### 

For: GoldenSource Professional Services

Project: Best Practices

Subject: Automated Build and Deploy

Table of Contents

[1 Introduction 4](#_Toc30671394)

[1.1 Objectives 4](#_Toc30671395)

[1.2 Intended Audience 4](#_Toc30671396)

[1.3 Best Practice Governance 4](#_Toc30671397)

[2 What is DevOps? 5](#_Toc30671398)

[2.1 Third Party Tools 5](#_Toc30671399)

[2.2 In-house Tools 5](#_Toc30671400)

[3 Current Process 6](#_Toc30671401)

[3.1 Problems 8](#_Toc30671402)

[4 Proposed Process 9](#_Toc30671403)

[5 Requirement & Analysis 10](#_Toc30671404)

[6 Development 11](#_Toc30671405)

[7 OOB Tools 12](#_Toc30671406)

[7.2 UI Installation Archive 12](#_Toc30671407)

[7.3 GEM 12](#_Toc30671408)

[7.4 EDMA 13](#_Toc30671409)

[8 Jenkins 14](#_Toc30671410)

[8.1 Plugins 14](#_Toc30671411)

[8.2 Creating a Pipeline 14](#_Toc30671412)

[8.3 Pipeline Script (Groovy) 15](#_Toc30671413)

[8.4 Pipeline Steps 15](#_Toc30671414)

[8.5 Build Parameters / Running the Pipeline 16](#_Toc30671415)

[9 Subversion 20](#_Toc30671416)

[9.2 SVN Structure 20](#_Toc30671417)

[9.2.1 build 20](#_Toc30671418)

[9.2.2 workstation 20](#_Toc30671419)

[9.2.3 engines 20](#_Toc30671420)

[9.2.4 EDMA 20](#_Toc30671421)

[9.2.5 configuration 21](#_Toc30671422)

[9.2.5.1 tasks/workflows/events/engines/queries 21](#_Toc30671423)

[9.2.5.2 vendordefinitions 21](#_Toc30671424)

[9.2.5.3 reference 21](#_Toc30671425)

[9.2.5.4 resources 22](#_Toc30671426)

[9.2.5.5 sql 22](#_Toc30671427)

[9.2.5.6 GEM 23](#_Toc30671428)

[9.2.5.7 EDMA 23](#_Toc30671429)

[9.2.5.8 Environment Configuration 23](#_Toc30671430)

[9.3 Branch & Merge 25](#_Toc30671431)

[9.3.1 Make a Branch of “trunk/custom” 25](#_Toc30671432)

[9.3.2 Merge changes back to “trunk/custom” 27](#_Toc30671433)

[9.4 Strategy 29](#_Toc30671434)

[9.4.1 Release Merge 29](#_Toc30671435)

[9.4.2 Production Merge 29](#_Toc30671436)

[9.5 PSG-OD Collaboration 29](#_Toc30671437)

[9.5.1 Merge Responsibility in Cases involving common components by PSG/OD as below 30](#_Toc30671438)

[10 Template Subject Section 31](#_Toc30671439)

[10.1 Summary 31](#_Toc30671440)

[10.2 Contributors 31](#_Toc30671441)

# Introduction

The Professional Services – Configuring Web Services Best Practices Guide is the primary source for GoldenSource PSG team members reference before configuring web services

## Objectives

This document provides instructions on how to best configure web services

## Intended Audience

The intended audience for this is primarily for the GS PSG team members although can also be helpful for our product management and development teams as well

## Best Practice Governance

This section describes the process in place to ensure the Best Practices themselves are:

* Documented and Understood
* Maintained
* Used by the Professional Services Group
* Audited to ensure consistency across implementations

# What is DevOps?

**DevOps** is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably.

## Third Party Tools

**JIRA** - Bug tracking and agile project management.

**Jenkins** - Automation server that helps to automate the non-human part of the software development process.

**Tortoise SVN** – Subversion maintains current and historical versions of source code.

**ANT** - Automating software build processes.

**Docker** - Software platform to create, deploy and manage virtualized application containers on a common operating system.

**Jboss** – Application Server on which the GoldenSource software runs.

**Nexus** - Manages binaries and build artifacts across the software supply chain, making it easy to distribute the software.

## In-house Tools

**GEM** – GoldenSource Environment Manager. To facilitate the automation of build and deployment of GS components/packages.

**UI Installation Archive** – To facilitate the deployment of the application ears on to the application server (Jboss)

**Download Center** – Online Repository of OOB GoldenSource Packages

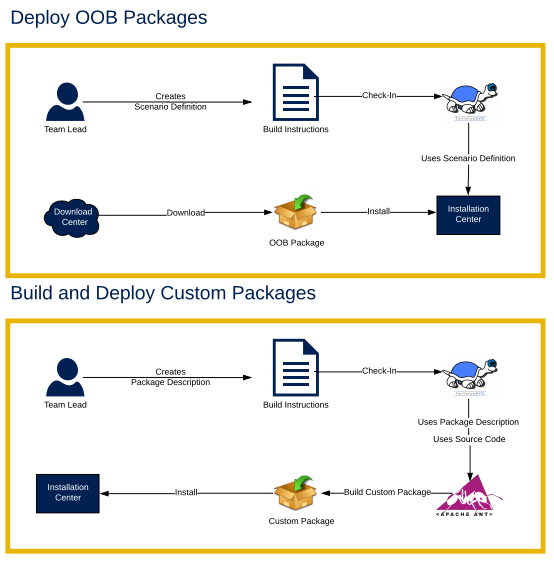
**Workstation** – User Interface or Front-End application of GoldenSource

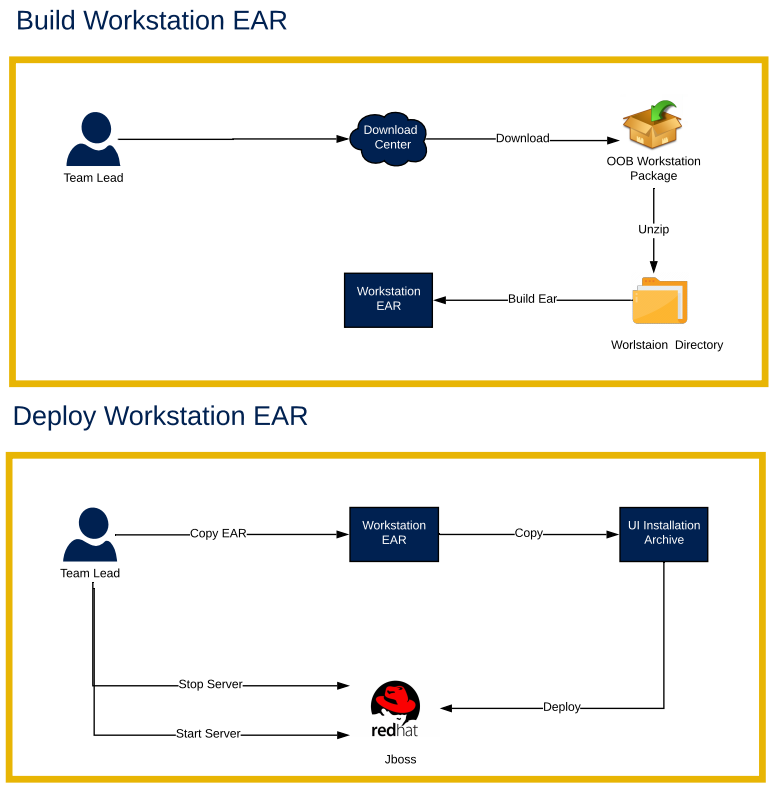
**Orchestrator** – Back – End application of GoldenSource

