our practice on behalf of copyleft-using development communities, we encounter such problems not frequently, but regularly. To the best of our knowledge, not one such instance has ever resulted in compliance litigation by a community party. These issues are regularly settled in an amicable and cooperative fashion.

The Key to Compliance is Governance

"Software governance" is the phrase we use to describe the processes by which businesses document and control what software they take in, what software they distribute, and what license terms they incur or offer on those inbound and outbound transactions. Whether the business is selling physical products with software embedded, or software products and services, good software governance is the key to minimum-cost preparation to meet compliance obligations.

Let's take a hypothetical manufacturer of consumer electronic devices, the Acme Company. Acme embeds Android—containing a copylefted OS kernel, Linux, and some custom applications containing LGPL'd libraries, along with some GPL'd utilities—in some devices. It embeds custom software stacks—also containing a copylefted kernel, modified Linux, Busybox under GPLv2, and a suite of GNU utilities under GPLv3—in others. It provides firmware upgrades, as digital files, to intermediaries who resell some of their devices. Acme chooses not to provide CDs or USB memory sticks containing GPL'd source code along with its products, preferring instead to place a written offer for relevant source code in product packaging or documentation. As a result, Acme has obligations to provide complete and corresponding source code for all copylefted programs in its products to any requesting party.

In order to meet these obligations, Acme's governance process begins from the "bill of materials" listing the component parts of the software stack in each product it makes. "License analysis," a process for which good FOSS tools exist, is used to confirm the licenses applicable to all the items on the bill of materials before development begins, and in an iterated process as development continues and the bill of materials is modified. A set of license texts required to be reproduced in product documentation is built from the output of the license analysis step, and is included in the product packaging and online documentation. For any copylefted software Acme takes in—whether embedded in hardware components or provided by third-party software suppliers—it automatically requests complete and corresponding source code. By exercising its own rights as a user, Acme prepares to meet its own obligations, and improves the compliance responsiveness of entities in its supply chain.

All source code taken in by Acme under copyleft licenses is carried, modified or unmodified, through the build process. At every stage, from the first trial build to the final build for release to manufacturing, the complete and corresponding source code for the copylefted binaries is itself part of the build target. When the product containing this software stack is released, the complete and corresponding source for all copylefted components is placed on a public-facing website, indexed by the product model number, a revision code, or an arbitrary hash code specific to the device and software version. This "stackmark" is also printed in the product documentation, and is represented by a QR-code on the exterior product packaging. A similar stack mark accompanies firmware upgrades delivered as digital files to sales intermediaries for placement on their own websites.

Legal review before product release verifies that Acme's forbearance obligations are also met. Any EULA, or umbrella license Acme wraps around all the software in its products, and any services agreement it makes for "software as a service" deployments in connection with its products, is checked to assure that it explicitly defers to the terms of any FOSS licenses in the product software stack, thus assuring that no additional restrictions are imposed in violation of copyleft.

As a result of these steps in its software governance process, Acme is confident that any user of its products who wishes to exercise her or his rights under the GNU family of licenses can request and receive source code instantly, even without opening the box. Acme also can be sure that each purchaser of its products has been provided a copy of the relevant copyleft licenses, and that no licensing of its own proprietary products interferes with or vitiates rights users have concerning copylefted software in their products, with minimum friction and at minimum cost.

These steps do not guarantee that Acme has avoided any accidental inclusion of copylefted code in a proprietary program it distributes as part of the product stack. But Acme has assured itself against the forms of compliance failure that have caused 100% of all the compliance litigation ever brought by community parties.

Principles of Prepared Compliance

Of course, the real world is not as simple as our schematic description of the Acme Company. Contemporary products, from cars to home appliances to networking or entertainment devices, may each contain dozens of computers, each containing in turn a full stack of embedded software, mixing copyleft FOSS, other FOSS, and proprietary programs. But a few basic principles remain valid across the entire complex domain:

1. Measure your compliance from the position of the user downstream from you trying to exercise rights conveyed by the licenses. Has the user received notice of the copylefted software intentionally included in your product? Is complete, corresponding source code and applicable installation information available to the user easily, preferably by automated means? Tools that measure what you deliver are more valuable than tools that only measure what you build.

2. Always exercise your own right to request complete and corresponding source code for all copylefted works from all your providers of software and of components embedding software, preferably in an automated process directly feeding your overall software governance system. Where possible, reject as non-conforming components provided to you containing copylefted software for which complete and corresponding source code is not furnished in response to your request or which is not accompanied by a "stackmark" for automated provisioning of source code. If you rely on an upstream provider for your software you cannot ignore your GPL compliance requirements simply because someone else packaged the software that you distribute.

