Bug Fix and Release Policy

This document describes the process for how and when bug fixes will be published in any released versions of sipXecs:

Release Procedure

Products go through varying procedures to get released. It's important to understand our terminology when communicating outward. From a 50,000 foot view, the different stages look like:

Development -> Unstable -> Stable -> Release Candidate -> Release

Before a build is pushed to release, the following steps are performed:

1. Product Management coordinates with support and developers and decides there are bugs, enhancements or new features (these are all broken down into 'issues') that need to be delivered to customers.
2. Product Management determines the issues that will get assembled into a release based on severity, need and desired timeline for a release.
3. Developers build code to solve the identified issue(s).
4. Developers test their code and then merge their code into an Unstable build.
5. Code undergoes testing in Unstable by QA.
6. QA accepts the code as working or notes issues with it and sends it back to the developer.
7. Once code is accepted the developer merges the new code into the Stable build.
8. In the Stable build the new code is tested with all other code new & old.
9. As significant additions are moved to Stable, eZuce installs Stable build on our internal communications system to work out issues in a production environment.
10. Once all Issues for a release have been tested in Stable, Product Management informs testing/release engineering to prepare a release candidate.
11. Release Engineering initiates a Release Candidate build from the Stable code repository.
12. Automated Testing team lead coordinates with manual testing team lead on test plan.
13. Automated testing initiates what is called a smoke test.
14. Manual testing initiates manual testing according to test plan.
15. openUC Release Candidate is installed on eZuce's corporate system.
16. Automated testing initiates sanity, regression and load tests (take about a week)
17. If any regressions or bugs are found, Product Management is consulted and development is consulted for a fix or feature clarification.
18. If further coding is required it is completed, tested, merged back into stable and the process starts again at step 10.
19. Once all parties are satisfied with Release Candidate the binaries are Released to customers on the appropriate servers.

Explanation of Unstable:

A build that contains features that are still in development and slated for an upcoming release. Features or bug fixes that make "large" changes are developed on separate code (git) branch called a "feature branch" away from all other code. Once they pass a certain level of testing, then the branch is merged into the Stable branch. This allows features to be integrated with the rest of the system code when they are determined to be in a stable state.

Unstable build code is available to customers for testing from the following binary repositories:

SF binaries: http://download.sipfoundry.org/pub

Look for repo with 'unstable' after the version number.

Explanation of Stable:

Stable builds are essentially the precursor to a Release Candidate. These builds represent the most stable code produced by development to their knowledge. The Stable build includes all bug fixes and any features that have passed an initial testing process. The fixes and features may not however been tested with the system as a whole.

Stable build code is available for testing from the following binary repositories:

SF binaries: http://download.sipfoundry.org/pub

Look for the repo with 'stable' after the version number.

Explanation of Release Candidate:

Once a Release Candidate build is chosen, it goes through an outlined testing process. If no regressions are found during the testing process, the exact set of binaries are copied to the appropriate release directory and user and customers are notified. If bugs are found that are not regressions, release process can continue as normal depending on what the bug is. This decision to continue with a known bug is up to Product Management.

Release Candidate code is available for testing from the following binary repositories:

SF binaries: http://download.sipfoundry.org/pub/stage

Release Policies

1. Source code submissions will have to pass the requirements listed the section **Source Code Submission Requirements and Policies** below
2. Version number will not change. Example: Release 1.2.3 will remain 1.2.3. Each rpm will have a release number that will increment so you will be able to identify when a file is newer. See Version Numbers for more info.
3. Change log will be published describing each bug fix stating explicitly the impact of the bug fix. Administrators should be able to easily decide if and/or when they need to install the update. **TBD: how to build change log and where to publish?**
4. ISOs will **not** be rebuilt for each bug fix. Building ISOs is easy, but testing each of the ISO is not. As a general rule, administrators would be required to install the last released ISO and run yum update to get the latest bug fixes. This is the same policy Linux distributions like CentOS and Fedora follow.
5. After a bug fix is made to source code repository and binaries have been built but before binaries have been published to download.sipfoundry.org, the binaries will be made available for a verification process. There will be a simple installation process any customer can do to install the build and verify bug fix. Once the fix has been verified and published to download.sipfoundry.org, customer's system will not require another update.
6. When a bug fix release is uploaded to download.sipfoundry.org the previous RPMs will no longer be made available for download. Having hundreds of repositories is simply not feasible. It is recommended you keep a local copy of the RPMs you've installed into production should you need them.

Source Code Submission Requirements and Policies

1. Source code for bug fixes will be rejected if they would restrict administrators from using any other set of binaries in the same major.minor version scheme.
2. Contributors will be required to sign a contri
3. Each source code fix made to a released branch will go through a code review process by a predetermined team. The team will look for validation of the fix, but more importantly look for regressions or areas the could cause system crashes. If a code fix can be refactored to be safer, then it will be recommended as such.
    a. Release 4.4.0 Source code Review Team: George Niculae, Joegen Baclor, Douglas Hubler