Will Google Break the GPL?[i]

Open Source Licensing = Confusion

Confusion surrounds compliance with open source licensing. Reactions vary from rigorous compliance to adopting a hands-off approach especially when the open source license at issue is the General Public License (GPL) or a GPL derivative. These licenses popularized the concept of copyleft, which leverages copyright law (providing, in the United States, statutory damages and injunctive relief for infringement of a registered copyright), to promote their philosophy of "freeing" code from intellectual property restrictions. **Even more confusing to those analyzing issues with code covered by open source licenses is the right to make derivative works.** Derivative works can be confusing enough with more traditional art forms but become even more difficult to apply when the subject matter involves computer code. For example, let's say that Company X wishes to utilize a package, licensed under the GPL, with its proprietary codebase. But, Company X doesn't want to trigger the requirement that it open up its proprietary code upon distribution. How does one interpret the concept of a derivative work or a work "based on" code covered by a GPL license?

Sockets …a Possible Answer

The Free Software Foundation (FSF) permits aggregation (separate programs distributed on the same media) but proposes that more intimate forms of combination (e.g., running via the same executable file or linked and running in the same address space) between two programs may result in a derivative work.[ii] But, partially due to the vast array of possibilities when architecting a computer solution, FSF is less certain when it comes to the use of sockets:

> *By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs.* ***But if the semantics of the communication are intimate enough, exchanging complex internal data structures, that too could be a basis to consider the two parts as combined into a larger program.***[iii]

Thus, some conventions have developed based on the concept of utilizing two source programs, one of which is licensed under the GPL.[iv]

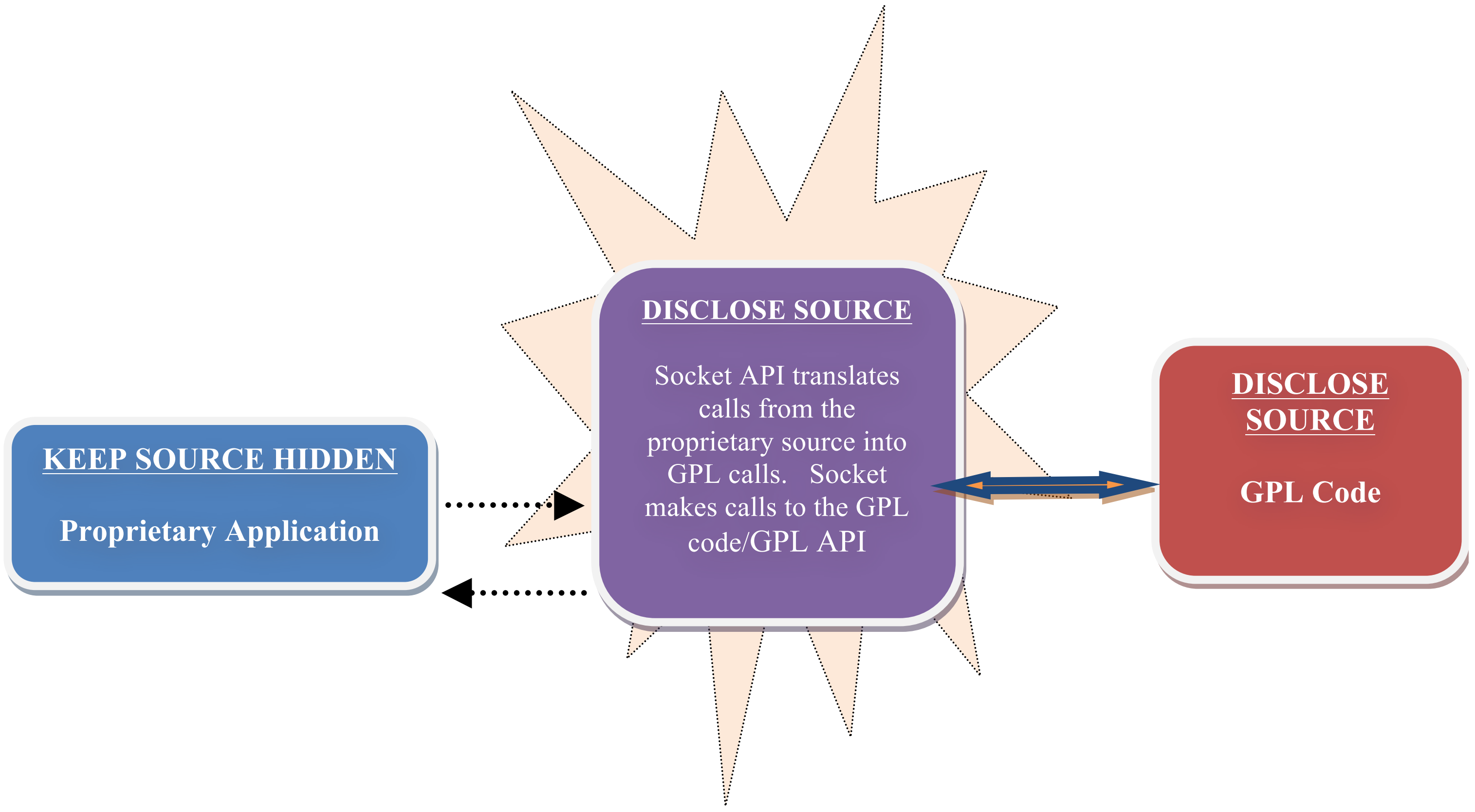
- If the two sources are compiled and statically linked, the resulting executable is a single machine code combination of those two sources. This is considered a derivative work.
- If the two sources are compiled and dynamically linked, the resulting executable may be considered a derivative work if designed to run in a shared address space even though the executables remain separate. It may be necessary to look at the specific use of the GPL-based code and the level of communication between that and the proprietary code. The argument that a derivative work may exist is strengthened if complex data structures are shared/exchanged (e.g., instantiation of objects defined in the GPL code).

