Committing patches to the Subversion Repository

> **Important**
>
> Currently under construction! Don't use these instructions yet as these are currently based upon Derby's instructions from http://mail-archives.apache.org/mod_mbox/db-derby-dev/200603.mbox/%3c20060302115045.GA2590@atum01.norway.sun.com%3e which may not be entirely suitable and requires discussion.

Committing contributions from others is not as straightforward as one might think. This mainly stems from the fact that there is an svn diff command, but there is no svn patch command (lack of symmetry - see this and this Subversion issue). There are a number of things to remember; if not done carefully, you might end up doing partial commits that may break the build. This page attempts to give a recipe for safely committing code contributions.

1. Make sure you have a clean sandbox

In the trunk directory, run svn status - it should not list anything. If it does, you may want to run svn revert -R . to remove all local modifications, or use /check out a different sandbox.

Do we need to run maven clean ?

2. Check your sandbox' svn revision with svn info

example goes here..

3. Align you sandbox' svn revision with the patch's

1. If the contributor has indicated on which revision the patch was created (with svn diff), run svn update -r REVISION.

2. Otherwise, you may choose to run svn update -r { DATE }, using the date of the contribution of the patch. NOTE: svn is very picky about date formats. See http://svnbook.red-bean.com/en/1.0/ch03s03.html and scan down for valid date formats.
3. If you think there has been no changes in the patch's area since it was contributed, you can try on the head: svn update.

4. Apply the patch in the trunk directory

patch -p0 < PATCH_FILE.

5. Ensure there were no conflicts when applying the patch

On unix, you can run find . -name "*.rej" -print. If a *.rej file is found, you should either resolve the conflict or ask the submitter to submit a new patch against the newest revision. If you omit this step, you'll still catch this problem before committing (when you compare your svn status output to the contributor's), but it's a good idea to catch this early on.

6. Run svn status

example..

7. Add Newly Added Files

Newly added files will show up with ?, indicating that Subversion does not know anything about them. You will have to add these yourself using svn add.

8. Ensure svn properties are set for newly added files

You will have to set svn properties for added files. In particular for all text files. Run svn propset svn:eol-style native FILE(S). If you have your subversion configuration setting this should be done automatically. Add this list to your ~/.subversion/config file, as described in (TBD need to move from old wiki).

9. Compare your sandbox to contributor's

Now compare your sandbox to the contributor's.

Compare contributors status file with yours

run svn status | diff - CONTRIBUTOR'S_SVN_STATUS_FILE. This should be clean, except that the ordering may be different. You can overcome this by sorting both your own svn status output and the contributor's file with sort before comparing: svn status | sort > MY_SVN_STATUS_FILE; cat CONTRIBUTOR'S_SVN_STATUS_FILE | sort | diff - MY_SVN_STATUS_FILE.
1. If the svn status output was made on Windows vs. Unix, you may need to adjust formatting such as spacing and forward versus backward slashes.

2. If the diff contains new, empty files, patch will not create them for you; you will have to create them yourself using touch FILENAME.
1. After creating such new, empty files, you will have to add them, too, with svn add.

Compare your differences to the patch file

By now your svn status should be equal to the contributor's. Now compare the diff by running svn diff | diff - PATCH_FILE.

The diffs you see could be the following:
1. Revision numbers, if your sandbox was not aligned with the contributor's (see above)

2. Subversion file properties: You will have to set svn properties for added files (see above). Do not automatically take for granted the properties in the contributor's patch file, he/she may have got it wrong; use your own judgement.

3. Actual code diffs: Inspect and figure out what/why.
1. Files may be examined in different order by svn diff on your machine and the contributor's; not sure if there's anything we can do about that.

10 Update to the head with svn update

Check that there are no conflicts.

11. Build the code, running tests

more detailed instructions here..

Run tests if you are not confident that the contributor's/reviewer's actions are sufficient.
1. Check that tests pass. If they do not:
1. Check if the same failure is in the continum build reports (if so the problem wasn't introduced by the patch).

2. Verify that the failure is not caused by the patch.

12. Commit the patch

Commit the patch with svn commit. Use either --message, --file or --editor-cmd
1. Include the following in the commit message:

1. The ID of the JIRA issue. Make sure you use the format GERONIMO-NNN so that JIRA picks it up.

2. Some text explaining what the patch does (typically snipped from the JIRA issue).

3. The contributor's name/email.

13 Send out email

Send out an email to let people know you have committed the patch.