

DCMF OSS Information

IBM Rochester Messaging Team — Version 1.2

Maintenance Policies

Scope of the Policies

As the IBM DCMF messaging team begins to move the source code into an open community, the initial external repository is the primary repository for the production code (the external prod), and it will be the central authoritative code resource for the DCMF code. Not all repositories need conform to any given plan, but the external prod needs to be managed in a way that allows developers to follow the progress of the project.

There should be branches to handle all the interesting code distribution. It is possible to ensure that the repository avoids an over-zealous focus on the BGP tree—though that is the current platform for this OSS effort. This discusses a naming system to avoid confusion, aiding the IBM messaging team's goal of making this an effective cross-platform messaging layer.

Branch names

This project should have a straight-forward and documented method for creating the branches to be housed on the external prod repositories. It would be a silly idea to create branches whose purpose isn't clear to those looking at the code. This is the suggested list for the initial branches:

- master

To the best of my knowledge, this is the only default branch. Without it, I'm not sure how the default is selected after a clone operation. Since this is the first branch new developers will see, it should be relatively stable and easy to build. Documentation changes and most anything committed early in a release will be added to this branch, since it wouldn't be a serious threat to system stability.

- BGP/next

This would be the home of most new and novel development. As the team works to bring in major patches, they would go into this branch for stability testing.

- BGP/new

The new branch would be a quickly changing branch that would be for quick integration of untested and/or unfinalized patches. This would not be a serious development branch, but merely a convenient place to dump new patch submissions.

- BGP/IBM_V1R2M1 (for example)

Branches of this form would be for official IBM production releases. These branches will help the project managers to track progress, and they may help control changes. While there is no true requirement that IBM do its release work in the public eye, it should not involve much extra work—certainly not from the technical end. Developers will likely appreciate the advanced feature warning that will come from watching the branch develop. It should be possible to create tags for the official release and subsequent fixes (e.g. BGP/IBM_V1R2M1_fix_2001).

- *BGP/topics/whatever*

For each of the major development topics, the maintainers can create a branch for review. This doesn't need to be done for small patches or one-time changes, but the big ideas with much discussion would do well to have a special branch.

- *platform/whatever*

Branches for other platforms would be managed at the behest of the members of that development team.

Patch Submission Guide

When patches are submitted, they effectively have two sets of rules. The first is the generic procedural and legal issues, which are detailed elsewhere. The second are the rules that relate specifically to the use of git in this effort. These guidelines are designed to follow the common systems of other healthy communities using git, and to ensure that everyone is treated equally.

- Patches are generated using the “git format-patch” command (<http://www.kernel.org/pub/software/scm/git/docs/git-format-patch.html>).
- The patch should contain a “Signed off by” line that indicates your support for the code in question. This is generally added by using the “-s” flag with the commit and format-patch commands.
- At the start of the comment, there should be an indication of the issue/requirement/bugtrack report that lets the Patch Butlers (PBs) reference the error tracking system to understand the context for the patch. This also helps the Project Managers follow the progress being made on certain problems. Currently, the commit message should match these two equivalent regular expressions:

```
/^Issue [0-9][0-9]*(,[0-9][0-9]*)*: /
/^Issue \d+(,\d+)*: /
```

At this time, external contributors will need to work with a PB to find a valid issue number to use.

- The included comment should explain the purpose for and method to the changes being made. Since the commit comment is the sole textual explanation included in the email, it is reasonable to require that it communicate the point of the patch.
- The patch should be made against the latest master or */next branch to ensure that it can be easily applied by others. Unless there is a good reason for the not doing this, it will help avoid conflict headaches.
- The patch should be publicly sent to the mailing list (<mailto:dcmf@lists.anl-external.org>) to ensure that everyone with an interest in the topic will be able to comment. The git tool “send-email” (<http://www.kernel.org/pub/software/scm/git/docs/git-send-email.html>) handles the email. Note: it is best to use the --smtp-server option, since sendmail emails may get blocked by the list software.
- Once you pull the email off the list—or store it into an mbox via your favorite command-line client—you can use “am” (<http://www.kernel.org/pub/software/scm/git/docs/git-am.html>) to apply the patch to your tree.