Practical Advice to Scale Open Source Legal Support

Ibrahim Haddad, Ph.D.

June 2013

www.linuxfoundation.org

The paper offers practical advice to allow an organization to scale its open source legal support. Please note that the author is not a legal counsel and the paper does not offer legal advice.

Introduction

In its simplest definition, FOSS compliance means that users of FOSS must observe all the copyright notices and satisfy all the license obligations for the FOSS they use in commercial products. The complexity of achieving FOSS compliance increases slightly because you also want to protect your intellectual property and that of any third party suppliers (whose source code included in your product) from unintended disclosure.

Still FOSS compliance is more of an operational challenge related to execution and scaling than a legal challenge. Achieving compliance requires the aggregation of policies and processes, training, tools and proper staffing that enables an organization to effectively use FOSS and contribute to open source projects and communities while respecting copyrights of their respecting holders, complying with license obligations, and protecting the organization's intellectual property and that of its customers and suppliers.

Most companies create a FOSS compliance program and set up a core team, usually called the Open Source Review Board (OSRB) to ensure proper compliance. The OSRB often consists of representatives from engineering, product teams and legal in addition to the Compliance Officer (sometimes called Director of Open Source). In addition to the core team, an extended team that consists of various individuals across multiple departments (Documentation, Supply Chain, Corporate Development, IT, Localization, etc.) also contributes on an on-going basis to the compliance efforts.

In this article, we look closely at the role of the Legal Counsel in ensuring FOSS compliance and examine practical advice that a Legal Counsel can provide to the software development team. Such practical advice will enable software developers to make daily decisions related to open source licenses without having to go back to the Legal Counsel for every single question.

Legal Counsels and Open Source Compliance

The Legal Counsel is a core member of the Open Source Review Board, the committee that ensures compliance with FOSS licenses.

The Legal Counsel focuses on four essential duties:

1. **Provide approval around the use of FOSS in products:**
   The approval of the Legal Counsel is required when using FOSS in a commercial product. Typically, the Legal Counsel reviews the compliance ticket, the source code scan report, and the license information provided with the source code package. They then evaluate any risk factors based on the feedback provided by engineering and the open source compliance officer. As part of this exercise, the Legal Counsel decides on the incoming and outgoing licenses of the software component in question.

2. **Advise on FOSS licensing:**
   a. Offer guidance about FOSS license obligations that must be fulfilled.
   b. Advise on licensing conflicts in relation to incompatible or conflicting licenses..
   c. Advise on IP issues associated with the use of FOSS – this is especially the case when the company is about to release proprietary source code under FOSS license.
   d. Provide recommendations and guidance to engineering teams on FOSS questions and concerns.

3. **Review and approve updates to end-user documentation:**
   This form of legal support is related to ensuring that appropriate FOSS notices are provided to consumers in relation to any FOSS included in the product along with a written offer on how to obtain the source code when applicable.

4. **Contribute to establishing and running the FOSS compliance program:**
   a. Establish and maintain the FOSS policy and process.
   b. Handle compliance inquiries sent to the company in relation to FOSS compliance.
   c. Provide training around FOSS licenses, company policies and guidelines.

Practical Advice

The practical advice from the Legal Counsel to the software developers consist of:

1. **License Playbooks:** An easy to read and digest summary of FOSS licenses intended for software developers.
2. **License compatibility matrix:** An easy method to learn if License-A is compatible with License-B. Such a matrix can be used by software developers as they merge code incoming from different projects under different license into a single body of code.
3. **License classification:** An easy way to understand the different and the course of action needed when using source code provided under these licenses.
4. **Software interaction methods:** A guide to understand how software components available under different licenses interact, and if the method of interaction is allowed per company compliance policies.
5. **Checklists:** An easy way to remember what needs to be done at every gate in the development and compliance processes.

