Finalizing EVSS Components for Sustainment Releases

GitHub/Jira Version – March 30, 2022

# Overview

This SOP describes the process for preparing an EVSS component for a New Sustainment Release after Dev Testing is complete, and then for ongoing development. The following steps are included and must be performed in this order:

1. Create Jira Work Item
2. Finalize EVSS Release
3. Merge Changes and Build Components
4. Deploy & Release Ready Notification
5. Post-Release Preparation for New Builds

In order to accomplish this, the following are prerequisites:

* GitHub EC and GitHub.com access and basic familiarity
* Basic familiarity with GitHub Pull Requests usage
* Authenticated access to the CI Jenkins server and basic familiarity with Jenkins
  + See <https://jenkins.io/doc/book/getting-started/> if needed

# Create Jira Work Item

The first step is to create a Jira work item which will be used by commits related to finalizing the release.

The following process should be used:

1. Create a Jira work item which will be used for version changes in POM files.
   1. Set the Summary to: “Finalize Development Projects for EVSS R<X.Y> Sprint/Release” where <X.Y> is the Release number (for example, at this time the current Sustainment Release is “17.24”.)
   2. Set the work item type to: Task
   3. Set the Component field to: Development
   4. Set the Environment to “CM, INT1”
   5. Set the Release Notes to include: “Description: Finalize components in R<X.Y> for release builds” where <X.Y> is the Release number
   6. Set the label to: CM
   7. Set the Description to the same value as the Summary
   8. Set the Story Points to: 8
   9. Set the Estimated time to: 1d
2. Save the draft work item, then change the Status to Start Working and save again.
   1. Note the Jira work item number to be used later (below.)

**NOTE**: You can also clone the work item from the previous release as a starting point / short-cut if desired.

# Finalizing EVSS Release

At the end of development for each release, the SNAPSHOT versions identifying components that have been updated for this release need to be revised to reflect the release (non-SNAPSHOT) version. This generally involves removing ‘-SNAPSHOT’ from the end in both the component POM files, and the release-definitions master index.

**NOTE**: All changes, SNAPSHOT builds and dev testing MUST be completed BEFORE beginning this process!

## Update Release Definitions

If a component is NOT going to be delivered as part of the release, the version should be reverted to the previous Production release value. If needed, **downgrading versions of any such components is a manual task which must be performed before continuing!** Edit the appropriate release-definition parent-pom file to complete this task.

**NOTE**: There are two release definition packages for each release: the ‘main’ package used by most services and webparts components and the true baseline for the release is names ‘R17.X.Y’. The second one used by a few webparts components and the Portal tier is named ‘R17.X.WLS.Y’. The components using this are behind on some TRM packages and must use older versions of several components for compatibility (but this list should decrease over time.)

If there are components set to SNAPSHOT versions in the release definitions which are NOT going to be delivered, they must be corrected by manually editing the parent POM file(s). Set the version number for each back to the previous Production release version instead of the SNAPSHOT version being worked on by developers. The following process can be used to perform this change:  
  
**NOTE**: If no version changes are needed you can skip ahead as there is no need to edit the file.

1. Browse to the release definitions parent POM folder for the release branch that is ready for builds. These can be found in GitHub.com under the path:  
     
   https://github.com/department-of-veterans-affairs/evss-release-definition/tree/branches/R17.X.Y/evss-release-parent
2. View the ‘pom.xml’ file in your browser by clicking on the ‘pom.xml’ file name link.  
     
   Graphical user interface, text, application, email

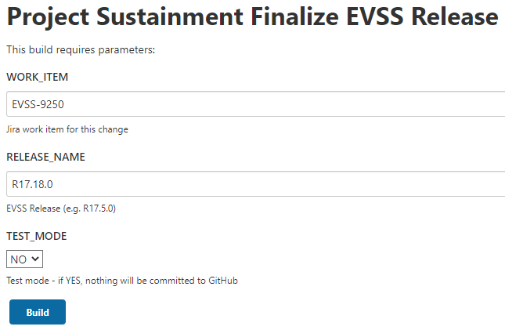
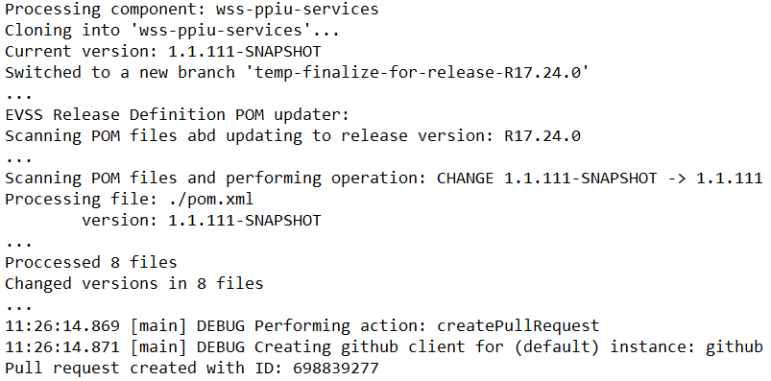
   Description automatically generated
3. Click the pencil icon to edit this file in the browser.  
     