3. Concentrate on the copylefted software you know you are using. Historically, the risk from a copylefted code snippet that some programmer dropped in your proprietary product careless of the consequences

is a problem far more infrequent and less difficult to resolve. Efficient management of the risks of higher concern lies in making sure you can provide, for example, precisely corresponding source code and makefiles for a copy of the Coreboot bootloader, Linux kernel, Busybox, or GNU tar that you included in a product you shipped two years ago.

4. Don't rely blindly on code scanners as they work too late in the process to improve your governance and too early in the process to catch problems in your delivery and post-sale provisioning. They do less important parts of the job expensively, and more important parts of the job not at all. Use them, where they are cost-effective, as a supplement to your own governance and verification processes, not as a primary tool of risk management.

Handling Compliance Inquiries

Between us, the authors have spent almost thirty person-years enforcing the GPL. We have, individually or collectively through SFLC, participated in every community enforcement of the GNU copyleft licenses ever brought to court in the United States. We have helped to settle dozens of compliance disputes for every one that has ever reached the point of litigation.

In this context, too, we have seen the consequences of mutual misunderstanding. Community parties bring forward complaints of non-compliance in order to achieve compliance. Commercial parties often expect compliance disputes to result in monetary demands or efforts to interfere with trade secret treatment of proprietary software, and respond defensively in consequence. Community parties, accustomed to the software engineering practices of the FOSS world, sometimes assume that commercial parties who cannot swiftly produce complete and corresponding source code for copylefted programs they intentionally included in their products are engaged in deliberate obfuscation.

In our experience, skilled facilitation of communication between parties at the early stages of the process can prevent these misunderstandings from escalating. A few guidelines about what to expect, accompanied by some historical examples, may help:

1. *Return the call with the right person.* The single most important rule of successful handling of compliance complaints is to maintain commu-

nication. Routing FOSS compliance issues to a particular individual who understands the internal software governance mechanisms, and can serve as the key public contact with the community when compliance concerns arise, may be the most effective way to resolve compliance matters. No community party has ever brought a compliance enforcement lawsuit against a party who responded cooperatively to its initial communications.

In one instance, a major multinational consumer electronics manufacturer which had repeatedly failed to respond to requests for fulfillment of source code obligations over many months was removed from a multi-defendant compliance enforcement lawsuit hours before the complaint was filed, as a result of mere verbal assurances of swift cooperation made personally by the corporation's general counsel in a telephone conversation with one of the present authors, who was acting on behalf of the complaining copyright holder.

2. *Assume preparation on the complainant's side.* The organizations traditionally bringing complaints of copyleft non-compliance (in historical order, the Free Software Foundation, GPL-violations.org, the Software Freedom Law Center, and the Software Freedom Conservancy) all fully investigate and verify complaints referred to them before making contact with apparently non-complying parties. Complainants will be prepared to substantiate the facts on which their complaint is based.

   In an unintended inclusion case arising some years ago, a global manufacturer used an entire copylefted library to provide essential features in one of its flagship proprietary software products. When we contacted them on behalf of the copyright holder, the corporation's legal counsel for FOSS matters repeatedly denied that such an event could have occurred, or that the code which our engineers could clearly see in their product was present there. We had to insist, three times, on their rechecking with their own engineers before they agreed that, indeed, such a mistake had occurred. Once our view of the facts had been verified, the matter was swiftly settled, without litigation and without payment of monetary damages.

3. *Let engineers be a part of the process.* The most time-consuming and difficult part of resolving most compliance matters, in our experience, is verifying that source code is indeed complete and corresponding. Without direct contact between software engineers on both sides, the resolution of the technical issues involved in demonstrating that the

binary distributed was built from the source provided is likely to be tortuous, expensive, and potentially tense. Counsel are understandably reluctant to expose their client's employees to direct inquiry from potentially hostile parties. But facilitated exchanges of information among software engineers communicating on technical subjects shortens the time to resolution, substantially reduces the cost of reaching resolution, and prevents unnecessary escalation due to mutual misunderstanding.

4. *Offer compliance.* Under the GPLv3 family of licenses, license termination is automatically reversed by prompt action to cure noncompliance, except in cases of repeated infringement. Although license termination is automatic under the terms of GPLv2, most of the GPLv2 licensors whose programs are widely used in commerce have a policy of immediately restoring distribution rights for non-compliant parties who come promptly into compliance when notified. An offer of compliance, subject to verification and mutual agreement on the details, is likely under all but the most exceptional circumstances to form the basis of an amicable settlement.

   In cases of mistaken inclusion of copylefted code in proprietary programs distributed as software products, prompt offer to remove the copylefted code and to redistribute a compliant version to customers and other reasonably reachable recipients has historically been the linchpin of arrangements to avoid litigation by amicable settlement.

