

BPM Developer’s Guide for SVN

Version 1.2.1

August 14, 2020

# Table of Contents

[Table of Contents 2](#_Toc38619710)

[Summary of Changes 3](#_Toc38619711)

[1 Summary 4](#_Toc38619712)

[2 Core Code 4](#_Toc38619713)

[3 Corp and Project Code 4](#_Toc38619714)

[4 Locks 4](#_Toc38619715)

[5 SVN Keywords 5](#_Toc38619716)

[5.1 PL/SQL 5](#_Toc38619717)

[5.2 Unix scripts 5](#_Toc38619718)

[5.3 Kettle Code 6](#_Toc38619719)

[5.4 Enabling Keyword Substitution 7](#_Toc38619720)

[6 Folders 7](#_Toc38619721)

[6.1 /BPM 7](#_Toc38619722)

[/createdb 7](#_Toc38619723)

[/patch 7](#_Toc38619724)

[6.2 /trunk 7](#_Toc38619725)

[/archive 7](#_Toc38619726)

[Kettle8 7](#_Toc38619727)

[7 Contact Center Code 8](#_Toc38619728)

[7.1 Contact Center Product 8](#_Toc38619729)

[DB Scripts 8](#_Toc38619730)

[Patch 8](#_Toc38619731)

[ETL 8](#_Toc38619732)

[Unix 8](#_Toc38619733)

[Project specific 8](#_Toc38619734)

[7.2 Trunk 8](#_Toc38619735)

[/archive 8](#_Toc38619736)

[Kettle8 8](#_Toc38619737)

# Summary of Changes

|  |  |  |
| --- | --- | --- |
| Change | Version | Owner |
| Initial Version | 1.0 | Devin Heimbuch |
| Add Contact Center section. | 1.1 | Lavanya Gopal |
| Improve clarity, formatting, and examples. | 1.2 | Randall Kolb |
| Correct misspelling. | 1.2.1 | Randall Kolb |

# Summary

The purpose of using SVN is to have a current copy and history of every script used in the production environment.

# Core Code

This is code that is installed on all projects part of database set up, BPM Queue, Admin packages, etc.

# Corp and Project Code

Both the database and ETL have Corp code, this is shared code that is deployed to multiple projects without modifications. When looking for a file, look in both “corp” and the project (example: “ILEB”, “NYHIX”, etc.) directories. If a file exists in the project, the project version should be used, if it does not, then the corp version of the file should be used.

You should also look for the “SVN\_FILE\_URL” keyword on the Production server, it will tell you where that file came from in SVN. See examples in section 5 - Versioning below. Before making changes, you should check that the file in SVN is indeed the correct file, correct version, etc.

# Locks

Before making any changes to a file, you should lock it in SVN (SmartSVN -> Locks -> Lock). If a lock already exists, you must first resolve the lock with the developer that currently has it locked. After Production deployment you must release the lock (SmartSVN -> Locks -> Unlock) and write a log entry. The log entry should include the Jira number and a brief description of the change. For KTRs give the steps that was modified, for a package give the procedure or function change. Example “PAXTECH-1234 – Added feature X to package.”

# SVN Keywords

Adding a SVN Keyword header and enabling SVN Keyword Substitution allows you to verify which SVN version of a file actually is deployed to a database or server. You should ensure that the header is added and SVN Keyword Substitution is set for every PL/SQL package, Unix shell script or Kettle code you to deploy.

## PL/SQL

Add this immediately after the package specification header:

*-- Do not edit these four SVN\_\* variable values. They are populated when you commit code to SVN and used later to identify deployed code.*

*SVN\_FILE\_URL varchar2(200) := '$URL$';*

*SVN\_REVISION varchar2(20) := '$Revision$';*

*SVN\_REVISION\_DATE varchar2(60) := '$Date$';*

*SVN\_REVISION\_AUTHOR varchar2(20) := '$Author$';*

Other SQL files such as create table, alter and updates do not need this header since script comments do not persist in the database for these types of SQL.

## **Unix scripts**

Add this to the top of the script:

*# ================================================================================*

*# Do not edit these four SVN\_\* variable values.  They are populated when you commit code to SVN and used later to identify deployed code.*

*#*

*#   SVN\_FILE\_URL = $URL$*

*#   SVN\_REVISION = $Revision$*

*#   SVN\_REVISION\_DATE = $Date$*

*#   SVN\_REVISION\_AUTHOR = $Author$*

*# ================================================================================*

## Kettle Code

Add this in Extended Description (Right click on job or transformation, select properties)

*----------------------------------------------------------------*

*Do not edit these four SVN\_\* variable values.  They are populated when you commit code to SVN and used later to identify deployed code.*

