

# JDK Updates Guidance

## Goal

As an OpenJDK developer I'd like to fix a bug in a released JDK version, for example JDK 8u201.

## Getting Your Fix into an Updates Release Train

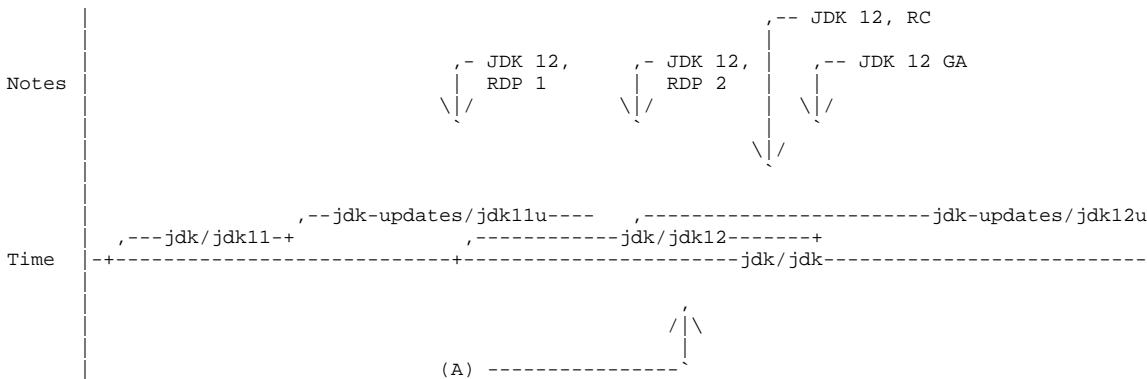
The first step to get your bug fixed in a released JDK version is to attempt to reproduce it with latest JDK: <http://hg.openjdk.java.net/jdk/jdk/> (jdk/jdk). If that's the case, the next step is to fix the bug in JDK head[1] and then follow the backport-chain which is usually in this order:

1. jdk/jdk (JDK N)
2. jdk-updates/jdkXu (where X == N-1)
3. jdk-updates/jdkYu (where Y is the current LTS version)
4. jdk8u/jdk8u

Note that it is advised to let the fix in latest JDK receive a certain amount of testing before a backport is being requested for an updates release.

## Current Fast-Release-Cadence Development Model

Understanding the fast release cadence development model helps when trying to get bugfixes into a released JDK version. Below is a ascii-art picture of JDK 13 at a point where JDK 12 has not been released yet. That would be a time around February 2019.



## Example: Getting Bug Fixed at Time (A)

Considering the above example we'll walk you through the process of getting the bug fixed at point in time (A) as depicted above. At that point the jdk/jdk tree is the development tree for JDK 13, the JDK 12 stabilization tree has been branched already at point RDP 1. What's more, a JDK 12u updates tree has been created as well at point RDP 2.

Let our example bug have priority 3. Once we have pushed the fix to jdk/jdk, what happens next?

According to the JDK Release Process Rules[2], RDP 2, means that only priority 1 and priority 2 bugs may be fixed beyond RDP 2 point. Note that RDP 2 refers to Ramp Down 2 Phase. That means that the JDK 12 stabilization tree is frozen for our P3 bug. Instead, we will get the bug-fix into an JDK 12u update by following the JDK updates rules[3]. In our case we add label 'jdk12u-fix-request' to the master bug and add a comment with heading "Fix Request" with some reasoning, risk assessment, testing done, etc. Once, JDK 12 maintainers approve the backport by adding the 'jdk12u-fix-yes' label, we can now push the fix to the jdk-updates/jdk12u tree.

Once we have the bug fixed in jdk/jdk and jdk-updates/jdk12u we may request a backport to the latest LTS release, JDK 11 in our example. Note that persuing this backport may happen in parallel to JDK 12 backports. Rules as determined by the relevant updates project apply.

Finally, once the fixes were pushed to JDK 12 and JDK 11 updates, the bug may be backported to JDK 8u. Again, project specific approval rules apply.

## Appendix

-----

Where should I push backport patches to?

JDK 12u backports tree: <http://hg.openjdk.java.net/jdk-updates/jdk12u>  
JDK 11u backports tree: <http://hg.openjdk.java.net/jdk-updates/jdk11u-dev>  
JDK 8u backports tree: <http://hg.openjdk.java.net/jdk8u/jdk8u-dev/>

Which rules do apply for my backport?

JDK updates rules: <http://openjdk.java.net/projects/jdk-updates/approval.html>  
JDK 11u updates rules: <https://wiki.openjdk.java.net/display/JDKUpdates/JDK11u>  
JDK 8u updates rules: <https://wiki.openjdk.java.net/display/jdk8u/Guidelines+for+working+on+jdk8u>

## References

-----

- [1] <http://openjdk.java.net/projects/jdk-updates/groundrules.html>
- [2] <http://openjdk.java.net/jeps/3>
- [3] <http://openjdk.java.net/projects/jdk-updates/approval.html>