5. *Use compliance discussions to improve relationships.* Development communities make software to benefit users, which includes you. When you use copylefted community software in your products, you are an important and valuable part of the commons, from the developers' point of view. Resolving a compliance matter is an occasion to strengthen your relationship to the commons, by increasing communication between your engineers and the project whose output you use for business benefit.

   In our experience, companies have often formed beneficial consulting or employment relationships with project developers they first encountered through compliance inquiries. In some cases, working together to alter the mode of use of the project's code in the company's products was an explicit element in dispute resolution. More often, the communication channels opened in the course of the inquiry served other and more fruitful purposes later.

6. *When the dust settles, offer to cover costs.* Most of the work investigating compliance complaints and verifying remediation is performed by non-profit entities that represent development communities without revenue. The ethic of these community parties against rent-seeking and monetization of others' mistakes means that the substantial engineering efforts they put into preparing compliance inquiries and resolving disputes are often paid for from their own scant means.[9] When a compliance issue has been successfully resolved, an offer to cover the cost of the engineering time involved on the community's side will do much to ensure the continuance of good relations in future. Requests for the repayment of those costs are entirely reasonable and foreseeable; a prompt offer precludes the escalation of dispute, an outcome we have seen more than once in our practice.

[9] SFLC provides free legal assistance to non-profit parties who make and distribute FOSS. Our efforts, including our work on compliance matters, are funded not from proceeds of settlement, but from the donations of companies and individuals who recognize the value of those services. SFLC has occasionally participated in monetary settlements made with non-compliant parties on behalf of projects we represent. Such settlements represent less than 2% of SFLC's historical revenues, and have never amounted to as much as 10% of our income in any fiscal year. We have not recouped in any settlement the cost of our efforts in that particular case, let alone the cost of our efforts on behalf of our clients overall.

Appendix 1: Offer of Source Code

From *A Practical Guide to GPL Compliance*[10]

Many distributors prefer to ship only an offer for source with the binary distribution, rather than the complete source package. This option has value when the cost of source distribution is a true per-unit cost. For example, this option might be a good choice for embedded products with permanent storage too small to fit the source, and which are not otherwise shipped with a CD but *are* shipped with a manual or other printed material.

However, this option increases the duration of your obligations dramatically. An offer for source must be good for three full years from your last binary distribution (under GPLv2), or your last binary or spare part distribution (under GPLv3). Your source code request and provisioning system must be designed to last much longer than your product life cycle.

In addition, if you are required to comply with the terms of GPLv2, you **cannot** use a network service to provide the source code. For GPLv2, the source code offer is fulfilled only with physical media. This usually means that you must continue to produce an up-to-date "source code CD" for years after the product's end-of-life.

Under GPLv2, it is acceptable and advisable for your offer for source code to include an Internet link for downloadable source *in addition* to offering source on a physical medium. This practice enables those with fast network connections to get the source more quickly, and typically decreases the number of physical media fulfillment requests. (GPLv3 §6(b) permits provision of source with a public network-accessible distribution only and no physical media. We discuss this in detail at the end of this section.)

The following is a suggested compliant offer for source under GPLv2 (and is also acceptable for GPLv3) that you would include in your printed materials accompanying each binary distribution:

> The software included in this product contains copyrighted software that is licensed under the GPL. A copy of that license is included in this document on page X. You may obtain the complete Corresponding Source code from us for a period of three years after our last shipment of this product, which will be no earlier than 2011-08-01, by sending a money order or check for

[10] `http://softwarefreedom.org/resources/2008/compliance-guide.html`.

> $5 to:
> GPL Compliance Division
> Our Company
> Any Town, US 99999
>
> Please write "source for product *Y*" in the memo line of your payment.
>
> You may also find a copy of the source at `http://www.example.com/sources/Y/`.
>
> This offer is valid to anyone in receipt of this information.

There are a few important details about this offer. First, it requires a copying fee. GPLv2 permits "a charge no more than your cost of physically performing source distribution". This fee must be reasonable. If your cost of copying and mailing a CD is more than around $10, you should perhaps find a cheaper CD stock and shipment method. It is simply not in your interest to try to overcharge the community. Abuse of this provision in order to make a for-profit enterprise of source code provision will likely trigger enforcement action.

Second, note that the last line makes the offer valid to anyone who requests the source. This is because v2 §3(b) requires that offers be "to give any third party" a copy of the Corresponding Source. GPLv3 has a similar requirement, stating that an offer must be valid for "anyone who possesses the object code". These requirements indicated in v2 §3(c) and v3 §6(c) are so that non-commercial redistributors may pass these offers along with their distributions. Therefore, the offers must be valid not only to your customers, but also to anyone who received a copy of the binaries from them. Many distributors overlook this requirement and assume that they are only required to fulfill a request from their direct customers.

