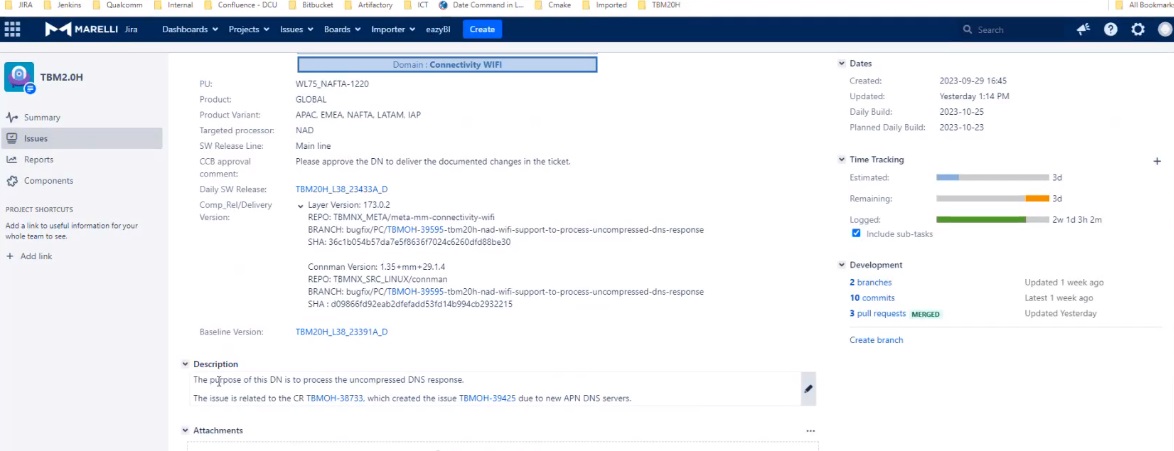
**SI on NAD Build**

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**

Change all the DN READY to PLANNED State.