   Graphical user interface, text, application, email

   Description automatically generated
4. Locate the lines for any components set to a new SNAPSHOT version which are NOT ready for Production and update those to reflect the (non-SNAPSHOT) Production version instead of this development version.
5. Once the edits are complete, scroll to the bottom to the GitHub commit change form.  
     
   Graphical user interface, text, application, email

   Description automatically generated
6. Add a summary with the format ‘[#EVSS-NNNN] Short description’ which references the Jira ticket number created above.
7. Add additional description information in the extended description input box (if desired.)
8. Leave the radio button set to ‘Commit directly to the branches/R17… branch’ choice.
9. Click ‘Commit Changes’ to submit your update.
   1. No Review/Approval step will be needed since this change is applied to a version specific branch and not the master branch.
10. Repeat steps 1-9 for the ‘R17.X.WLS.Y’ branch (if changes are needed in that file.) The parent Pom folder for this release will be found at:  
      
    https://github.com/department-of-veterans-affairs/evss-release-definition/tree/branches/R17.X.WLS.Y/evss-release-parent

## Finalize Versions

Once the release-definitions parent POM files accurately reflect the components to be delivered in this release, the following procedure should be used to update all versions for this release:

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to the “998-Stream Management” tab.
3. Select the “Sustainment Finalize EVSS Release” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.
5. Enter the number for the work item created above in the WORK\_ITEM input field.
6. Enter the name for this Sustainment Release in the RELEASE\_NAME input field.
   1. This should use the same release number as in the Summary for the Jira ticket above.
   2. This should be one “word” (no whitespace, e.g. “R17.18.0”.)
7. Set the “TEST\_MODE” field to “NO” unless you are testing this process (for the developer only!)
8. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
   2. This job will update the version for every component to a non-SNAPSHOT version and submit GitHub Pull Requests (PRs) to merge the changes into the master branch.
9. Click the hyperlink on the left-hand column to open this build job.
   1. The job overview screen will be displayed.
10. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
11. Once the build finishes:
    1. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it or use the manual process.
12. After a successful build, verify that each component expected for this release was identified and updated by the automation script.
    1. Locate each line that starts with “Processing component:” in the console output. You should fine one match for each expected component, and no *other* components should be processed.
    2. At the end of each “processing” section there should be a Pull Request submit line showing the PR number that was created.
    3. If any errors or missing/extra components are identified, resolve these problems before continuing.

If no problems are identified, you are ready to merge changes and build each component.

# Merge Changes and Build Components

Finalizing the EVSS release above resulted in changes for the release-definitions as well as every component to be delivered in this release. These changes need to be approved and merged into the GitHub master branch and then a build is needed for each.

There are several steps in this overall process. These are:

1. Update and Build Framework (if needed)
2. Build Release Definitions (always, twice)
3. Update and Build Other Release Components (in order)

## Update and build Framework

If the Framework (evss-framework) component is being updated as part of the release, it must be built FIRST because the release-definitions component itself depends on framework (as does everything else in EVSS.)

Merge the framework version changes using the process described in Appendix 1: Approving and Merging Changes in GitHub.

**NOTE: Wait for framework to finish building successfully before continuing since most components depend on it.**

Jenkins is configured to build the master branch automatically within 5 minutes of a check-in, so you can wait up to 5 minutes for this automatic build or trigger the build manually following the steps in Appendix 2: Building Components. In either case, you must wait for the framework build to be completed (successfully) before continuing.

## Build Release Definitions

After changes are made to the release definitions component branches (either or both two) you must (manually) trigger a build for each using the following process.

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to the “10-Dependnecy-CUT” tab.
3. Select the “cut-evss-release-definition” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.  
     
   Graphical user interface, text, application, email

   Description automatically generated
5. Leave the JKD\_VERSION set to the default value.
6. Leave the MAVEN\_GOALS set to the default value.
7. Scroll through the list of branches and locate the “R17.X.Y’ branch for the target release.   
     