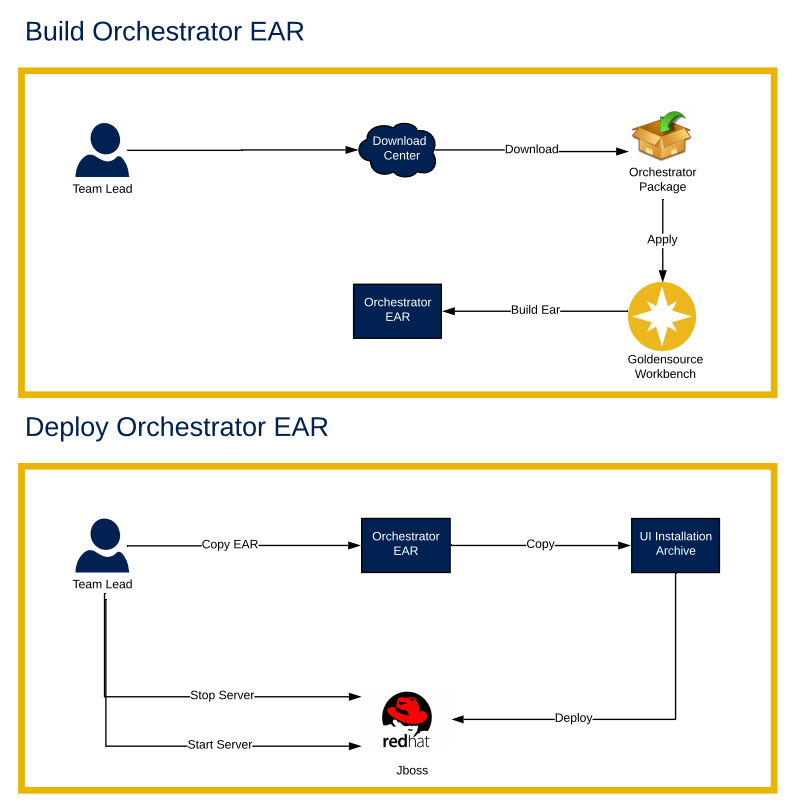
**EDMA** - Automation testing tool for GS

# Current Process

The current process is manual build and deployment of code from one environment to the other



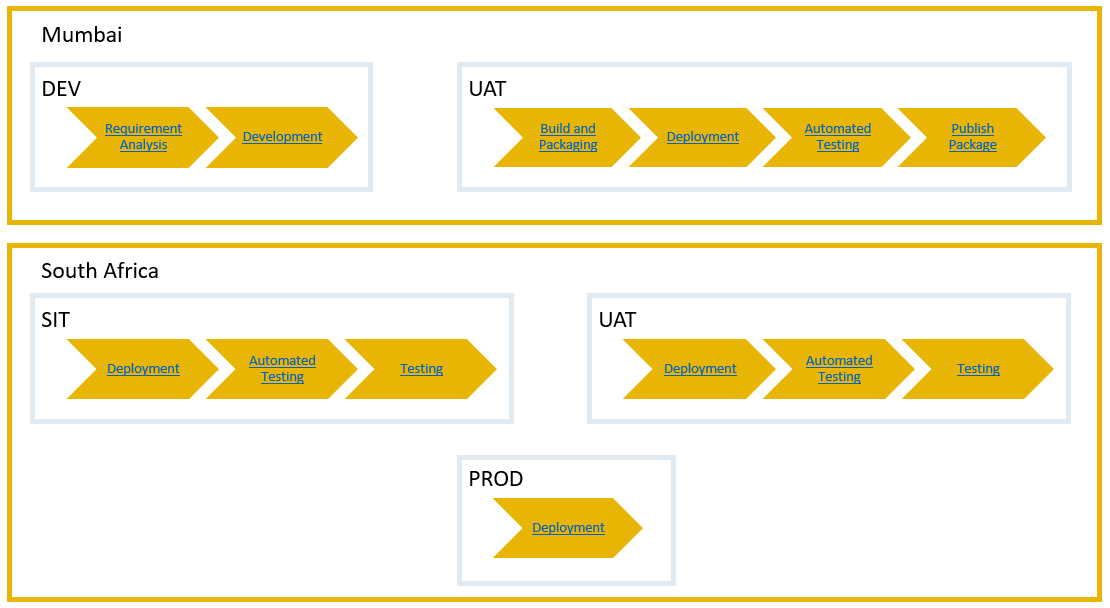




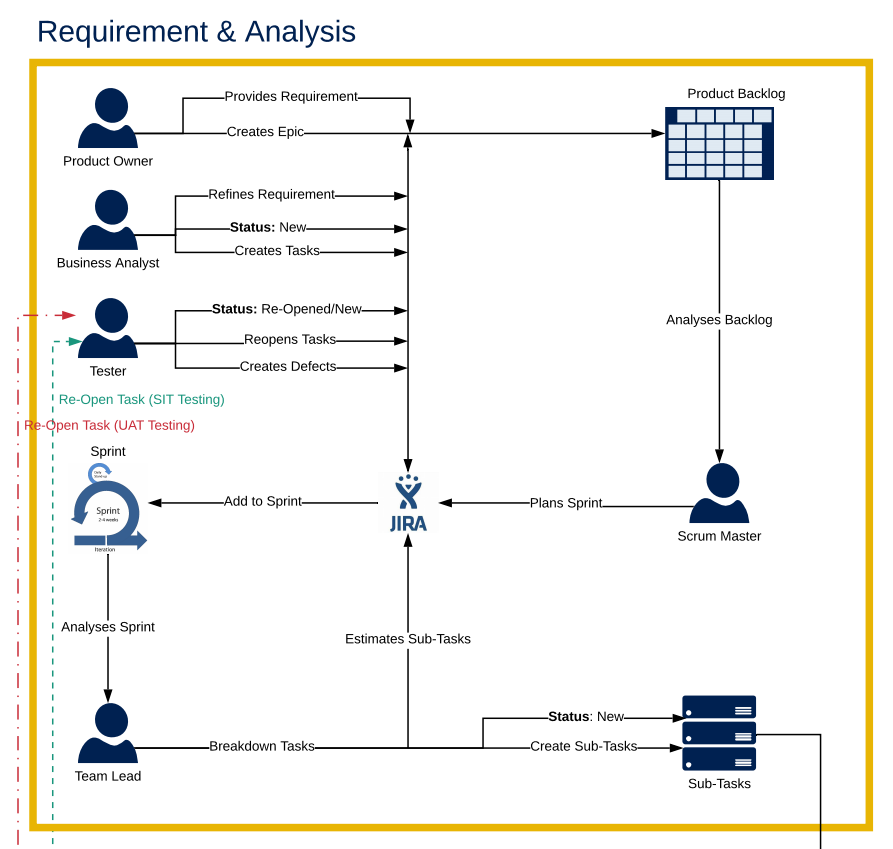
## Problems

|  |  |
| --- | --- |
|  | Manual |
| Human Intervention | High |
| Risk | High |
| Deployment Time | High |
| Deployment Effort | Tedious |
| Productivity | Decreased |
| Quality | Low |