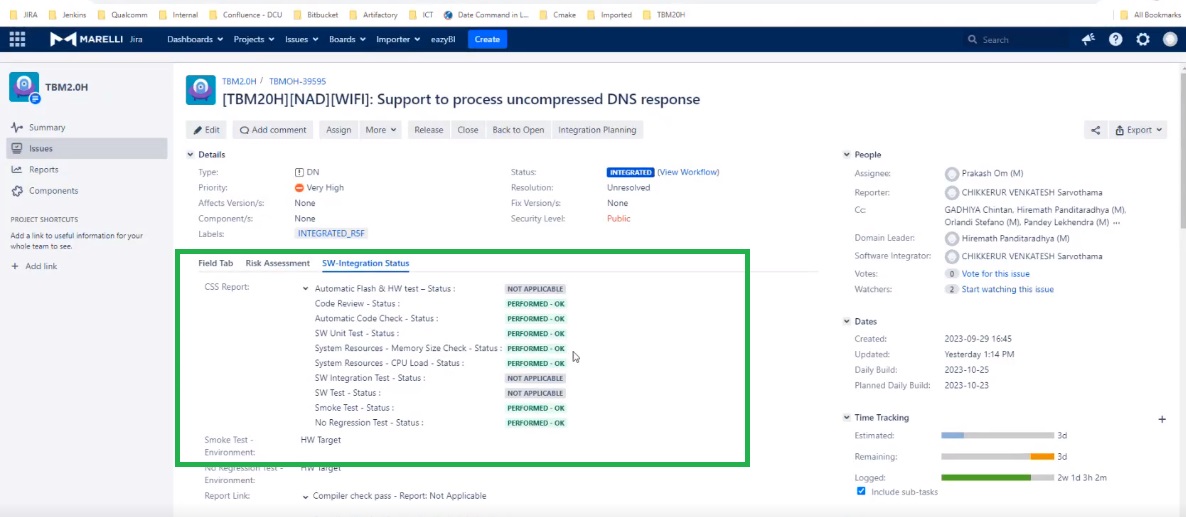
1. **DN**
2. Check the description



ii)Check all PR are merged to the DN

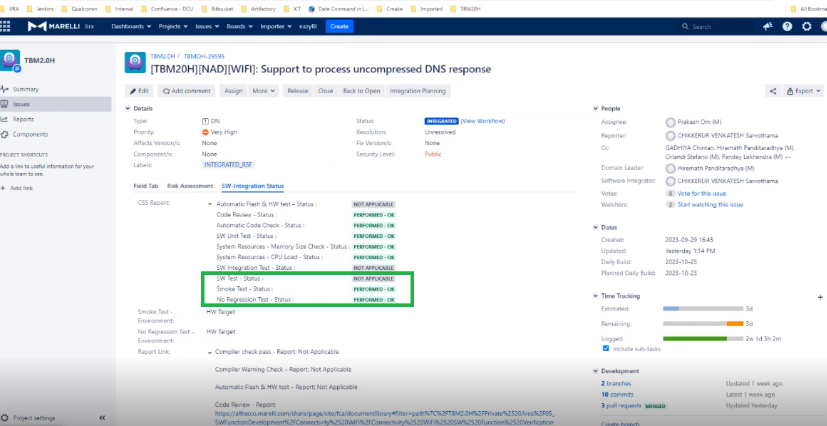
iii)Check all PR are fast forward with the integration line.

iv) Check that all reports are linked in the CSS section of the DN.

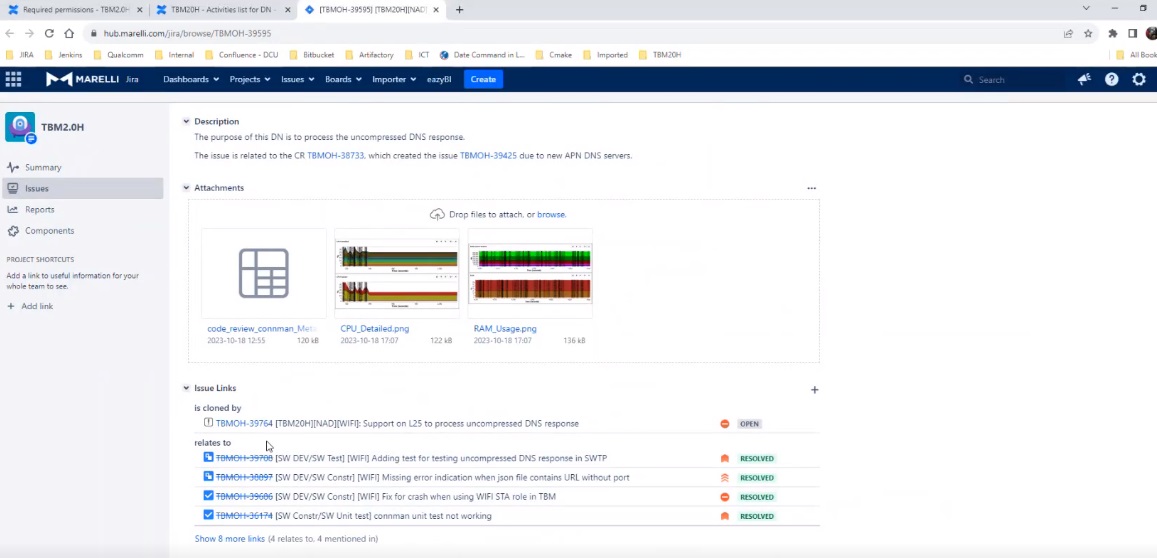


v) Check if severity 1 violations are shown in the Parasoft report and if so link the tracking defect.

vi) Check if regressions or test not performed are justified



vii) Check all dependencies are reported with links



viii) ) Check defects fixed are linked and in the state "System integration"(Bugfix DN only)

ix) (Porting DNs only) Check for original DN link. If two or more DN links are there.

x) (When releasing more than one DN) Check integration sequence and notify integration order

In the DN check for the target processor NAD/VuC

A screenshot of a computer

Description automatically generated

1. **Integrate DNs into mainline**

Merge all NAD pull requests to the release line.

1. **Download Yocto baseline**
2. **Create 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  $git checkout -b daily/ |

1. **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> |

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

Tag the version name in the folder

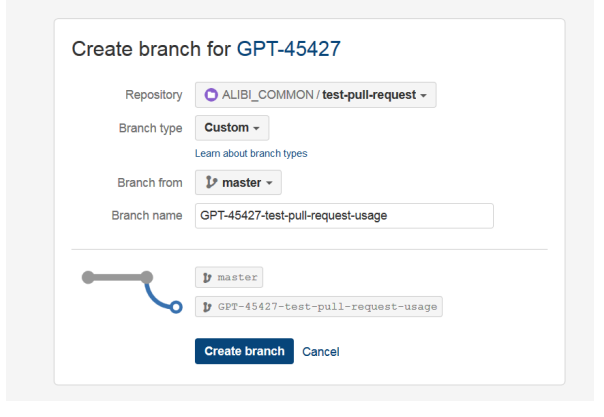
|  |
| --- |
| $ cd <repository path>  $ git tag <Release Name> |

