Committer's Rules

Commits and Release Engineering

Committers should follow these guidelines when deciding, which branch to use for committing the patches and when to commit.

Branches and Release Engineering

1. The SVN repository consists of the following areas:
    a. **trunk** (equivalent to CVS HEAD), where the current development code base is found. This area is not always guaranteed to be in usable state, some occasional breakage may occur, some parts of the code base may not work properly or at all. This is the area for developers, the "bleeding edge", and usually not suitable for stable production - average users are discouraged to use it, unless they miss some functionality available only here, and are prepared to face some hardships (such as the lack of documentation, the issue of setting up a development environment, bugs, etc).
    b. **branch-x.x** branches, where the code from each release is put for further maintenance. These areas contain code, which is considered "stable", i.e. at the point of release it was known to be working well, *within the limits of functionality available for that release*. The code here is also maintained in a well-working state for a certain period after release, but only minor fixes are applied here in order to provide a solid product with the functionality of the given release. Normally, no new functionality should be added to the maintenance branches. It is unacceptable to introduce changes to this branch, which would break the compatibility with the earlier code within the same branch.
    c. any other temporary branches (such as e.g. "mapred"), which serve as temporary repository for the work to be merged with the trunk at a later stage. You should not expect anything functional here, unless the developers explicitly ask for help in testing and integration.

2. The trunk is the area, where active current development occurs. New features and enhancements are first committed here.

1. This requirement helps to minimize the risk of losing new features and enhancements somewhere on the branches, because as the time goes it is more and more difficult to forward-port them from the past branches to the trunk.
2. If some changes are invasive and would result in prolonged periods of breakage, they probably need more development time before they are integrated with the trunk. If you want other developers to join you in work, it's a good idea to put these changes on a temporary branch to be merged later with the trunk.

3. If there are important features or fixes, which will benefit majority of users, these can be back-ported to release branches, after they have been committed to the trunk (if appropriate). The back-porting process should involve extensive testing to ensure that the code on the Release branch remains stable and production-quality. It is unacceptable to commit code, which breaks the build process, or is known to be unstable. Users will expect from the Release branches to be stable and working with production quality at all times.

Backward compatibility

Committer's checklist

Things to check before commit.