   **NOTE** this sometimes will have a prefix like “origin/branches/R17.X.Y”. This list is automatically loaded from GitHub and should always include all possible choices.
8. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
9. Click the hyperlink on the left-hand column to open this build job.
   1. The job overview screen will be displayed.
10. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
11. Once the build finishes:
    1. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it or notify the developer(s) to address the issue.
12. Once complete repeat steps 4-11 for the “R17.X.WLS.Y” release-definitions branch.
    1. For example, the 17.22.0 release required builds of “R17.22.0” and “R17.22.WLS.0” branches of the release definitions.

**NOTE**: The order of building these two is NOT actually important. They are independent and either can be built first so long as you do BOTH before continuing.

## Update and Build Other Release Components

For each component updated by the Finalize step above, it created a temporary branch and submitted a PR to merge the version changes into the master branch in GitHub. Those PRs need to be approved and merged into the master branch, after which the temporary branch for each component can be deleted.

Follow the order in which components appear in the release-definitions parent pom file. The order is important for some of them, and this file lists the correct order. This same order will be used in the output from the Finalize step so you can also follow the order of updates shown in the console output form that step.

**NOTE: If common-services has changed for the release, wait for a successful build from the PR approval before continuing with additional components.** Then approve PRs for document-services, claims-services, and 686-services sequentially rather than in parallel (if they are changing for the release). Wait for each to build successfully before approving the next PR. The remaining components can be built sequentially following the order in the release-definitions parent pom file.

Merge each component version change using the process described in Appendix 1: Approving and Merging Changes in GitHub.

Jenkins is configured to build the master branch automatically within 5 minutes of a check-in, so you can wait up to 5 minutes for this automatic build or trigger the build manually following the steps in Appendix 2: Building Components.

For dependency, partner library and the common-services components you should wait for each build to complete before continuing to the next component. For others you can continue without waiting if you come back and confirm that each build was completed successfully.

# Deploy & Release Ready Notification

Once all the builds are done and artifacts have been uploaded into the Nexus repository, deploy to the INT1 environment using the “deploy-evss-release” automation job to establish a baseline for timing. Review the DIBORG if direction is needed for execution of this task. Then send an email notifying the Release Manager and test lead that everything is ready. The following is a sample message showing the updated artifacts and versions for R17.16.0:

|  |
| --- |
| The release builds for R17.16.0 are ready.  The following were updated and built:  Dependency projects:   * evss-release-definition                               R17.16.0, R17.16.WLS.0 * evss-partner-services-intenttofile              1.1.7   Services:   * evss-env (apache config)                           10.6.24 (same for all tiers) * evss-common-services                               11.6.117 * evss-document-services                             3.7.113 * evss-claims-services                                   3.6.105 * evss-686-services                                        2.6.108 * wss-lettergenerator-services                      1.1.110 * wss-intenttofile-services                            1.0.106     Webparts:   * evss-env (apache config)                           10.6.24 (same for all tiers) * evss-common-webparts                             11.6.112 * evss-claims-webparts                                 3.6.106 * evss-vdc                                                      15.6.113   Portal:   * evss-env (apache config)                           10.6.24 (same for all tiers)   This is now deployed to INT1. Using the Jenkins automated job took 19 minutes for this deployment. |

# Post-Release Preparation for New Builds

After all the release builds are done, baseline branches must be created, and then new versions set to prepare components for ongoing development. These changes can be made manually, but since there are numerous updates, a Jenkins process has been created to simplify the process.

The following process should be used:

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to the “998-Stream Management” tab.
3. Select the “Sustainment\_Post\_Release\_Processing” task.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.
5. Enter the number for the work item created above in the WORK\_ITEM input field.
6. Enter the name for this Sustainment Release in the RELEASE\_NAME input field.
   1. This should use the same release number as in the Summary for the Jira work item above.
   2. This should be one “word” (no whitespace, e.g. “R17.18.0”.)
7. Set the “TEST\_MODE” field to “NO” unless you are testing this process (for the developer only!)
8. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
   2. This job will make a branch of each component, tag it with the Release Name, and then update the version for every component to a the next SNAPSHOT version and submit GitHub Pull Requests (PRs) to merge the changes into the master branch.
9. Click the hyperlink on the left-hand column to open this build job.
   1. The job overview screen will be displayed.
10. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
11. Once the build finishes:
    1. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it.
12. If the job finished successfully, you are ready to approve the PRs and merge the changes into the master branch for each component.   
      
