**System integration NAD daily build procedure**

Below are Steps need to follow for NAD Build:

1.Prepare build plan

2.DN

3.Integrate DNs into mainline

4.Download Yocto baseline

5.Create daily build branch for the manifest

6.Integrate meta data changes or other changes impacting the repo manifest

7.Tag the manifest with the version name

8.Create System integration pull requests

9.Update repo to the new daily build manifest

10.Launch the NAD build

11. Change integrated DNs to "Integrated" state

12.Perform NAD build checklist

1. **Prepare build plan**

* First task is to move all DN from Ready to Planned State.

**2.**  **DN**

* Check the DN’s Descripition means what is the purpose of this DN
* DN is releated to which issue.
* Need to check all PR are merged to the DN.
* Need to check PR are fast forward with the integration line.
* Check that all reports are linked in the CSS section of the DN
* Check if severity 1 violations are shown in the Parasoft report and if so link the tracking defect.
* Check if regressions or test not performed are justified
* Need to Check all dependencies are reported with links
* Check defects fixed are linked and in the state "System integration"(Bugfix DN only)(Need to check with Team onces).
* (Porting DNs only) Check for original DN link. If two or more DN links are there.(Need to check with Team onces)
* (When releasing more than one DN) Check integration sequence and notify integration order.
* In DN Check Target type whether this DN is for NAD or VUC.

3.**Integrate DNs into mainline:**

Merge all NAD pull requests into the release line.

4.**Download Yocto baseline:**

Needto Refer to NAD build – v1.0#v1.0-GettingSources

(Need to check ).

5.**Need to do Daily build branch for the manifest**

Create a branch in the manifest repository starting from the release baseline

|  |
| --- |
| $cd <repository path>  $git fetch  $git checkout / # Or Line branch if applicable (Line Branch Means )  $git checkout -b daily/ |

6.**Integrate meta data changes or other changes impacting the repo manifest .**

If there was any new changes, it should be staged, commited and pushed.

$git commit -m 'Integrated for <Release Name>

>TBMOH-xyz <summary>

>TBMOH-xyw <summary>

7.**Tag the manifest with the version name**

Tag the version name in the folder

$ cd <repository path>

$ git tag <Release Name>

8.**Create System integration pull requests**

**Creating new branches**

**1.**Go to the JIRA issue you are working on and select "create branch”

2.Then choose branch type and customize branch name if needed (choose "Custom" for legacy TBM2 branch naming convention DEV\_\*, PC\_\*).

**Note:** Add JIRA issue key to every commit: git commit -m "GPT-45427: commit message...

**Opening a pull request**

* Go to the JIRA issue you're working on
* Identify the branch you want to merge in release branch
* Click on Create pull request option in jira.
* Configure pull request selecting destination branch

**Updating pull request**

* To update a pull request just **push new commits** on the same branch.
* If the pull request was already approved, Bitbucket will remove the approval and reviewers shall approve it again, upon new inspection.

**Merging pull requests**

There are different merging strategies.

* Merge commit
* Fast-forward
* Fast-forward only
* Rebase, merge
* Rebase, fast-forward
* Squash
* Squash, fast-forward only

For **pre integration Fast-forward** is the best one, but also rebase could be useful.

**Note**: **SI will update the branch or ask Domain to update the branch in case of conflicts, then SI will merge the updated source branch using Fast-forward only method.**

If we **face conflict in merging**, follow below steps:

* Checkout the source branch and merge in the changes from the target branch. Resolve conflicts.

|  |
| --- |
| git checkout  git pull origin comp\_rel\_tbm2\_l1 |

* After the merge conflicts are resolved, stage the changes, accordingly, commit the changes and push
* Merge the updated pull request.

**Update repo to the new daily build manifest**

* Change directory to the Yocto build folder and fetch the new manifest.

|  |
| --- |
| $cd /opt/builds//  $repo init --no-repo-verify -u <https://hub.marelli.com/bitbucket/scm/tbmnx_common/manifest.git> -m mm-layers.xml -b < New daily build Manifest HASH>  $repo sync --no-tags -c -q -j 4  Doing this, all metadata is updated to the new daily build contents. |

**Launch the NAD build**

* Follow the standard procedure to compile the full NAD images:

|  |
| --- |
| $cd /opt/builds/<Username>/<Build name>  $source poky/build/conf/set\_cust\_bb\_env.sh  $build-9607-perf-image |

**Execute post build scripts:**

* Scripts are all stored in the **toolchain-tools repository** cloned by the repo tool in the yocto build folder. All script shall be executed in order.
* **post-00\_package-build.sh\*:** This script packages the binaries needed for **flashing a modem image**,
* **Note**: use with -a option to automatically get the version name.

**Output is stored in:**

/opt/builds/<User name>/<Build name>/poky/build/deploy/<build name>

* **post-01\_documentation.sh\*:** This script generates a documentation data file (Python pickle) used to **automate release document generation**. This can be imported by the release manager tool.
* **post-02-sendToArtifactory.sh\*:** This script **uploads all files to the project's Artifactory** database.
* It deploys the flashing binary and documentation file in the release tree structure for the daily build.

**Output is stored in:**

Artifactory release path.

* **post-03\_tag-release.sh\*:** This script **tags all meta layers** with the release name.

**Output is stored in:**

No binary **only tags pushed to the respective Git repositories**

**Note:**

* **For NAD V0.0 Only above 3 scripts.**
* **For NAD V1.0 build need to run above script and also below Scripts .**
* **post-04\_package-symbols.sh:** This script creates an archive that merges the target rootfs with a rootfs made of all -dbg packages.

This is needed for debugging with gdb.

* **post-05-deploy-symbols.sh:** This script deploys to artifactory the **debugfs archive.**

**Output is stored in:**

* **Artifactory debug symbols path:**

<https://artifactory.marelli.com/artifactory/TBMOH/TBMOH_TOOLS/debug/symbol/>

* **post-06\_push-metadata.sh\*:** This scripts pushes NAD release information to a central git repository.

This serves for documentation and other release automation process.

**Change integrated DNs to "Integrated" state:**

* DNs shall be moved from the "Planned" state to "Integrated".
* The Day and Daily build shall be written checked and changed if needed.
* DNs rejected shall be moved to "Open"

**Perform NAD build checklist:**

* Verify that the release manifest is pushed to the remote and tagged
* Verify that all pull requests are merged and metadata are tagged
* Verify that flashing binaries are available in Artifactory
* Double check NAD version name was changed correctly
* Verify that the documentation file was deployed in Artifactory
* Check if src\_cache and sstate\_cache was updated

Prerequisites:

1.sourcetree (Branch Analysis tool)

As per Confluence page Document for NAD build procedure, they have not Mentioned any tools.