Further separating the programs through the use of a **wrapper (a thin layer of code placed between two programs which acts like a translator** to receive a function call from one program and invoke the appropriate function in the second program) may strengthen the argument that no combination occurred if a communication mechanism, such as an socket, is utilized between two computers where one has the compiled GPL code and the other has the proprietary executables stored.

Many copyright theories may come into play here but, given the inherent uncertainty of applying these doctrines, sockets may provide one possible answer to deterring infringement claims for creating a derivative work. **Sockets work, effectively, as a rest stop between a client program and a server program.** A client program sends a request to the socket where it "rests" until it is picked up by the server program. The server program returns the result to the socket where it "rests" until it is picked back up by the client program. Sockets do create certain amount of overhead but defining a clear line between two processing spaces to avoid "contamination" may be worth it. Naturally, the FSF still asserts that **if complex data structures are exchanged then this mechanism may not save you from disclosure requirements triggered by distribution** but it does reduce some fear of contamination and, at a minimum, will help your tech/legal team walk through whether or not an issue may exist.[v] Programmers already use wrappers/sockets as a workaround for the GPL (see the next section on "Traditional Wrappers) but a case (Oracle America v. Google), decided in May 2012, in the Northern District of California, made this solution more comfortable …. at least until Oracle filed an appeal last October.[vi] If the decision is upheld, however, here's how the decision may affect programming practices.