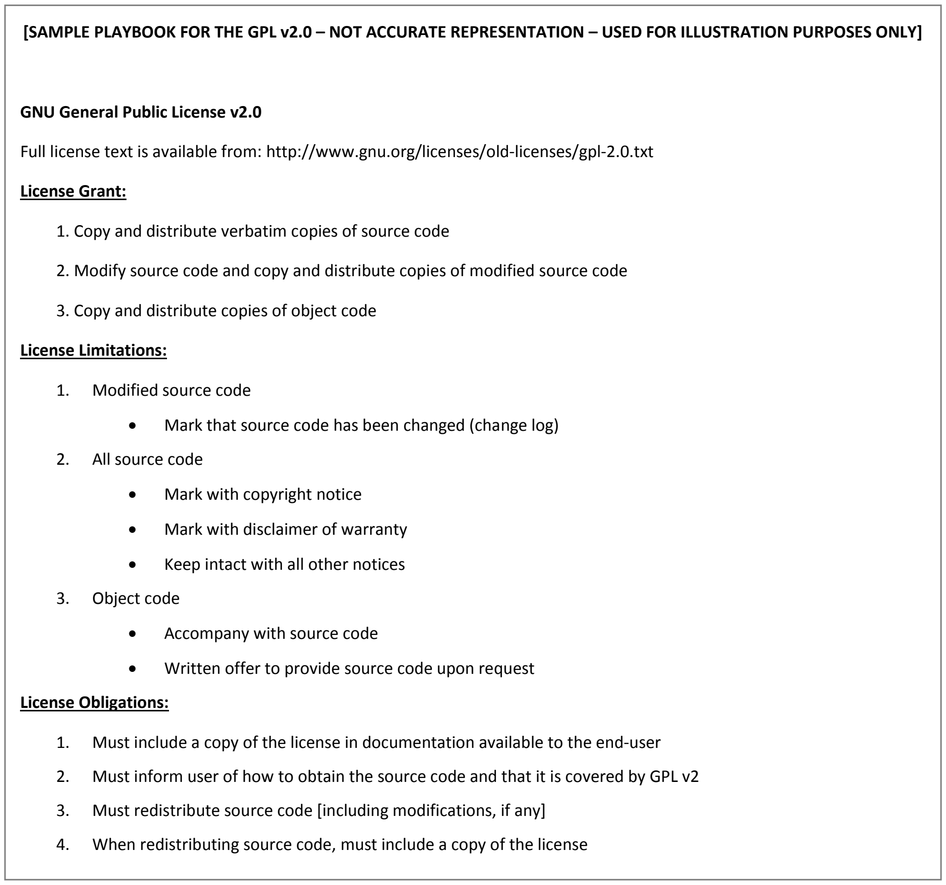
In the following sections, we examine these five advice areas, provide examples and discuss how such practical advice helps software developers working with FOSS.

License Playbooks

License Playbooks are summaries of most used or most popular FOSS licenses. They provide easy to understand information about these licenses such as license grants, restrictions, obligations, patent impact and more.

License playbooks minimize the number of basic questions sent to Legal Counsels and provide developers with immediate legal information about these most used licenses.

Figure 1 provides an example license playbook for the GPL v2. Please note that this playbook is provided for illustration purposes only and its content should not be considered accurate.



[SAMPLE PLAYBOOK FOR THE GPL v2.0 – NOT ACCURATE REPRESENTATION – USED FOR ILLUSTRATION PURPOSES ONLY]

GNU General Public License v2.0

Full license text is available from: http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt

License Grant:

1. Copy and distribute verbatim copies of source code
2. Modify source code and copy and distribute copies of modified source code
3. Copy and distribute copies of object code

License Limitations:

1. Modified source code
   - Mark that source code has been changed (change log)
2. All source code
   - Mark with copyright notice
   - Mark with disclaimer of warranty
   - Keep intact with all other notices
3. Object code
   - Accompany with source code
   - Written offer to provide source code upon request

License Obligations:

