


# SIPLA 2005 ANNUAL SYMPOSIUM

## Law and Technology:
## Current Developments in
## Free Software and Open Source

**DATE:** **March 26, 2005**
**TIME** **8:30 AM – 4:30 PM**
**LOCATION:** **Franklin Pierce Law Center**
**2 White Street**
**Concord, NH 03301**
**Rich Room**

## AGENDA

**OPENING REMARKS**
9:00 AM – 9:10 AM Dean John Hutson

**INTRODUCTION TO FREE SOFTWARE AND OPEN SOURCE**
9:10 AM – 9:20 AM Professor William Hennessey

**WHY FREE AND OPEN SOURCE SOFTWARE MATTERS AND WHAT COMMERCIAL ENTERPRISES MIGHT DO ABOUT IT……P. 3**
9:20 AM – 10:05 AM Scott K. Peterson

**PATENTS AND FREE SOFTWARE……P. 4**
10:10 AM - 10:55 AM Dan Ravicher

**FREE SOFTWARE AND BEYOND……...P. 6**
11:00 AM - 12:15 PM Richard Stallman

**LUNCH** 12:15 PM – 12:55 PM

**OPEN SOURCE: THE LEGAL FRONT……..P. 10**
1:00 PM - 1:45 PM Edmund J. Walsh

**OPEN SOURCE LICENSES: THE LATEST ISSUES……P. 20**
1:50 PM - 2:35 PM Peter M. Moldave

**WORKING WITH OPEN SOURCE……P. 50**
2:40 PM - 3:25 PM Karen Copenhaver

**RECEPTION** 3:30 PM - 4:30 PM

**Available CLE Credit: 5 Hrs**

## SPONSORS




http://www.students.piercelaw.edu/sipla/sipla.htm

# Why free and open source software matters and what commercial enterprises might do about it

*Scott Peterson, Hewlett-Packard Company*

ABSTRACT:
Technical and economic factors have fostered the widespread availability of software under licenses that do not include conventional restrictions. The role that theses licenses play can be understood by observing the process of aggregation of effort that is central to development of this software. Seen from the perspective of that role, these 'free' licenses fall into two major categories.

Software licensed in this way is too economically significant for commercial enterprises to ignore. From a legal perspective, this software can be viewed as third party software under unusually permissive license terms. However, for a commercial enterprise to move to a more complete awareness, in addition to coming to understand the obvious differences in the license terms, it is necessary to appreciate the differences in the typical patterns of software development and distribution. With those understandings, then the enterprise can decide how it will relate to that software world, and can appreciate how it should adapt its existing technical and legal processes.

BIO - Scott Peterson
Senior Counsel, Intellectual Property
Hewlett-Packard Company
Cambridge, MA
Mr. Peterson has practiced as an intellectual property attorney since 1983, focusing on information technologies. He joined HP in 1991, and has provided intellectual property support for a wide range of HP's businesses.

Beginning in 1992, Mr. Peterson has provided legal advice relating to the GNU GPL and other free and open source software licenses for a wide range of projects. He has been counsel to HP's Open Source Review Board since its formation in 2001.

In recent years, Mr. Peterson has been active in policy issues concerning the interplay between intellectual property and standards setting.

Mr. Peterson received bachelor's and master's degrees in electrical engineering from the Massachusetts Institute of Technology and a juris doctorate from the Franklin Pierce Law Center.

# Patents and Free Software

*Dan Ravicher, Free Software Foundation & Public Patent Foundation*

ABSTRACT:

This discussion will address the relationship between patents and free software, and particularly whether they are antithetical to one another or actually aligned in their purposes. The first part of the discussion will address the licensing of patents for use in free software, and particularly the implied patent license that arises in conjunction with the GNU General Public License. The second part will address why free software might actually have less to fear from patents than non-free software.

BIO - Dan Ravicher
Senior Counsel, Free Software Foundation
Executive Director, Public Patent Foundation
New York, NY
http://pubpat.org
Dan Ravicher is a registered patent attorney with experience litigating, licensing, prosecuting and counseling clients with respect to patents. Prior to founding PUBPAT, Dan was associated with the patent law practice groups of Skadden, Arps, Slate, Meagher & Flom LLP, Brobeck, Phleger & Harrison, LLP and Patterson, Belknap, Webb & Tyler, LLP in New York, and served the Honorable Randall R. Rader, Circuit Judge for the U.S. Court of Appeals for the Federal Circuit in Washington, D.C.

Dan has published several legal articles and given numerous presentations regarding patent law. He received his law degree from the University of Virginia School of Law, where he was the Class of 2000 Franklin O'Blechman Scholar, a Mortimer Caplin Public Service Award recipient and Editor of the Virginia Journal of Law and Technology and his bachelors degree in materials science magna cum laude with University Honors from the University of South Florida.

# FRANKLIN PIERCE LAW CENTER

## SIPLA ANNUAL SYMPOSIUM

## LAW AND TECHNOLOGY: CURRENT DEVELOPMENTS IN THE OPEN SOURCE AND FREE SOFTWARE MOVEMENTS

MARCH 26, 2005
Concord, NH

# Patents and Free Software

**by**

**Daniel B. Ravicher**
**Executive Director , Public Patent Foundation**
**Legal Director , Software Freedom Law Center**
**1375 Broadway, Suite 600**
**New York, New York 10018**
**dan@pubpat.org - 917-843-3425**

THESE MATERIALS

These materials addressing the relationship between patents and free software, and particularly whether they are antithetical to one another or actually aligned in their purposes, consist of two parts. The first part discusses the implied patent grant that arises in connection with free software licensing under the GNU General Public License ("GPL"). The second part of these materials addresses the relationship between patents and free software.

THE PUBLIC PATENT FOUNDATION

The Public Patent Foundation ("PUBPAT") is a not-for-profit legal services organization working to protect the public from the harms caused by the patent system. PUBPAT provides the general public, particularly those persons or businesses otherwise deprived of access to the system governing patents, with representation, advocacy, and education.

THE SOFTWARE FREEDOM LAW CENTER

Free and Open Source Software ("FOSS") is maturing at a rapid pace. The FOSS production ecosystem, once dominated by a few small not-for-profit entities and individual contributors, now includes a global array of individuals, not-for-profit entities, and commercial developers and redistributors. In this mixed-model organizational environment, all FOSS developers must have an environment where liability and other legal issues do not impede their important public service work. The Software Freedom Law Center ("SFLC") provides legal representation and other law related services to protect and advance FOSS.

DANIEL B. RAVICHER

Dan Ravicher is Executive Director and Founder of the Public Patent Foundation ("PUBPAT"), Legal Director of the Software Freedom Law Center ("SFLC") and a registered patent attorney. A significant portion of Mr. Ravicher's practice involves Free and Open Source Software legal issues, most particularly licensing and patent counseling. In addition to representing clients of the SFLC, such as the Free Software Foundation, on these matters, Mr. Ravicher has published numerous legal articles and given dozens of presentations regarding both Free and Open Source Software legal issues and patent law. Prior to founding PUBPAT, Mr. Ravicher was associated with Skadden, Arps, Slate, Meagher & Flom LLP, Brobeck, Phleger & Harrison, LLP, and Patterson, Belknap, Webb & Tyler, LLP, all in New York, and served the Honorable Randall R. Rader, Circuit Judge for the U.S. Court of Appeals for the Federal Circuit in Washington, D.C. Mr. Ravicher received his law degree from the University of Virginia School of Law, where he was the Franklin O. Blechman Scholar for his class, a Mortimer Caplin Public Service Award recipient and Editor of the Virginia Journal of Law and Technology, and his bachelors degree in materials science magna cum laude with University Honors from the University of South Florida. He may be reached at (917) 843-3425 or dan@pubpat.org.

# PART 1

## THE IMPLIED PATENT GRANT IN GPL

Daniel B. Ravicher
Executive Director, Public Patent Foundation
Legal Director, Software Freedom Law Center

The GNU General Public License ("GPL"), despite being silent with respect to patents, may actually confer on its licensees more rights to a licensor's patents than other free software licenses that purport to address the issue. This is the case because patent law, under the doctrine of implied license, gives to each distributee of a patented article a license from the distributor to practice any patent claims owned or held by the distributor that cover the distributed article. The implied license also extends to any patent claims owned or held by the distributor that cover "reasonably contemplated uses" of the patented article.

To quote the Federal Circuit Court of Appeals, the highest court for patent cases other than the Supreme Court:

> "Generally, when a seller sells a product without restriction, it in effect promises the purchaser that in exchange for the price paid, it will not interfere with the purchaser's full enjoyment of the product purchased. The buyer has an implied license under any patents of the seller that dominate the product or any uses of the product to which the parties might reasonably contemplate the product will be put."

*Hewlett-Packard Co. v. Repeat-O-Type Stencil Mfg. Corp., Inc.*, 123 F.3d 1445 (Fed. Cir. 1997).

Of course, free software is licensed, not sold, and there are indeed restrictions placed on the licensee, but those differences are not likely to prevent the application of the implied license doctrine to free software, because software licensed under the GPL grants the licensee the right to make, use, and sell the software, each of which are exclusive rights of a patent holder. Therefore, although the GPL does not expressly grant the licensee the right to do those things under any patents the licensor may have that cover the software or its reasonably contemplated uses, by licensing the software under the GPL, the distributor impliedly licenses those patents to the GPL licensee with respect to the GPL licensed software.

An interesting issue regarding this implied patent license of GPL'd software is what would be considered "uses of the [software] to which the parties might reasonably contemplate the product will be put." A clever advocate may argue that the implied license granted by GPL is larger in scope than the express license in other free software licenses that contain express patent grants, in that, the patent license clause in the majority of those other free software licenses are specifically limited to patent claims covered by the code as licensed by the patentee.

To the contrary, GPL's implied patent license grants the GPL licensee a patent license to do much more than just that because the GPL licensee, under the doctrine of implied patent license, is free to practice any patent claims held by the licensor that cover "reasonably contemplated uses" of the GPL'd code, which may very well include creation and distribution of derivative works since the GPL's terms, under which the patented code is distributed, expressly permits such activity.

Further supporting this result is the Federal Circuit's pronouncement that the recipient of a patented article has, not only an implied license to make, use, and sell the article, but also an implied patent license to repair the article to enable it to function properly. *Bottom Line Mgmt., Inc. v. Pan Man, Inc.*, 228 F.3d 1352 (Fed. Cir. 2000). Additionally, the Federal Circuit extended that rule to include any future recipients of the patented article, not just the direct recipient from the distributor. This theory comports well with the idea of free software, whereby software is distributed amongst many entities within the community for the purpose of constant evolution and improvement. In this way, the law of implied patent license used by the GPL ensures that the community mutually benefits from the licensing of patents to any single community member.

Note that simply because GPL'd software has an implied patent license does not mean that any patents held by a distributor of GPL'd code become worthless. To the contrary, the patents are still valid and enforceable against either (a) any software other than that licensed under the GPL by the patent holder, and (b) any party that does not comply with the GPL with respect to the licensed software.

For example, if Company A has a patent on web browsing, but also licenses a web browsing software program under the GPL, then it cannot assert the patent against any party that takes a license to its program under the GPL. However, if a party uses that program without complying with the GPL, then Company A can assert, not just copyright infringement claims against the non-GPL-compliant party, but also infringement of the patent, because the implied patent license only extends to use of the software in accordance with the GPL. Further, if Company B distributes a competitive web browsing program, Company A is free to assert its patent against Company B or any user of Company B's product. It is irrelevant whether Company B's program is distributed under the GPL, as Company B can not grant implied licenses to Company A's patent.

This result also reassures companies that they need not fear loosing their proprietary value in patents to competitors through the GPL implied patent license, as only those competitors who adopt their technology and comply with the GPL's terms can benefit from the implied patent license. To continue the example above, Company B does not receive a free ride on Company A's patent, as Company B has not licensed-in and then redistributed Company A's web browser under the GPL. If Company B does do that, however, Company A still has not lost competitive advantage against Company B, as Company B must then, when it re-distributes Company A's web browser grant an implied license to any of its patents that cover the web browser.

Further, if Company B improves the web browser a distributes that improved version of Company A's program, it must do so under the GPL, meaning that any patents it holds that cover not just the original version, but also the improved version, are impliedly licensed to any of Company B's licensees. As such, the only way Company B can benefit from Company A's implied patent license, is if it, itself, distributes Company A's software program and grants an implied patent license to any of its own patents that cover that program.

## PART 2

PATENTS – WHY FREE / OPEN SOURCE SOFTWARE

MIGHT HAVE LESS TO FEAR THAN NON-FREE SOFTWARE

Daniel B. Ravicher
Executive Director, Public Patent Foundation
Legal Director, Software Freedom Law Center

Many believe that patents pose a large threat to Free / Open Source Software. Although this may be true, Free Software is no more concerned about, and actually might be less concerned about, software patents than most other software developers.

First recognize that any patent that covers Free Software is going to cover non-free software, because licensing terms are irrelevant to a patent's scope. So, there aren't any patents that only threaten Free Software; they all threaten all software, regardless of license terms.

Now, assume there exists a patent that arguably covers Free Software and the patent holder brings a patent infringement law suit to stop such infringement. It is highly unlikely that the patent holder would receive a preliminary injunction, as they are highly unusual in patent infringement cases to begin with. Further, the equities and public harm would very rarely be in favor of the patent holder, because Free Software is a public good, on which many individuals, businesses, and government rely. Lastly, the wide spread distribution of Free Software would render any such injunction meaningless, unless it attached to every single possessor of a copy of the allegedly infringing code, a highly unlikely scenario for any Free Software project of significance.