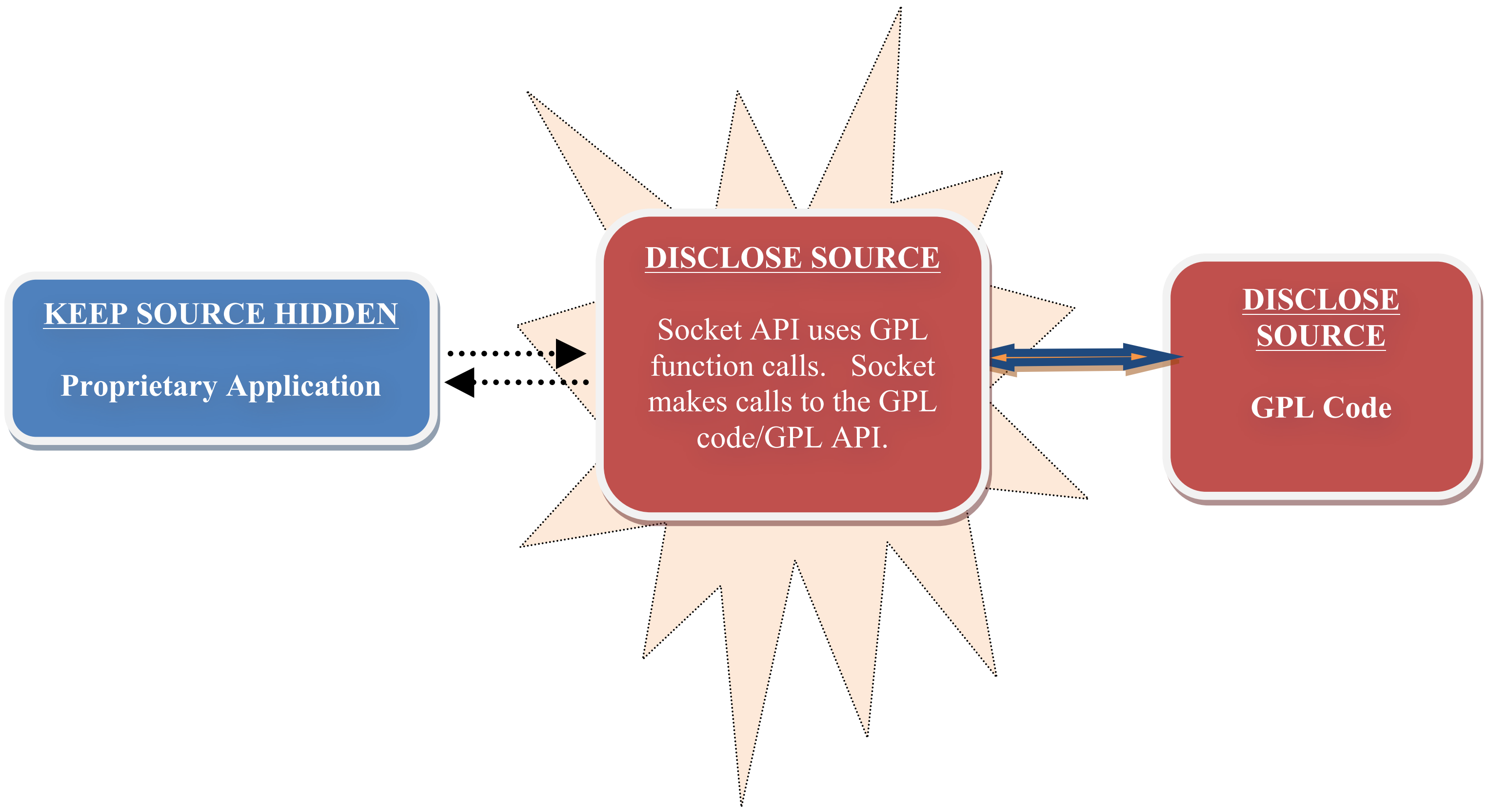
Delving Into Traditional Wrappers

A traditional wrapper includes placing a socket, a method for communication between a client program (e.g., proprietary code) and a server program (e.g., GPL code) in a network. A programmer configures the socket to make calls to the GPL code (the socket and the GPL code may be linked). These may be connected enough that the source to the socket may need to be distributed. The Proprietary Application can send requests via a Socket Application Program Interface (API) through a communication protocol (e.g., TCP/IP) to the Socket. **The Proprietary Application and the Socket are not programmatically linked although they do communicate with one another.** Both the Socket and Socket API must be coded and tested before they can be deployed so, in addition to overhead, there may be some delay to market using this technique.