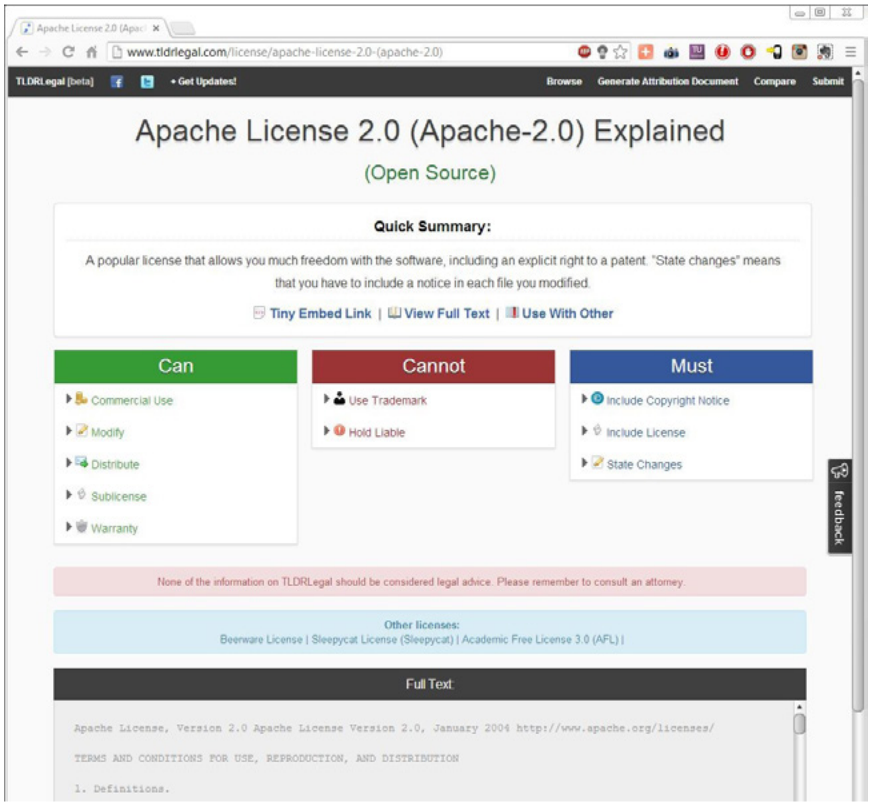
1. Must include a copy of the license in documentation available to the end-user
2. Must inform user of how to obtain the source code and that it is covered by GPL v2
3. Must redistribute source code [including modifications, if any]
4. When redistributing source code, must include a copy of the license

Figure 1: Example license playbook for GPL v2 (for illustration purposes only)

Practical Guide to the GPL:

On August 26, 2008, the Software Freedom Law Center published a guide on how to be compliant with the GNU General Public License (GPL) and related licenses. The guide entitled "Practical Guide to GPL" focuses on avoiding compliance actions and minimizing the negative impact when enforcement actions occur. It is an excellent reference to understand the GPL license obligations, and it is available from:

http://www.softwarefreedom.org/resources/2008/compliance-guide.html



Apache License 2.0 (Apac ×

www.tldrlegal.com/license/apache-license-2.0-(apache-2.0)

TLDRLegal [beta] + Get Updates! Browse Generate Attribution Document Compare Submit

Apache License 2.0 (Apache-2.0) Explained

(Open Source)

Quick Summary:

A popular license that allows you much freedom with the software, including an explicit right to a patent. "State changes" means that you have to include a notice in each file you modified.

Tiny Embed Link | View Full Text | Use With Other

Can	Cannot	Must
Commercial Use	Use Trademark	Include Copyright Notice
Modify	Hold Liable	Include License
Distribute		State Changes
Sublicense		
Warranty		

feedback

None of the information on TLDRLegal should be considered legal advice. Please remember to consult an attorney.

Other licenses:
Beerware License | Sleepycat License (Sleepycat) | Academic Free License 3.0 (AFL) |

Full Text

```
Apache License, Version 2.0 Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.
```

Figure 2: Example license playbook from http://www.tldrlegal.com/ that shows what the license allows and restricts. This example is provided for illustration purposes only. (This is not an endorsement of the web site or its content.)

