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 3.

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 an OpenJDK Author, that is, you don't have a JBS user account, you'll have to ask for help for steps 5 and 6 (working with JBS to put appropriate metadata). The regular place to ask for help is the JDK Updates mailing list. Regular contributors would eventually gain the necessary privileges to avoid this overhead.

OpenJDK 11u uses SKARA (Git) for backporting fixes from later JDK releases since June 2021. The old Mercurial-based workflow is no longer described. Please reach out to the JDK Updates mailing list in case you need help with that.

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 a clean backport, consider backporting those first (within reason).

2. Create the backport commit
   - Option 1 - Use the /backport comment command on GitHub:
     a. Make sure GitHub Actions is enabled for you on the OpenJDK Bots jdk11u-dev repository.
     b. Open the link of the original commit in GitHub and issue the /backport command by adding the comment: "`/backport jdk11u-dev`"
     c. For a clean backport the bot will provide you with a branch in his own repository and a link to create a pull request. If the backport needs manual resolving, it will provide you with instructions, similar to option 3.
   - Option 2 - Use SKARA CLI tooling:
     a. Create a branch for your backport, e.g. "`git checkout -b my-backport-branch master`"
     b. **"git backport --from `https://github.com/openjdk/jdk` <commit-sha>"**. See the SKARA Wiki for more info.
     c. If necessary, resolve changes and follow the instructions.
   - Option 3 - Use plain Git to create the change:
     a. Create a branch for your backport, e.g. "`git checkout -b my-backport-branch master`"
     b. **"git fetch --no-tags `https://github.com/openjdk/jdk` <commit-sha>"**
     c. "`git cherry-pick --no-commit <commit-sha>`"
     d. If necessary, resolve changes.
     e. "`git commit -a -m "Backport <commit sha>"`

3. 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
   e. Enabling GitHub Actions for your personal fork of the jdk11u-dev repository before publishing your branch will provide you with a tier1 test run via GitHub Actions

4. Create a pull request at https://github.com/openjdk/jdk11u-dev
   a. If you have created a backport via Option1, the /backport command, and the backport was clean, you can use the provided link to create a PR
   b. In all other cases, push the new branch to your fork of https://github.com/openjdk/jdk11u-dev and open a PR. You can do this in one step via the SKARA command "`git pr create --publish`". If it is a backport, make sure the title of the PR is "`Backport <SHA hash of original commit>`" to have the bots correctly recognize your change as a backport.

5. If your patch is not a clean backport (labeled as `clean` by the bots), get the change reviewed by some jdk-updates reviewer
   a. **Note:** the change review *is not* the approval, which you would get at the next step
   b. The PR message is automatically posted at the jdk-updates-dev mailing list. You might optionally cc the original mailing list or other OpenJDK mailing lists to get some more attention to your PR by using the `/label` command.
   c. It is helpful to state in the PR description what changes were needed and why: the difference against original patch, motivations for doing things differently, etc.

```
Hi,

This is a backport of JDK-8888888: My Hovercraft Is Full Of Eels

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+.

Testing: x86_64 build, affected tests, tier1

Thanks,
-Monty
```

6. Request and await approval for the fix (if the issue is not public, go to 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, 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 an informed decision for inclusion into the update release.
    b. Wait for maintainer approval, which would manifest as `jdk11u-fix-yes` label on the issue.

```
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. PR was
acked by monty.
```

```
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 JBS issue, create a backport JBS issue ("More""Create Backport"). Target the backport issue to `11-pool`. This issue will be resolved when the change is pushed.
    b. From that **11-pool** JBS 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 Git 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 Git "**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**"). After adding this link, it will be not visible. You can only verify that you added it by looking at the history.
    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 2.

9. If and only if everything is approved, push the change. This is done using the /integrate command of SKARA on the backport pull request. If you are not an jdk-updates project committer you'll need a committer to sponsor the PR for you.