Dhanush-Advanced Check-in Script User Guide

Software Version: 8.0
February 17, 2015

Introduction:

Check-in script is used to build and sanity the developer code changes and then check-in the code in to SVN through a build server. The Check-in script will only check-in the code changes after the build and sanity are passed, otherwise mail will be sent to development by providing appropriate build or sanity fail information informing that current check-in request is rejected.

Check-in script Steps:

Please follow the below steps to check-in the code in to the SVN:

Step-1:

1. Create a patch by executing patch creation script(Create_Patch.pl) provided by build team at user PC(terminal). User needs to enter the local code path where user wants to create a patch.(script will ask to enter the path)
   Note: User entering path must point to any of the code base name.
   Ex: http://insvn01:9090/svn/swdepot/Dhanush/SW/Micro-trunk/MSDK/DHANUSH_M_B0_2.1/Native
2. Patch creating script(Create_patch.pl) generates three outputs after execution: *.patch, add_information.log, repository_url.log.(* - indicates code base)

Step-2:

Login to check-in request site(http://192.168.42.46/cgi-bin/login.pl)with the SVN username and password.

Step-3:

Select the request as needed Build Request or Sanity Request by clicking on the Build or Sanity scrollings.

Step-4:

If you select build request, the **Build Process** will be executed. (Check out the code, merge with the user submitted patch and build the code, if any problem found in it mail will be sent)

Step-5:

Enter the appropriate fields on the next page(Build Request Form) – BugID, CodeReview-ID, patchfile, add_information.log, repository_url.log generated from *Step-1*.

Step-6:

Check the No-Conflicts box in the page indicating that no conflicts are identified while generating a patch.(simply code which is submitting has No conflicts occured with the SVN code).

Step-7:

Submit the build request, to perform the build on the required code base. Mail will be send on the build information, whether merge or build is passed or failed.

If the Merge is failed user needs to resolve the conflicts and needs to resubmit as a new request.

If the Merge is passed it continues the check-in process to build the requested code base.

If the Build is failed user needs to rectify the error and needs to resubmit as a new request.
If the Build is passed it holds the check-in process to get the sanity result from the user.

Step-8:
Perform the sanity on the build image shared at certain path provided in the build pass mail. Then Submit the sanity result to build server to check-in the code changes in to SVN.

Step-9:
After Sanity done, login to check-in request site(http://192.168.42.46/cgi-bin/login.pl) with the SVN username and SVN password.

Step-10:
Select the Sanity Request form, then the **Sanity Process** will be executed.

Step-11:
Enter the appropriate information in the fields on the next page(Sanity Request Form) – Check-in ID and provide Sanity Result – Pass or Fail option, Submit the Sanity result log, SVN Check-in comments and select the code base from the drop-down list.
If the Sanity is failed user needs to resolve the issue and needs to submit as a new check-in request.
If the Sanity is passed, code will be check-in to SVN, when SVN revision and code base revisions are same. Otherwise user needs to resubmit a new patch.

Check-in script Steps for other codes which do not require build or sanity processes like docs, backup codes:

Please follow the below steps to check-in the code in to the SVN:

Step-1:

1. Create a Repository info log by running svn info command.
   Ex: svn info some_working_copy > repository_url.log.
2. Create a compressed file (tar or zip or tar.gz)for the local working copy which has all local modifications like svn addition, deletions which are need to go in to SVN at user terminal.

Step-2:
Login to check-in request site(http://192.168.42.46/cgi-bin/login.pl)with the SVN username and password.

Step-3:
Select the request "Others" by clicking on the Build or Sanity or Others scrollings.

Step-4:
Enter the appropriate fields on the next page(Check-in Request Form) – BugID, CodeReview-ID, compressed file, repository_url.log generated from ***Step-1*** and check-in comments.

Step-5:

If the user entered working copy and repository info log are correct then the modified changes with in the working copy will check-in to SVN. Other wise user needs to reenter with correct information.

Advantages:

- Code Check-in's will happen only after sanity and build is done.
- It can accept multiple build requests from multiple users at a time.
- It can handle multiple builds on the build server for the same code base build - from different users.
- Build image shared will have only one build request change, one developer change, one bug fix change.

Limitations:

1. Binary files are not allowed in SVN patches, empty binary file will check-in to SVN.
2. In case of docs, scripts or other code bases which do not need build or sanity, user needs to compress whole directory and this causes the problem in the case of larger size directories.

Suggestions:

This portion is to intimate the user about the wrong possiblities that may be happen, because of the difference from manual to automation.

1. User needs to cross check the code changes, once the patch and logs are generated(status log has the information) before the build request submission.
2. Once the check-in is done, please cross check the list of files that got modified on the SVN.
3. Please do not refresh the page for submitting the same build request, please re-enter the fields.