License Compatibility Matrix

License compatibility is a term that refers to the situation of determining if a certain license has compatible terms with another license. GPL compatibility refers to the situation of determining if a certain license has compatible terms with the GPL. The license compatibility problem is often encountered when combining source code originating from different software components licensed under incompatible licenses making it impossible to combine the source code to create a new software component.

An example of license compatibility is the X11 license, which is compatible with the GPL version 2, because works licensed under the X11 license, can be distributed under the terms of the GPL.

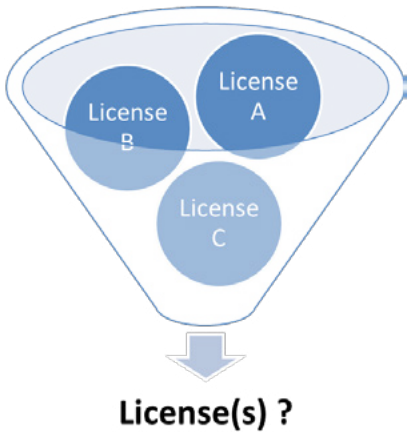


Figure 3: Combining source coming under different licenses into a single body of code to be licensed under a single license assuming the differences are compatible.

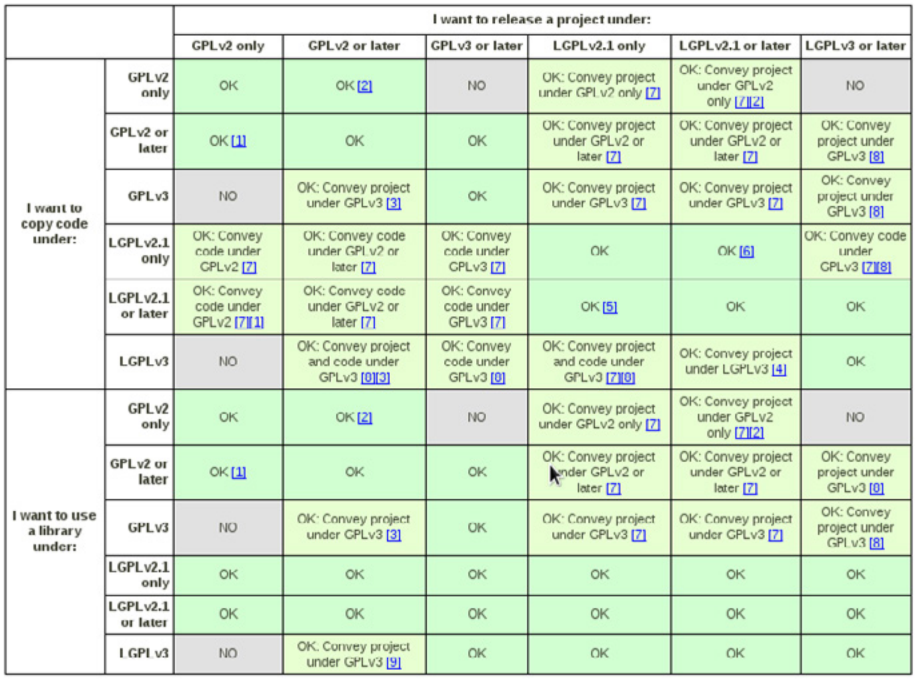
Many of the FOSS licenses, such as the BSD license and the LGPL, are GPL-compatible, meaning that their source code can be combined with a source code that is licensed under the GPL without conflict; the new program resulting from the combination would have to be licensed under the GPL applied. Other FOSS and proprietary software licenses are not GPL-compatible since they have conflicting terms and conditions.

This is one area where development teams need guidance from Legal Counsels that can provide a License Compatibility Matrix that would look as follows (Table 1):

Is Compatible With:	License-A	License-B	License-C	License-D	License-E	License-F	License-G
License-A	X				X	X	
License-B		X					
License-C			X				
License-D		X		X			X
License-E					X		
License-F			X			X	
License-G	X						X

Table 1: Example license compatibility matrix (for illustration purposes only)

When the development team is combining code incoming under different licenses, they can refer to this matrix to verify if there is a licensing conflict joining the source code in a single software component. If a license is used and it is not covered in the matrix, it can be added by the Legal Counsel advising on open source licensing.