Example for release Name

A close up of a screen

Description automatically generated

1. **Create System integration pull requests**
2. **Creating new branches**
3. got to the JIRA issue you are working on and select "create branch
4. then choose branch type and customize branch name if needed (choose "Custom" for legacy TBM2 branch naming convention DEV\_\*, PC\_\*)



**Committing**

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

1. **Opening a pull request**
2. go to the JIRA issue you're working on
3. identify the branch you want to merge in release branch
4. Create pull request

A screenshot of a social media account

Description automatically generated

1. configure pull request selecting destination branch

A screenshot of a computer

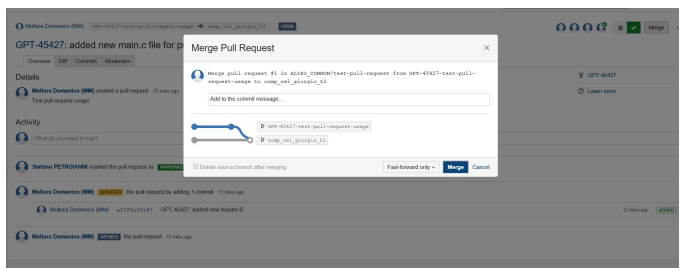
Description automatically generated

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

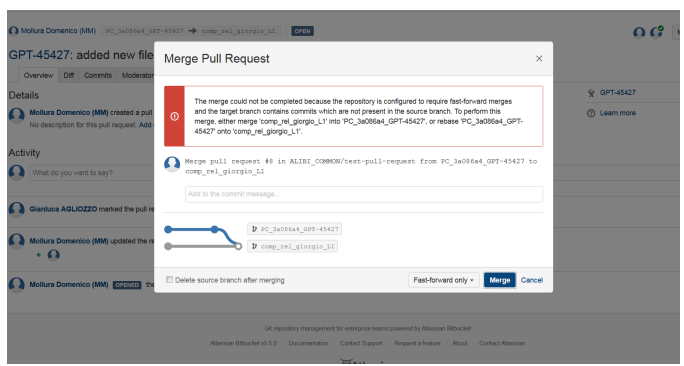
There are different merging strategies.

* 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



In the above image the PR is merged. That have no conflicts.



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:

**Step 1:** Checkout the source branch and merge in the changes from the target branch. Resolve conflicts.

git checkout

git pull origin comp\_rel\_tbm2\_l1

**Step 2:** After the merge conflicts are resolved, stage the changes accordingly, commit the changes and push.

git commit -a

git push origin HEAD

**Step 3:** Merge the updated pull request.

1. **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 |

1. **Launch the NAD build**

**9.1 Need to build in Yocto baseline.**

**OS :** Ubuntu 16.04.6 LTS

**Needed packages:**

sudo apt-get install build-essential chrpath coreutils cvs desktop-file-utils diffstat \

docbook-utils fakeroot g++ gawk gcc git-core help2man libgmp3-dev libmpfr-dev libreadline6-dev \ libtool libxml2-dev make python-pip python-pysqlite2 quilt sed subversion texi2html texinfo unzip wget sudo apt-get install python-jinja2 ninja-build cmake doxygen graphviz

**gerrit-repo tool:**

**repo version:** v1.13.4

**repo launcher version:** 1.25

**Folder to store the tool:** https://storage.googleapis.com/git-repo-downloads/repo > /usr/local/bin/repo

**Verify repo version:**

|  |
| --- |
| $ repo --version |

repo version v1.13.4 (from https://hub.marelli.com/bitbucket/scm/dmd\_cm/gerrit-repo.git)

repo launcher version 1.25 (from /usr/local/bin/repo)

git 2.7.4

Python 2.7.12 (default, Oct 8 2019, 14:14:10) [GCC 5.4.0 20160609]

**9.2 To Launch Build**

1. Retrieve manifest sha from Daily page (e.g.TBM20H\_L1\_19305A\_D Modem manifest SHA)
2. create a new folder
3. get sources with repo
4. configure git parameters and credentials timeout

git config --global user.name "Name Surname"

git config --global user.email "user@example.com"

git config --global credential.helper 'cache --timeout=86400'

