Freeze and Slush

This is a draft (more of a collection of ideas than a plan/guideline)

As a growing community, to be efficient, we must have a common understanding of Where to Commit, but we must also get a common understanding of **When** to commit.

This page will attempt to describe a consensus on when to commit so we can all converge and be ready to release according to the Lifecycle.

See also: Where to commit

Reverse-planning / Work-Back Schedule

- x time before, what needs to be done

This part needs to be much precise

when	what
1 month before	technical release (feature-freeze) where all translations (with automatic syntax checking) and promotional material can be prepared.
2 months before	All minor new features should be in trunk so they can be tested in time for the release.
3 months before	All major new features should be in trunk so they can be tested in time for the release.

The big underlying questions:

- Will it be good enough in time for the next scheduled release?
- What are the risks of introducing regressions? The closer we are to the release, the lower the risk your contributions should be

In case of doubt, reach out to the release coordinator. These are questions that come to mind:

- If you do introduce a bug, will we find quickly and will you fix very quickly? (ex.: within 48h) (or are you the dump-and-disappear-and-let-someone-else-cleanup-my-mess type of developer)
- If you do introduce a bug, is it for something localized (ex: change css for one theme) ? or can it affect everyone (ex.: login process)
- Are you an experienced developer and you "know" where the risk areas are?
- Are you affecting a lot of files?
  - Did you test every single file? (mass search and replace sometimes go wrong)
  - Which can make it difficult to review
  - Which can make the merge from stable to trunk more difficult
- Are you introducing a database change?
  - In general, DB changes should be done early in the process
- Has this code been used in production for a reasonable time-frame?

Types of changes

Major changes that affect the whole application.

These should be done fairly early in a cycle. So we have several months to iron-out all the issues. A backward-compatibility layer should be provided. Ex.: Permission Revamp, new modules system, etc. It's a good idea to start these in an Experimental branch.

Major change to existing features.

How will upgrades be handled?

New self-contained features

These can arrive quite late. If they are not ready for prime-time, just tag as experimental.

Inspiration

https://wiki.gnome.org/ReleasePlanning/Freezes

Related

- Version Lifecycle
- When to Branch
- When to release
- Where to commit
- Tools for Merge Request reviewers

Alias

- Freeze
- Slush
- Risk Freeze
- Bug fix vs New feature