This is where free software is very different from non-free software, and why patents are more of a threat to the latter, than the former. A non-free software product will have a harder time defending against a preliminary injunction as, although they might be able to argue equities, they will have much more difficulty arguing a public harm. Further, their product can indeed be easy stopped because they have complete distribution control over it.

Having virtually no chance at preliminary relief, the patent holder will then seek damages, but here they are in a catch-22 when it comes to Free Software. If they go after deep pockets, the defendant will, by the fact that they are a "deep pocket," be more than capable of defending itself against the patent assertion. Such defendants have a very high success rate in patent cases, as about half of all litigated patents are held invalid and many of those that are held valid are nonetheless held not infringed.

If, instead, the patent holder sues a little guy, there will be no money to recover, because the defendant is a "little guy." Additionally, there are several reasons why a deep pocket may step up to protect any such little guy. First, the Free Software product involved may be very important to the deep pocket. For instance, if a patent holder has a patent on Apache, the most used web sever in existence today, they could sue the Apache Foundation, a small non-profit Free Software development organization. See Netcraft Web Server Survey Archives, http://news.netcraft.com/archives/web_server_survey.html (showing Apache having almost 70% market share in the web server market in February 2004). However, IBM and all of the other major distributors of Apache would have a vested interest in ensuring Apache, the product, wasn't defeated.

Second, a deep pocket competitor may be worried about a favorable claim construction, the pre-trial process of determining what ambiguous or arguable terms in a patent mean, resulting in the case with the little guy because it may believe that it is next on the patent holder's "to-be-sued" list. As such, the deep pocket may very well wish to impact the first claim construction where the little guy is involved to ensure that the claim construction is not a lopsided victory for the patent holder, because future litigations involving the asserted patent may simply adopt or be significantly impacted by that claim construction.

Therefore, a permanent injunction is the only truly threatening remedy available for a patent holder bringing a patent infringement suit against Free Software. However, knowing that patents cannot cover functionality, and can only cover certain structure that accomplishes functionality, it is highly likely that before a patent infringement case is tried and appealed, the Free Software at issue can be designed around the asserted patent. Further, it is also highly likely that the Free Software community, a very participatory and technically sophisticated group, will be quite capable at finding prior art to challenge the patent's validity.

In essence, the common theme is that, because of the network effects inherent with Free Software, no party or project of any significance can be singled out for attack on the basis of patent infringement. It is the opposite of a free rider problem.

However, this does not mean that patents pose no threat to Free Software. The burden placed on parties defending themselves from assertions of patent infringement is indeed substantial, in terms of both costs and distraction. Significant resources will be required to defend Free Software from patents, but the battle can – and will – be won.

# Free Software and Beyond

*Richard Stallman, lead developer, GNU Operating System*

ABSTRACT:

People often ask whether the ideals of the Free Software Movement apply to anything other than software. In this talk I will explain how I have extended them to additional kinds of works, and where I think they do not extend.

USEFUL INFORMATION:

The GNU web site: www.gnu.org.

The philosophy of free software: http://www.gnu.org/philosophy

Free Software Directory (~4000 free software packages):
http://www.gnu.org/directory

BIO - Richard Stallman
Mr. Stallman launched the development of the GNU operating system (see www.gnu.org) in 1984. GNU is free software: everyone has the freedom to copy it and redistribute it, as well as to make changes either large or small. The GNU/Linux system, basically the GNU operating system with Linux added, is used on tens of millions of computers today. Stallman has received the ACM Grace Hopper Award, a MacArthur Foundation fellowship, the Electronic Frontier Foundation's Pioneer award, and the the Takeda Award for Social/Economic Betterment, as well as several honorary doctorates.

---

## The Free Software Definition

(from http://www.gnu.org/philosophy/free-sw.html)

We maintain this free software definition to show clearly what must be true about a particular software program for it to be considered free software.

``Free software'' is a matter of liberty, not price. To understand the concept, you should think of ``free'' as in ``free speech,'' not as in ``free beer.''

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose (freedom 0).

- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission.

You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be required to notify anyone in particular, or in any particular way.

The freedom to use a program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job, and without being required to communicate subsequently with the developer or any other specific entity.

The freedom to redistribute copies must include binary or executable forms of the program, as well as source code, for both modified and unmodified versions. (Distributing programs in runnable form is necessary for conveniently installable free operating systems.) It is ok if there is no way to produce a binary or executable form for a certain program (since some languages don't support that feature), but you must have the freedom to redistribute such forms should you find or develop a way to make them.

In order for the freedoms to make changes, and to publish improved versions, to be meaningful, you must have access to the source code of the program. Therefore, accessibility of source code is a necessary condition for free software.

In order for these freedoms to be real, they must be irrevocable as long as you do nothing wrong; if the developer of the software has the power to revoke the license, without your doing anything to give cause, the software is not free.

However, certain kinds of rules about the manner of distributing free software are acceptable, when they don't conflict with the central freedoms. For example, copyleft (very simply stated) is the rule that when redistributing the program, you cannot add restrictions to deny other people the central freedoms. This rule does not conflict with the central freedoms; rather it protects them.

You may have paid money to get copies of free software, or you may have obtained copies at no charge. But regardless of how you got your copies, you always have the freedom to copy and change the software, even to sell copies.

``Free software" does not mean ``non-commercial". A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important.

Rules about how to package a modified version are acceptable, if they don't substantively block your freedom to release modified versions. Rules that ``if you make the program available in this way, you must make it available in that way also" can be acceptable too, on the same condition. (Note that such a rule still leaves you the choice of whether to publish the program or not.) It is also acceptable for the license to require that, if you have distributed a modified version and a previous developer asks for a copy of it, you must send one, or that you identify yourself on your modifications.

In the GNU project, we use ``copyleft" to protect these freedoms legally for everyone. But non-copylefted free software also exists. We believe there are important reasons why it is better to use copyleft, but if your program is non-copylefted free software, we can still use it.

See Categories of Free Software for a description of how ``free software," ``copylefted software" and other categories of software relate to each other.

Sometimes government export control regulations and trade sanctions can constrain your freedom to distribute copies of programs internationally. Software developers do not have the power to eliminate or override these restrictions, but what they can and must do is refuse to impose them as conditions of use of the program. In this way, the restrictions will not affect activities and people outside the jurisdictions of these governments.

Most free software licenses are based on copyright, and there are limits on what kinds of requirements can be imposed through copyright. If a copyright-based license respects freedom in the ways described above, it is unlikely to have some other sort of problem that we never anticipated (though this does happen occasionally). However, some free software licenses are based on contracts, and contracts can impose a much larger range of possible restrictions. That means there are many possible ways such a license could be unacceptably restrictive and non-free.

We can't possibly list all the possible contract restrictions that would be unacceptable. If a contract-based license restricts the user in an unusual way that copyright-based licenses cannot, and which isn't mentioned here as legitimate, we will have to think about it, and we will probably decide it is non-free.

When talking about free software, it is best to avoid using terms like ``give away" or ``for free", because those terms imply that the issue is about price, not freedom. Some common terms such as ``piracy" embody opinions we hope you won't endorse. See Confusing Words and Phrases that are Worth Avoiding for a discussion of these terms. We also have a list of translations of "free software" into various languages.

Finally, note that criteria such as those stated in this free software definition require careful thought for their interpretation. To decide whether a specific software license qualifies as a free software license, we judge it based on these criteria to determine whether it fits their spirit as well as the precise words. If a license includes unconscionable restrictions, we reject it, even if we did not anticipate the issue in these criteria. Sometimes a license requirement raises an issue that calls for extensive thought, including discussions with a lawyer, before we can decide if the requirement is acceptable. When we reach a conclusion about a new issue, we often update these criteria to make it easier to see why certain licenses do or don't qualify.

If you are interested in whether a specific license qualifies as a free software license, see our list of licenses. If the license you are concerned with is not listed there, you can ask us about it by sending us email at <licensing@gnu.org>.

If you are contemplating writing a new license, please contact the FSF by writing to that address. The proliferation of different free software licenses means increased work for users in understanding the licenses; we may be able to help you find an existing Free Software license that meets your needs.

If that isn't possible, if you really need a new license, with our help you can ensure that the license really is a Free Software license and avoid various practical problems.

---

Another group has started using the term "open source" to mean something close (but not identical) to "free software". We prefer the term "free software" because, once you have heard it refers to freedom rather than price, it calls to mind freedom. The word "open" never does that.

# Open Source - The Legal Front

*Ed Walsh, Wolf Greenfield & Sachs*

ABSTRACT:

Many people were jolted into awareness of open source software by accusations from The SCO Group that IBM owed it many billions of dollars because IBM's participation in the open source movement violated SCO's intellectual property rights. The panel will discuss the technological origins of the dispute and trace its legal history. The specific dispute between The SCO Group and IBM will be used to highlight legal concerns that have been raised about open source software in general.

BIO - Edmund J. Walsh
Counsel
Wolf Greenfield & Sacks, P.C.
Boston, MA
http://www.wolfgreenfield.com

Ed Walsh is a Counsel at Wolf Greenfield & Sacks and is a member of the Electrical and Computer Technologies Practice Group. He has more than 14 years experience serving as in-house counsel for technology companies. Most recently, Ed served as Chief Intellectual Property Counsel for Teradyne, a position he held for ten years.

Electrical and computer-related clients seek him for IP strategy development and execution, including patent prosecution, clearances and counseling, IP portfolio management, litigation, licensing, consulting agreements and joint development agreements. His areas of technical expertise include semiconductor processing and high-speed circuit design, software, networks and network management, connectors and interconnection technology and various types of test equipment.

Prior to Wolf Greenfield, Ed was the Division IP Counsel for Textron Specialty Materials and Patent Attorney for Raytheon.

Ed has taught at Boston University as an Adjunct Professor and Lecturer in electrical engineering. He was also a Fellow of the National Science Foundation and has also been a faculty member for both the Massachusetts Continuing Legal Education IP Program and Suffolk University Law School's Advanced Legal Studies Program

# OPEN SOURCE LITIGATION
## *PAST, PRESENT AND FUTURE*

## EXECUTIVE SUMMARY

Litigation initiated by The SCO Group against IBM raised fears that Linux, one of the most popular open source programs, would be derailed by claims of IP infringement. Though that suit persists, the initial hype has died away, making way for more moderate reflection on whether open source poses a greater risk of IP infringement than proprietary software. There is scant data to suggest that the SCO suit is the beginning of a trend rather than an aberrant event. Yet, there are differences between open source and proprietary software such that the factors shaping the risk of IP infringement are different. With widespread use of open source both relatively recent and still evolving, it is too early to declare that concerns of IP infringement should be ignored completely.

The zealous public debate from participants and bystanders on both sides of the SCO litigation also left many with a greater awareness that adopting open source does present greater risks that a business could lose IP rights. Though some open source licenses may contain clauses of concern, internal use of open source does not inherently create unique IP risks. But, participating in the open source movement by distributing open source products requires some IP rights to be surrendered.

Though the open source software movement has seen incredible advances in a very short time, there are still areas of risk and uncertainty. These risk and uncertainty can be the source of disputes that generate litigation as open source software is adapted by a broader community including for-profit businesses. Accordingly, current risks and areas of uncertainty are indicators of possible future litigation.

## RISK OF IP CLAIMS

Many in the mainstream legal community were jolted into awareness of open source software by a multi-***billion*** dollar suit filed by The SCO Group against IBM.[1] It is that legal battle, and the collateral litigation it has spawned, that many think of when they think of the risk of IP claims.

The litigation centers around charges that IBM breached contracts with SCO to develop a new version of UNIX. Instead, it is alleged, IBM surreptitiously devoted its own resources, and SCO's proprietary technology, to developing the open source operating

[1] *SCO v. IBM,*, Case No. 2:03CV294 DAK (D. Utah).

system, Linux. SCO expanded its legal campaign to users[2] of Linux and to licensees of UNIX who have begun to use Linux.[3]

Counter charges abound that SCO never owned IP rights to UNIX[4], or, if it did, gave them away by distributing open source software itself. Red Hat, a Linux distributor, has seized the initiative and sued SCO, seeking a ruling that Linux does not infringe SCO's copyrights.[5] Discerning where the truth lies is no simple task.

But, SCO has not faired well so far in proving that its initial hype of the suit was warranted. In litigation in Germany, SCO was ordered to stop claiming that Linux infringes its intellectual property rights until it is prepared to prove that claim.[6] In the US, its suit against a UNIX user has fizzled and the judge presiding over the suit with IBM has expressed reservations that SCO will be able to prove its case.[7] But, the IBM case is still in the discovery phase, with SCO only recently getting permission to access much of IBM's code and development records, which SCO contends will show SCO's rights were violated by IBM's role in furthering Linux development.

But regardless of how it ends, the SCO litigation is unlikely to have a direct impact on the open source movement. Even if SCO prevails on its breach of contract claims, others – who were not parties to the contract – would feel few consequences. The effect of any copyright infringement can be deflected by rewriting code subject to SCO's copyright, which Linux developers and distributors have committed to do. Further, SCO has never owned patents in the UNIX code and so has no IP rights to preclude use of the functionality of UNIX or any other open source code. The most lasting impact of the SCO suit may be indirect: sensitizing people to the risks of IP claims.

