Contributions

Contributing Documentation

If you want to help us improve our User Guide and other online documentation, please contact a Steering Committee Member and ask for editing permissions for our wiki space.

Contributing Code

If you have extended Jikes RVM and would like to contribute your extension back to the community, please use the patch tracker to submit your contribution. Please include the following:

- Your contribution in the form of patches or bundles (see below)
- The appropriate Statement of Origin (see below)
- A description of the functionality you are contributing
- The version of Jikes RVM used to create your patch

Your contribution will be licensed under the EPL (Eclipse Public License), the license used for Jikes RVM. The license has been approved by the OSI (Open Source Initiative) as a fully certified open source license. If your contribution is included in the system, you will be acknowledged on the contributors web page, along with getting the satisfaction of making the world a better place.

Getting your code contribution merged

You should make it easy for members of the Jikes RVM team to merge your code contribution.

At minimum, you must do the following:

1. Make sure that your contribution follows our coding style and the coding conventions. Use Checkstyle (e.g. via "ant checkstyle") to check for common errors.
2. After verifying that there are no checkstyle errors, run the pre-commit tests on your machine.
3. After the pre-commit run is done, you can find the report at results/tests/pre-commit/Report.html in your jikesrvm directory. It should read something like "Total Success Rate: 128/128" (numbers from August 2012). The report will also display the revision number, if applicable. If the revision number ends with a +, you have uncommited changes in your working copy. Check that the uncommited changes are not supposed to be in your patch.

The Jikes RVM team will check for those points.

You can do the following things to further increase the chances that your contribution will be merged:

- Split your contribution in a set of patches with each patch representing a commit. The size of the patches is not important. However, it is important that each patch contains exactly one conceptual change. One way to check this is to think of a summary message for each patch. If your summary message contains an "and", you should check if there is a sensible way to split up your patch. The set of patches should show the development of the feature as it would have been ideally, i.e. you should remove dead ends. Put the reasoning for the changes (and why alternative solutions are not appropriate) in the commit messages. If you are unfamiliar with preparing patches for open source projects, take a look at this article from the Linux Foundation.
- Write useful commit messages. Use the first sentence to give a summary of the issue. If you are fixing a JIRA issue, please mention it in the description.

  

  Example commit message
  RVM-XXXX : Fixed a bug in foo.

  Longer description of the change, the reasons for the change and the consequences. This part can span multiple paragraphs and is optional: you will need to decide what's important for people looking at the commit.

  Consider providing a README if your chosen patch format does not support commit messages.
- Run appropriate existing tests other than pre-commit. This is especially useful if your changes touch an area of the code that is not covered by pre-commit (e.g. alternative garbage collectors).
- Provide automated tests for your contribution.
- Follow up on your patch until it is in the mainline. This may include answering questions, reworking your patch and gently reminding the developers of your contribution.

Contributing patches

Patches should apply against a revision of the main repository. This ensures that your patch can always be applied easily. You can use `hg export` to create patch files from your commits.

Contributing bundles

If you contribute your changes in the form of mercurial bundles, you must make sure that the parent changeset of your first changeset is in the main repository. If it is not, `hg unbundle` will fail and your bundles cannot be imported.

Statement of origin

All contributions must include one of the Statements of Origin below. Insert your name(s) in the first blank(s) and a high-level summary in the blank in a (i) . Examples of a high-level summary are "Fixed bug in scheduler", "Extended type propagation in optimizing compiler", or "Added new garbage collector".

If your contribution is owned by your employer, someone authorized by your employer to make such a decision must add a comment to the patch in the tracker stating that you have permission to contribute it.

Statement of Origin: Single Contributor Single Contributor for all Contributions Multiple Contributors