Ramblings

Precession of the Perihelion of Mercury

Preface

Everybody agrees that the coastline of Australia is longer than the coastline of Cuba. Similarly, the coastline of Japan is longer than the coastline of Australia. Wait... what? The CIA says so, so it must be true. Furthermore, it turns out that the length of a coastline depends on how you measure it.

What does that have to do with licensing? Most third party licensing questions are of the form:

Can ASF Project "A" do "B" with "C", which is licensed as "D"?

The answer is that simple rules and gross approximations often gives the right answer in both cases. Like comparing Cuba and Australia, for example. In some cases, however, such rules and approximations give the wrong answer. And in rare cases, the answer itself depends on how the question was asked.

This is meant to be a lighthearted and sometimes sideways introduction to what is a complex and often contentious subject.

Approximation 1

The first approximation to the Third Party Licensing Policy is that the ASF must be able to distribute works created within an Apache Project solely under the Apache License, Version 2. And furthermore, any dependencies that such Works might have must be licensed under similar terms. This is an ideal that is both highly desirable and yet not 100% consistently met – at least not in full, when you consider recursive dependencies.

For this approximation, the sole determinant is the license (the "D") in the statement above, and the question arises: what are "similar" terms?

I look at it this way: Apache type licenses are a universal donor. Such code can be given to anybody. GPL type licenses are closer to being a universal recipient, at least within the domain of open source. Such code can be combined with a wide range of open source code.

The FSF provides us with a good visual aid for this. Note how all the arrows are one way.

For us to remain a universal donor, our products must remain antigen free. This leads naturally to the definition of Category A: Authorized Licenses.

Approximation 2

The first approximation gives a lot of good answers, but a few wrong answers. For starters, antigens aren't all bad. The Apache License, Version 2 has a patent termination clause. That clause coupled with Contributor License Agreements is part of a larger strategy which keeps the blood supply safe from other dangers.

This means that while category A licenses are perfectly acceptable in small doses, heavy dependencies on works made available under such licenses will dilute the patent liability limitations that the ASF so carefully engineered into its license policy.

The flip side of this is also true. Non-category A licenses may be perfectly acceptable, depending on *how* they are used. In case you hadn't noticed, that's the "B" in the statement above. Incorporating a Work in order to implement a core function in a way that is difficult to extract is a very different thing that having an optional plugin that allows integration with another product. That continues to be true even if that product that is made available under an excluded license. Or even a proprietary, non-open source one.

A concrete example of this is the Jakarta NT Service. A JBOSS deployer is another. To borrow a phrase from another aspect of law, as long as there is a substantial non-infringing use without these plugins, these uses are not only OK, they are positively aligned with the goal of being a universal donor. Some care must be taken in documenting how this is done, of course.

So the second approximation is a matrix. Rows are licenses. Columns are uses. Initially the matrix is sparse, but over time it can be filled in with "always OK" and "never OK". Some of the answers will have an asterisk beside it. When organized this way, you can start to see some licenses are inert gases, while others are positively radioactive, and categories will emerge.

And just when you think you can get a handle on this, people invent new licenses, new clauses, and new exclusions. And new uses.

Approximation 3

Asterisks were mentioned above. On a few licenses and uses, the asterisks seem to multiply like dandelions. Examples: a column that deals with hard dependencies on proprietary products that are distributed free of cost for a wide range of uses; and a row that deals with GPL based licenses with classpath or usage as a library exceptions.

Java and C# are prime examples.

At first, it is tempting to try to define this in terms of the types of usages. But increasingly you find that it is difficult to distinguish the acceptable usages from the unacceptable usages in this manner. Ultimately, it turns out to be simpler to simply enumerate the acceptable products. That's the "C" in the description above.

The distinguishing characteristic tends to be something entirely outside of the license and outside of our control. Charateristics that deal with attributes like *ubiquity* and *standards*. Is the product ubiquitous enough to be something that it can be presumed to already be installed in the target environment? Is the ASF code interacting with this product using "standard" interfaces that were explicitly designed to enable what the ASF product is trying to do? Note: I put "standard" in scare quotes as this question needs to be explored without the burden of trying to figure out whether or not the ASF endorses any particular standards body. For these purposes, any architected interface will do. More simply: are we employing or extending the product in the way it was intended?

So log4j depending on Java is clearly OK. It even is clearly marked as such.

When faced with the question of whether or not Hibernate could be considered a system dependency or not, the Incubator simply stated that Roller could not bundle and ship Hibernate. This lead to this issue, which was based on the observation that much of the target audience for Roller did not have Hibernate already installed, and merely were interested in installing it for the purposes of supporting Roller.