---

[2] *SCO v. AutoZone*, Case No. CV-S-04-0237-RCJ (LRL) (D. Nev).

[3] *The SCO Group, Inc., vs. DaimlerChrysler Corp,* No. 04-056587-CK Circuit Court for Oakland County, Michigan.

[4] *The SCO Group, Inc. v. Novell, Inc.*, CV No.: 2:04CV00139 (D.Utah).

[5] *Red Hat, Inc. v. The SCO Group*, C.A. No. 03-772-SLR (D.Del).

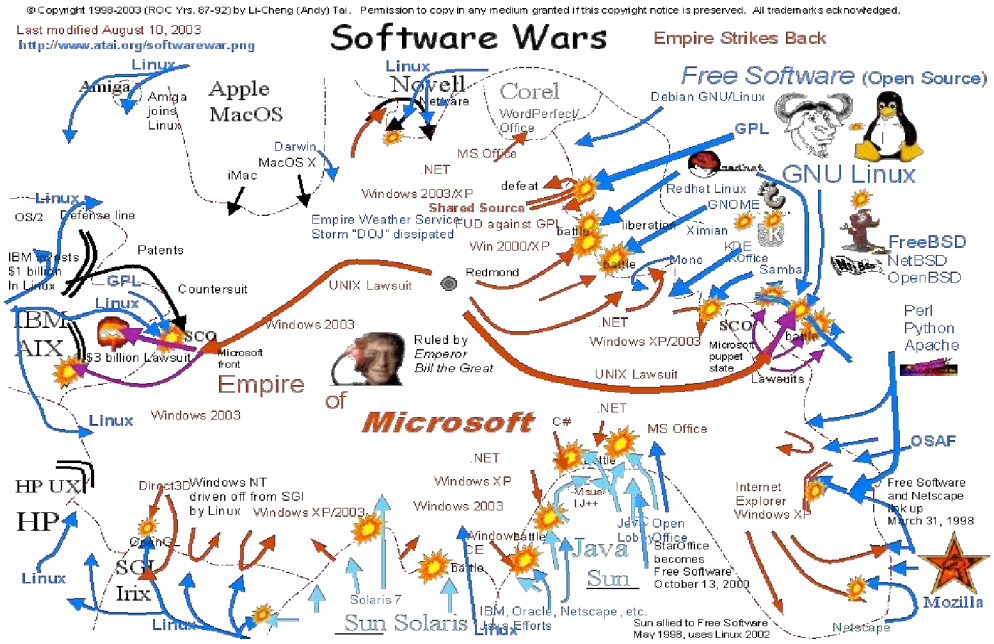
[6] *Tarent v. The SCO Group* (District Court Munich I) June 5, 2003.

[7]Order by Judge Kimball dated February 11, 2005, denying IBM's request for partial summary judgment, giving SCO an opportunity to take discovery and prove its case at trial, but nonetheless commenting on the current state of SCO's case. *SCO v. IBM,*, Case No. 2:03CV294 DAK (D. Utah). ("Viewed against the backdrop of SCO's plethora of public statements concerning IBM's and others' infringement of SCO's purported copyrights to the Unix software, it is astonishing that SCO has not offered any competent evidence to create a disputed fact regarding whether IBM has infringed SCO's alleged copyrights through IBM's Linux activities").

IP claims are not unique to open source code. A relevant question, therefore, is whether the risk is greater than for proprietary code. Some argue that an open source development model premised on contributions from many, sometimes anonymous contributors with disdain for intellectual property laws, increases the risk that code subject to third-party IP rights will be used.[8] Others argue that the open collaboration has the opposite effect by vetting improper code.

Little practical guidance is likely to result from probing whether open source programmers can be stereotyped as a group that is more or less unethical and therefore more or less prone to copy code than proprietary code developers. Nonetheless, there is evidence to support an argument that there is more risk.

A graphic prepared by fervent supporters of the open source movement highlights one risk factor – much open source code has been developed to displace existing proprietary software. Patents in those existing products can be infringed without any unethical intent to infringe or even knowledge of the patent.




Do these "battle plans" suggest a risk of conflict?

As the graphic illustrates, open source programmers often target Microsoft – with their code and their jokes. So far, Microsoft has seemed willing to license its patent portfolio,

[8] *See, e.g., Davidson & Assoc. v. Internet Gateway,* 334 Fed.Supp.2d 1164 (E.D. MO, 2004)(Users of an on-line, multi-player game, frustrated with poor service of commercial game company and philosophically opposed to advertisements that appeared when the game was played on-line, created an open source version of the server needed to play the game on-line, in the process infringing the game developer's copyrights, violating license agreements and violating the Digital Millennium Copyright Act).

which limits the monetary exposure of open source users and therefore reduces risk.[9] But, the risks posed by products developed to target a company with a formidable arsenal of patents should not be underestimated. Further, Microsoft is by no means the only holder of software patents. For example, a recent study concluded that Linux potentially infringes 283 patents – only 27 of which are owned by Microsoft.[10]

In counterpoint, the open source experiment in collaborative vetting of code for IP violations has taken tangible form. The Public Patent Foundation has asked the Patent Office to revoke one of the patents that Microsoft alleges covers the file access table (FAT) widely used in desktop operating systems.[11] The Public Patent Foundation has won the first round in this fight, but is still a long way from prevailing on this one patent or any other.

Given this competing evidence, it is premature to predict whether IP infringement will be more prevalent in open source code. It also seems premature to predict, if an infringement occurs, whether the consequences will be more severe in situations were open source code is involved.
But, many factors that govern whether and the extent to which IP rights are enforced are different in an open source context. In most commercial markets, enforcing IP rights does not make economic sense when all participants have similar cost structures and make comparable investment in the generation of intellectual property. Consequently, portfolio cross-licensing is relatively common among commercial companies. But, the open source "cost" model can be disruptive to for-profit businesses. And, to date, there has been little organized effort to patent open source products. As a result, the cross-licensing that reduces IP litigation in many commercial markets is less likely to exist. A further risk factor is that publication of source code makes infringements easier to detect, thereby removing one factor that restrains enforcement.

Further, in a traditional commercial context, infringements are often dealt with by suits against the originator of an infringing product. Those who acquire the product are generally insulated from the dispute. Though many open source distributors have begun to promise some form of protection from IP claims, traditional open source licenses includes a disclaimer of all liability. Those who adopt open source will not, therefore be able to seek indemnity from suppliers. While many who use open source will not present economically viable targets for enforcement actions, for-profit business should expect that, if they adopt open source code, they are more likely to be forced to deal with infringements caused by others.

---

[9] *See, e.g.* http://www.microsoft.com/mscorp/ip/tech/fat.asp

[10] "Group: Linux potentially infringes 283 patents" NEWS.COM, August 1, 2004. http://news.com.com/Group+Linux+potentially+infringes+283+patents/2100-7344_3-5291403.html?tag=nl

[11] Reexamination Application Serial Number 90/007,007 filed with the U.S. Patent And Trademark Office.

In the aggregate, these factors raise the specter that broad-based licensing programs, such as the ones proposed by SCO for Linux or Microsoft based on its FAT patents, are possible with open source code. But, as with many issues, time will reveal the consequences of this risk.

## LOSS OF IP RIGHTS

The heated public debate surrounding The SCO v. IBM litigation – even if exaggerated – highlighted the risk to businesses wishing to retain some IP rights while adopting open source. The risk to IP rights arises because open source licenses, as a generalization, require that any distribution of open source code, and any derivative work based on open source code, include the entire source code and a license allowing the recipient to use, modify and further distribute the software.

**"The GPL has this sucking effect of grabbing your IP, sucking it in and destroying your property rights."**

Darl McBride

**"Having a hole in your head has this sucking effect."**

Linus Torvalds

Exchange between Darl McBride, CEO of SCO, whose company is suing IBM for allegedly putting its intellectual property into Linux and Linus Torvalds, originator of Linux, as reported in *The Washington Post,* Thursday, June 10, 2004; p. E01

Because loss of IP rights is only triggered by distribution of open source software, merely using open source code does not create any special risk to IP rights (unless the license includes additional terms conditioning use of the software on a greater relinquishment of rights). But, when a developer creates a new product that is a derivative work of open source software, the new product becomes subject to open source terms upon distribution. A developer may, for business or philosophical reasons, also decide to voluntarily distribute a product using open source terms. Expressly or by creating a derivative work placing all or a portion of the product under open source terms destroys most IP rights to that portion of the product. This loss of IP rights is both intended[12] and potentially significant.

Of course, whether one sees this effect as good or bad depends on one's point of view. But, many businesses depend on IP rights as part of their business strategy. It is common for for-profit businesses to rely on IP rights to ensure they have an advantage in providing upgrades, maintenance and support services or entirely new products that leverage technology from prior products. Further, IP rights are often the basis of

[12] "[F]ree software, far from being a marginal participant in the commercial software market, is the vital first step in the withering away of the intellectual property system" Anarchism Triumphant: Free Software and the Death of Copyright, by Eben Molgen. http://firstmonday.org/issues/issue4_8/moglen/index.html#author. Eben Moglen is general counsel for the Free Software Foundation.

restrictions in agreements with suppliers, development partners and employees. Not being able to enforce those rights can be bad; losing those rights and attempting to enforce them anyway can be worse, potentially creating exposure under the antitrust laws. [13]

IP rights, whether protected by copyright, trade secret or patents, can be impacted by distribution of proprietary works that become subject to the open source terms. Copyright is lost through the license which is expressly granted to whomever receives open source code when distributed. Open source licenses allow direct copying and further distribution of any open source program. These licenses also allow the program to be modified and distributed in modified form.

Trade secret rights are lost through the publication of the source code that is required for compliance with open source terms. Disclosing a trade secret without restriction destroys it. Distributing source code discloses, and therefore destroys, trade secrets such as algorithms, formulas or data, that can be discerned from the source.

Distribution of software under open source terms also requires a relinquishment of patent rights, though the extent to which patent rights are relinquished varies under different licenses. Two of the most widely used open source licenses take different approaches on the question of patent licenses. The Mozilla Public License contains an express grant of a patent license.[14] The express license is limited, generally applying only to contributions made by the patent holder.

In addition, the Mozilla Public License includes an economic disincentive for any person using or distributing open source software to assert patents against other contributors. According to its terms, all or parts of the license terminate if the user of the licensed code asserts patent rights against any contributor to the open source code. License termination exposes a user of the open source who elects to assert patent rights to claims of infringement for using the open source code that was previously licensed, thereby imposing a penalty for asserting patent rights. This clause provides an important exception to the general rule that IP rights are not diminished merely by using open source code.

In contrast there is no express patent license in the current version of the GPL. [15] However, two legal doctrines come into play: "exhaustion" [16] and "implied license." [17]

---

[13] *CVD, Inc. v. Raytheon Co.*, 769 F.2d 842 (1st Cir. 1985), *cert. denied*, 475 U.S. 1016 (1986)(Threatening to enforce "trade secret" rights in information that was not, in fact a trade secret, was an antitrust violation).

[14] http://www.mozilla.org/MPL/MPL-1.1.html

[15] The Free Software Foundation, publisher of the GPL, is reported to be tackling the question of what patent rights are granted or –from the opposite perspective -- what patent rights are relinquished as it drafts the next version of the GPL. "GPL 3 to Take on IP, Patents,"

Traditionally, these doctrines prevent a patent holder who has sold a product from using patents to block use or transfer of the product. They, thus preclude the patent holder from blocking the purchaser of a product from any activity which the circumstances plainly indicate ought to be allowed.

But, these notions from traditional business transactions do not map cleanly to an open source community in which open source products are intended for widespread use and adaptation for other applications. There are no clear answers to questions, such as: Who benefits from the implied license? What technology is covered by the license? What actions are protected by the license?

The risk from loss of IP rights or the uncertainty of not knowing for sure what rights have been given away to competitors could be a source of future disputes that are resolved through litigation. Litigation could be brought be advocates of open source software, seeking enforcement of the open source terms.[18] Litigation of open source issues may arise as companies who distribute open source code seek to enforce IP rights against competitors who have copied their products.[19]

## Just Derivative Works

An area of uncertainty, and therefore a possible issue for litigation, is the extent to which IP rights are lost as a consequence of distributing open source software. IP rights are lost only in the contributions that a developer makes and then distributes under open source terms, which includes derivative works of an open source code. If a single product contains both open source and code that a developer wants to keep as proprietary, IP rights are lost only if the proprietary code is part of a derivative work of open source code.

---

eWeek, November 22, 2004
http://www.eweek.com/article2/0,1759,1730102,00.asp

[16] *See, e.g. United States v. Univis Lens Co., Inc*. 316 U.S. 241, 250 (1942).

[17] *See, e.g. Anton/Bauer, Inc. v. PAG, Ltd.,* 329 F.3d 1343 (Fed. Cir. 2003).

[18] See, e.g. *Netfilter v. Sitecom* (Germany) (GPL enforced blocking use of open source code without distributing source code).

[19] See, e.g., *Computer Associates International v. Quest Software,* 333 F.Supp.2d 688, 2004 (N.D.Ill., Jun 28, 2004)(Defendant asserted that plaintiff had no right to sue for copying of its software because it had been distributed under open source terms). A similar defense was asserted by IBM in its litigation with SCO.