The option to provide an offer for source rather than direct source distribution is a special benefit to companies equipped to handle a fulfillment process. GPLv2 §3(c) and GPLv3 §6(c) avoid burdening noncommercial, occasional redistributors with fulfillment request obligations by allowing them to pass along the offer for source as they received it.

Note that commercial redistributors cannot avail themselves of the option (c) exception, and so while your offer for source must be good to anyone who receives the offer (under v2) or the object code (under v3), it *cannot* extinguish the obligations of anyone who commercially redistributes your product. The license terms apply to anyone who distributes GPL'd software, regardless of whether they are the original distributor. Take the example of Vendor V, who develops a software platform from GPL'd sources for use in embedded devices. Manufacturer M contracts with V to install the software as firmware in M's device. V provides the software to M, along with a compliant offer for source. In this situation, M cannot simply pass V's offer for source along to its customers. M also distributes the GPL'd software commercially, so M too must comply with the GPL and provide source (or M's *own* offer for source) to M's customers.

This situation illustrates that the offer for source is often a poor choice for products that your customers will likely redistribute. If you include the source itself with the products, then your distribution to your customers is compliant, and their (unmodified) distribution to their customers is likewise compliant, because both include source. If you include only an offer for source, your distribution is compliant but your customer's distribution does not "inherit" that compliance, because they have not made their own offer to accompany their distribution.

The terms related to the offer for source are quite different if you distribute under GPLv3. Under v3, you may make source available only over a network server, as long as it is available to the general public and remains active for three years from the last distribution of your product or related spare part. Accordingly, you may satisfy your fulfillment obligations via Internet-only distribution. This makes the "offer for source" option less troublesome for v3-only distributions, easing compliance for commercial redistributors. However, before you switch to a purely Internet-based fulfillment process, you must first confirm that you can actually distribute *all* of the software under GPLv3. Some programs are indeed licensed under "GPLv2, *or any later version*" (often abbreviated "GPLv2-or-later"). Such licensing gives you the option to redistribute under GPLv3. However, a few popular programs are only licensed under GPLv2 and not "or any later version" ("GPLv2-only"). You cannot provide only Internet-based source request fulfillment for the latter programs.

If you determine that all GPL'd works in your whole product allow upgrade to GPLv3 (or were already GPLv3'd to start), your offer for source may be as simple as this:

> The software included in this product contains copyrighted software that is licensed under the GPLv3. A copy of that license is included in this document on page X. You may obtain the com-

> plete Corresponding Source code from us for a period of three years after our last shipment of this product and/or spare parts therefor, which will be no earlier than 2011-08-01, on our website at `http://www.example.com/sources/productnum/`.

Under both GPLv2 and GPLv3, source offers must be accompanied by a copy of the license itself, either electronically or in print, with every distribution.

References

1. Free Software Foundation, FSF FAQs about the GNU Licenses. http://www.gnu.org/licenses/gpl-faq.html, October 2014. (Accessed Oct 23, 2014).
2. Free Software Foundation, How to use GNU licenses for your own software. http://www.gnu.org/licenses/gpl-howto.html, April 2014. (Accessed Oct 23, 2014).
3. Free Software Foundation, Various Licenses and Comments about Them. http://www.gnu.org/licenses/license-list.html, October 2014. (Accessed Oct 23, 2014).
4. Bradley M. Kuhn, Aaron Williamson, Karen M. Sandler, A Practical Guide to GPL Compliance. http://softwarefreedom.org/resources/2008/compliance-guide.html, August 2008. (Accessed Oct 23, 2014).
5. Software Freedom Law Center, Managing Copyright Information Within a Free Software Project. http://softwarefreedom.org/resources/2012/ManagingCopyrightInformation.html, September 2012. (Accessed Oct 23, 2014).
6. A History of the GPLv3 Revision Process, Eben Moglen. https://www.softwarefreedom.org/resources/2013/A_History_of_the_GPLv3_Revision_Process.pdf, July 2013. (Accessed Oct 23, 2014).
7. Free Software Foundation, GPLv3 Discussion Draft 1. http://gplv3.fsf.org/comments/gplv3-draft-1, January 2006. (Accessed Oct 23, 2014).
8. Free Software Foundation, GPLv3 Discussion Draft 2. http://gplv3.fsf.org/comments/gplv3-draft-2, July 2006. (Accessed Oct 23, 2014).
9. Free Software Foundation, GPLv3 Discussion Draft 3. http://gplv3.fsf.org/comments/gplv3-draft-3, March 2007. (Accessed Oct 23, 2014).
10. Free Software Foundation, GPLv3 Discussion Draft 4. http://gplv3.fsf.org/comments/gplv3-draft-4, May 2007. (Accessed Oct 23, 2014).