		I want to release a project under:					
		GPLv2 only	GPLv2 or later	GPLv3 or later	LGPLv2.1 only	LGPLv2.1 or later	LGPLv3 or later
I want to copy code under:	GPLv2 only	OK	OK [2]	NO	OK: Convey project under GPLv2 only [7]	OK: Convey project under GPLv2 only [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK: Convey project under GPLv2 or later [7]	OK: Convey project under GPLv2 or later [7]	OK: Convey project under GPLv3 [8]
	GPLv3	NO	OK: Convey project under GPLv3 [3]	OK	OK: Convey project under GPLv3 [7]	OK: Convey project under GPLv3 [7]	OK: Convey project under GPLv3 [8]
	LGPLv2.1 only	OK: Convey code under GPLv2 [7]	OK: Convey code under GPLv2 or later [7]	OK: Convey code under GPLv3 [7]	OK	OK [6]	OK: Convey code under GPLv3 [7][8]
	LGPLv2.1 or later	OK: Convey code under GPLv2 [7][1]	OK: Convey code under GPLv2 or later [7]	OK: Convey code under GPLv3 [7]	OK [5]	OK	OK
	LGPLv3	NO	OK: Convey project and code under GPLv3 [8][3]	OK: Convey code under GPLv3 [8]	OK: Convey project and code under GPLv3 [7][8]	OK: Convey project under LGPLv3 [4]	OK
I want to use a library under:	GPLv2 only	OK	OK [2]	NO	OK: Convey project under GPLv2 only [7]	OK: Convey project under GPLv2 only [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK: Convey project under GPLv2 or later [7]	OK: Convey project under GPLv2 or later [7]	OK: Convey project under GPLv3 [8]
	GPLv3	NO	OK: Convey project under GPLv3 [3]	OK	OK: Convey project under GPLv3 [7]	OK: Convey project under GPLv3 [7]	OK: Convey project under GPLv3 [8]
	LGPLv2.1 only	OK	OK	OK	OK	OK	OK
	LGPLv2.1 or later	OK	OK	OK	OK	OK	OK
	LGPLv3	NO	OK: Convey project under GPLv3 [9]	OK	OK	OK	OK

Figure 4: Example of license compatibility matrix for the GPL/LGPL license (Source: http://www.gnu.org)

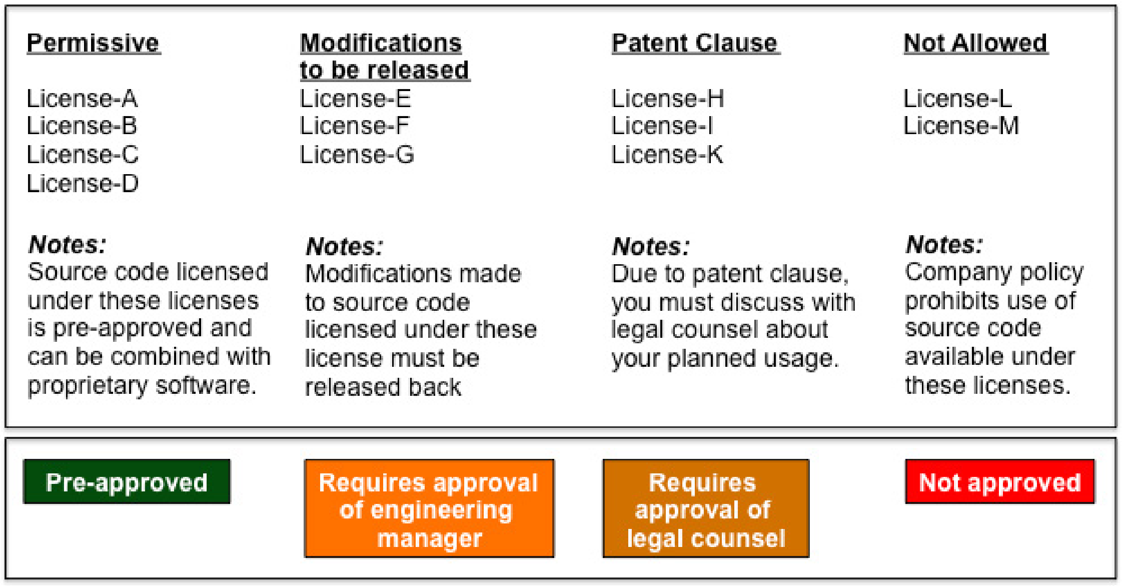
GPL and LGPL Compatibility Guide:

For information about GPL and LGPL compatibility, the Free Software Foundation maintains reference information online on the topic, available from:

http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility

License Classification

In an effort to reduce the number of questions for Legal Counsels and to increase license and compliance process education, some companies opt to classify the most used licenses in their products under few categories. Figure 5 presents an example license classification, where most used licenses are divided into four categories.



Permissive	Modifications to be released	Patent Clause	Not Allowed
License-A License-B License-C License-D	License-E License-F License-G	License-H License-I License-K	License-L License-M
Notes: Source code licensed under these licenses is pre-approved and can be combined with proprietary software.	***Notes:*** Modifications made to source code licensed under these license must be released back	***Notes:*** Due to patent clause, you must discuss with legal counsel about your planned usage.	***Notes:*** Company policy prohibits use of source code available under these licenses.
Pre-approved	**Requires approval of engineering manager**	**Requires approval of legal counsel**	**Not approved**

Figure 5: Example license categories (for illustration purposes only)

Pre-approved Licenses

FOSS permissive licenses fall under this category. Source code available under these licenses is pre-approved for use by developers without having to go through the approval process with their manager and/or Legal counsel. Such pre-approvals are usually conditional to capturing the notices and making sure they are sent to the documentation team.

Licenses Requiring Manager's Approval

For source code licensed under these licenses, manager's approval is required since you have the obligation to release any source code modifications, in addition to notices fulfillment (publishing license text, attribution notice, copyright notice, etc.).

Licenses Requiring Legal Counsel Approval

Source code available under these licenses requires legal review and approval. This usually applies to licenses that have a patent clause.

Prohibited Licenses

Some companies flag certain licenses as "not allowed" and they communicate that as part of training, all hands meetings, or other form of employee communication.

How can classifying licenses be helpful?

The above license categories are an example of a way to classify licenses and make it easier for software developers to know the course of action to take when they decide to integrate incoming source into the build system depending on its incoming license. Furthermore, it makes it easy to create an associate between a license and what need to be done. An example of that would be a software developer that downloads source code licensed under:

- License A - Action: I can use it with no problem
- License E - Action: I need to get my manager's approval
- License I - Action: I need to consult with the Legal Counsel
- License M - Action: I can't use this source code
- Other - Action: Ask my manager on course of action

Please note that these different scenarios are used for illustration purposes. You can setup a different classification model with different actions depending on your company policies and guidelines.

Software Interaction Methods

As part of the compliance process, there is usually a step in the process where the OSRB conducts an architecture review of the software component in question. The goal of the architecture review is to understand how that specific software component interacts with other software components and the method of interaction to identify:

- Components that are Open Source (used "as is" or modified)
- Components that are proprietary
- Components that are originating from third party software providers
- Components dependencies
- Communication protocols
- Linkage method Dynamic versus static linking
- Components that live in kernel space versus user space
- Use of shared header files
- Etc.

Figure 6 illustrates an example architecture interaction template that software developers can fill out to demonstrate to the Legal Counsel how the specific software component interacts in the software stack.