The Mozilla Public License defines derivative works to be any "addition to or deletion from the substance or structure of" the code. In cases where code is distributed as a series of files, any file that contains any open source code is a derivative work. Conversely, code distributed in a file without any open source code is not a derivative work – creating a "safe harbor" -- code not intended to be subject to open source terms should be distributed in a separate file.

The GPL contains no comparable safe harbor. It relies on the definition of "derivative works" in US copyright law.[20] Despite seeming clarity, there is uncertainty in how this definition should be applied in practice. Some courts deciding issues unrelated to open source have deemed a work to be "derivative" only if it recasts or transforms the original. Under this rationale, where an original work is included – without being recast or in some way transformed -- along with added material, the added material is a separate work rather than a derivative work.[21] If this definition is accepted, proprietary code accepting outputs from or providing inputs to open source code, without changing the open source code, would not be a derivative work. Anecdotally, it appears as if many in the open source community have accepted a version of this definition of "derivative works," treating code that is dynamically linked to open source code as a separate work rather than a derivative work.

But, the issue is far from settled. One of the leading treatises on copyright law describes "a bewildering number of options for conceptualizing derivative works."[22] The author points to a court decision premised on the "ancient insight that all works are, in some sense, derivative."[23] Under this definition, any combination of proprietary code and open source code could be deemed to create a derivative work. One of the few US cases to consider (but not decide) the meaning of "derivative works" in the open source context seemed to accept this view.[24] Such uncertainty, unfortunately, is itself a form of risk.

---

[20] A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a "derivative work" 17 USC §101.

[21] *Lewis Galoob Toys, Inc. v. Nintendo of America, Inc.* 964 F.2d 965 (9tth Cir, 1992). ("Game Genie" device that modifies the output of a Nintendo game system to create new displays did not change the underlying Nintendo code and therefore did not to create a derivative work.) *SHL v. Artisan*, 117 F.Supp.2d 301 (S.D.N.Y. 2000) (Photograph of ornamental frames for pictures and mirrors is not a derivative work of the frames. Though the original work – the frame – is included in the photograph, it has not been recast or transformed and is not a derivative work. However, the arrangement, lighting, etc. of the items in the photograph, if sufficiently creative, might be a separate work that is copyrightable.).

[22] Nimmer on Copyright §3.03[C][3].

[23] *quoting, Emerson v. Davies*, 8 F.Cas. 615, 621, No. 4436 (C.C. Mass. 1845).

[24] *Progress Software v. MySQL,* 195 F.Supp2d 328 (D. Mass 2002) (In a preliminary motion, the judge

## GOING FORWARD

Open source software is still a newcomer in the software community. Adoption of open source by for-profit businesses is even more recent. Open source is premised on a philosophy and economic model that challenges the *status quo* in the business world. These factors generate areas of risk and uncertainty. While there are no obvious indicators of a flood of litigation surrounding open source, there is enough risk and uncertainty that some issues will almost certainly be resolved through litigation.

*About the Author*

*Ed Walsh is currently of counsel to the intellectual property specialty firm Wolf, Greenfield and Sacks in Boston, working predominately with clients in the firm's electrical and computer technologies practice group. His perspectives reflect more than 14 years as in-house counsel for technology companies. Ed established intellectual property programs for Teradyne, Inc.-- where he served as its Chief Intellectual Property Counsel -- and Textron Specialty Materials. Before devoting himself full time to the practice of law, he managed a software development group and graduated from Stanford University, where he studied as a Fellow of the National Science Foundation, and Boston University College of Engineering. He has previously written and spoken on Open Source Software for managers, lawyers and technologists in forums such as the Massachusetts Software Counsel legal issues series, Lawyer's Weekly and the Embedded Systems Developers conference.*

---

acknowledged that MySQL, arguing that a proprietary program interacting with an open source library is a derivative work, had the "better argument." ).

# Open Source Licenses - The Latest Issues

*Peter Moldave, Gesmer Updegrove*

ABSTRACT:

In order to understand the rights and restrictions relating the use of open source software you have to understand the license agreements. Different constituencies (developers, users, investors) have different objectives. The choice of an open source license and the consequence of that choice can make a significant difference in the viability or appeal of an open source project to different constituencies. The panel will discuss the strategy behind major open source licenses (GPL, BSD, Apache and others) and their use by each of these constituencies; recent and potential future changes to the terms of some of these licenses (for example the GPL and Apache licenses); different treatment of such things as patent license and ASP services; and expansion of the open source concept to non-software areas (for example, the Creative Commons licenses).

BIO - Peter M. Moldave
Gesmer Updegrove LLP
Boston, MA
http://www.gesmer.com

Peter Moldave concentrates his practice in the areas of computer software licensing transactions and corporate and securities law. He advises a variety of clients in the software industry, including emerging software companies in the internet and enterprise software area, investors in software companies and individual executives, authors and consultants, on issues including business formation, angel and venture capital financing, mergers and acquisitions, equity-based compensation and software development, distribution, publishing and licensing arrangements.

Since Spring 2002, Peter has been an adjunct faculty member at Suffolk University Law School, where he teaches a course on licensing intellectual property.

Prior to joining the firm, he worked at Apple Computer, Inc., where he was responsible for federal securities law compliance and legal aspects of corporate finance transactions and technology joint ventures, as well as general corporate legal advice, and at Skadden, Arps, Slate, Meagher & Flom, where he concentrated in corporate and securities transactions, including mergers and acquisition of publicly and privately held companies and public and private placements of securities.

Peter has a J.D. from the University of California (Boalt Hall) and a B.A. in Chemistry and Physics from Harvard College. He is a member of the California and Massachusetts Bars.

## 1. GPL v. MPL v. BSD

| | GPL | Mozilla 1.1 | BSD |
|---|---|---|---|
| **Notice** | Must include copyright notice and disclaimer of warranty on all copies<br>Must distribute a copy of the license with any copies | Must include copy of license with all licenses, and terms cannot be offered or imposed that would alter or restrict the license. | Only need to include copyright notice and permission notice in supporting documentation. |
| **Additional Independent Work** | Includes otherwise independent and separate works if distributed "as part of a whole which is a work based on the Program"; complex definition what is "as part of"<br>Combination on media does not in itself cause parts to be a single product | Free software may be include in a larger work, but the requirements of the license must be fulfilled for the free software. | Allowed without restriction |
| **Modifications to base** | Modifications must include statement referencing contributor and documenting the changes | Modifications must include statement referencing contributor and documenting the changes | Allowed without restriction |
| **Effect of Third Party Rights** | May not be able to distribute | Notice requirement if contributor is aware of third party licenses required | No restriction |
| **Ability to charge** | May charge a fee for the physical act of transferring a copy<br>May offer warranty protection for a fee<br>No fee based on for further redistribution by third party | May charge a fee for warranty, support, indemnity or liability obligations, but must indemnify contributors for resulting liability. | No restriction |
| **Disclaimers** | AS IS; NO LIABILITY | AS IS; NO LIABILITY | AS IS; NO LIABILITY |

| | | | |
|---|---|---|---|
| **Provision of Source Code** | Source code must be provided with (or made available for nominal additional cost) all licenses | Modifications of source code must be made available | No requirement to provide; however, if code is provided, copyright and permission statement needs to be included |
| **Contribution Back** | no reference | Each licensee who creates or contributes to the creation of Modifications gives a license to other licensees to use the Modifications and a license under patent claims | no reference |
| **Termination** | Terminates if licensee copies, modifies, sublicenses, links with or distributes in violation of the License | Terminates if uncured breach.<br>Terminates if licensee asserts patent infringement against contributor relating to the free software, unless licensee agrees to pay royalty for free software or drop the claim<br>License rights from a participant terminate if licensee asserts patent claim against the participant for *anything* (effective as of first use) | no reference |

## 2. AFFERO GENERAL PUBLIC LICENSE

Version 1, March 2002

Copyright © 2002 Affero Inc.
510 Third Street - Suite 225, San Francisco, CA 94107, USA

http://www.affero.org/oagpl.html

* * *

[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

### 3. The Apache Software License

```
/* ====================================================================
 * The Apache Software License, Version 1.1
 *
 * Copyright (c) 2000 The Apache Software Foundation.  All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution,
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowledgment may appear in the software
itself,
 *    if and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Apache" and "Apache Software Foundation" must
 *    not be used to endorse or promote products derived from this
 *    software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *    nor may "Apache" appear in their name, without prior written
 *    permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
```

```
 * SUCH DAMAGE.
 * ====================================================================
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation.  For more
 * information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 *
 * Portions of this software are based upon public domain software
 * originally written at the National Center for Supercomputing
Applications,
 * University of Illinois, Urbana-Champaign.
 */
```

## *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**.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

**2. Grant of Copyright License**. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

**3. Grant of Patent License**. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

**4. Redistribution**. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

a. You must give any other recipients of the Work or Derivative Works a copy of this License; and

b. You must cause any modified files to carry prominent notices stating that You changed the files; and

c. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

d. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

**5. Submission of Contributions**. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

**6. Trademarks**. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

**7. Disclaimer of Warranty**. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

**8. Limitation of Liability**. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

**9. Accepting Warranty or Additional Liability**. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

## *APPENDIX: How to apply the Apache License to your work*

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner] Licensed under the Apache
License, Version 2.0 (the "License"); you may not use this file except
in compliance with the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0 Unless required by
applicable law or agreed to in writing, software distributed under the
License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for
```

the specific language governing permissions and limitations under the License.

## 4. CDDL

## COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

http://www.opensource.org/licenses/cddl1.php

- **1. Definitions.**
  - **1.1. Contributor** means each individual or entity that creates or contributes to the creation of Modifications.
  - **1.2. Contributor Version** means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.
  - **1.3. Covered Software** means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.
  - **1.4. Executable** means the Covered Software in any form other than Source Code.
  - **1.5. Initial Developer** means the individual or entity that first makes Original Software available under this License.
  - **1.6. Larger Work** means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.
  - **1.7. License** means this document.
  - **1.8. Licensable** means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.
  - **1.9. Modifications** means the Source Code and Executable form of any of the following:
    - **A.** Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;
    - **B.** Any new file that contains any part of the Original Software or previous Modification; or
    - **C.** Any new file that is contributed or otherwise made available under the terms of this License.
  - **1.10. Original Software** means the Source Code and Executable form of computer software code that is originally released under this License.
  - **1.11. Patent Claims** means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

- **1.12. Source Code** means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.
- **1.13. You (or Your)** means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, You includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, control means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

- **2. License Grants.**
  - **2.1. The Initial Developer Grant.**

    Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

    - **(a)** under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and
    - **(b)** under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).
    - **(c)** The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.
    - **(d)** Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.
  - **2.2. Contributor Grant.**

    Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

    - **(a)** under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications

created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

- **(b)** under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).
- **(c)** The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.
- **(d)** Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

- **3. Distribution Obligations.**
  - **3.1. Availability of Source Code.**

    Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

  - **3.2. Modifications.**

    The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

  - **3.3. Required Notices.**

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

- **3.4. Application of Additional Terms.**

  You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

- **3.5. Distribution of Executable Versions.**

  You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipients rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

- **3.6. Larger Works.**

  You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

- **4. Versions of the License.**
  - **4.1. New Versions.**

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

- **4.2. Effect of New Versions.**

  You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

- **4.3. Modified Versions.**

  When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

- **5. DISCLAIMER OF WARRANTY.**

  COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN AS IS BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

- **6. TERMINATION.**
  - **6.1.** This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.
  - **6.2.** If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as Participant) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.
  - **6.3.** In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.
- **7. LIMITATION OF LIABILITY.**

  UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTYS NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

- **8. U.S. GOVERNMENT END USERS.**

  The Covered Software is a commercial item, as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of commercial computer software (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and commercial computer software documentation as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

- **9. MISCELLANEOUS.**

  This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdictions conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

- **10. RESPONSIBILITY FOR CLAIMS.**

  As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

**5. CDDL Changes Summary**

***Common Development and Distribution License (CDDL) Description and Rationale***

http://www.sun.com/cddl/CDDL_why_details.html

## Executive Summary

We have carefully reviewed the existing OSI approved licenses and found none of them to meet our needs, and thus have reluctantly drafted a new open source license based on the Mozilla Public License, version 1.1 ("MPL"). We do appreciate the issue of license proliferation, however, and have worked hard to make the Common Development and Distribution License ("CDDL") as reusable as possible. Additionally, we have attempted to address the problems we perceived in existing open source licenses that led us to conclude that reusing those existing licenses was impractical.

We chose to use the MPL as a base, since it already had a number of attributes we were looking for in a license:

- Requirement that the source of modifications be made available; this helps encourage code reuse and a vibrant community.
- Requirement that the source of modifications be shared under the open source license; this helps encourage that open source remains open source.
- Ability to distribute executables under a different license; this gives businesses more flexibility in licensing their derived works.
- A "files based" definition of modifications and covered software; this makes it easy to combine code with code distributed under different licenses.
- Explicit patent license; this makes it easier for businesses to use the software without concerns over patent rights.
- Provision for termination of rights in response to patent claims (i.e. "patent peace" provision); this helps discourage patent litigation amongst the community, which we felt to be A Good Thing.

The MPL came closest to meeting our needs. We felt that it had a number of issues, though, which prevented us from simply using that license or one of its variants.

Once we embarked down the path of creating a new license, we focused on certain high level goals. We wished to create a license that was simpler, less burdensome for contributors, clear and consistent in the use of terms and language, and that was as reusable and general as possible. In addition, we made a number of specific improvements, a few of which are highlighted here.

- Fixed the "Effect of New Versions" problem. We added an option to make Covered Software available under a specific version of the license, rather than allowing the use of future license versions. This change was made to make the license more reusable by others: it addresses the concern that the license steward could change the terms of the license in ways that are not compatible with a community's (and the Initial Developer's) values and objectives.
- Focused the "patent peace" provisions to cover only software released under this license. We felt that this would make the license more acceptable to a diverse community of contributors, whether large or small. We strengthened the the

penalties of the remaining provision, however, since we believe that patent peace has an important role to play in open source licenses.
- Simplified many of the Required Notices (or in a few cases, removed the requirement) since they seemed overly specific and burdensome.
- Clarified the definition of Modifications, to make it easier for readers to understand what is covered by the license and what is not.

## General Changes made throughout the license

What follows is a detailed description of changes from the MPL. Terms defined in the CDDL are used whenever possible; these are indicated with capital letters. All section references are to the CDDL unless otherwise indicated.

We made a number of general changes, which are propagated throughout the license. These include:

- Changed use of the term "Code" to "Software". This was done to ensure that it was clear that this license can be used for both Source Code and Executables, not simply Source Code.
- Eliminated the definition of "Commercial Use". This term was thought to be misleading, since software may be distributed for both commercial and non-commercial purposes, and we didn't want anyone to see the definition and conclude that the license makes a distinction between commercial and non-commercial use. Since this definition was only used in one place (2.2(c)), we simply eliminated the definition, and embedded the descriptive text in the place where the definition had been used.
- Wherever software distribution is mentioned, added the phrase "or otherwise makes available" to cover passive types of distribution, such as with ASPs.
- Eliminated the definition of "Electronic Distribution Mechanism"; this was no longer necessary, since we simplified the requirements around how Source Code is to be made available.
- Clarified wording related to the file-based approach to Modifications and Covered Software. We wanted to make it very clear to the community what is covered and what is not.
- Added option of making Covered Software available under a specific version of the license, rather than allowing the use of future license versions. This change was made to make the license more reusable by others.
- Eliminated Exhibit A, in order to simplify the process of contributing software under the license.

## Specific Changes

Sections that have not changed (except for numbering) are not listed.

Name: changed to a generic name to allow (and encourage) use of the license by groups other than Sun.

Removed 1.0.1 of the MPL (Commercial Use), as mentioned under "General Changes."

1.1. Contributor: Clarified that an individual can be a Contributor.

1.2. Contributor Version: Added an "if any" clause, to clarify that the Contributor Version may be derived from the Original Software without any prior Modifications by other Contributors. Also changed "Code" to "Software," as mentioned under General Changes.

1.3. Covered Software: Clarified wording regarding file-based approach to Modifications and Covered Software, as mentioned under General Changes.

Section 1.4 of the MPL (Electronic Distribution Mechanism) was deleted; it is no longer needed due to simplified language regarding how source code for modifications must be made available. See 3.1 (Availability of Source Code).

1.4. Executable: Changed "Code" to "Software", as mentioned under "General Changes."

1.5. Initial Developer: Exhibit A notice has been removed, so the definition of Initial Developer had to change as a result. The new definition is based on the individual or entity that initially releases the Original Software.

1.6. Larger Work: Changed "Code" to "Software," as mentioned under "General Changes."

1.9. Modifications: Clarification of files-based approach, as mentioned in "General Changes." Specifically, removed "substance or structure of" phrase to avoid confusion, because it suggests that the definition of Modifications might go beyond modified files to something more.

Changed "Code" to "Software," as mentioned under "General Changes."

Added 1.9(c), which takes into account the common practice of contributing new source files to an open source distribution.

1.10. Original Software: Removed Exhibit A references, and extended to cover both Source Code and Executables. Also changed "Code" to "Software," as mentioned under "General Changes."

1.12. Source Code: Simplified the definition; the details regarding scripts, interface definition files, etc., didn't seem necessary and are not specified in other common licenses (e.g., CPL and OSL).

1.13. You: Since Section 4.2 (6.2 in the MPL) has been changed to allow the Initial Developer for a particular project to specify use of a specific version of the License (excluding future versions of the License), the "future versions" clause was removed here.

2. License Grants: Changed title to make it clear that the license grant is for Executables as well as Source Code. In the MPL, it was unclear whether the license grant covered Executables.

2.1. Initial Developer Grant: Clarified language to make sure that those using the license grant to modify and redistribute software under this license would also be bound to make the Source Code available. Moved "subject to third party intellectual property claims" so that the structure of the license grant in 2.1 was the same as in 2.2.

2.1(a). Changed "Code" to "Software," as mentioned under "General Changes."

2.1(b). Changed "Code" to "Software," as mentioned under "General Changes."

2.1(c). Added "otherwise makes available" as mentioned in General Changes. Changed "Code" to "Software," as mentioned under "General Changes."

2.1(d). The "separate from the Original Code" clause is ambiguous, and was felt by some to be inconsistent with other aspects of the license grant and definitions that specify "portions thereof". In order to avoid confusion, we chose to simply eliminate this clause. Also changed "Code" to "Software," as mentioned under "General Changes."

2.2. Contributor Grant: As in 2.1.

2.2(a) As in 2.1(a).

2.2(c) Removed "Commercial Use". Replaced with "distributes or otherwise makes available"; this matches the wording in 2.1(c). See "General Changes" for a more detailed explanation.

2.2(d) As in 2.1(d).

3. Distribution Obligations: This section was substantially reorganized, including some simplifications and deletions. See below for details.

3.1. Availability of Source Code: This section is based on Sections 3.1 and 3.2 of the MPL. We tried to merge the various obligations about Source Code into one section for clarity. Aside from rewording due to the restructuring of the section, the primary change is to the requirements around Source Code availability. Since access to Source Code is a critical requirement for any open source community, we felt that it was important to make it easy for community members to get access to the Source Code in a usable form.

The MPL requires that those making (and distributing) Modifications need only make the source for those Modifications available, which might result in the need to assemble the source changes from a number of different contributors to get a source base that matches a particular distribution. Instead, we chose to require that anyone distributing Covered Software under this license make the Source Code available for all Covered Software they are distributing (rather than just the Source Code for their Modifications). Note that if someone is just distributing a component of the Covered Software, only the Source Code for that component needs to be made available.

In addition, the specific mechanism and timing requirements for making Source Code available and references to "Electronic Distribution Mechanisms" were removed and replaced with notions of "reasonable" and "medium customarily used for software exchange."

We also removed the reference to the "future version of the License" since Section 4.2 of CDDL (Effect of New Versions) is different from Section 6.2 of MPL (Effect of New Versions). See "General Changes" for a more detailed explanation.

3.2. Modifications: This section is based on Sections 3.1 and 3.4(c) of the MPL. The required notices in the MPL regarding third party claims and patents (formerly in MPL Section 3.4(a) and 3.4(b)) have been eliminated; they seemed overly burdensome and likely to prevent wider acceptance of the license by by the community. Additionally, none of the other major open source licenses (e.g., GPL, BSD, CPL, OSL) require such disclosures.

3.3. Required Notices: This section is based on Sections 3.3 and 3.5 of the MPL, though the form by which a Contributor identifies Modifications is no longer specified. References to Exhibit A were also removed. This was done because the requirements of the MPL seemed overly specific and burdensome; even the Mozilla community doesn't seem to consistently adhere to this practice. Since failure to follow requirements of the license can have significant legal repercussions, it seemed best to be pragmatic and leave the details of how notice is provided to the community and its developers. We note that our requirement is similar to that of other open source licenses (e.g., CPL, OSL).

A requirement that you "may not remove or alter any copyright, patent or trademark notice contained within ..." was also added; this matches similar language in the OSL.

3.4. Application of Additional Terms: This section is from the next to last sentence of MPL Section 3.1 ("Application of License") combined with the latter half of MPL Section 3.5 ("Required Notices".) We merged these two sections as part of the overall reorganization of Section 3, in an attempt to simplify and group things more coherently. We also added "Covered Software" to clarify that the obligations related to the Source Code of the Covered Software. Also fixed a typo present in the MPL.

3.5. Distribution of Executable Versions: This section is from MPL Section 3.6 ("Distribution of Executable Versions".) We removed the redundancies related to the required notices. Also, "Code" was replaced with "Software" (as mentioned under General Changes), and the reference to MPL Section 3.2 and its specific required notices was also removed.

3.6. Larger Works: Replaced "Code" with "Software," as mentioned in "General Changes."

Sections 4 and 5 of the MPL ("Inability to Comply Due to Statute or Regulation" and "Application of this License") were seen as unnecessary or redundant, and have been removed from the CDDL.

4.1. New Versions (of the License): Added concept of a license steward, in order to make the license reusable by others.

4.2. Effect of New Versions: The MPL's option of allowing Contributors to use new versions of a license, while often quite useful (as it makes it easy to improve the license without relicensing all covered code), can cause problems when the Initial Developer is not also the license steward because the license steward could change the terms of the license in ways that are not compatible with the community's (and the Initial Developer's) values and objectives.

