**Why, oh why, must I enter one CQ per jar file?**

The common thread in all scenarios to follow is that Committer Joe downloads source code for Kewl Software. Kewl's standard source code download contains some java files at the top level and a bunch of jar files delivered at source. Our tools only review source code, so our review is limited to the source code that is presented to us at that top level.

1. Joe enters one CQ for Kewl's standard source code download. However, our review would be limited to the source code at the top level. We would ask Joe to enter individual CQs for any jars that are bundled-with-source-distribution (we'll call them "JarBWSD" from now on to make it simple) that he may require.

* We ask for one CQ per JarBWSD for a number of reasons, including:
a) a jar file is a reasonable level of granularity (not the minimum level, but a reasonable one) that a project might choose to consume; and
b) if nested jars were to be allowed on each CQ, the CQ summary with the top level source does not give us an accurate view of the IP that has actually been reviewed, which could lead to projects not knowing about some applications they could otherwise piggyback on for re-use, duplication of IP Team's review of JarsBWSD, and as a result of the duplication, longer wait times for CQ resolution in general.

2. Joe knows our tools only review source code so before he attaches Kewl to the CQ, he helpfully opens all the jar files, extracts the source for each, zips it all up with the top level source in one zip file and attaches it to the CQ as one unique amorphous blob of Kewl. Joe thinks that in addition to helping us, he only has to enter one CQ…win/win!! IP Team will thank Joe for his thoughtfulness, but ask him to please revert the package to the manner in which he received the source download and enter individual CQs for the JarsBWSD that he may require, for the reasons cited in scenario 1a and 1b above.

3. Joe only needs the top level source and enters one CQ for it. During the build process, Kewl's top level source builds a bunch of jar files. Because our policy is to review top level source code from the standard download, we will have defacto reviewed all the source code associated with the generated jar files. We do not need separate CQs for jars built from / generated by top level source: Joe's one CQ for Kewl at the top level will suffice.

These aren't the only 3 scenarios, I'm sure. But they are the ones that come to mind for this exchange.

As a footnote, you may ask "Why did Eclipse standardize on JarsBWSD as distinct levels of granularity? Why don't you lump some jars together for one CQ?" We needed an objective delineation of code because we aren't users of the applications in question, and given the re-use we see, this seems to be a reasonable demarcation. The bandwidth and consultation required to make a subjective call on which jars should be included vs. which jars should be broken out, on a case-by-case basis, would have the effect of slowing down the actual review work and time for CQ resolution.

Whew.