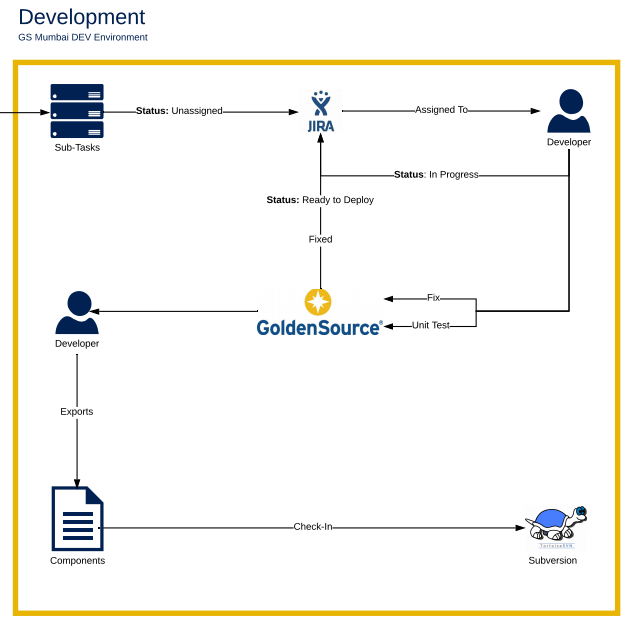
# Proposed Process



# Requirement & Analysis



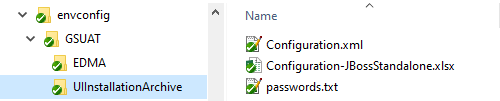
# Development



# OOB Tools

## UI Installation Archive

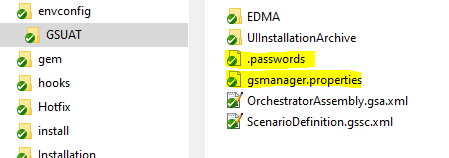
* UI Installation archive is the in-house deployment software provided by GS
* It currently supports deployment of GS application on 3 different application servers
  + Jboss
  + Websphere
  + Weblogic
* Environment specific configuration can be fed into the Configuration.xslx provided to perform deployment on the application server
* OOB UI Installation copy in use should be maintained in SVN under
  + trunk\UIInstallationArchive
* Environment specific configuration related to that environment should be stored under
  + trunk\envconfig**\<ENV\_NAME>\**UIInstallationArchive



* This document does not contain information on how to configure UIInstallation Archive. For more information on this refer to UI Installation Archive OOB documentation

## GEM

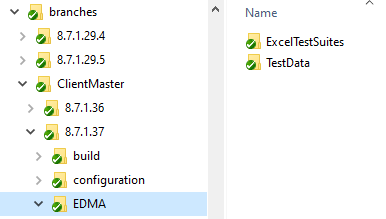
* GEM or Goldensource Enterprise Manager is the in-house build-installation-deployment software provided by GS
* GEM can perform the following tasks
  + Install OOB Packages
    - via a custom scenario definition file stored under trunk\envconfig\**<ENV\_NAME>**\ScenarioDefinition.gssc.xml
  + Install Custom Packages
  + Install or Upgrade Workstation Container
  + Install or Upgrade Workbench Container
  + Build Workstation Container
    - Apply Ant scripts like apply all sqls load models etc
  + Build EAR’s
    - Workstation
    - GSO Designer
    - Drools
    - Orchestrator
  + Copy EAR’s to Application Server
  + Deploy EAR’s via UI Installation Archive
  + Start/Stop Servers
* OOB Installation of GEM should be maintained in SVN under
  + trunk\gem
* Environment specific configuration related to gem is stored under
  + trunk\envconfig**\<ENV\_NAME>**



* GEM properties are stored in gsmanager.properties
  + GEM properties are self-explanatory + have some write up beside it.
* Configurable GEM Properties have been exposed to the individual performing the deployment on the Jenkins Pipeline screen. This will be explained in detailed in the Running the pipeline process

## EDMA

* EDMA is Automation Test Suite provide by GS
* OOB Installation of EDMA should be maintained in SVN under
  + trunk\EDMA
* Environment specific configuration related to EDMA is stored under
  + trunk\envconfig**\<ENV\_NAME>\**EDMA
* Test Cases related to the release will be stored under
  + Branches\<Project Name>\<Release Version>\EDMA



* This document does not contain information on how to configure EDMA Tool. For more information on this refer to EDMA OOB documentation

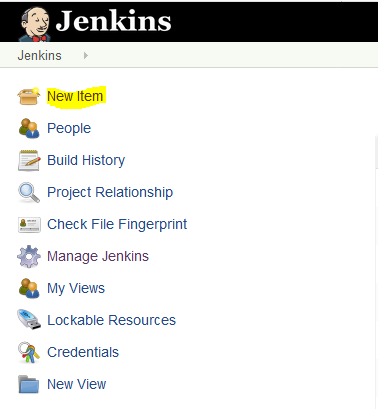
# Jenkins

* Automation server that helps to automate the non-human part of the software development process.

