03. Features Proposal

Features

- **Feature:** A logical grouping of code and functionality in a project. A Feature is usually any component or grouping.
- **Top-Level Feature:** A Feature that provides one of the major pieces of functionality delivered by a project. In general, this should not require an understanding of the project's internals to know when/how to install it. Most projects will have a small number of, maybe even only one, Top-Level Features. In many cases the Top-Level Feature will actually only be a meta-feature grouping together lower-level features, which would be less obvious for an outsider to consume.
- **User-Facing Feature:** A Top-Level Feature that somebody looking to install and run TF should know about. They should be able to install the Feature and should be able to tell that it's been installed in the form of new user interface elements, support for new southbound devices, or other mechanisms.

Stable and Extended Features (optional)

- **Stable Feature:** A Top-Level Feature that meets the following criteria. The TSC is expected to review each potentially stable feature during the project's release review and ensure that it meets these requirements. **Note:** Any of these requirements can be waived by the TSC for a given feature. However, the TSC is encouraged to find a way to make the test fit the situation before granting a waiver.
    - The feature must lie within the declared scope of the project.
    - The feature is part of a Mature project (as defined in the project lifecycle).
    - The feature was present in the previous release.
    - The feature is only dependent on other stable features (or sub-components of other stable features).
    - The feature provides adequate documentation.
    - The feature has 75% or higher test coverage.
    - The feature must have at least one automated test for each of:
        - *functionality* (previously called a system test) to show the basic functionality works
        - *cluster compatibility* to show the feature works in a 3-node cluster using the clustered datastore
        - *scalability* to show how large a system, e.g., number of hosts, switches, or links, the feature can handle
        - *performance* to show how many operations, e.g., transactions, flows, linkstate events, per second the feature can handle
        - *longevity/stability* to show the feature can run for a period of time under load without ill effect
        - In each case, the tests must show no unexplained regressions from previous releases.
    - The feature is backward compatible with the previous release of the feature, e.g., any APIs that were not deprecated in the previous release still exist with the same signatures.
    - The feature has no known vulnerabilities that are older than a week and classified as important by the security response team or high by their CVSS score in a CVE. If a fix for such a vulnerability lies outside of TF, the TSC may choose to relax the requirement on a case-by-case basis.
    - The feature commits to providing a migration strategy from the previous release. This will ideally take the form of scripts or automatic upgrade support, but could also come in the form of documentation.
- **Service Release:** A top-level feature that is a part of the release and does not meet the Stable Feature criteria.