1. cache git credentials:

git ls-remote <https://hub.marelli.com/bitbucket/scm/tbmnx_common/manifest.git>

1. **Launch Repo commands**

**New Repo Version**

|  |
| --- |
| repo init --no-clone-bundle --repo-url https://hub.marelli.com/bitbucket/scm/tbmnx\_common/gerrit-repo.git -u https://hub.marelli.com/bitbucket/scm /tbmnx\_common/manifest.git -m mm-layers.xml -b < Manifest HASH>  repo sync -c -q -j 4 |

**Old Repo version**

|  |
| --- |
| repo init --no-repo-verify -u https://hub.marelli.com/bitbucket/scm/tbmnx\_common/manifest.git -m mm-layers.xml -b < Manifest HASH>  repo sync -c -q -j 4 |

1. Example

**New Repo Version:**

create a new folder (e.g. myfolder)

|  |
| --- |
| cd myfolder  repo init --no-clone-bundle --repo-url https://hub.marelli.com/bitbucket/scm/tbmnx\_common/gerrit-repo.git -u https://hub.marelli.com/bitbucket/scm /tbmnx\_common/manifest.git -m mm-layers.xml -b 214585cac30de7ab0844e485ef2eb4b5fb8d5942  repo sync --no-tags -c -q -j 4 |

**Old Repo Version:**

|  |
| --- |
| cd myfolder  repo init --no-repo-verify -u https://hub.marelli.com/bitbucket/scm/tbmnx\_common/manifest.git -m mm-layers.xml -b 214585cac30de7ab0844e485ef2eb4b5fb8d5942    repo sync --no-tags -c -q -j 4 |

**9.3 Building**

To setup Yocto build environment use this command:

|  |
| --- |
| source poky/build/conf/set\_mm\_env.sh  bitbake machine-image |

**9.3.1 Generating flash package**

**o/p files in:** tmp-glibc/deploy/images/mdm9607-perf/

**Folder Name :** poky/build/

**Execute the command**:

|  |
| --- |
| post-00\_package-build.sh |

**Files created :** Bin\_NAD\_BUILDNAME.tar.gz

**Files created location**: poky/build/mm-deploy/BUILDNAME

**9.3.2 Speeding up the build**

1. Using source cache Yocto to avoid downloading code from internet
2. create a new folder to store source cache
3. uncompress the archives in that folder, using the following command

cat source-cache-TBM20H\_L1\_19355A\_D.tar.bz2\*| tar xjvf –

1. Configure Yocto to use Source cache:

/opt/builds/source-cache-TBM20H\_L1\_19355A\_D/downloads

Edit the file(local.conf) in YOUR\_BUILD\_FOLDER/poky/build/conf/

Modify the variable in file :

SOURCE\_MIRROR\_URL = “file:///opt/builds/source-cache TBM20H\_L1\_19355A\_D/downloads”

1. Using sstate cache Yocto will rebuild just changed contents
2. create a new folder to store source cache
3. uncompress the archives in that folder, using the following command

cat sstate-cache-TBM20H\_L1\_19312B\_D.tar.bz2\*| tar xjvf –

1. configure Yocto to use local-sstate folder

Edit the file(local.conf) in YOUR\_BUILD\_FOLDER/poky/build/conf/

Modify the variable in file :

SSTATE\_MIRRORS ?= "\ file://.\* [file:///opt/builds/local-sstate/PATH](../../../../../D:/opt/builds/local-sstate/PATH)"

**9.3.3 Post Build Scripts**

* **post-00\_package-build.sh\*** - This script packages the binaries needed for flashing a modem image

i)Output is stored in: /opt/builds///poky/build/deploy/

ii) Artifacts: Flashing package: Bin\_NAD\_.tar.gz

* **post-01\_documentation.sh\*** - This script generates a documentation data file (Python pickle) used to automate release document generation.

i) Output is stored in: /opt/builds///poky/build/deploy/

ii) Artifacts: NAD documentation file: documentation\_NAD\_.pkl

* **post-02-sendToArtifactory.sh\*** - This script uploads all files to the project's Artifactory database.

i) Output is stored in: Artifactory release path

ii) Artifacts: No artifacts are generated.

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

i) Output is stored in: No binary only tags pushed to the respective Git repositories

ii)Artifacts: No artifacts are generated.