The approach we have taken is from the GPL; the CDDL allows an Initial Developer to decide whether they wish to license their Original Software solely under a specific license version, or allow the use of a later version of the same license as well. This ensures that other communities can use the license.

4.3. Modified Versions: Retitled this subsection (in MPL, Section 6.3 was called "Derivative Works") to clarify that this is about modifications to the license, not the code. Also simplified the requirements. While we do not want to contribute to a proliferation of open source licenses, we want other businesses and communities to be able to use this license as the basis for future licenses if industry or business needs warrant.

5. Disclaimer of Warranty: Replaced "Code" with "Software," as mentioned in "General Changes."

6.1. Removed the sublicense statement, since it was redundant with a similar statement in 6.4.

6.2. Changed trigger from initiation of litigation to assertion of a patent infringement claim, for clarity. Also, created an inline definition of Participant Software to account for the fact that the claim might be against either a Contributor Version or against the Original Software and used this definition throughout the section. Clarified a potential ambiguity by adding "directly or indirectly" on line 6 in the first sentence, so that if your

rights from this license are terminated due to your assertion of a patent infringement claim, you cannot reacquire those same rights indirectly. Removed the royalty provision, which was seen as confusing and unnecessary. Broadened the penalty to terminate rights from any Contributor (and the Initial Developer), not just the Participant.

We removed MPL's 8.2(b), which revoked license rights if patent claims are made against **any** product of a Participant, not just code released under this license. We're trying to build a community of diverse contributors, large and small, including commercial contributors, and felt that this section would be a hindrance to commercial adoption. However, we believe that "patent peace" has an important role to play in open source licenses, which is why we have strengthened the provisions in 6.2(a) and 6.3.

Removed MPL's 8.3, since this no longer made sense after removal of the royalty provision in 6.2.

6.3. As in 6.2(a), clarified a potential ambiguity so that if your rights from this license are terminated due to assertion of a patent infringement claim, you cannot reacquire those same rights from a distributor. Removed the parenthetical "(excluding distributors and resellers)", which was seen as confusing and unnecessary.

7. LIMITATION OF LIABILITY: Changed "Code" to "Software," as mentioned in "General Changes." Also added "lost profits" as a clarification to protect Contributors and other licensees.

8. U.S. GOVERNMENT END USERS: Updated to reflect the current U.S. Government regulations and to avoid any ambiguity regarding the application of any other regulations. Also changed "Code" to "Software," as mentioned under "General Changes."

9. MISCELLANEOUS: To make the license more general, we removed the requirement that the License be governed by California law provisions, and we removed the restriction that litigation be subject to the jurisdiction of the Federal Courts of the Northern District of California. Rather, choice of law, venue and jurisdiction can now be specified in a notice in the Original Software. Added last sentence to clarify the party responsible for complying with export control obligations.

Deleted Section 13 of the MPL because it was confusing and unnecessary.

Exhibit A from the MPL was removed, since 3.3 no longer requires a specific format for the notice of modifications.

### 6. Microsoft IP Indemnification

## Microsoft Intellectual Property Indemnification

Published: November 10, 2004

http://www.microsoft.com/windowsserversystem/facts/indemnification/policy.mspx

## *Helping Protect End-User Customers With Strong Intellectual Property Protection*

Companies understand the importance of protecting themselves from legal risk. Many have asked Microsoft about ways they can reduce their risk of being caught in the middle of an intellectual property legal dispute.

Software today is built not just on a complex array of technical code but also on complicated sets of intellectual property rights. Sometimes, those rights need to be acquired from a number of third parties. Because of this, software is occasionally subject to allegations of intellectual property infringement.

By licensing software that comes with strong end-user intellectual property protection, individuals and organizations help shield themselves from costs commonly associated with being involved in these types of disputes.

Intellectual property protection is a valuable benefit Microsoft offers end users of its server and desktop Microsoft software. If someone alleges that your end use of this Microsoft software infringes their intellectual property rights, then Microsoft will help protect you from legal costs and liability associated with the dispute.

Not all software companies offer intellectual property protection. Few offer the same high levels of protection provided by Microsoft.

Microsoft's intellectual property protection:

- **Is provided to end users of Microsoft's server and client software offerings.** Microsoft offers the same high level of intellectual property protection for all end users, regardless of whether they have acquired their license through one of Microsoft's volume licensing programs or through other means, such as a retail outlet or computer manufacturer.
- **Covers patent disputes, as well as copyright, trademark, and trade secret claims.** Intellectual property disputes can come in many forms. Microsoft's end-user protection includes coverage for most patent, copyright, trade secret, and trademark disputes.
- **Helps protect users from legal costs, without pre-set caps.** In the event of a covered infringement claim against a Microsoft software offering, Microsoft will help protect the end user from the costs of defending the claim and from court-awarded damages.

Interested customers should review the specific language that describes Microsoft's legal commitment by reviewing their Volume Licensing agreements or, if they have acquired Microsoft software through another channel, by reviewing the legal commitment below. If you have a contractual indemnity from Microsoft, you are free to rely on either those

terms or the policy below. You don't have to do anything to indicate your choice. If you ever want to assert the terms below, you can—the choice is yours.

## *Microsoft's Intellectual Property Commitment for End Users of Covered Software*

**1. Microsoft's commitment to defend infringement and misappropriation claims.** This commitment covers any Microsoft software licensed for a fee for your end use, except for embedded (industry or task-specific) software. When we refer to "covered software" within this commitment, we are referring to those Microsoft software offerings only.

For any covered software, we will:

- defend you against any claims made by an unaffiliated third party that the covered software infringes its patent, copyright, or trademark or misappropriates its trade secret, and
- pay the amount of any resulting adverse final judgment against you (after any appeals) or settlement to which we consent.

For purposes of this commitment, the terms "misappropriation" and "trade secret" will have the meanings defined in the Uniform Trade Secret Act.

You must notify us promptly in writing of the claim. You also must give us sole control over its defense or settlement. You agree to provide us with reasonable assistance in defending the claim. We will reimburse you for reasonable out-of-pocket expenses that you incur in providing that assistance. The claim might fall outside the scope of our commitment, but send it to us anyway. We may choose to treat it as if it were covered by this commitment.

Our obligations will not apply to the extent that the claim or adverse final judgment is based on:

(i) your running of the covered software after we notify you to discontinue running due to such a claim;
(ii) the combination of the covered software with a non-Microsoft product, data, or business process;
(iii) damages attributable to the value of the use of a non-Microsoft product, data, or business process;
(iv) your altering the covered software;
(v) your distribution of the covered software to, or its use for the benefit of, any third party;
(vi) your use of our trademark(s) without express written consent to do so; or
(vii) for any trade secret claim, your acquiring a trade secret (a) through improper means;

(b) under circumstances giving rise to a duty to maintain its secrecy or limit its use; or (c) from a person (other than us or our affiliates) who owed to the party asserting the claim a duty to maintain the secrecy or limit the use of the trade secret.

You will reimburse us for any costs or damages that result from these actions.

If we receive information about an infringement claim related to covered software, we may do any of the following, at our expense and without obligation to do so:

- procure the right to continue its use; or
- replace it with a functional equivalent, or modify it to make it non-infringing (including disabling the challenged functionality). If we do that, you will stop running the allegedly infringing software immediately.

If, as a result of an infringement claim, a court of competent jurisdiction enjoins your use of covered software, we will do one of the following, at our option:

- procure the right to continue its use,
- replace it with a functional equivalent,
- modify it to make it non-infringing (including disabling the challenged functionality), or
- refund the amount paid for the infringing software and terminate the license for it.

This commitment provides your exclusive remedy for third-party infringement and trade secret misappropriation claims.

**2. Limitation of liability.** The limitation of liability in your license agreement for the covered software will not apply to our defense and payment obligations under this commitment.

To learn more about Microsoft's end-user intellectual property protection and to compare the coverage provided by Microsoft to that offered by other software companies, please visit http://www.microsoft.com/getthefacts.

## 7. Sun Patent Grants

**Sun Grants Global Open Source Community Access to More than 1,600 Patents**
http://www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.2.html

**Largest Single Grant in Patent History Spurs Software Innovation**

**SANTA CLARA, Calif. - January 25, 2005 - Sun Microsystems, Inc. [NASDAQ: SUNW] today announced the largest single release of patent innovations into the open source community by any organization to date, marking a significant shift in the way Sun positions its intellectual property portfolio. By giving open source developers free access to Sun OpenSolaris related patents under the Common**

**Development and Distribution License (CDDL), the company is fostering open innovation and establishing a leadership role in the framework of a patent commons that will be recognized across the globe.**

"As the largest business contributor to the open source community, Sun has always been an ardent believer in open standards and the open source process going back to the inception of this company," said Scott McNealy, Chairman and CEO, Sun Microsystems, Inc. "The release of more than 1,600 patents associated with the Solaris OS far eclipses any other vendor's contribution. Today represents a huge milestone for Sun, for the community, for developers and for customers."

Sun's goal in offering access to these patents is to help facilitate innovation and help users get new open source products and technologies to market faster without having to obtain patent licenses from Sun. The new approach underscores Sun's belief that license agreements for software are not as significant as the company who stands behind its products. Sun is also addressing current issues and increased scrutiny in U.S. and international patent law which has increasingly granted overly broad patents on abstract processes.

In removing the emphasis on intellectual-property rights as an inhibitor to innovation, Sun is leveling the playing field in key emerging markets and helping to revive an innovation system that is straining under a record number of patent filings globally. More markets are looking for ways to monetize their knowledge economy and patents are becoming the profit center. With growing attention on locking up intellectual property in countries like China - which has seen a five-fold increase in the number of patent filings from 1991 to 2001 - Sun is ensuring that software will be available to open source developers and that progress continues unabated.

"By gaining access to these Solaris OS patents, participants in the open-source community now have a tremendous opportunity to build unique and innovative technologies for a wide range of markets," said Stacey Quandt, Senior Business Analyst, Open Source Practice Leader, Robert Frances Group. "An IP contribution of this magnitude has the potential to deliver exceptional value to developers and strengthens the overall open source community."

Addressing the patent system that is under siege, Sun's pledge of open access reduces the quagmire for developers who previously had to walk through a minefield to avoid infringement and enables them to confidently produce derivative works without fear of reprisal or patent claims.

Radically reducing risks associated with using and developing open source software, Sun is firmly standing behind our products and the worldwide development community. Armed with access to Solaris OS platform intellectual property, OpenSolaris developers and customers alike no longer need patent protection or indemnity from Sun's and other

participants in the OpenSolaris community for use of Solaris-based technologies under the CDDL and OpenSolaris community process.

By releasing the OpenSolaris OS platform under the CDDL, the open source community will immediately gain access to 1,600 active Sun patents for all aspects of operating system technologies that encompass features ranging from kernel technology and file systems to network management, to name a few. Patents for Sun's newest technologies, such as the anticipated Dynamic Tracing technology, will also be available under the open access program.

Historically, Sun has contributed more code to open source initiatives than all other organizations with the exception of UC Berkeley, and remains committed to providing engineering support for Apache, Mozilla, Gnome, OpenOffice, Grid, JXTA, ODSL and other open source projects. Previously, Sun donated the source code of StarOffice software, which drives the OpenOffice suite bundled with most versions of Linux and was awarded a Product Excellence Award at the 2004 LinuxWorld Conference & Expo for Best Productivity/Business Application.

* * *

### 8. MySQL Licensing Policy

MySQL Licensing Policy

http://www.mysql.com/company/legal/licensing/

MySQL AB has two licensing options. No matter which license you choose, the overall business benefit of the MySQL database is a dramatically lower **total cost of ownership**.

The guiding business principle of MySQL is one of fair exchange, or *Quid pro Quo* ("something for something"). From a licensing perspective, we have two different products depending on usage and distribution, though technically they have the same source code.

Thanks to our licensing model, we are capable of producing superior software at a much lower cost than proprietary software vendors. That's why MySQL is the world's most popular open source database. Our licensing options include:

- The **Commercial License**, which allows you to provide commercial software licenses to your customers or distribute MySQL-based applications within your organization. This is for organizations that do not want to release the source code for their applications as open source / free software; in other words they do not want to comply with the GNU General Public License (GPL). For more detail on the Commercial non-GPL License, click here. Or, if you want more information on pricing, click here.

- For those developing open source applications, the **Open Source License** allows you to offer your software under an open source / free software license to all who wish to use, modify, and distribute it freely. The Open Source License allows you to use the software at no charge under the condition that if you use MySQL in an application you redistribute, the complete source code for your application must be available and freely redistributable under reasonable conditions. MySQL AB bases its interpretation of the GPL on the Free Software Foundation's Frequently Asked Questions. For more information on MySQL's Open Source License, click here.

At MySQL AB, we believe in open source / free software and we welcome all initiatives to publish more software under the GPL license, just as we have done.

In their simplest form, the following are general licensing guidelines:

- If your software is licensed under either the GPL-compatible Free Software License as defined by the Free Software Foundation or approved by OSI, then use our GPL licensed version.
- If you distribute a proprietary application in any way, and you are not licensing and distributing your source code under GPL, you need to purchase a commercial license of MySQL
- If you are unsure, we recommend that you buy our cost effective commercial licenses. That is the safest solution. Licensing questions can submitted online for our advice, and we encourage you to refer to the Free Software Foundation or a lawyer as appropriate.

Commercially licensed customers get commercially supported product with assurances from MySQL. Commercially licensed users are also free from the requirement of making their own application open source.

For OEM's, ISVs, corporate, and government users, a commercial license is the proper solution because it provides you with assurance from the vendor and releases you from the strict requirements of the GPL license.

Nevertheless, you can test MySQL under the GPL license and inspect the source code before you purchase a commercial non-GPL license.

We also have created a license exception which enables Free and Open Source software ("FLOSS") to be able to include the GPL-licensed MySQL client libraries despite the fact that not all open source licenses are compatible with the GPL. Read more about the exception.

*Version 4.1, 12 March 2004*

The MySQL AB Exception for Free/Libre and Open Source Software-only Applications Using MySQL Client Libraries (the "FLOSS Exception").

http://www.mysql.com/company/legal/licensing/foss-exception.html

### *Exception Intent*

We want specified Free/Libre and Open Source Software ("FLOSS") applications to be able to use specified GPL-licensed MySQL client libraries (the "Program") despite the fact that not all FLOSS licenses are compatible with version 2 of the GNU General Public License (the "GPL").

### *Legal Terms and Conditions*

As a special exception to the terms and conditions of version 2.0 of the GPL:

You are free to distribute a Derivative Work that is formed entirely from the Program and one or more works (each, a "FLOSS Work") licensed under one or more of the licenses listed below in section 1, as long as:

1. You obey the GPL in all respects for the Program and the Derivative Work, except for identifiable sections of the Derivative Work which are not derived from the Program, and which can reasonably be considered independent and separate works in themselves,
2. all identifiable sections of the Derivative Work which are not derived from the Program, and which can reasonably be considered independent and separate works in themselves,
    1. are distributed subject to one of the FLOSS licenses listed below, and
    2. the object code or executable form of those sections are accompanied by the complete corresponding machine-readable source code for those sections on the same medium and under the same FLOSS license as the corresponding object code or executable forms of those sections, and
3. any works which are aggregated with the Program or with a Derivative Work on a volume of a storage or distribution medium in accordance with the GPL, can reasonably be considered independent and separate works in themselves which are not derivatives of either the Program, a Derivative Work or a FLOSS Work.

If the above conditions are not met, then the Program may only be copied, modified, distributed or used under the terms and conditions of the GPL or another valid licensing option from MySQL AB.

### *1. FLOSS License List*

| License name | Version(s)/Copyright Date |
|---|---|
| Academic Free License | 2.0 |

| | |
|---|---|
| Apache Software License | 1.0/1.1/2.0 |
| Apple Public Source License | 2.0 |
| Artistic license | From Perl 5.8.0 |
| BSD license | "July 22 1999" |
| Common Public License | 1.0 |
| GNU Library or "Lesser" General Public License (LGPL) | 2.0/2.1 |
| Jabber Open Source License | 1.0 |
| MIT License (As listed in file MIT-License.txt) | - |
| Mozilla Public License (MPL) | 1.0/1.1 |
| Open Software License | 2.0 |
| OpenSSL license (with original SSLeay license) | "2003" ("1998") |
| PHP License | 3.0 |
| Python license (CNRI Python License) | - |
| Python Software Foundation License | 2.1.1 |
| Sleepycat License | "1999" |
| W3C License | "2001" |
| X11 License | "2001" |
| Zlib/libpng License | - |
| Zope Public License | 2.0 |

Due to the many variants of some of the above licenses, we require that any version follow the 2003 version of the Free Software Foundation's Free Software Definition (http://www.gnu.org/philosophy/free-sw.html) or version 1.9 of the Open Source Definition by the Open Source Initiative (http://www.opensource.org/docs/definition.php).

### *2. Definitions*

1. Terms used, but not defined, herein shall have the meaning provided in the GPL.
2. Derivative Work means a derivative work under copyright law.

### *3. Applicability*

This FLOSS Exception applies to all Programs that contain a notice placed by MySQL AB saying that the Program may be distributed under the terms of this FLOSS Exception. If you create or distribute a work which is a Derivative Work of both the Program and any other work licensed under the GPL, then this FLOSS Exception is not available for

that work; thus, you must remove the FLOSS Exception notice from that work and comply with the GPL in all respects, including by retaining all GPL notices. You may choose to redistribute a copy of the Program exclusively under the terms of the GPL by removing the FLOSS Exception notice from that copy of the Program, provided that the copy has never been modified by you or any third party.

Version 0.3, 11 February 2004

# Working With Open Source Software Compliance Management

*Karen Copenhaver, Black Duck Software*

ABSTRACT:

Acknowledging that the benefits of open source software ensure its increasingly widespread deployment, software developers and CIOs are implementing Software Compliance Management processes to identify the source of the software in their code base and to ensure compliance with the license obligations applicable to the code in use. The panel will discuss the elements of an ideal process that: enables developers to embrace open source; identifies issue early in development cycle; facilitates efficient engagement of legal counsel in the development process to address issues as they arise; and efficiently tracks compliance with license obligations from development through deployment or distribution.

BIO - Karen Faulds Copenhaver
Executive Vice President and General Counsel
Black Duck Software, Inc.
Waltham, MA
http://www.blackducksoftware.com
Karen Copenhaver is the Executive Vice President and General Counsel of Black Duck Software. Prior to Black Duck, Karen was a partner at Testa, Hurwitz & Thibeault LLP in the Patent and Intellectual Property Group practicing in the area of technology licensing. Karen was also a partner with Brown & Bain. Prior to her firm experience, Karen spent 14 years as inhouse counsel at IBM.

She has served as an adjunct professor at Suffolk Law School and is a frequent speaker on topics relating to strategic alliances and licensing of intellectual property, and in particular, issues relating to open source licensing.

Karen is a graduate of Dickinson School of Law (JD, 1979), is admitted to practice in New York, Vermont, California and Massachusetts.

# Working with Open Source[25] Software Compliance Management

**"It's the alligators that you don't see that will get you."**

***IT Control Objectives for Sarbanes-Oxley***
***IT Governance Institute***
***www.itgi.org***

Virtually all companies that develop software are now working in a "mixed-IP environment." Software is created on ever increasing layers of previous work, mixing internally developed software assets with licensed proprietary components and including important open source elements. The re-use of internally and externally available software components is inevitable and should not be discouraged. Not only does it reduce the time and cost of development, re-use of components that have withstood the tests of time and the many eyeballs of critical peer review provides assurance of quality and reliability. Open source software is often necessary to meet important open standards. Thus we are rapidly moving to a time when "Just Say No" to the use of open source software is no longer a viable option for most companies developing software.

If we are honest with ourselves, we will admit that the "Just Say No" policies of the past were about as effective in controlling the use of open source software as they have been with the prevention of drug use among teenagers. It was convenient to establish an absolute prohibition as the official company policy, and a simple ban worked fairly well as long as there was nothing that could be done to actively monitor or determine compliance. If the source code had been subject to a monetary obligation,

---

[25] I must start with a definition. There are very important distinctions that are drawn with respect to Free and Open Source Software (FOSS), Open Source Software (OSS), Freeware, Shareware, etc. While I acknowledge that these are very important distinctions, in this paper I am using "open source" without initial caps to refer to any software that is made widely available in source code form without requiring payment in advance. The term open source in this context does not say anything about the license terms that may be applicable to its permitted use. It is only useful in referring to software that is easily incorporated into a development effort without triggering the review of contract terms that generally occurs in conjunction with a transfer of monetary consideration.

even a small one, the agreements related to its use would have gone through the procurement or finance departments and ended up on a lawyer's desk for review. But without any required payment, its use easily evaded all internal business controls. As a colleague described the situation, "There was a lot of concern about managing everything that happened in the front office, but it was very easy to ignore the loading dock with the big open door in the back where the source code was being downloaded." When situations are very complex and would require enormous energy to address, it is easier to focus on problems that are more easily managed.

Once I was asked to conduct a seminar for developers at a small company to impress upon them the importance of having every license reviewed prior to downloading any source code from the internet. This was just a reminder of practices and policies that had always been in place at the company. We had a good discussion. There were lots of questions about the enforceability of licenses that did not require signature or that did not have click-through "I accept" buttons. We talked about the GPL and the LGPL and software that was posted on university websites. At the close of the class, while all of the developers were still finishing the excellent cookies that were provided to entice everyone to come, the manager of the department came over and thanked me for coming. Then he added, in a stage whisper that was clearly intended to be heard by everyone in the room, "You know, these fellows can't get everything they need to get done every day and worry about all of this stuff." And I think this sums up where we have been on this issue. There are so many things to worry about in modern life. We are supposed to do so many things that no one, or hardly anyone, actually does. We don't drain our water heaters annually, or turn our mattresses quarterly, or keep monthly photocopies of everything in our wallets. And we can't worry about our kids copying their friend's CDs onto their iPods or downloading obviously pirated copies of videogames. We are bombarded by admonitions and prohibitions. We have to cull through them and allow some to rise to the top while we ignore the rest in order to survive. There are much more

serious issues that sap our time and energy. And aren't there questions about whether these licenses are even enforceable?[26]

Let me pause here and compare this with our experience and practice as lawyers. Do we ever start to write a contract with a clean sheet of paper? Or do we borrow from models that reduce the time and cost involved? Don't we look for samples that come from major suppliers because we value the amount of peer review that we know went into the development of their forms? Don't we cut and paste selected provisions and then focus our time on the higher level process of coordinating the interaction between the individual components to assure that the end product meets the project specifications? We don't spend too much time worrying about the copyright in the contract. Programmers do the same thing, under the same sorts of time and cost pressures.

So why is use of open source software an issue that should rise to the top of our collective conscious and become one of those issues that requires our attention? The short answer is because we now have the technology available to determine compliance. We can no longer assume that no one will ever know. The longer answer would require consideration of all of the recent events, and the regulatory responses to those events, that have caused our society to be much more attuned to compliance concerns. It is helpful to consider who might ask the questions and what the result might be if the answers were not satisfactory. Public companies are performing extensive due diligence reviews relating to the use of open source software prior to acquiring the intellectual property assets of a target company. Strategic partners want more than words of assurance that the companies that they are allying with are in compliance with their obligations. Customers are demanding broader intellectual property warranties and indemnities. Employees want to know that their employer is fulfilling its obligations to the open source community. Investors want to know that there are no skeletons in the closet that will affect shareholder value.

[26] Although I still hear questions regarding the enforceability of open source licenses, lawyers who are familiar with the issues assume that the licenses are enforceable and counsel their clients accordingly.

Thus, we have a new field of endeavor known as Software Compliance Management. Over time, it will become a normal part of the DNA of every company that uses software. In general, Software Compliance Management means:

- Knowing what is in the code base and controlling the introduction of licensed materials into the code base;
- Knowing what obligations are related to the use of licensed materials, knowing whether the license obligations for combined components are compatible, and managing fulfillment of those obligations; and
- Managing this process from product architecture all the way through deployment or distribution.

Although the immediate focus is on Software Compliance Management relating to the use of open source software, the same asset management issues arise in relation to proprietary code licensed from third parties. The trend is toward delivery of more source code, making the inadvertent migration of proprietary source code a very real possibility. The trend is also toward re-use of internally developed components, which requires effective methods of ensuring that the use of the asset is consistent with overarching business objectives. And the replication of components across divisions and business units must be carefully tracked for both support and security purposes. All of these issues are addressed through Software Compliance Management because they all start with knowing what is in the code base and managing compliance with restrictions on use.

So where do companies begin? The first step most companies take is to implement due diligence procedures for acquisitions of software assets. This makes sense for a number of reasons. First, it is wise to make sure that you are not inadvertently adding to the problems that will have to be addressed while you are preparing for your internal review. Second, there is a lot of learning that takes place in the due diligence process, and that learning will make the internal review go much more smoothly. Third, the education process is much less painful when you are not dealing with your own products or people. It is difficult to be objective and to use a presented issue as an opportunity to reach a consensus on proper policies and procedures going forward when

that issue is tied to specific members of the management team and existing customer commitments.

Companies and their counsel use many different methods of conducting due diligence. In most ways, due diligence relating to use of open source software is the same as due diligence relating to any other issue. Draft representations and warranties and due diligence checklists are used to elicit information. Oral interviews delve more deeply into the issues that merit further attention. But, in addition to these common due diligence procedures, there are methods of inquiry that are uniquely suited to reviewing the contents of the code base.

There is a broad range of code review techniques available to acquirors. Some companies routinely engage forensic computer consultants to review a target's code base. These experts are highly skilled in the art of programming and scour the code line by line for clues as to origin and dependency. Many consultants and companies have developed their own internal tools that scan code, usually in source code form, for indications of copied code. The scan looks for words in the text such as "copyright" or "license" or "free" or "open" or the names of individuals, such as Richard Stallman, who are closely connected with the development of free software. The fact that these scans so often produce interesting results is a good indication of how poor the communication has been between management and developers. Executives pronounce with great assurance that there is no third party code in the code base prior to being handed a long list of license references produced by the scan. The developers, of course, know what is in the code base. And the results of these scans make it obvious that it did not even occur to the developers that there was any reason to worry about or to remove the license indications.

Newer automated tools analyze the code at a much deeper level and pick up copying even if all of the textual clues have been removed. These tools look for matches in the source code or indications that binary files have similar characteristics. These tools look below the component level and pick up the equivalent of copied sentences, and even sentences that have copied material that has been reordered or modified. The results of the scan can present very difficult questions for consideration. When does merger of

function and expression occur in source code such that we are no longer looking at copyrightable subject matter? What factors must be taken into consideration in drawing a conclusion that the copied material is de minimus? And how much investigation is required before you can accept a developer's representation that the identical code resulted not from copying but from the limited number of possibilities available to represent a given function?[27]

Once the code review has been completed, companies are faced with difficult decisions regarding remediation. If the compliance failure is limited to replacing limited amounts of non-critical but obviously copied code, most companies feel comfortable proceeding with the transaction and letting the acquiring entity implement a remediation plan. The cost of the remediation efforts or the loss of functionality due to the removal of code may be a factor in valuation, but the delay in closing the transaction, which is often much more problematic, is minimized. In some situations, the companies agree to a remediation plan to be completed prior to closing and, occasionally, use escrows or holdbacks to secure the remediation obligations. If the compliance failure is more serious, however, difficult questions arise as to who will perform the remediation work. In at least one situation that I am familiar with, the entity acquiring the asset was very concerned about either performing the remediation work itself or directing the remediation effort prior to closing. Even if it were possible to avoid direct infringement, there seemed to be no way to avoid vicarious liability or being held responsible as a contributory infringer. The resulting delay eventually caused the parties to pursue other options. Obviously, it is far better for a company that might be a target in an acquisition

[27] Another method of due diligence that I have often seen is the preparation of a Certificate of Originality. This is a list of detailed questions to be filled out, usually by the developers who originally contributed to the code. The downside to this approach is that the answers to the questions are all arguably admissions and it is difficult to argue that any of the preliminary draft answers would be covered by privilege. It is very difficult to prepare answers to detailed questions in a manner that is absolutely accurate for all purposes and in all contexts, but this is particularly so when the questions pertain to very complex subject matter and cover a time period that may be months or years in the past. If very sophisticated intellectual property attorneys with deep technical expertise are involved in the development of the responses at the same level they would be involved in the creation of litigation related materials, this is an effective method. If the certificates are handed to developers for completion, I think this method of gathering information can be damaging to everyone involved - the target company as well as the eventual acquiror who will be forever bound by the admissions as a successor in interest.