Needless to say, these types of discussions and decisions are difficult for everybody involved. And should only be reopened based on a specific request from a PMC.

Approximation 4

If you see the pattern, there is only one letter left. "A". Given that we are all ASF projects here, there can't be a case where two ASF product wanting to use the same product (with obviously the same license) in the same manner ends up with the answer that it is OK in one and not OK in the other, can there? Actually, it is rare, but possible.

Lucene.Net has a hard dependency on the Microsoft .NET Framework. As described above, that's quite OK. And again, the dependency is clearly marked, so those not interested in this podling will simply not participate.

On the other hand, changes to httpd to make use of a similar dependency would likely result in a fork. For a while, Open Office faced a similar controversy, and that lead to the mail merge component being rewritten in Python and ultimately may have contributed in a small way to the formation of the OpenJDK.

And, clearly, Tomcat is exactly one such a "fork". One that is doing quite well. The point here is that there will be times when protecting the option to NOT make use of a third party library is as important as protecting the option TO make use of a third party library.

Approximation 5

OK, I'm out of letters. But still not everything quite fits. Santuario has to deal with regulations that aren't copyright, patent, or even license related. HTTPD has dealt with a similar situation by providing separate downloads.

The title at the top of this document pays homage to an observed phenomenon that helped cause Newtonian mechanics to be replaced by General Relativity. An ugly fact as it were. Note that Newtonian mechanics is still taught today as it still produces quite useful answers in a large range of applications, but where the answers that General Relativity provides are different than the ones that Newtonian mechanics provides, the former is more often the correct one.

Now I don't pretend to presume that this brain fart compares in even the slightest way to the significance of General Relativity. To be honest, I simply sprinkled in science metaphors to spice up an otherwise dry subject, and also to give credibility to the notion that some things that may seem initially to defy common sense from one perspective may actually be the **only** sensible position from a completely different perspective.

And on that topic, while Newton's Gravity was the "downfall" of Newtonian Mechanics, Quantum mechanics will likely revolutionize Relativity. In Quantum mechanics, the role of the observer often affects the properties of the observed. The ASF's decision to pursue Java and XML have had a profound affect on both of those subjects. (Perhaps not as much as we might have hoped, but substantial nevertheless). This means that from time to time we should allow ourselves to pursue carefully selected initiatives that defy all of the above criteria.

Epilogue

Even with the above disclaimer, this entire ramble is too pretentious for my tastes. What I really want to do is to make quick work of questions on this list. I believe that in order to do so, having the flexibility of being able to merely say that "Java based projects are welcome here", or "anything written in Ruby and made available under the Ruby license is OK as a dependency for a project written in Ruby".

The reason I want to do so is simple: the boundary between what is and what is not a part of the Ruby runtime is blurry; the intentions of the authors of any such Gem is clear; and frankly, such a statement is very much in line with what licensees of Ruby code expect. Such a statement also avoids spending any energy trying to fit the either the Java or Ruby licenses into one of the existing categories (or, **shudder** alternately, modifying the definition of one or more categories to accommodate Ruby), which would be tantamount to trying to decide how we should handle TCL code that is licensed under the Ruby license.

Differences

To a high degree of approximation, this approach described by this page matches the Draft Third-Party Licensing Policy. Key differences:

- This page contains a complete lack of phrases such as "must be removed before the earlier of one year or two major releases". If an existing project has been making releases for an extended period of time without significant objections by either contributors or licensees, I feel that there is no reason to intercede.
- This page lacks any specifics on Category B or how to handle third party scripts. I'm not prepared to declare consensus on these issues, and the appropriate way to handle these cases may very well be context specific.
- This page explicitly acknowledges that decisions may not be consistent across all projects and all time, while at the same time attempting to minimize and constrain the times that this may occur.

Statement of Intent

- The overriding goal is to remain a universal donor. While PMCs will be expected to balance developers' and licensees' needs, in questions relating to third party licensing matters, the Legal Affairs Committee will focus mostly on ensuring that the entire rage of the needs of licensees, ranging from the most risk adverse corporations to the most activist members of the Free Software Movement, are met.
- The ASF is all about voluntary contributions, and intends to obey the constraints expressed by the copyright holder when redistributing their work.
- All decisions will be made and posted publicly.
- As much as humanly possible, everything will be by consensus of the Legal Affairs Committee, though possibly via Lazy Consensus. If there ever is a time that consensus can't be reached, the decision will fall to the Chair who will keep the board informed on any such issues.
- PMCs will be encouraged to provide specific questions. Questions of the form listed above will take priority over "here's a problem, solve it for me" types of open ended questions.
- A key part of the third party licensing policy will be the establishment and policing of specific policies for NOTICE, LICENSE, and README files.