*----------------------------------------------------------------*

*---------------------------------*

*SVN\_FILE\_URL = $URL$*

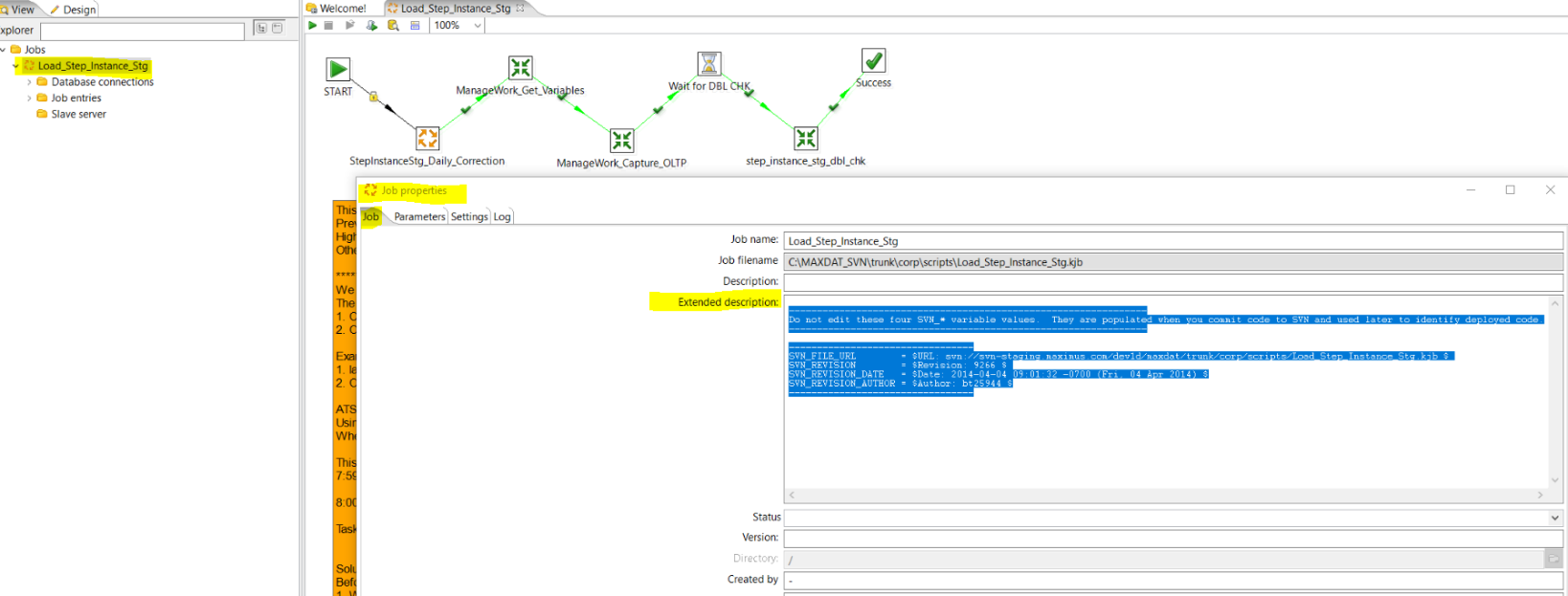
*SVN\_REVISION = $Revision$*

*SVN\_REVISION\_DATE = $Date$*

*SVN\_REVISION\_AUTHOR = $Author$*

*---------------------------------*

Example –



## Enabling Keyword Substitution

You must enable keyword substitution in SVN for new files checked in that use SVN Keywords.

When committing the code to SVN, enable the SVN keyword substitution. This step should be done next after you first commit this file to SVN. Values are populated after commit of any file revisions or initial enabling of Keyword Substitution for this file.

SmartSVN

right-click new file -> Properties -> Keyword Substitution

check Set for:

Author

Date

Revision

URL

OK

Commit changed settings to SVN

# Folders

## /BPM

/createdb - This is for code required to build an exact new replica of production today. Any changes showing in a patch created to add, change, remove database objects, must also be updated in these files.

/patch - This is for code required to update objects or data in the database. Any objects changes made here must also be added to the createdb script. There are specific naming conventions required.

Example: (date/time is to make it unique and sortable, doesn’t have to be exact)

YYYYMMDD\_HHMM\_{optional: ticket #}\_{object\_name\_or\_description}

20191018\_0858\_1234\_EMRS\_D\_PLAN\_update\_ABC.sql

## /trunk

/archive - Will hold all Kettle v4.2 code that is no longer needed. This is going to be done soon.

Kettle8 – All code for ETL server including cron, kettle.properties, shared.xml, Unix scripts, KJB, KTR.

# Contact Center Code

## Contact Center Product

DB Scripts – All code required to build an exact replica of production today should be under svn://svn-staging.maximus.com/dev1d/maxdat/ContactCenter/trunk/kettle/MAXDAT/product/main/scripts directory. Any patch created to add, change, remove database objects, as a part of Enhancement-Defects must also be updated in these files once the deployment in production is successful.

Patch - This are files required to update objects or data in the database, any objects changes made here must also be added to the DB Scripts. There are specific naming conventions required

Example: (date/time is to make it unique and sortable, doesn’t have to be exact)

YYYYMMDD\_HHMM\_{optional: ticket #}\_{object\_name\_or\_description}

20191018\_0937\_1234\_ CC\_C\_CONTACT\_QUEUE \_update\_ABC.sql

ETL - All ETL scripts are under svn://svn-staging.maximus.com/dev1d/maxdat/ContactCenter/trunk/kettle/MAXDAT/product/main/jobs and svn://svn-staging.maximus.com/dev1d/maxdat/ContactCenter/trunk/kettle/MAXDAT/product/main/transforms directories

Unix – All product specific unix scripts are under svn://svn-staging.maximus.com/dev1d/maxdat/ContactCenter/trunk/kettle/MAXDAT/main/bin

Project specific – Project specific scripts would be under svn://svn-staging.maximus.com/dev1d/maxdat/ContactCenter/trunk/kettle/MAXDAT/implementation/CiscoEnterprise/implementation directory under specific project name.

## Trunk

/archive - Will hold all Kettle v4.2 code that is no longer needed. This is going to be done soon.

Kettle8 – All code for ETL server including Cron, kettle.properties, shared.xml, Unix Scripts, KJB, KTR.