**Creation of SVN Stream and general SVN info**

**CONFIDENTIALITY & COPYRIGHT**

|  |
| --- |
| This document contains information confidential to the GT product/project.  Any form of reproduction, transmission for use of this document or its contents is not permitted without prior written approval from GridNode Pte Ltd.  All rights are reserved |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 4 October 2008 | 1.0 | Initial | Tam Wei Xiang |
| 30 Jan 2013 | 1.1 | Touch up section 4, 5, 6 | Tam Wei Xiang |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table Of Content**

Table of Contents

1 Introduction 4

1.1 Purpose 4

1.2 Reference 4

2 Creation of GT4.2.X 5

3 Type of Stream 5

4 General flow when working on particular task of a project 5

5 General flow when making release 5

6 Appendix 6

6.1 Organization of the tag (Baseline) 6

6.2 Organization of the Binary files 6

# Introduction

# Purpose

This document shows the high level overview on creation the GT4.2.X stream. It will also touch on some general information regarding working in the SVN, the step of release etc.

# Reference

# Creation of GT4.2.X

The GT4.2.X stream is branched off from GT4.0.2 (Tag 4.0.2) as the base and we merges the existing latest changes from various stream including 4.0.2.X, 4.X and 4.1.1.

# Type of Stream

We usually create two stream when we working on a particular project. In 4.2 case, we have **integration stream 4.2.X** and **development stream 4.2.x\_dev**. This can help to minimize the changes of a particular developer to commit the malfunction code into the project main stream directly.

The development stream can be branched off from the integration stream we created earlier.

# General flow when working on particular task of a project

1. The developer will be working on the development stream meaning they will check out a working copy from the project development stream.
2. The team lead will create the ticket for the developer of a particular task
3. The developer will accept the ticket (The status of the ticket will be changed to \*)
4. The developer completed the work and his/her own unit testing, he/she will commit the code into development stream of that project. (Appropriate comment is required: EG delivered ticket #88)
5. The developer attached SVN revision number he/she committed into the ticket and assign back the ticket back to the project integrator for verifying the changes.
6. If the project integrator found any bug/undesired behavior of the delivered task, he/she will re-assign back the ticket to that developer.

# General flow when making release

1. The project integrator will obtain all the changes from the developer into his/her development working copy.
2. Validate all changes is met the requirement
3. Merge all changes from the development stream to the working copy of integration view
4. Make release from the local copy of integration view
5. Go through some testing prior releasing to QA. Testing on installer is required in this stage.
6. Quick check/integration test passed, commit the changes into the integration stream.
7. Tag the current project source code as alpha release eg 4.2.0**a** under “*https://dcvssn02.asianconnect.com/crimsonlogic/etrade1/gtke/****tags/built***” directory.
8. Release a binary for QA to test.
9. QA Passed. QA team will notify the Project Integrator the test is completed.
10. Project integrator will tag the project source code as tested eg 4.2.0 (*NOTE: without the 'a'*) under “*https://dcvssn02.asianconnect.com/crimsonlogic/etrade1/gtke/****tags/tested***”.

# Appendix

# Organization of the tag (Baseline)

**Initial** – Similar to the clear case initial baseline → Incremental baseline.

**Built** – This version has gone through the Unit Testing and integration testing. This version will be released for QA testing. It will be tagged as 4.2a (alpha release), 4.2a1 (if QA failed)

**Tested** – QA has completed the testing, and it passed

**Released** – The version will be released for the customer.

*This is needed for GridLite team since after they release from QA, they need to go through with Gridnode PS. If passed, they will switch to this stage.*

# Organization of the Binary files

Place the binary release under [\\nas-etrade\releases\_prd\gridtalk](file:///\\nas-etrade\releases_prd\gridtalk). Organize the binary release based on the product version.