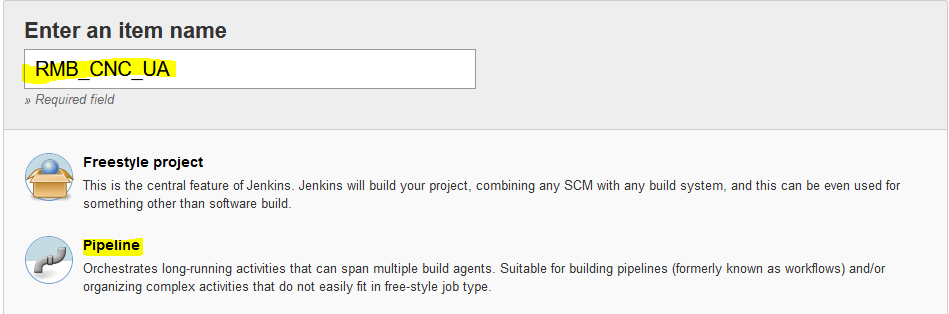
## Plugins

|  |  |
| --- | --- |
| **Name** | **Version** |
| [Parameter Separator Plugin](https://wiki.jenkins-ci.org/x/W4BWB) | 1.0 |
|  |  |

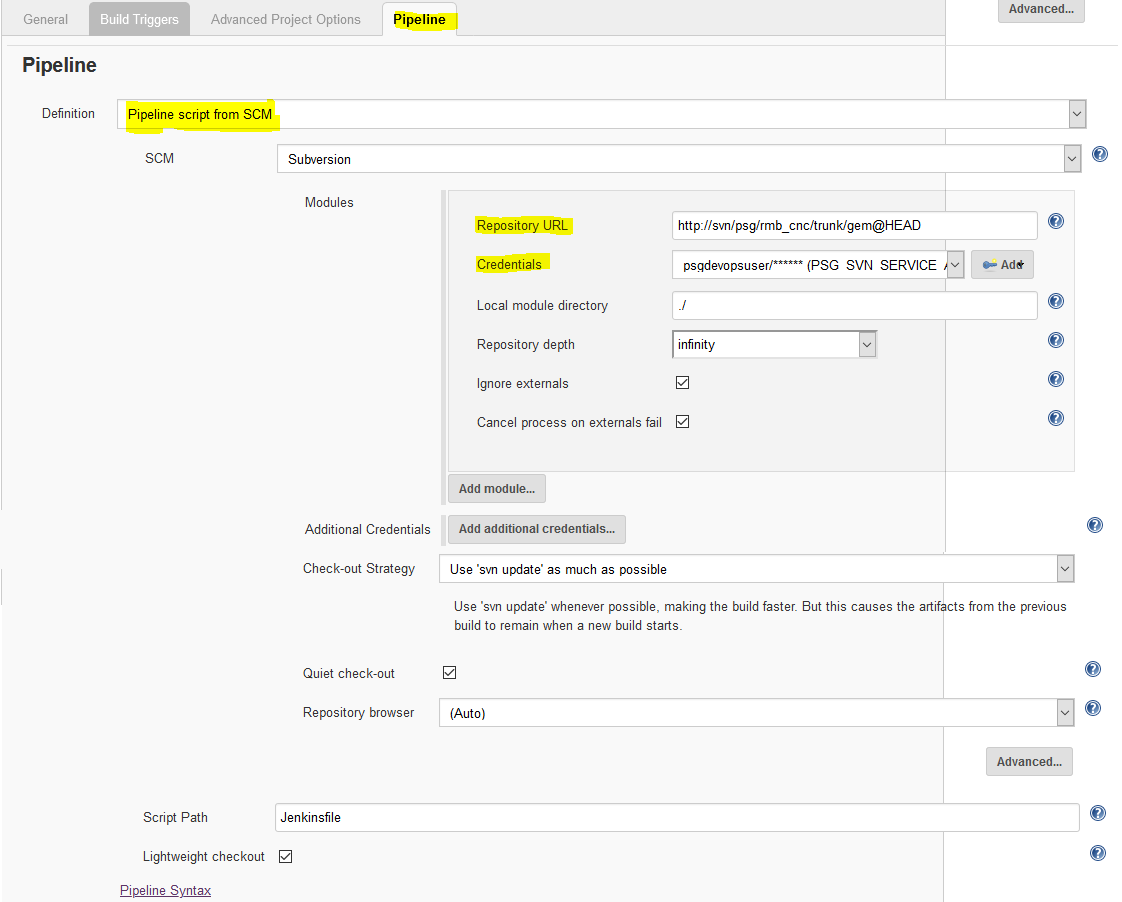
## Creating a Pipeline



* Click on New Item



* Enter a name of the Project and select Pipeline project



* Click on the Pipeline Tab
* Select Pipeline script from SCM (Source Control Management)
* Select SCM as Subversion
* Provide the Repository URL and Credentials

## Pipeline Script (Groovy)

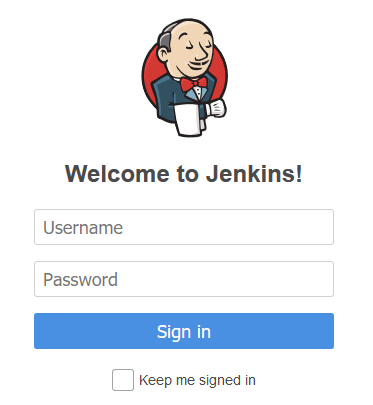
* Pipeline script is written in groovy and will be placed on SVN under
  + trunk\gem\Jenkinsfile

## Pipeline Steps

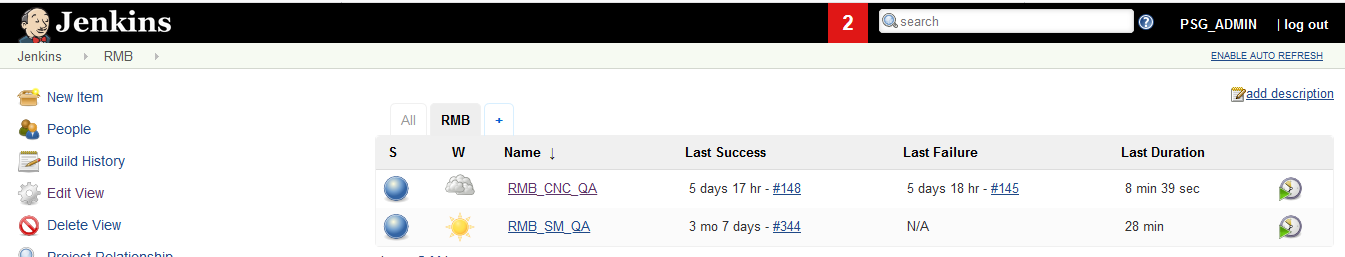
* Pipeline can perform the following tasks
  + Build Custom Package
  + Copy Custom/Environment Specific Configuration from SVN to the Build Server
  + Update GEM Properties depending on user selection
  + Invoke GEM
    - Install OOB Packages
    - Install Custom Packages
    - Install or Upgrade Workstation Container
    - Install or Upgrade Workbench Container
    - Build Workstation Container
    - Build EAR’s
    - Copy EAR’s to Application Server
    - Deploy EAR’s via UI Installation Archive
    - Start/Stop Servers
  + Invoke EDMA Sanity
  + Invoke EDMA Patch
  + Invoke EDMA Regression
  + Send Emails on Build Status