* **post-04\_package-symbols.sh** - This script creates an archive that merges the target rootfs with a rootfs made of all -dbg packages.

i) Output is stored in: opt/builds///poky/build/mm-deploy/

ii) Artifacts: NAD debugfs: debug\_symbols..tar.bz2

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

i) Output is stored in: Artifactory debug symbols path: https://artifactory.marelli.com/artifactory/TBMOH/TBMOH\_TOOLS/debug/symbols/

ii) Artifacts: No artifacts are generated

* **post-06\_push-metadata.sh\***

i) This scripts pushes NAD release information to a central git repository.

ii) This serves for documentation and other release automation process

* 1. **Flashing procedure**
     + - 1. **Interacting with the target**

1. Connect a serial terminal to the AT serial port from the board (see windows device manager to find the port). The speed of the serial connection is 115200bauds.
2. Send the following command

|  |
| --- |
| at+qcfg="usbcfg" |

1. If the last parameter in reply is not 1, send:

|  |
| --- |
| at+qcfg="usbcfg",0x2C7C,0x0435,1,1,1,1,1,1 |

1. reboot module ADB interface should be detected by your Windows host.
2. If the board is recognize as a Network interface or an USB Modem instead of the ADB interface use the following at command and reboot the board:

|  |
| --- |
| AT+QCFG="usbnet",1 |

* + 1. **ADB for Windows**

Get it from: [https://artifactory.marelli.com/artifactory/webapp/#/artifacts/browse/tree/General/TBMOH/TBMOH\_TOOLS/ADB/platform-tools-latest-windows.zip](https://artifactory.marelli.com/artifactory/webapp/" \l "/artifacts/browse/tree/General/TBMOH/TBMOH_TOOLS/ADB/platform-tools-latest-windows.zip)

* + - 1. **Serial connection**

NAD serial line can be accessed using debug adapter and FTDI serial to usb converter

**Settings:**

speed: 115200

data bits: 8

stop bits: 1

parity: none f

low control: XON/XOFF

**Linux shell:**

user: root

password: oelinux123

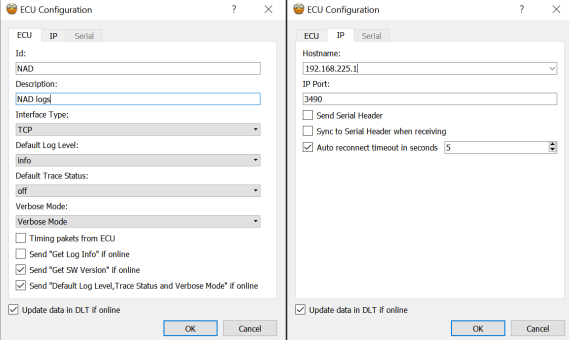
* + - 1. **Collecting NAD DLT logs**
  1. Media Converter
  2. DLT viewer with Thelematics plugins: [https://artifactory.marelli.com/artifactory/webapp/#/artifacts/browse/tree/General/TBMOH/TBMOH\_TOOLS/DLT/2.18.0](https://artifactory.marelli.com/artifactory/webapp/" \l "/artifacts/browse/tree/General/TBMOH/TBMOH_TOOLS/DLT/2.18.0)
  3. Connect Media converter as in these pictures



* 1. Configure your PC network interface with an IP in same subnetwork as your board (192.168.225.1):

e.g. assign to your pc network interface the ip 192.168.225.100 (you can use an eth to USB adapter as well)

e. Open DLT viewer and configure ECU interface: go to menù Config Ecu Add and select option as in these pictures



* 1. Verify build version Click on NAD, then click on "Get Software Version", then search for the string "TBM20H"

A screenshot of a computer

Description automatically generated

1. **Change integrated DNs to "Integrated" state**

DNs shall be moved from the "Planned" state to "Integrated

1. **Perform NAD build checklist**
2. Verify that the release manifest is pushed to the remote and tagged
3. Verify that all pull requests are merged and metadata are tagged
4. Verify that flashing binaries are available in Artifactory
5. Double check NAD version name was changed correctly
6. Verify that the documentation file was deployed in Artifactory
7. Check if src\_cache and sstate\_cache was updated

**Questions:**

* + - 1. How to download Yocto baseline?
      2. If two or more DN’s link are there how to find the original DN?
      3. What is the server path in which build needs to done?
      4. How to configure yocto with our build code?
      5. Is Static code Analysis should be done in SI?

**TBD**

Generate and Deploy SDK - TBD

Other tasks to be added:

TASKS TO BE DONE

src\_cache upload - NFS shall be updated with the latests mirrors. Currently done manually. Due to NFS issues archives are been uploaded to Artifactory as alternative solution

sstate-cache NFS shall be updated with the latests mirrors. Currently done manually. Due to NFS issues archives are been uploaded to Artifactory as alternative solution

FOTA and CAN flashing input files Currently not deployed.