or subject to due diligence review by a strategic or OEM partner to conduct their own code review and complete all necessary remediation in advance.

The level of sensitivity an acquiring company will have towards open source code that is lurking in the code base greatly depends on the intended use of the software assets. If the acquisition is for the purpose of temporarily filling a hole in the company's product line and the assets will remain in the subsidiary and not be rolled up into the parent, then there may be relatively little sensitivity to problems that might be disclosed through an extensive due diligence. The intent may be for the acquired code to be replaced in its entirety by code developed as an integral part of the company's existing product line. This is obviously not an ideal valuation scenario for the target company, but it will result in less focus on code scans or other methods of detailed due diligence review. If the intention is to merge the acquired code into the acquiring company's existing product line and combine it with other valuable software assets, then the sensitivity will be extremely high and the discovery of even minor problems may be sufficient to tip the balance toward "make" in a "make versus buy" analysis.

It is important to remember that it is not only what is found in the code scan that is important, but also how the discoveries jive with the representations made by the executives involved in the transaction and what the discoveries say about the effectiveness of business controls in place during development. At some point, confidence in the management team will be reduced if there are too many surprises.

If a company is actively engaged in due diligence of all acquired code, at some point it becomes obvious that these same techniques ought to be applied to assess the company's own code hygiene. This almost always starts with an assessment of the code base that is very similar to the assessment performed on code to be acquired, but it should not end there. Once the initial assessment is completed, the real value comes in the implementation of automated, auditable controls that enable the company at any time to answer questions regarding the content of its code base and its compliance status. An ideal process will have many benefits, including the following:

- Enable the company to embrace open source software by setting the proper tone regarding compliance;
- Reduce the time and cost of development;
- Identify issues early in development cycle when good options are still available;
- Facilitate efficient engagement of legal counsel in review process by gathering all available information automatically; and
- Efficiently track compliance all the way through to product deployment or distribution.

For most in-house attorneys, the prospect of conducting an assessment of their code base causes a certain queasy feeling. They know that copying excellent software off of the internet to meet development schedules is extremely enticing and that, in the absence of technical controls, the company has been relying entirely on the sensitivity and good will of developers and their managers to impose discipline on the development process. In-house counsels also know that the process of conducting the assessment will be neither easy nor straightforward. All investigations result in the creation of questionnaires, interim processing and preliminary conclusions that contain inaccurate information. It takes time to get to the point where enough information is available and there is sufficient corroboration of all important data to feel confident that the results should be considered final. This healthy hesitation to bring an investigation to a close too quickly is particularly wise when the investigation involves technical data. Cross-discipline communications between experts in a technical field and lawyers are often are conducted through the use of analogies. The analogy may work very well to assist the lawyer in understanding one particular aspect of the problem, but the lawyer often is drawn to extend the analogy to explain related matters for which the analogy does not hold. For in-house counsel at public companies, in particular, creating a lot of preliminary information regarding possible compliance failures requires consideration of complicated and exacting disclosure obligations.

Thus many attorneys consider whether there are steps that should be taken to preserve a possible claim of attorney client privilege over this investigatory phase. Although the availability of privilege will vary from state to state and will, of course, not protect the company from the facts, steps taken to preserve the possibility of a claim of privilege are wise counsel for reasons unrelated to avoiding disclosure of work product. There are real benefits to considering all of the elements of privilege and managing the investigation within the confines of the privilege analysis. For example, it should be clear in advance who will conduct and who will be involved in the investigation, who has requested the investigation and for what purpose, how the investigation will be conducted, what the schedule for the investigation is and when it will be brought to a conclusion. If a realistic schedule is established that provides for the conduct of a thorough investigation over a six month period, and a date is set for review of the final results with the requesting executive on a date seven months later, it will be much easier to answer questions about why the executive was not made aware of preliminary information that became available three months into the process.

Beginning an internal assessment of source code requires the commitment of valuable resources for as long as it takes to finish the assessment. This is not a project that can be begun and then put on hold. It is important to realize this up front and to make an honest assessment of the resources that will be necessary to follow through all the way to conclusion. And be realistic about the resources that are identified. It is very tempting to pull the very best development personnel into any remediation effort, but this is probably not the best use of those extremely valuable resources. By identifying the right people, you can limit the number of people involved and recognize the importance of their contribution.[28]

In planning the assessment, it is important to prioritize projects and to avoid assessing the entire code base at once. The first project provides a learning opportunity that enables the following projects to be completed more efficiently. Gather contextual information in advance for the project that is selected for review. For example, you will

[28] Adequate recognition and encouragement is very important. In one situation, a specific developer assigned to perform an extensive remediation project resigned the next day because he concluded that it was a career ending move.

want to understand what parts of the project are shipped to customers and what source code is made available to OEM partners before you begin to review the code. If concerns are raised by the results of the review, you will not want to be seeking this information in an emergency mode that will spread rumors throughout the development organization. The goal is to have a reasonable number of open issues at any one time and to avoid having more issues swirling around than can be addressed efficiently.

The hardest part of the assessment is performing triage. You cannot let the perfect be the enemy of the good and get bogged down in minutiae. Save energy for the issues that demand attention. Stay the course and avoid crying "wolf." The goal has to be a determination in good faith that the company is in compliance. The goal cannot be the eradication of all code that might arguably indicate some level of familiarity with source code owned by another. At times the proper course is determined based on fine distinctions regarding what is and what is not copyrightable subject matter, what is permitted as fair use or whether a use can fairly be deemed both quantitatively and qualitatively de minimus.

In planning the assessment, it is important to anticipate employee concerns and to limit the involvement to those employees who are necessary to the process. Many employees are very sensitive to fulfillment of obligations to the open source community and rumors regarding compliance issues raise issues not only regarding the company's legal obligations, but also its moral obligations and commitment to a fundamental sense of fairness. And remember that laws protecting employees who serve as whistleblowers may be applicable to allegations of copyright infringement.

The most challenging phase of the assessment is remediation. If you are an attorney who does not have deep technical knowledge, it is wise to line up outside counsel in advance who has sufficient expertise to talk with developers directly regarding remediation options. Ideally the counsel will have an historical perspective on open source, including knowledge of those issues on which community perspectives have been explored, will know where valuable repositories of information are located, and will have contacts in the community that could be called upon for clarification anonymously.

Remediation plans may include the removal of code or the release of source code, but they may involve other tasks as well. Sometimes simply re-architecting the interaction between product components is sufficient to achieve compliance. In many cases, the authors of open source projects have been contacted in order to obtain the rights necessary to continue shipment. Although remediation plans vary, they all require enormous cooperation between the development and legal communities.

In the development of a remediation plan, it is essential to win the right hearts and minds before charting a course of action. Avoid mandating any course of action constructed entirely by lawyers operating in a vacuum. The technical options will only be fully explored when there is a consensus that the issue must be addressed. Remember that any remediation plan always involves the painful re-direction of valuable resources. You should expect a firm pushback to any proposal and come prepared to convince everyone involved that remediation is required. And, once again, remember that developers will not want to be involved in any remediation effort unless they sense that they will be fully supported and that the value of their contribution will be fully recognized.

The most difficult remediation issues will be those that extend to software that has been distributed to customers or other third parties. Do you have an obligation to notify the third party? Is there a method for ensuring that the third party receives and uses replacement code following remediation? Has the third party further distributed the non-compliant code to others? Is the third party obligated to install updates that are delivered? Do the warranties or indemnities in your agreement require any action? Thorough preparation in advance of any internal or external discussion is extremely important and the focus should remain on compliance going forward.

When the assessment has been completed, bringing the process to real closure requires enormous discipline. If at all possible, report back to the executive that requested the review on the on pre-established date. Avoid previews or interim reviews that will leave the executive with information regarding compliance concerns that is not final and is unaccompanied by fully developed recommendations for remediation.

Generally, there will be no time or need to review minor remediation work that has been completed and the report will be limited to those issues requiring continued remediation efforts. Discussion of specific, fully developed recommendations for remediation and alternatives will avoid any hiatus during which management knows of risk but has no plan in place to address it.

After completion of a compliance review of the existing code base, the company will want to maintain control over the introduction of licensed materials into the code base in the future. The first step in developing an effective compliance program is to evaluate existing processes. There is generally at least some formal or informal process in place already that should be understood before any process is imposed from above. Identify opportunities to capitalize on good things that are already happening, assess attitudes among the development managers, and know who the champions are that will make a successful program possible. Gather the existing lore regarding compliance. Honest communication requires an acknowledgement of existing problems and challenges.

It is wise to involve your own financial compliance experts. CFOs and auditors are experts in the implementation of internal controls. They will be familiar with materials specifically prepared by commissions and organizations that have been given the mission to define, review and assess the sufficiency of internal control programs. (See, for example, the website for IT Governance Institute *www.itgi.org*.)

There are several approaches to pre-approval of the use of open source software that are commonly employed even in the absence of any formal, audit-ready compliance program. Some companies request developers to identify in advance those open source projects that they would like to use so that an identified individual or individuals – either in the legal department or in development – can analyze the project for appropriateness. The review conducted is often quite extensive, looking beyond the license that has been declared for the open source project. All information available about licenses that may apply to subcomponents of the project is brought to light and the compatibility of the licenses that apply to these subcomponents is fully explored. In some cases, the author of

an open source project is contacted to confirm licensing terms and to provide some assurance of originality. After the project has been approved, it is placed on an internal website which indicates that it is has been approved for use. This review may be conducted by a committee, which is sometimes called an Open Source Review Board. The creation of an internal group of individuals who are savvy about open source in general and who are knowledgeable about use of open source in the enterprise is an excellent way to communicate an organization's openness to use of open source. It fosters awareness, serves educational purposes; creates formal records, builds internal precedent that improves decision making, and establishes a process to bring issues regarding open source usage to closure.

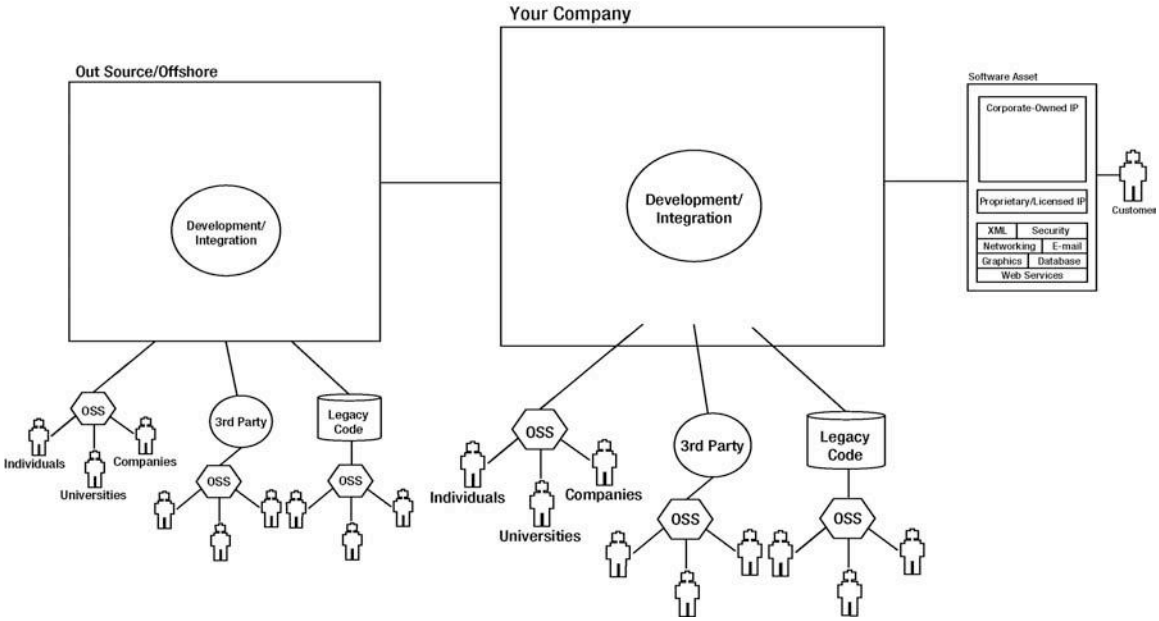
Another common method, but one that has fewer benefits, is to publish a list of approved licenses. Open source software that is made available under selected non-reciprocal licenses is pre-approved and may be freely used in development. This method ignores the fact that many extremely valuable projects are under reciprocal licenses and that these projects can, in some cases, be used without incurring obligations to release source code. It also can encourage "license laundering." Code originally made available under a reciprocal license may have been copied for use in projects made available under non-reciprocal terms. It is also possible, of course, to take code made available under a reciprocal license and repost it elsewhere on the web under an approved license. It is always important to consider, first and foremost, the source of open source software.

A final approach that is more common than one might imagine, is to analyze code origins as part of the final product quality review prior to product release. Not only can this create a log jam, it raises issues at a point in time where there are few good options and enormous pressure to proceed with product release and to try to remediate later. It is far preferable to raise the issues early in the development cycle, as the code is added to the source tree, when many good options remain viable.

To recap, a compliance program usually begins with establishment of a due diligence policy to assess code that is acquired or licensed from third parties. This is following by an assessment of the code base and remediation of compliance issues raised

in the review. When the internal review is complete, an evaluation of existing process is the first step in the development of a process that is tailored to meet the goals and culture of the organization. Finally, a compliance process implementing automated and auditable business controls is rolled out across the enterprise.

In order to fully understand the importance of the use of technology in any compliance program, consider the following diagram. In the current environment, a product offering is likely to be a complex, multi-layered program including many components from different sources – internally-developed legacy code, code obtained from commercial providers, and code obtained from the open source community. Code from each of these sources was developed by many individuals. Some of the contributors are employed by the developer; many are employed by other companies or by universities. All of the code must be tracked from its origins to the end product in order to understand its pedigree.



With respect to outsourcing, contractual provisions are used to establish code pedigree and to make sure that assignments of intellectual property rights are effective under local law. These include representations, warranties and indemnities relating to intellectual property. Nevertheless, concerns persist due to the lack of personal contact with the developers and concerns about different cultural perspectives on the importance of intellectual property rights. The implementation of a technical solution as part of the contracting process can both set the proper tone regarding controls and verify compliance.

Although compliance concerns are often raised in relation to open source software, the same concerns apply to the use or misuse of proprietary software. The amount of open source software that may have leaked into the code base may be greater due to the absence of controls over code that has been available for download without charge for more than a decade, but the same issues apply to the use of proprietary source code that is often readily available overseas.

Software compliance management is just one more business process that reasonable development managers, general counsel, financial officers, auditors and executives will embrace as a solution to the problems presented by a failure to manage the introduction of licensed materials into a code base. An effective program sets the proper tone regarding compliance, raise issues in a timely manner, and improve communications between developers and attorneys by providing immediate access to all of the information that is relevant to resolution.

## SIPLA 2004-2005 Officers and Contacts:

**Co-Chairs:**

Matt Saunders, asaunders@piercelaw.edu
Ronald Sia, rsia@piercelaw.edu

**Secretary:**

Michelle Busque, mbusque @piercelaw.edu

**Treasurer:**

Anthony Del Monaco, adelmonaco@piercelaw.edu

**Representatives:**

Max Green (3L), mgreen@piercelaw.edu
Yandie Fashu Kanu (2L), afashukanu@piercelaw.edu
Sirisha Kalicheti (1L), skalicheti@piercelaw.edu
Justin Lingard (1L), jlingard@piercelaw.edu
Arshad Auf (MIP/LLM), aauf@piercelaw.edu

**Faculty Advisor:**

John Orcutt, Professor of Law
2 White Street, Concord, NH 03301