## Build Parameters / Running the Pipeline

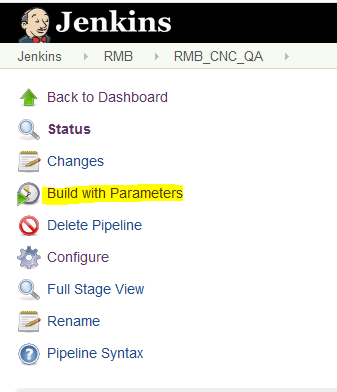
* Login to Jenkins

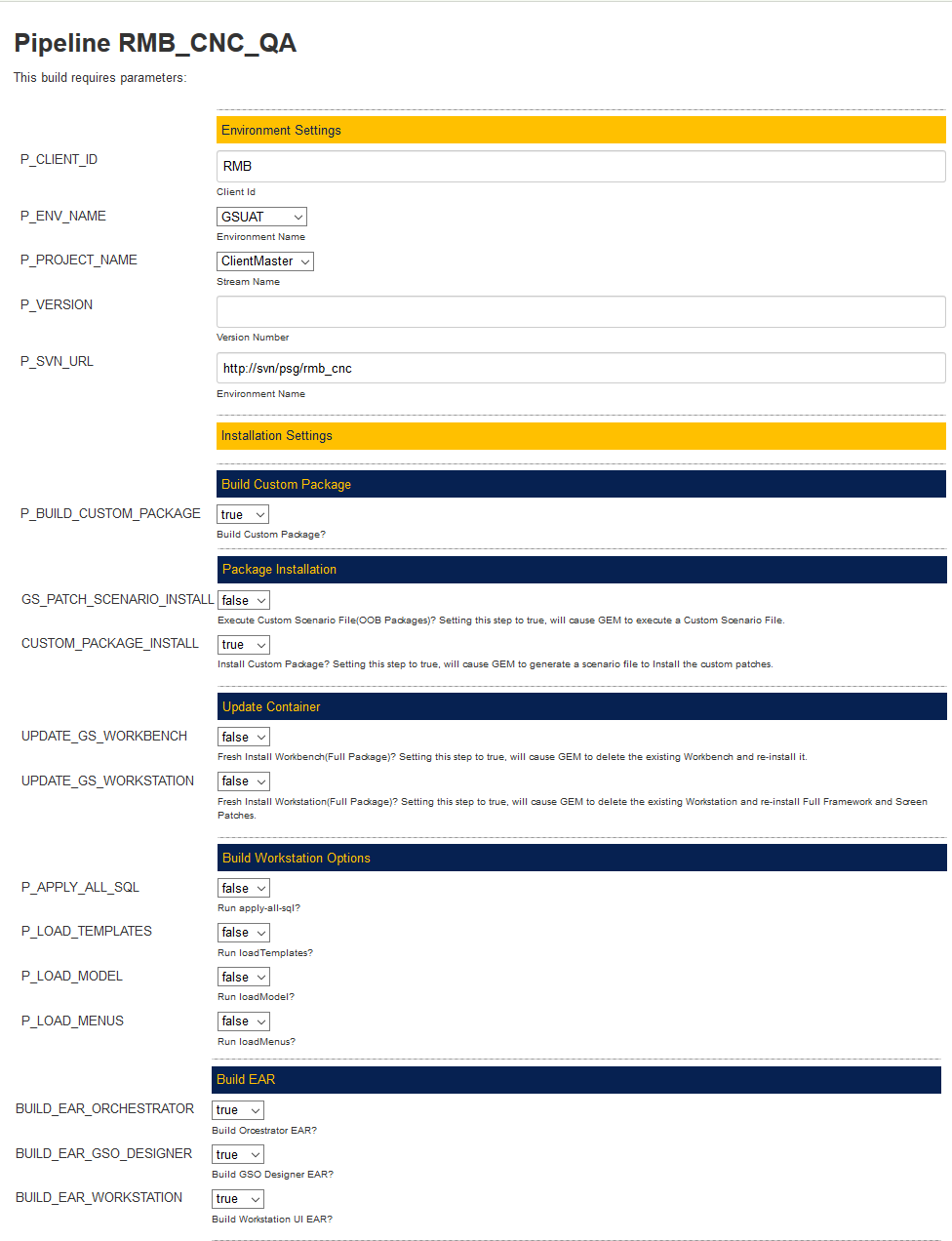


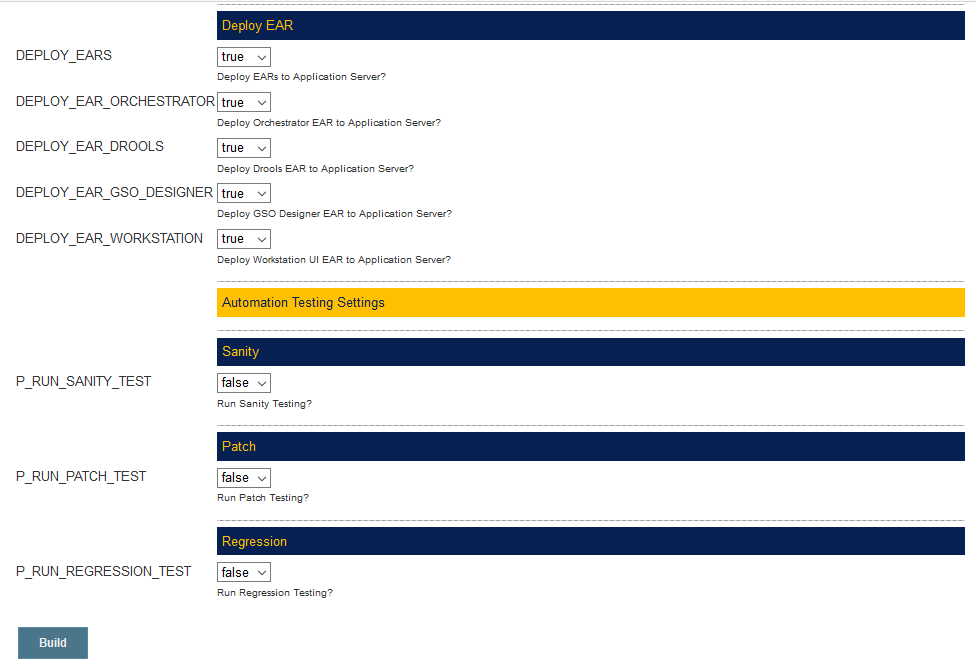
* Go to the desired Jenkins Project



* Click on Build with Parameters







|  |  |
| --- | --- |
| Property | Description |
| P\_CLIENT\_ID | Client Id. For Example RMB, ASX, DB etc  Used in the Emails sent out + Publish Package to Nexus |
| P\_ENV\_NAME | The Environment Name   * + - QA     - SIT     - UAT     - PROD   This is used to fetch environment specific configuration from SVN. Details like Database details, Application server details of that environment |
| P\_PROJECT\_NAME | Project Name  This is used to distinguish between two different running projects of the same client.  Under SVN it will be branches\<PROJECT\_NAME> |
| P\_VERSION | Version Number of the Release  This is used to provide the release number of the branch which needed to be built  Under SVN it will be branches\<PROJECT\_NAME>\<VERSION> |
| P\_SVN\_URL | URL of the SVN that you need to connect to in order to fetch all the code |
| P\_BUILD\_CUSTOM\_PACKAGE | Build Custom Package?  true/false |
| GS\_PATCH\_SCENARIO\_INSTALL | Execute Custom Scenario File (OOB Packages)? Setting this step to true, will cause GEM to execute a Custom Scenario File. |
| CUSTOM\_PACKAGE\_INSTALL | Install Custom Package? Setting this step to true, will cause GEM to generate a scenario file to Install the custom patches. |
| UPDATE\_GS\_WORKBENCH | Fresh Install Workbench(Full Package)? Setting this step to true, will cause GEM to delete the existing Workbench and re-install it. |
| UPDATE\_GS\_WORKSTATION | Fresh Install Workstation(Full Package)? Setting this step to true, will cause GEM to delete the existing Workstation and re-install Full Framework and Screen Patches. |
| P\_APPLY\_ALL\_SQL | Run apply-all-sql? |
| P\_LOAD\_TEMPLATES | Run loadTemplates? |
| P\_LOAD\_MODEL | Run loadModel? |
| P\_LOAD\_MENUS | Run loadMenus? |
| BUILD\_EAR\_ORCHESTRATOR | Build Orcestrator EAR? |
| BUILD\_EAR\_GSO\_DESIGNER | Build GSO Designer EAR? |
| BUILD\_EAR\_WORKSTATION | Build Workstation UI EAR? |
| DEPLOY\_EARS | Deploy EARs to Application Server? |
| DEPLOY\_EAR\_ORCHESTRATOR | Deploy Orchestrator EAR to Application Server? |
| DEPLOY\_EAR\_DROOLS | Deploy Drools EAR to Application Server? |
| DEPLOY\_EAR\_GSO\_DESIGNER | Deploy GSO Designer EAR to Application Server? |
| DEPLOY\_EAR\_WORKSTATION | Deploy Workstation UI EAR to Application Server? |
| P\_RUN\_SANITY\_TEST | Run Sanity Testing? |
| P\_RUN\_PATCH\_TEST | Run Patch Testing? |
| P\_RUN\_REGRESSION\_TEST | Run Regression Testing? |

* To execute the Pipeline fill in parameters
* Click on Build

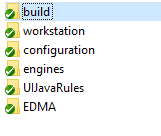


# Subversion

* Branch and Merge process will be used to version code.

## SVN Structure

* Automated package build uses the exact folder structure and names (case sensitive).



### build

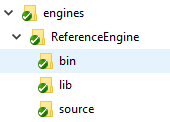
* Contains build instructions for building a package
* build.properties contains information on the package name and version

### workstation

* Contains the custom workstation components
* GEM will simple copy/replace the contents of this folder on top of the OOB workstation container

### engines

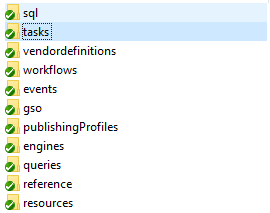
* Contains the custom engine jar and source code



### EDMA

* Contains the custom test cases that need to be executed for automation testing.

### configuration

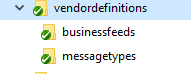


### tasks/workflows/events/engines/queries

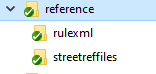
* + Store in .gsp format