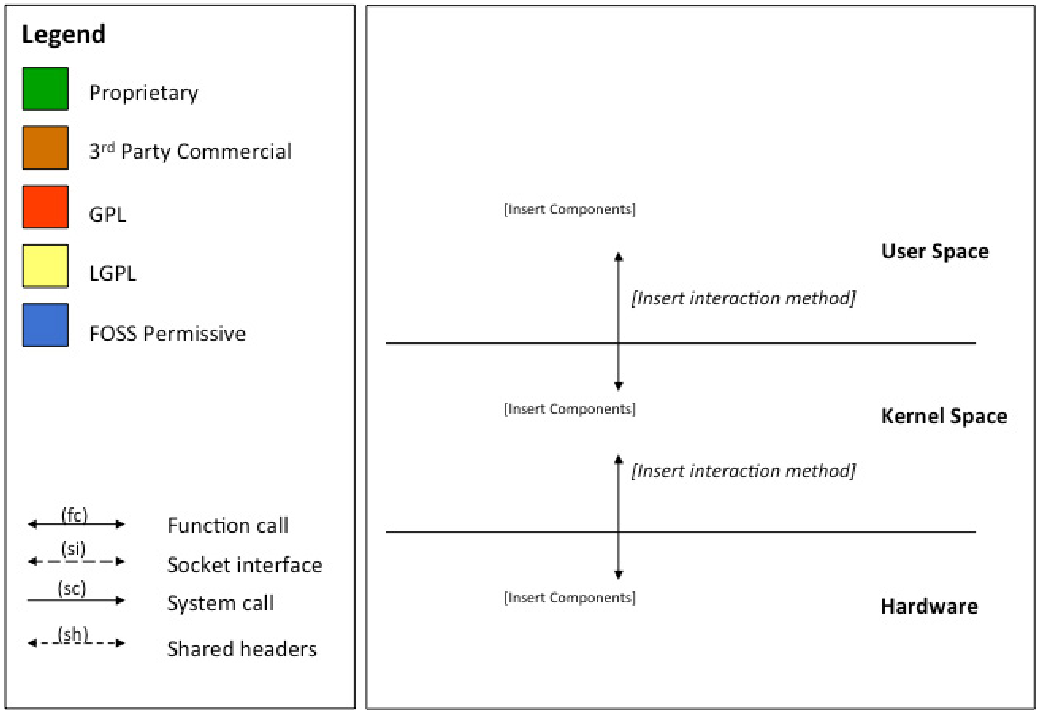


Figure 6: Example architecture interaction template (for illustration purposes only)

The result of the architecture review is an analysis of the licensing obligations that may extend from the FOSS to proprietary or 3rd party software components.

Table 2 and 3 provide additional information that Legal Counsels can provide to software developers. The tables are license matrices that illustrate what licenses can be used to dynamically and statically link while respecting company policies.

For example, looking at Table 2, source code licensed under License-B can dynamically link to source code license under License-D. However, source code licensed under License C cannot dynamically link to source code licensed under License-B.

Can Dynamically Link To	License-A	License-B	License-C	License-D
License-A	X	X	X	X
License-B		X		X
License-C	X		X	
License-D		X	[Requires Pre-Approval]	X

Table 2: Sample dynamic linkage matrix (for illustration purposes only)

Similarly, looking at Table 3, source code licensed under License-A can statically link to source code license under License-C. However, source code licensed under License A cannot statically link to source code licensed under License-B. Some linkage combination may be allowed on a case-by case basis, which is why we see in the table below (Table 3) the [Requires pre-approval].

Can Statically Link To	License-A	License-B	License-C	License-D
License-A	X		X	
License-B		X	[Requires Pre-Approval]	
License-C	X		X	
License-D	[Requires Pre-Approval]			X

Table 3: Sample static linkage matrix (for illustration purposes only)

In the event that the architecture review reveals any linkage issue (i.e. a static or dynamic linkage that does not follow company policy – as provided in the linkage matrices), then the person responsible for driving the architecture review (usually the compliance officer) would notify the software developer responsible for that software component and requests a correction.

Checklists

Most companies establish checklists that are used within the development and compliance processes at every major milestone. When it comes to open source compliance, several checklists can be developed and used before approving the integrating incoming open source code the product's source code repository.

