Guidelines for working on jdk8u

OpenJDK 8 is a long-term stable release of OpenJDK. Our primary goal is to maintain it, fixing significant bugs and especially security problems as we go along. The "first, do no harm" principle applies: we must not break things.

In general we'll use the process in https://openjdk.java.net/projects/jdk-updates/approval.html.

All fixes that significantly improve stability, security or performance and do not change behavioural compatibility [1]will be considered for jdk8u. To use the language of the JDK 6 project, by default such fixes are assumed to be applicable to jdk8u, especially if having "soaked" in later JDK releases for a time without incident. By "significant" I mean any bug that affects runtime behaviour in a way that either produces incorrect results, poor performance, or crashes the VM, especially TCK failures. Failures that are due solely to bizarre or unreasonable combinations of -XX: command-line parameters probably don't reach the bar of significance, and fixing them will carry a non-zero risk of breaking something, so we should err on the size of caution. For now it's mostly "bug fixes only".

Build failures on all platforms, including 32-bit ones, are assumed to be applicable. Also, there is a good deal of C++ code with Undefined Behaviour in HotSpot, and such bugs tend to cause failures with more recent C++ compilers. While all UB fixes may be applied to jdk8u, they should be submitted to the current jdk development tree first.

Occasionally we might have to make changes which raise compatibility issues. We will liaise with the Compatibility & Specification Review (CSR) Group.

We have two active trees, http://hg.openjdk.java.net/jdk8u/jdk8u and http://hg.openjdk.java.net/jdk8u/jdk8u-dev. jdk8u-dev is always open to all contributors, and that's where everyone should push their patches after maintainer approval. Patches pushed to jdk8u-dev will be copied to jdk8u at regular intervals. When the time comes for an update release, jdk8u will be closed for testing and stabilization. From that point onward only critical bug fixes may be applied to jdk8u, at the discretion of the maintainers.

To request maintainer approval for a backport, tag your entry in the bug database with "**jdk8u-fix-request**". A maintainer will reply by either tagging the bug entry with "**jdk8u-fix-yes**" or "**jdk8u-fix-no**".

Once a quarter, we will snapshot the jdk8u tree and prepare a Critical Patch Update (CPU) release. Once the snapshot has been taken the engineers working on the CPU will work in the dark, sharing the patches with only the OpenJDK Vulnerability Group. Any patches not committed to jdk8u at the time of the snapshot will probably have to wait for a later release. [I don't propose to make any non-CPU releases: one release a quarter should be quite enough for jdk8u. However, if an urgent problem arises we might need to make an intermediate release.]

Having said all of that, there is considerable customer demand for backports of features from later OpenJDK releases. I don't intend to forbid such backports, but strict rules will apply. Features which apply to ports in jdk8u must have the property that they can be disabled altogether by the use of a command-line switch. This switch should turn the feature into a NOP, so that it does not affect the rest of the system in any way. Reviewers should ensure that every hunk in such a changeset is guarded by an if (Feature_enabled) statement or something similar. This will also allow Feature_enabled to be made a compile-time constant, and if set to false this will allow images to be created without the feature.

With regard to the likely feature backports, there are several possibilities, in particular the Java Flight Recorder (JFR). There is a tree http://hg.openjdk.java.net/jdk8u/jdk8u-jfr-incubator/ for this work. Patches to this tree must be approved in the usual way, with an RFR email to the jdk8u-dev mailing list.

[1] https://blogs.oracle.com/darcy/kinds-of-compatibility:-source,-binary,-and-behavioral