    Follow the same process as described Appending 2: Approving and Merging Changes in GitHub section, with *one important addition*: the evss-release-definition component will also have changes, and when merging into the release-definitions master branch, **DO NOT DELETE THE RELEASE-DEFINITIONS SOURCE BRANCH (note the branch name does not start with temp.)**

**NOTE: If common-services has changed for the release, wait for a successful build from the PR approval before continuing with additional components.** Then approve PRs for document-services, claims-services, and 686-services sequentially rather than in parallel (if they are changing for the release). Wait for each to build successfully before approving the next PR. The remaining components can be built sequentially following the order in the release-definitions parent pom file.

**NOTE**: In this case also note that the POM file edits will be to increment the version number and ADD the “-SNAPSHOT” suffix for the new development component versions (e.g. the version of common-services might go from the 11.6.121 CUT version to 11.6.122-SNAPSHOT version ready for continued development.)

# Appendix 1: Approving and Merging Changes in GitHub

For each component updated one of the steps above (except release-definitions), jenkins created a temporary branch and submitted a PR to merge the version changes into the master branch in GitHub. Those PRs need to be approved and merged into the master branch, after which the temporary branch for each component can be deleted. Follow these steps to complete this process.

1. Graphical user interface, text, chat or text message

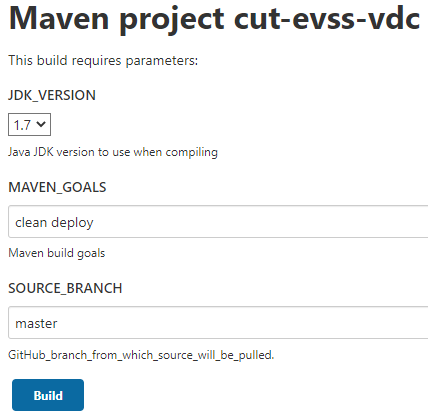
   Description automatically generatedNavigate to the component repository in GitHub (ex. Common-services, as shown here.)
2. Click on the “Pull Requests” link/tab near the top of the page.
3. Graphical user interface, text, application, email

   Description automatically generatedSelect the PR with the description “Finalize release version updates” from the list.  
     
   **NOTE**: if any other OPEN PRs are listed, confirm they are not required for this release!
4. From the PR overview page, click the “Add your review” link to review and approve this PR.  
     
   Graphical user interface, text, application, email

   Description automatically generated
5. Verify the changes in this PR. In general, two types of changes are expected. Each should be a one line change:
   1. The release-definitions version in the (ONE) parent pom should have the version changed to a release version with no “-SNAPSHOT” suffix.
   2. The component version in EVERY pom file should have the version changed to a release version updated.
6. Approve and Submit your review for the update.
7. Upon return to the PR overview page, click the “Merge pull request” and then “Confirm merge” buttons to complete the process and merge the changes into the master branch.
8. Confirm you see the message “Pull request successfully merged and closed”. If not, resolve this problem before continuing!
9. Click the “Delete branch” button to delete the temporary branch used to create these changes, since it is no longer needed.

# Appendix 2: Building Components

Jenkins is configured to build the master branch automatically within 5 minutes of a check-in, so no manual intervention is normally required to build components. If you do not want to wait 5 minutes builds cab be started manually through the EVSS CI Jenkins server. The following procedure should be used:

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to one of the “CUT” tabs depending on which component you plan to build:
   1. For dependency projects not directly associated with any particular application/tier, select the “10-Dependency-CUT” tab.
   2. For Partner service libraries, select the “20-Partner-CUT” tab.
   3. For EBN/Services tier projects, select the “30-Services-CUT” tab.
   4. For SEP/Portal tier projects, select the “40-Portal-CUT” tab.
   5. For VDC/Webparts tier projects, select the “50-WebParts-CUT” tab.
   6. For VAS/VA.gov Services tier projects, select the “990-VETS.GOV-CUT” tab.
3. Select the “cut-xxx” project you intend to build. For example, to build the vdc (evss-vdc) project, go to the “50-WebParts-CUT” tab and select the “cut-evss-vdc” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.  
     
   
5. Leave all settings at the default value.
6. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
7. Click the hyperlink on the left-hand column to open this build job.
   1. The job overview screen will be displayed.
8. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
   1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
9. Once the build finishes:
   1. Maven dependency information collected during the build will be available from the Project overview page.
   2. Following completion of the CUT build, a Fortify job will automatically be initiated to scan the code base for security issues. The Fortify jobs and resulting reports can be found under the “91-Fortify” tab in Jenkins. A link to this job is also available on the Project overview page.
   3. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it or notify the developer(s) to address the issue.