An example of such checklists is the following checklist used before making source code available on the web site (license obligation fulfillment):

- All source code components have a corresponding compliance ticket
- All compliance tickets have been approved by engineering and legal
- All compliance tickets are clear from any sub-tasks attached to them
- Notices for all of the software components have been sent to Documentation team and included in product documentation (including written offer)
- Legal has approved the written offer notice and overall compliance documentation
- Source code packages have been prepared and tested to compile on a standard development machine
- Source code provided is complete and corresponds to the binaries in the product

Such checklists minimize the probability of error and ensure everyone involved in the compliance process is aware of what needs to be done before moving to the next step in the process.

Conclusion

Software developers need to be educated about the licenses of the various FOSS components they use. Having Legal Counsels provide such education in a very practical way is extremely helpful as it allows software developers to have access to documented practical advices that will help answer most of their daily legal related questions.

These practical advices usually revolve around:

- Inclusion of FOSS into proprietary (or 3rd party), or vice versa.
- Linking of FOSS into proprietary (or 3rd party) source code, or vice versa.
- Interaction methods between various software components (proprietary, 3rd party, FOSS).
- License obligations that must be met when using FOSS components.

FOSS compliance is easy to achieve once you have built up your compliance program, created a compliance policy and process, established staffing to ensure execution, and enabled your team with various tools to assist in the compliance automation aspect.

The Open Compliance Program at the Linux Foundation aims to help organization achieve compliance faster and cheaper by providing a number of resources that are accessible via http://compliance.linuxfoundation.org.

About the Author

Dr. Ibrahim Haddad is the Head of Open Source Group at Samsung Research America, the North American R&D Center of Samsung Electronics. Prior to joining Samsung, Dr. Haddad was a member of the management team at The Linux Foundation responsible for technical, legal and compliance projects and initiatives. At the Linux Foundation, Dr. Haddad worked with the largest technology companies and with the Open Source community to facilitate a vendor-neutral environment for advancing the Linux platform for next-generation computing devices.

Dr. Haddad's career started at Ericsson Research where he spent five years focusing on advanced research for system architecture of 3G wireless IP networks and on adopting Linux and Open Source software in carrier grade environments. He then joined Motorola as Director of Technology Portfolio managing the Open Source Technology Group and driving Motorola's Open Source initiatives. After Motorola, Dr. Haddad ran the Open Source Office at Palm and was responsible for all Open Source activities related to webOS, and later supported HP with open sourcing webOS to become the open webOS project.

Dr. Haddad graduated with Honors from Concordia University (Montréal, Canada) with a Ph.D. in Computer Science. He is a Contributing Editor to the Linux Journal, Co-Author of two books on Red Hat Linux and Fedora, and Technical Editor for four books on Linux System Administration, Fedora Linux and Ubuntu Linux. He is fluent in Arabic, English and French.

Linux Foundation Compliance Resources

- Open Compliance Program: http://www.linuxfoundation.org/programs/legal/compliance
- Professional and Comprehensive Compliance Training: The Linux Foundation offers hands-on training from compliance experts for individuals and companies responsible for achieving compliance with FOSS licenses and establishing a FOSS compliance program, as well as for those who simply want to learn more about compliance. Options available include live on-site training in addition to instructor-led live remote training. http://www.linuxfoundation.org/programs/legal/compliance/training-and-education
- Open Compliance Program Self-Assessment Checklist: An extensive checklist of practices found in industry-leading compliance programs. Companies can use this checklist as a confidential internal tool to assess their progress in implementing a rigorous compliance process and help them prioritize their process improvement efforts. http://www.linuxfoundation.org/programs/legal/compliance/self-assessment-checklist
- Compliance Publications: http://www.linuxfoundation.org/publications/compliance
- Open Compliance Directory and Rapid Alert System: http://www.linuxfoundation.org/programs/legal/compliance/directory
- Compliance Tools: http://www.linuxfoundation.org/programs/legal/compliance/tools
- The Software Package Data Exchange™: The SPDX™ specification is a standard format for communicating the components, licenses and copyrights associated with a software package. http://www.spdx.org/