# Plan for release of the tranSMART Package V1.1 (aka Postgres Version with Oracle support)

Here is an outline of the major elements of a release of the tranSMART "package": that is, transmartApp, the ETL support, instillation (scripts, documentation and supporting material), test data, test codes and scripts, documentation, and tutorials. This is followed by draft description of the narrative of the release process.

# Outline.

### Schedule

* Code Freeze - As soon as possible (May 15 was proposed; but not possible)
* Alpha Release - "immediately after" the code freeze
* Beta Release - June 15
* Final Release - Sept 15 (also Beta release of V1.2?)

### What Elements does the Beta Release cover?

* Web Application: transmartApp (core-integration branch, Rmodules, transmart-core-api, transmart-code-db, transmartApp-config (postgresMigrate branch)
* Database and Loading: tranSMART-ETL, transmartApp-DB (postgresMigrate branch)
* Standard test/demo data

### What Additional Elements does the Final Release cover?

* Instillation package (scripts, documentation, demo data, other supporting material)
* Publication of test data, test code, test scripts.
* User, Administrator Documentation
* Usage Tutorials

### Remaining Tasks; Open Questions; Known Risks

* Minor fixes
  + transmartApp-config (postgresMigrate -> master)
  + transmartApp (core-integration -> master; Imperial -> master)
* Pressing needs
  + need a complete release plan
  + need more tests (GUI tests especially)
  + need to review/gather bug reports
  + need to fix bugs
* High risk - ETE
  + Note clear who the audience is or what the "out of the box" methods should do
  + Documentation is out of date!
  + What methods should be used (Kettle, Python, etc.)
  + No User-friendly interfaces. ??? - Is this needed? (lots of work)
* High risk - Administrative Interface
  + What is the status of this?
* Testing - Continuous Integration
  + Fixed test data required and known
    - several options
    - need to identify, document, and make part of archive
* Documentation
  + documentation and tutorials needed by Sept 15
  + list of known elements that need documentation
    - code and data architecture
    - install process
    - test/demo procedures/expectations
    - administrative interfaces and processes
    - user experience and workflow examples
    - tutorials

## **Release Process.**

### Roles in the release process.

* Developers
* Testers
* Certification Manager
* Documenters
* Beta Committer (Committee)
* Release Manager

### Release process narrative

All elements that belong in the code freeze and also are part of the main core development will have to be merged into either the master branch or a clearly defined release branch (i.e. "release-candidate-summer-2013") of their respective archives.

Once the release candidate branches have been identified, the code freeze will be implemented by creating a new branch in all the git hub archives (something obvious like "code-freeze-summer-2013"). After that, generally, only bug fixes can be committed to that code freeze branch; other commits must be justified in a group meeting, and voted in by a consensus of the developers in the meeting.

When the code freeze goes into effect, it becomes the alpha release.

Testing of the alpha release has three major phases: (1) Instillation of test data, web application, and the ETL process and code; (2) test design; and (3) test certification. Obviously, testing is iterative and bugs will be found. Bugs will be reported in JIRA.

The JIAR tickets are the task list for fixes to the Alpha release. Large fixes taking more then a day to implement, test, and commit should be done on a separate branch. From a practical point of view, it would be best to always do fixes on a separate branch with a merge/commit at the end of the day, that way the fix branch exists in the case the fix process becomes unexpectedly long.

From time to time, roughly weekly, the alpha release will be retagged.

As appropriate, alpha release fixes will be merged with the ongoing master branches.

When the Alpha release is certified on all the tests; then it is beta-ready.

The beta release will be done into a separate read-only archive. In practice, each contributing branch will be tagged as the v1.1 beta release and a fork of archive will be created with that tagged "release" version as the head. This archive will be read-only and have only a small number of committers (three?).

Fixes against the beta release will continue in the main archive using the same bug-report-in-JIRA method as with the alpha release. But, with special attention to the fact that GUI, and application functionality and behavior should remain as stable as possible to support the creating of documentation and tutorials.