# Submitting Changes

## Lustre Quality

A lot of work has been put into stabilizing Lustre over the years, and we have evolved processes to ensure we maintain the stability that so many sites worldwide now rely on for their site-wide production filesystems. In many sites, when Lustre stops working many millions of dollars of computing equipment sits idle, so stability of Lustre is the top criterion for all development.

The processes described below ensure that careful attention is paid prior to proposed changes being landed, whether the engineer works for Intel or some other community organization.

This presentation by Eric Barton (from SC10) elaborates on these themes.

## Landing a Feature to a Community Release

The community produces feature releases every 6 months. Check the Community Lustre Roadmap to find out when the next one is scheduled. At the beginning of the development cycle, the features that will be included into the upcoming release are decided, and a landing schedule is worked out to ensure that not all of the features try to land in the week before the code freeze. The feature code freeze may be as early as 3 months prior to the release date, depending on the number and scope of features that are to be landed.

### First Steps

Before starting to think about the logistics associated with developing your feature **it is imperative to share your plans with the Lustre community before you start work.**

You should check the community development wiki to see if anyone is already working on something similar. If someone is then add yourself as a watcher to the JIRA ticket and offer to collaborate.

If you are unable to find a match, then open a new JIRA ticket outlining your plans, including the intended purpose of the development and any initial thoughts on design. Then, mail the lustre-devel mailing list to draw attention to the ticket. This will alert other community members of your intentions and may well result in potential collaborators stepping forward.

If you choose not to do this you may find that you are either duplicating work with someone else, or that your code needs to be reworked to accommodate other changes occurring in the same part of Lustre code.

While features will not be scheduled for landing into a release until they are already close to completion, it is still important that the features themselves be discussed before or during early development. This allows developers to take into account other changes that are being worked on, to avoid conflicts in network protocol changes, code restructuring, and to ensure interoperability between releases.

It is also strongly suggested that you gain experience in the Lustre landing process by fixing one or more bugs for a maintenance release before attempting to tackle writing a Lustre feature. Feel free to ask for suggestions on the lustre-devel mailing list for a suitable bug to get started with.

### Schedule and Timing

Community Lustre releases operate to a "train model". The schedule is fixed and will not wait for features that are not ready in time - they are deferred to the next release.

History has shown that a lengthy stabilization period is needed after all features have landed to work through any bugs introduced by the new code that were not caught by normal regression testing. If there is sufficient testing of intermediate development releases at a large enough scale, and the release branch is stable, additional features may be landed as time permits.

For a feature release scheduled for release in month **T** the schedule is roughly as follows. For more precise dates, keep up to date on the lustre-devel mailing list .

1. **T-7** A call for features is sent out to the lustre-devel mailing list. The amount of change that can be landed for a given release is limited so it is prudent to respond early if you feel that you will have a feature that warrants consideration for inclusion. Expect to be asked to provide the information on the Feature Landing Checklist below - either completed or with estimates as to when any missing portions will be completed. Typically, feature development is already well underway before a feature is scheduled for landing.
2. **T-6** Initial review of candidate features to define the scope of the release. A test plan is created and the Community Lustre Roadmap is updated. A landing schedule is created so that feature landings are spaced out to make it easier for intermediate testing to identify when features introduce regressions. If serious regressions are found when a feature is landed then it will be reverted from master until the problems have been addressed. It should be obvious that not all changes can land in the last weeks before the feature freeze, hence the requirement that features already be close to completion before scheduling them for landing.
3. **T-3** Feature Freeze - feature landing is finished, bug fixes only from now on. The Community Lustre Roadmap is updated.
4. **T-1** Code Freeze - critical bug fixes only from now on. A release candidate (RC) is tagged and release testing commences
5. **T0** GA announced and RPMs available for download from the Whamcloud download site

### Feature Landing Checklist

1. High level **design** has been reviewed and signed off by a senior Intel engineer
2. **Test plan** has been reviewed and signed off by a senior Intel engineer. The test plan should include performance testing, interoperability, and any feature-specific tests that may fall outside normal testing.
3. **Results** from executing the test plan uploaded into Maloo
4. Proposed revisions to the **manual** have been provided
5. The criteria from the Patch Landing Process Summary are met

## Seeking Guidance

Please alert the community via the lustre-devel mailing list if you need some extra guidance in getting your patch submitted for the release.