  + 1 component per file
  + File name of the package should be same as the component name

### vendordefinitions

* + Store in .gsp format
  + 1 component per file
  + File name of the package should be same as the component name
  + Contains 2 sub folders
    - messagetypes
    - businessfeeds

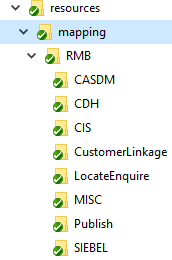


### reference



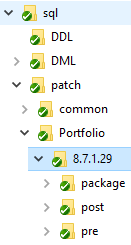
* rulexml folder contains the
  + StreetRefMsgSet.xml and the msgtype.dtd
* Streetreffiles
  + XCFG files like
    - IDUniquenessCheckMatchKeySet
    - VSH
    - TimeSeries
    - etc

### resources



* Contains the complete path for the mdx or resource in the database

### sql



* + Patch sql will follow the below structure
    - sql/patch**/<Stream>/<version>/**
    - e.g. sql/patch/Portfolio/8.7.1.25/
  + SQLs will be categorized into 3 folder depending on the time these sqls have to be executed
    - **pre** – SQLs that need to be installed prior to patch installation. Generally for **cleanup** scripts to prevent the failure of patch installation
    - **package** – SQLs that need to be installed along with the custom patch. Generally for **starterset** scripts.
      * End the sql file name with “\_GC” if you want it to run only in Goldencopy.
        + PM-121\_SQL1\_GC.sql
      * End the sql file name with “\_VD” if you want it to run only in VendorDetails.
        + PM-127\_ALID\_VD.sql
      * If no postfix is found in the end of the file it will be run in GC and VDDB.
      * Package folder has 3 sub-folders
        + DDL – executed using sql task via IC
        + DML– executed using sql task via IC
        + PLSQL – executed using sqloperations task via IC
      * Package folder contains **ExecutionOrder.txt** which is used for the order of the SQLs in the PackageDescription.xml
    - **post** – SQLs that need to be installed after the custom patch is deployed. Generally for **migration** scripts.
  + Every SQL should have the below pattern
    - **<JIRA\_ID>\_<ScriptName>.**sql
    - e.g. CDU-620\_MigrationScript.sql
  + SQLs in pre and post will have 3 **sqlplus** compatible scripts
    - runAll\_IR.sql
    - runAll\_GC.sql
    - runAll\_VD.sql
  + These SQLs will be run directly by GEM via **sqlplus**
  + Scripts that need to be run only once will have the below coding. The below script adds an entry to SCTL and prevents it from re executing, the same way OOB operates.

DECLARE

scriptExecuted CHAR (1);

BEGIN

BEGIN

SELECT 'Y'

INTO scriptExecuted

FROM FT\_O\_SCTL

WHERE PATCH\_ID = 'CDU-620\_MigrationScript.sql';

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN

scriptExecuted := 'N';

END;

IF scriptExecuted = 'N'

THEN

\*\*\*\*Actual Script Here\*\*\*

*--SCTL Entry*

INSERT INTO FT\_O\_SCTL (PATCH\_ID,PATCH\_SEQ\_NUM,RELEASE\_TMS,PATCH\_APPLIED\_TMS,PATCH\_STAT\_TYP,PATCH\_ID\_CTXT\_TYP)

VALUES('CDU-620\_MigrationScript.sql',1,sysdate,sysdate,'A','RMB:CSTM');

ELSE

***DBMS\_OUTPUT.put\_line*** ('Script CDU-620\_MigrationScript.sql already executed');

END IF;

END;

/

### GEM

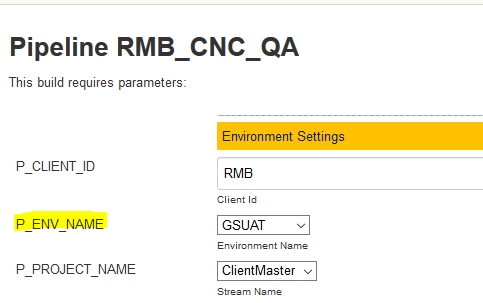
* + svn: **trunk\gem**
  + This folder will contain the OOB gem installation
  + Jenkins will fetch the GEM installation from this folder

### EDMA

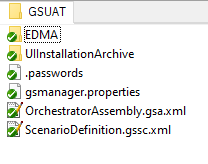
* + svn: **trunk\EDMA**
  + This folder will contain the OOB EDMA installation
  + Jenkins will fetch the EDMA installation from this folder

### Environment Configuration

* + svn: trunk\envconfig\***<Environment Name>***
    - Environment Name should be same as parameter **“P\_ENV\_NAME“** chosen in Jenkins
    - Example: **GSUAT**



* + Contains Environment specific configuration.