KEEP SOURCE HIDDEN
Proprietary Application
DISCLOSE SOURCE
Socket API translates calls from the proprietary source into GPL calls. Socket makes calls to the GPL code/GPL API
DISCLOSE SOURCE
GPL Code

Post Oracle/Google Wrapper

If the decision from the Oracle/Google case stands, one may be able to use the previous method with the actual GPL function calls in the Socket API and auto-generate the communication protocol source code. Even better, initial testing can be done with calls directly placed in the proprietary code to figure out which "pieces" of the GPL code may be necessary to achieve the desired functionality. This will work if the names/organization of functions in the APIs are upheld as functional thus eviscerating the copyrightability of those command calls.[vii]


KEEP SOURCE HIDDEN
Proprietary Application
DISCLOSE SOURCE
Socket API uses GPL function calls. Socket makes calls to the GPL code/GPL API.
DISCLOSE SOURCE
GPL Code

Caveats

This case only defines interpretation of APIs in the Northern District of California, so far, and Oracle filed an appeal on October 4, 2012. That said, the holding[viii] in Oracle v. Google aligns well with that of Lotus v. Borland (1996). In Lotus, the Supreme Court affirmed an appellate court's ruling that the menu structures were a "method of operation" and not copyrightable. Some of the amicus briefs, filed on Oracle's behalf, attempt to distinguish the JAVA APIs from the decision in Lotus.[ix]

However, just like a menu structure which calls a particular operation (e.g., "File" brings up a sub-menu of choices such as "New", "Save", "Close"), the JAVA APIs provide a set of methods (operations) organized under various classes that belong to specific packages. A programmer using the JAVA package java.lang, which contains the class "string", can invoke a method named "toUpperCase" to convert the characters in a "string" to uppercase.[x] The 37 packages at issue in the appeal may have more operations organized into various classes associated with each of those packages but, fundamentally, they are a glorified menu structure for programmers and should not be considered copyrightable subject matter as a method of operation.[xi] .

Additionally, the law tends to abhor barriers to entry (e.g., unreasonable non-compete clauses in employment contracts) and, in this case, if Oracle wins the appeal and the APIs qualify as copyrightable subject matter, they would become a barrier to entry for a JAVA programmer to switch to the Android platform.[xii] Sort of like moving a programmer from one house plan to another model … they will still be able to find the master bedroom but they may have hunt around for it first. So, a programmer functionally loses their ability to "operate" in the JAVA language without the use of the APIs.

Finally, Judge Alsup held that:

> *This order does not hold that JAVA API packages are free for all to use without license. It does not hold that the structure, sequence and organization of all computer programs may be stolen.*[xiii]

So, regardless of the ultimate decision, a determination as to whether the code of an API is functional or expressive must be made on a case-by-case basis perhaps with some sensitivity to the specific community involved with a given open source package.[xiv] Finally, while avoiding copyrighted elements of GPL code may not be as onerous in the future, there is always a possibility that the functionality utilized could be covered by a patent. So, perhaps we'll be seeing more patents filed on open source packages in the future.

About the Author

Ria Farrell Schalnat
Of Counsel
Cincinnati, OH
P: (513) 977-8348
F: (513) 977-8141
ria.schalnat@dinsmore.com
http://www.dinsmore.com/ria_farrell_schalnat/

Ria's passion for technology stems from her background as a computer programmer, which affords her unique perspectives when counseling clients on patent and licensing matters. She has helped clients secure patents for a wide range of technology-based systems, including billing, data management, customer relationship management and speech technology. She is adept at guiding clients throughout the entirety of the patent process, from reviewing disclosures and determining patentability to drafting and filing applications and interfacing with the patent office during the review process. She also has extensive experience in handling drafting and reviewing computer licensing agreements with an additional focus in open source licensing. Additionally, she has served as an adjunct professor at both the University of Dayton and University of Cincinnati Law Schools, teaching courses in open source licensing, computer and cyberspace law, and patent prosecution history analysis. Ria was the recipient of the Francis J. Conte Special Service Award in 2010.

Ria is particularly honored to serve as the pronouncer for the WCPO Regional Spelling Bee for Cincinnati, Northern Kentucky, and Southeast Indiana. She also enjoys playing chess and passes on her love of the game to children through the Hua Xia Chinese School where her family studies Mandarin Chinese. She managed several pro bono adoptions through the Volunteer Lawyers Project and recently welcomed a son from China to her own family through adoption.

