ReleaseNomenclature

Have you ever stumbled across a really nice looking project, downloaded and installed the binaries that were labelled "RC1" and got to work on integrating it in your application, only to find that "RC2" came out two weeks later with a host of feature changes, including some new ones and changes to the interfaces that you rely on? It's very frustrating for early adopters, and this has translated into less aggressive adoption of open source by larger customers because everyone is racing to have their code considered "release quality", often without putting in the effort to actually deliver that quality.

The goal of a documented release nomenclature is that the users of your project can intelligently choose a version at a glance, and know what they are getting in to when they do so. As in life, this is all about managing expectations. Exceeding expectations always leads to happy returns.

So without further ado, here is a proposed release nomenclatire for the Apache JDO project:

Stable Releases

Stable releases are numbered in the form of "x.y.z", where:

- *x* is the major version number of the library. This number is incremented when there are major changes to the underlying architecture or might be tied to the version of the specification that it is tied to, as in our case.
- *y* is the minor version number, and indicates new features and/or changes significant enough to affect compatibility with existing code.
- *z* indicates a bugfix release. No changes to features or reduced compatibility are indicated by such a release.

Unstable and Prerelease

- *SNAPSHOT* indicates that the code you are working with is whatever happened to be in the source repository when the snapshot was made. If the snapshot was made by an automated tool in a nightly process, you may be wise to check the results of the associated unit tests. If you are concerned about functionality that you rely on, you might consider finding the names of the unit test that most closely matches your code, and writing and contributing tests if you cannot find them. In doing so, you can not only check that a SNAPSHOT is suitable for you at a glance, but you tend to assert with interested developers that you are interested in the code working a certain way. For projects such as JDO, the unit tests are generally of the form of the TCK, in which case it is not practical to update the unit tests without also updating the specification itself. But your involvement there is encouraged as well!
- *d* or *development* releases are compilations that have been made of works in progress, but at select points where developers may have felt that the code was at a very natural point for people to work with it. Contrasted with an automatic snapshot, a development release is timed to coincide with the entire team at a point between tasks, affording maximum functionality to outside users.
- *a* or *alpha* indicates that the software is feature complete with no catastrophic bugs, but that the software may have bugs that in rare cases could be severe enough that unknown bugs could cause unexpected data loss. Projects with a strict release process generally implement peer-review change control at this point, ensuring that changes that are made do not introduce new functionality and are strictly tied to a documented bug. By doing this, users are assured that the number of bugs in the code is decreasing as the risk from new bugs in code with new features are eliminated. This will not stop developers from wanting to check in their changes for new features in the next release, so a branch is created for this code, named for the release that it will become, allowing developers to continue to work on HEAD if they choose.
- *b* or *beta* code has a quality level of "no known bugs that are not deferred". This means that if there are bugs (and there always are), they must be marked so as to defer them to the next release or they must be fixed before another release can happen. In doing so, users can understand what to expect from a final release and a time window exists for them to raise whatever objections they might have to the decisions that were made regarding code that they are personally interested in.
- *rc* or *release candidate* is the name that was traditionally given to code that was packaged as if it were going to the printers, with tens of thousands of copies prepared to be printed. Dozens of people were required to do this in larger shops such as Apple (where yours truly worked for some time), and decisions about change to a rc build were very lengthy and agonizing for all involved.
- In the age of printing on the internet, *"Golden Master"* and *"Release To Manufacturing"* builds are rarely seen, but are still very common in environments where large volumes of release media are printed and sent into distribution channels. These versions take on the appropriate name indicated in a stable release.