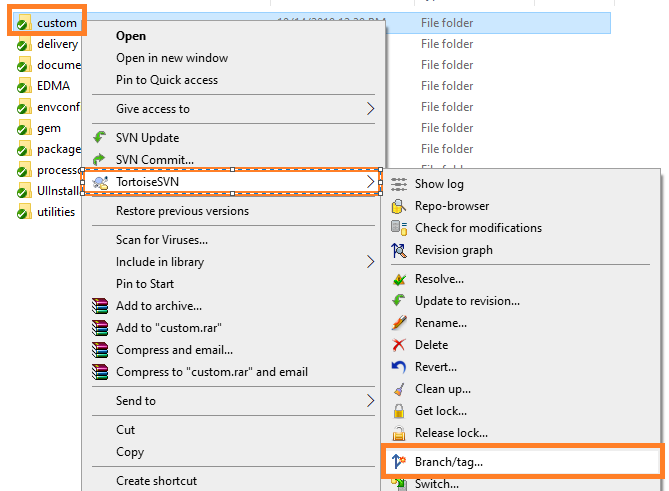
* + GEM
    - .password is the password file used in GEM
    - gsmanger.properties file used in GEM
  + OrchestratorAssembly File
  + ScenarioDefinition File
  + EDMA
    - EnvConfig.properties
    - Variables.properties
  + UIInstallationArchive
    - Configuration.xml
    - Configuration-JBossStandalone.xlsx
    - passwords.txt

## Branch & Merge

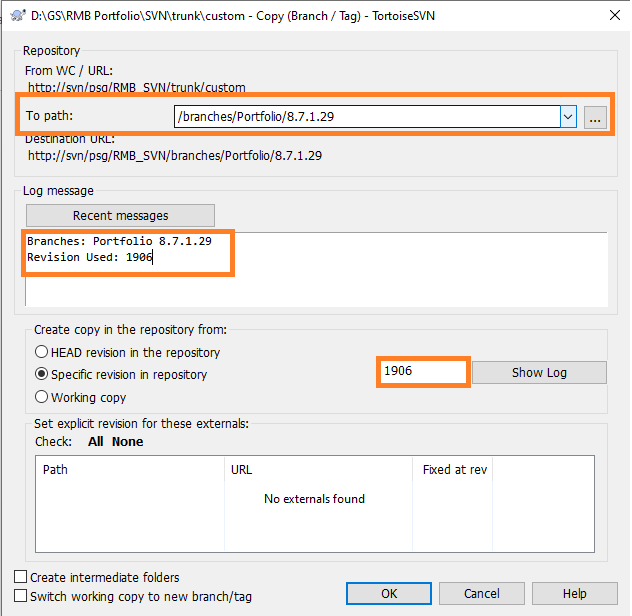
* Trunk – The **trunk** is the main line of development in a SVN repository.
* Branch - A **branch** is a side-line of development created to make larger, experimental or disrupting work without annoying users of the trunk version.
* Tag - **tags** are markers to highlight notable revisions in the history of the repository

### Make a Branch of “trunk/custom”

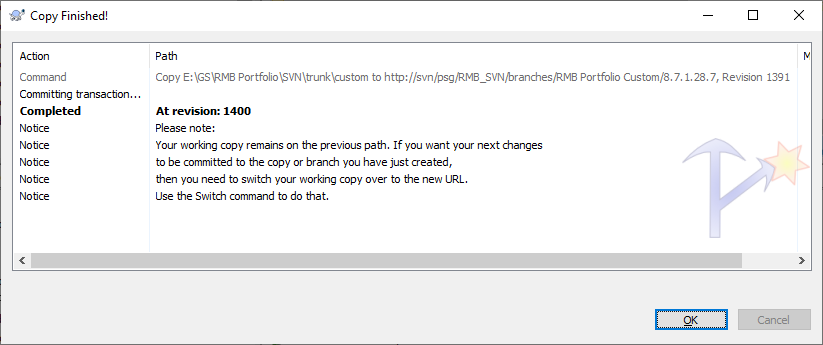
* + Right Click on custom folder > TortoiseSVN > Branch/Tag



* + Choose the folder you want to create the branch in. Use the Syntax: **/branches/<Stream>/<Version>**
* /branches/Portfolio/8.7.1.29 for Portfolio
* /branches/SecMaster/8.7.1.9 for SecMaster



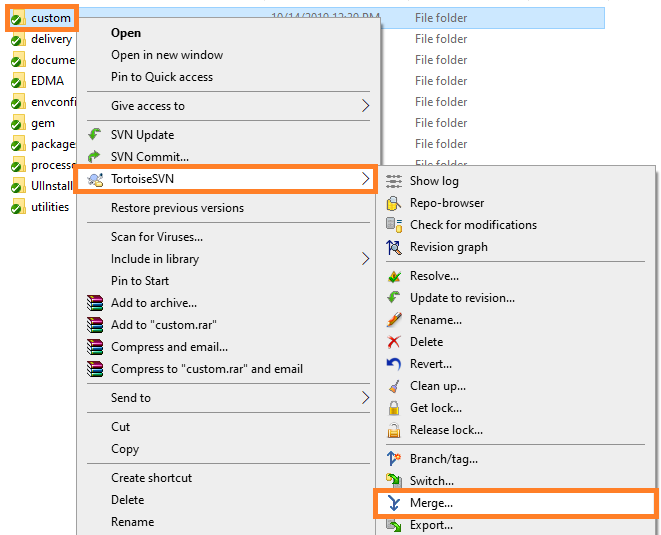
* + Add the Log Message (Mention the Revision Number for easier tracking later.)
  + Click OK



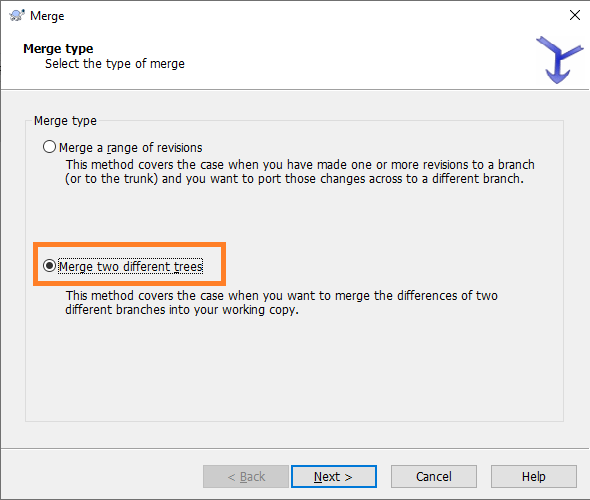
* + Update the branches folder. A new copy of the custom folder should be available now. All changes will be done on this folder.

