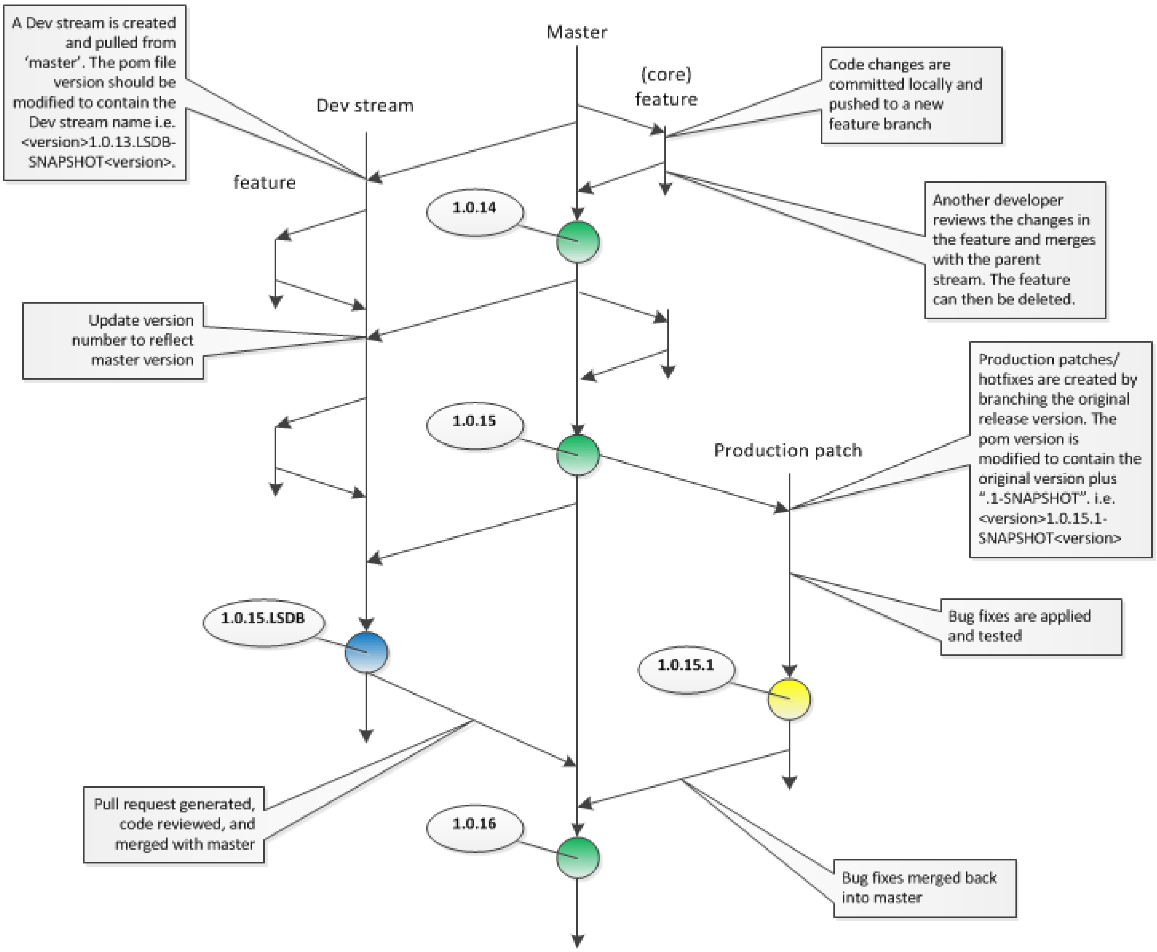
Branching Strategy

Git branching strategy



A Dev stream is created and pulled from ‘master’. The pom file version should be modified to contain the Dev stream name i.e. <version>1.0.13.LSDB-SNAPSHOT<version>.

Code changes are committed locally and pushed to a new feature branch

Another developer reviews the changes in the feature and merges with the parent stream. The feature can then be deleted.

Update version number to reflect master version

Production patches/ hotfixes are created by branching the original release version. The pom version is modified to contain the original version plus “.1-SNAPSHOT”. i.e. <version>1.0.15.1-SNAPSHOT<version>

Bug fixes are applied and tested

Pull request generated, code reviewed, and merged with master

Bug fixes merged back into master

The coloured circles denote a tagged release carried out by the maven release plugin on Team City. These releases are stored in Nexus RKYCTools Releases

All builds carried out between releases are stored as SNAPSHOT versions in Nexus RKYCTools Snapshots

It is assumed that we will only have one set of IDP instances running in Production and that all release builds will be taken from the master branch, except where there is a need for a Production patch/hotfix which will be taken from the Production patch branch.

POM versioning between branches

The pom versions need to be manually modified when pulling from master into a new branch (except for feature branches) and should not be merged back into the master branch at any time. Since Team City will build a Dev stream branch or a Production patch branch this ensures that artifacts generated from master are not overwritten in Nexus.

Master branch

The master branch is intended to be used for creating production candidates. Any feature branches destined for master will be for core functionality, bug fixes etc. All other functionality is merged into master from the Dev stream branches and from the Production patch branch.

Dev stream branches

The Dev stream branch allows new functionality to be developed without impacting on the master until all functionality is in place and has been tested. Multiple Dev streams can then be progressed in parallel, for example TPOS, Bond Pricing, LSDB, etc.

A new Dev Stream can be created and seeded with the latest code from master. The pom file version should be changed so as not to overwrite the master artifacts in Nexus.

Feature branches

When committing changes, these should be applied to a new feature branch and pushed. A pull request is then created allowing for a review of the changes before they are merged with the master branch or Dev stream branch.

After a successful merge the feature branch can be deleted.

A feature branch is created locally on the developers machine and starts with the name ***feature-*** . The significance of this is that it does not trigger a Team City build and will therefore not produce an artifact to upload to Nexus.

This process should be used with master and with any Dev stream branches in order to enable code reviews prior to entering the build mechanism.

Production patch branch

In order to support a production release with hotfixes, without introducing new features, we need to create a production release branch.

The bug fixes are applied to the production release branch and built in Team City. The hotfix should be tested in DEV and CI in order to prove it's validity and before it is released as a production candidate.

The production release branch should also be merged into Git 'master' to ensure the bug fix is also applied to future releases.