---

[i] Special thanks to Ted McCullough, co-chair of the IPO Subcommittee on Open Source and Senior IP Counsel-Hewlett-Packard-IP Transactions. Thanks also to Dave Marr, Vice President, Legal at Qualcomm. Both provided valuable advice on the direction of this article.

[ii] http://www.gnu.org/licenses/gpl-faq.html (visited September 12, 2012).

[iii] http://www.gnu.org/licenses/gpl-faq.html (visited September 12, 2012).

[iv] *See generally,* Bain, Malcolm (2010) Software Interactions and the GNU General Public License, IFOSS L. Rev, 2(2), pp 165 – 180 (an excellent discussion of static versus dynamic linking starts at page 175), See also, Working Paper on the Legal Implication of Certain Forms of Software Interactions (a.k.a Linking) Release 1 – July 2010 at http://wiki.fsfe.org/EuropeanLegalNetwork/LinkingDocument?action=AttachFile&do=get&target=software_interactions.pdf (*This document is directed to European Law but contains a detailed technical analysis of various linkages that could be easily ported to a U.S. Copyright analysis).*

[v] *See,* http://programmers.stackexchange.com/questions/50118/avoid-gpl-violation-by-moving-library-out-of-process (Note: As far as the author knows, this recommendation was NOT written by a lawyer but it does reflect, to an extent, the sensibilities of the programming community with regard to these questions. Also note the comment which criticizes this approach as being "morally reprehensible": "If you don't like the GPL, then the "proper" solution is not to use a GPL library.")

[vi] ORDER RE COPYRIGHTABILITY OF CERTAIN REPLICATED ELEMENTS OF THE JAVA APPLICATION PROGRAMMING INTERFACE. United States District Court for the Northern District of California. May 31, 2012. (Oracle America v. Google).

[vii] ORDER PARTIALLY GRANTING AND PARTIALLY DENYING DEFENDANT'S MOTION FOR SUMMARY JUDGMENT ON COPYRIGHT CLAIM. United States District Court for the Northern District of California. September 15, 2011 at page 13 ("This order finds that the names of the various items appearing in the disputed API package specifications are not protected by copyright.

[viii] "But the names are more than just names – they are symbols in a command structure wherein the commands take the form 'java.package.Class.method().' Each command calls into action a pre-assigned function. The overall name tree, of course, has creative elements but it is also a precise command structure – a utilitarian and functional set of symbols, each to carry out a preassigned function. The command structure is a system or method of operation under Section 102(b) of the Copyright Act and, therefore, cannot be copyrighted. Duplication of the command structure is necessary for interoperability." ORDER RE COPYRIGHTABILITY OF CERTAIN REPLICATED ELEMENTS OF THE JAVA APPLICATION PROGRAMMING INTERFACE at page 4.

[ix] *See*, Groklaw at http://www.groklaw.net/articlebasic.php?story=20130221153759232 (provides links to the amicus briefs filed on behalf of Oracle as well as an analysis of Microsoft's amicus brief).

[x] http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html.

[xi] Initially, I did not agree with the decision in Lotus v. Borland but that is the current law of the land and would require the United States Supreme Court to reverse their earlier position (which was actually 4-4) to render Oracle's position viable. Also, at the time Lotus v. Borland was decided, patent protection for software was ramping up so this may have been seen as a viable alternative. Patent protection for software has come under a significant amount of fire in the last few years which, from a policy perspective, makes this case ripe for another look by the United States Supreme Court.

[xii] Four packages in the JAVA API were deemed core to the JAVA language (including java.net, java.io, java.lang, and java.util). Some of these packages contained object and exception classes, without which the JAVA language would be virtually useless for coding programs. One has to wonder if there is such a thing as a non-core or non-functional API call?

[xiii] ORDER RE COPYRIGHTABILITY OF CERTAIN REPLICATED ELEMENTS OF THE JAVA APPLICATION PROGRAMMING INTERFACE at page 41.

[xiv] Remember, some open source communities may find this technique objectionable even if it is legally permissible. S*ee, supra,* the "morally reprehensible" comment in Footnote 5. So, if the goal is to avoid controversy, a quick check of a given community's feelings may be in order.