### Merge changes back to “trunk/custom”

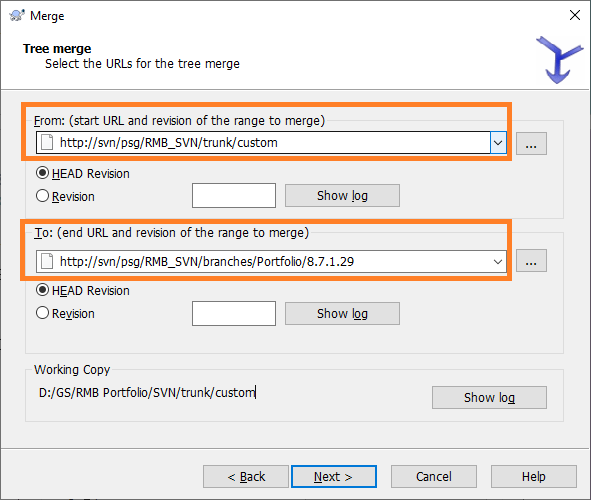
1. Right click on “trunk/custom” > Tortoise SVN > Merge



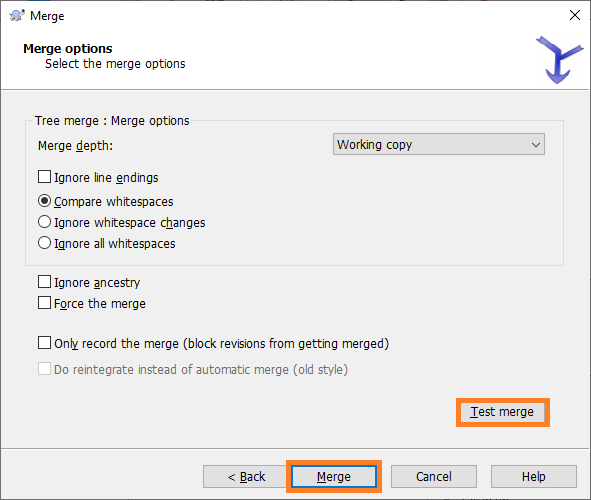
1. Select “Merge two different Trees”



1. Select the 2 revisions to be compared and merged
   * + **From:** /trunk/custom
     + **To:** /branches**/<Stream>/<Version>**



1. Click on Test Merge to validate the merging files



1. Click on Merge to commit the changes to trunk.

## Strategy

### Release Merge

* In this strategy branch will be merged into trunk when the code is delivered to the Client.

### Production Merge

* In this strategy branch will be merged into trunk only when the code is deployed into Production.
* This will be used on clients utilizing OnDemand as their implementation partner. In such cases OnDemand will also be making changes to Code. They will follow the same process of Branch and Merge. The only difference is Ondemand changes are more frequent than PSG changes and the delivery cycles far shorter. In “Release Merge”, code is committed to trunk after it is released to client. Having released code which is not in PROD but in a lower environment will cause issues as Ondemand will create a branch of trunk to make their fixes i.e. they will be working on copy of code that is not in Production.

## PSG-OD Collaboration

Whenever there is a Project Phase (PSG) running alongside a BAU Phase (OD) we will follow the Branch and Merge functionality of SVN

**Trunk = Production**

* Going forward SVN trunk folder will only maintain the latest copy of code that is currently in Production

**Branch**

* If PSG/OD needs to start a new deliverable, they need to take the latest copy of Productionalized code from Trunk and create a branch.

**Merge**

* As soon as Code is deployed via a Package into production it will have to be merged into trunk

Additionally

* PSG and OD to meet on Monday every week to discuss all foreseeable code changes moving into Production and the Release Dates into SIT,UAT and PROD
* All components being changed need to be vetted by PSG/OD to see if it is impacting the current release.
* If any common components are identified between the releases they need to be merged by the team going in on a later date into Production.

### Merge Responsibility in Cases involving common components by PSG/OD as below

**Case 1:**  Same Day Deployment PSG/OD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Team** | **Package Name** | **Release** | **SIT** | **UAT** | **PROD** |
| PSG | RMB Portfolio Master Custom | 8.7.1.29 | 10-Jul | 20-Jul | 30-Jul |
| OD | RMB Portfolio Master Custom OD | 8.7.1.1 | 28-Jul | 29-Jul | 30-Jul |

OD to merge the code into their Package. OD Package to run after PSG package.

**Case 2:**  Deployment PSG before OD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Team** | **Package Name** | **Release** | **SIT** | **UAT** | **PROD** |
| PSG | RMB Portfolio Master Custom | 8.7.1.29 | 10-Jul | 20-Jul | 30-Jul |
| OD | RMB Portfolio Master Custom OD | 8.7.1.1 | 28-Jul | 3-Aug | 7-Aug |

OD to merge the code into their Package.

**Case 3:**  Deployment OD before PSG

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Team** | **Package Name** | **Release** | **SIT** | **UAT** | **PROD** |
| PSG | RMB Portfolio Master Custom | 8.7.1.29 | 10-Jul | 20-Jul | 30-Jul |
| OD | RMB Portfolio Master Custom OD | 8.7.1.1 | 20-Jul | 21-Jul | 22-Jul |

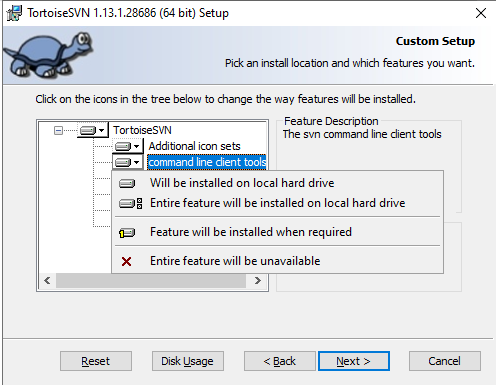
PSG to merge the OD code into the Package and Re-Release Package with Higher Version.

# Prerequisites

## Windows Automation Testing Machine

### SVN

* Make sure you add the “command line client tools” while installing SVN on the Windows Automation Testing machine



### Java JDK

* Install Java 1.8 or higher

### Jenkins Agent

# Template Subject Section

## Summary

## Contributors

|  |  |  |
| --- | --- | --- |
| **Source** | **Name** | **Date** |
| PSG | Desouza | 23-Jan-2020 |
|  |  |  |
|  |  |  |
|  |  |  |