How to contribute a fix

This page outlines the detailed recipe of what to do with a fix.

There are two general types of fixes:

- *backports:* most commonly, a change for OpenJDK 11 updates is a backport of a change that has been made in a higher OpenJDK version. Start the recipe from Step 1.
- *new fixes:* rarely, there is a need for a net new change for OpenJDK 11 updates, e.g. because a fix would not apply to higher OpenJDK versions that are in maintenance. Start the recipe from Step 4.

Important: The whole process is driven by the backport requester/contributor. Nothing here assumes that somebody else would do the work. If you are not the OpenJDK Author, that is, you don't have a JBS user account, you'll have to ask for help for steps 6 and 7 (working with JBS to put appropriate metadata). If you are not the OpenJDK Updates Project Committer, you'll have to do the same for step 8 (pushing the change). The regular place to ask for help is JDK Updates mailing list. Regular contributors would eventually gain the necessary privileges to avoid this overhead.

Contribution recipe:

1. Check the original JBS issue on https://bugs.openjdk.java.net/
    a. Carefully check linked issues whether there are follow-up fixes that need to be brought with the backport.
    b. If there are relevant issues that prevent clean backport, consider backporting those first (within reason).
    c. Open the link to the original commit and note its repository and changeset number.

2. Export the original commit from the original repository (hg-export assumes Skara CLI tooling)
    a. Locally: **`"git hg-export <commit-sha> > <bugid>.patch"`**
    b. From the remote repo: "**wget https://github.com/openjdk/jdk/commit/<commit-sha>.patch `-O <bugid>.patch".`** `You will have to change the format of the commit message (author name, reviewer name, etc.) manually so that it passes jcheck.`

3. Apply the exported patch to the target repository (mostly jdk11u-dev):
    a. Make sure that the changeset metadata is kept (original authors, Reviewed-by lines, etc.)
    b. Most convenient to use Mercurial mq extension: **`"hg qimport <bugid>.patch && hg qpush"`**
    c. Resolve the patch and make necessary adaptations if it doesn't apply cleanly.

4. Test the patch
    a. "tier1" tests should be passing at all times, use **`"make run-test TEST=tier1"`** to run
    b. "tier2" provides a larger coverage, if you have resources to run it. Use "**`make run-test TEST=tier2"`** to run
    c. Run tests from the area that the patch affects, use **`"make run-test TEST=<path-to-tests>"`** to run specific tests
    d. New regression tests that come with the patch should pass

5. If *(and only if)* the original patch was modified, get the change reviewed
    a. **Note:** the change review *is not* the approval, which you would get at the next step
    b. It is advised to do the review at the jdk-updates-dev mailing list and optionally cc the original mailing list.
    c. The review request should be clearly marked as such: "[11u] RFR <original-bug-id>: <synopsis>"
    d. It is helpful to state what changes were needed and why: the difference against original patch, motivations for doing things differently, etc.

Example RFR message

```
Subject: [11u] RFR 8888888: My Hovercraft Is Full Of Eels

Hi,

Original bug:
  https://bugs.openjdk.java.net/browse/JDK-8888888
  https://hg.openjdk.java.net/jdk/jdk/rev/88888888

Original patch does not apply cleanly to 11u, because eels are all different sizes
and shapes. Notably, I had to change the com/antioch/holy/Grenade.cpp to avoid API
that only exists in 12+.

11u webrev:
  https://cr.openjdk.java.net/~monty/8888888/webrev.01/

Testing: x86_64 build, affected tests, tier1

Thanks,
-Monty
```

6. Request and await approval for the fix (if the issue is not public, goto step 8 first)
    a. Put the **`jdk11u-fix-request`** label and add a *"Fix Request"* comment on the issue, that explains why the fix should be backported, contains the link to backport RFR (if applicable at step 5), the dependencies on other backports (if any), shows what testing was done to verify the backport, gives a risk estimate, etc. The goal for the *"Fix Request"* comment is to give maintainers all the information about the backport to make the informed decision for inclusion into update release.

   b. Wait for maintainer approval, which would manifest as **`jdk11u-fix-yes`** label on the issue.

Example Fix Request comment with RFR

```
Fix Request

Backporting this patch eliminates the critical eel overflow. Patch does not apply cleanly to 11u
and requires adjustments. Backport requires JDK-8423421 and JDK-8771177 to be applied first.
11u RFR: http://mail.openjdk.java.net/pipermail/holy-grail-dev/2019-August/00001.html
```

Example Fix Request comment without RFR

```
Fix Request

Backporting this patch eliminates the critical eel overflow. Patch applies cleanly to 11u. New test
fails without the product patch, and passes with it. Backport requires JDK-8423421 and JDK-8771177
to be applied first. tier1 and tier2 tests pass with the patch.
```

7. What if the change needs a CSR?
   a. From the original issue, create a backport issue ("More""Create Backport") in JBS. Target the backport issue to **`11-pool`**. This issue will be resolved when the change is pushed.
   b. From that **11-pool** issue, create a new CSR and copy/paste the contents from the original CSR. If the CSR is a straight copy of the original CSR, say so clearly in the issue text, otherwise point out differences. The new CSR should also have version **11-pool**.
   c. Run the CSR through its process to get it approved.

8. What if the change you want to downport is not public?
   a. Sometimes a change you want to downport is not public in JBS. This means you can see the change and its JBS ID in the Mercurial repository, but you will not find the corresponding issue in JBS.
   b. In such a case, you have to create the corresponding backport issue manually, according to the following recipe:
   c. Create a new issue in JBS with type "**Bug**" (you can't directly create an issue of type "**Backport**")
   d. The new issue should have exactly the same summary like the original change. You can take this from the Mercurial "**summary**" line by stripping the prefixed Bug ID.
   e. It's helpful if the bug description contains a link to the original commit.
   f. Affected version should be one of "**8**", "**11**", ...
   g. Under "**Linked issues**" choose "**backport of**" and enter the bug id of the change you want to downport into the "**Issue**" filed (in the format "**JDK-XXXXXXX**").
   h. Once you've filled out all the other fields, press "**Create**" and once the issue has been created, edit the issue "**Type**" and change it from "**Bug**" to "**Backport**".
   i. Continue as usual at step 6.

9. If and only if everything is approved, push the change.