Developer Training Using Gerrit

Created by Miller, Daniel, last modified by McGladdery, Jacob on Jun 07, 2022

Table of Contents

Introduction

We use Git, Repo, Gerrit, Jenkins, and a number of other tools to develop and build software here at Harman. This document explains how to get set up with Git, Repo, and Gerrit so you can get started with writing and committing code.

Git

An open source distributed version control system designed for tracking projects made up of mostly text files and of virtually any size. Generally, a Git repository (repo) tracks files and changes for a single software project. Since Git is distributed, no files are ever "locked" when other developers are working on them. Also, because of Git's distributed nature, you are free to make whatever changes you want to your local copy of a repo. This includes creating local branches which are lightweight compared to other version control systems and an important tool for keeping track of local development.

Here are some useful resources for getting started using Git:

- Git Homepage
- Pro Git Book
- Git Command Reference and Guides

Repo

A repository management tool developed by Google. Designed to manage multiple Git repos all at the same time. Used mainly by Google for managing and structuring the Android source tree. Also used by some organizations to manage multiple meta layers within a Yocto project. Repo generally relies on the concept that most or all of the Git repos it manages will each contain at least one branch with the same name. Each shared branch name is conceptually the branch name for an entire Repo project. Repo uses XML files called manifests to keep track of what Git repos to work with. For reference, the Repo tool is actually two tools, a launcher script for downloading and calling Repo, and the actual Repo tool itself. Older versions of the Repo tool run on Python 2. Newer versions use Python 3.

It's important to note that Repo is not a replacement for using Git. Repo mainly manages initializing and syncing workspaces made up of multiple Git repos. Repo can also be used to "start" and "upload" local branches. However, unless you actually work for Google on AOSP, only creating (starting) a branch seems to be useful in practice. Once you have a local branch to work on, you will be able to use Git normally.

Here are some useful resources for getting started using Repo:

- Repo Homepage
- Android Repo Command Reference

Gerrit Account Settings

The following sections in this document describe how to set up Git with a .gitconfig file and how to set up SSH and connect to Gerrit. Before proceeding though, please log in to the web interface of the Gerrit server you will be using (e.g. https://androidhub.harman.com) at least once. This will ensure your account on Gerrit is actually all set up.

If you need assistance with using Gerrit, such as with managing your access permissions, please reach out to STVBATBuildNA@harman.com

For reference, here's the official Gerrit user guide: https://gerrit-documentation.storage.googleapis.com/Documentation/3.5.0.1/intro-user.html

For historical reference, here's an older document describing how to get started with Gerrit: Set up your Gerrit Account.

Create Your ~/.gitconfig File

Detailed instructions are provided here: Set up your .gitconfig

For the sake of completeness, some information is also provided here. Essentially, you need to make sure you have a `$HOME/.gitconfig` file on your system, or in other words, in the environment where you will be running Git and related commands. Then, at a minimum, populate your .gitconfig file with the following:

- Your full name; first name followed by last name, no commas.
- Your Harman email address.
- A URL section for what Gerrit server to use when pushing (uploading) code.
- A URL section for what Gerrit server to use when fetching (downloading) code.

Below is a template for a minimal .gitconfig file for use with androidhub.harman.com here at Harman. Replace the text in angle brackets with your own information. If you are not located in the US, then also replace `androidhub-cg` with the Harman Gerrit mirror server which is closest to you. If you work with code from multiple Gerrit servers (i.e. androidhub, gerrit1, gerrit-audi, etc.), then you need to add URL sections for all of them in your .gitconfig file.

Pages / Build and Integration for Infotainment - North America Home / Programs and process

~/.gitcon

```
[user]
    name = <YOUR FULL NAME>
    email = <YOUR HARMAN EMAIL ADDRESS>

[url "ssh://<YOUR GERRIT USERNAME>@androidhub.harman.com:29418/"]
    pushInsteadOf = ssh://androidhub.harman.com:29418/
    pushInsteadOf = ssh://<YOUR GERRIT USERNAME>@androidhub.harman.com:29418/
    pushInsteadOf = ssh://androidhub-cg.harman.com:29418/
    pushInsteadOf = ssh://<YOUR GERRIT USERNAME>@androidhub-cg.harman.com:29418/
    pushInsteadOf = https://androidhub.harman.com/

[url "ssh://<YOUR GERRIT USERNAME>@androidhub-cg.harman.com:29418/"]
    insteadOf = ssh://androidhub.harman.com:29418/
    insteadOf = ssh://<YOUR GERRIT USERNAME>@androidhub.harman.com:29418/
    insteadOf = ssh://androidhub-cg.harman.com:29418/
    insteadOf = ssh://<YOUR GERRIT USERNAME>@androidhub-cg.harman.com:29418/
```

⚠ It's important to configure the Gerrit mirror for fetching correctly. Otherwise, you may end up trying to fetch code from the Gerrit master, which is not optimized for reading from. If that does happen, you may see Git's SSH connection to Gerrit being dropped and get the message "fatal: early EOF". Remember, configuring a different mirror for fetching in your .gitconfig file may be the solution if you experience dropped connections when using Git.

Generate SSH Keys

OpenSSH has deprecated SHA-1 signatures used with the RSA key type. In order to avoid confusion about keys vs signatures and how to generate a more secure RSA key with a SHA-2 signature, it is instead recommended to generate an SSH key pair using the Ed25519 key type. In terms of security and breakability, Ed25519 is comparable to an RSA key of a length around 3000 bits.

Generate an SSH key pair (private key + public key). Replace the comment string with either your Harman email or some other comment which can be used to help identify this key pair.

```
ssh-keygen -t ed25519 -C "<your Harman email or some other comment>"
```

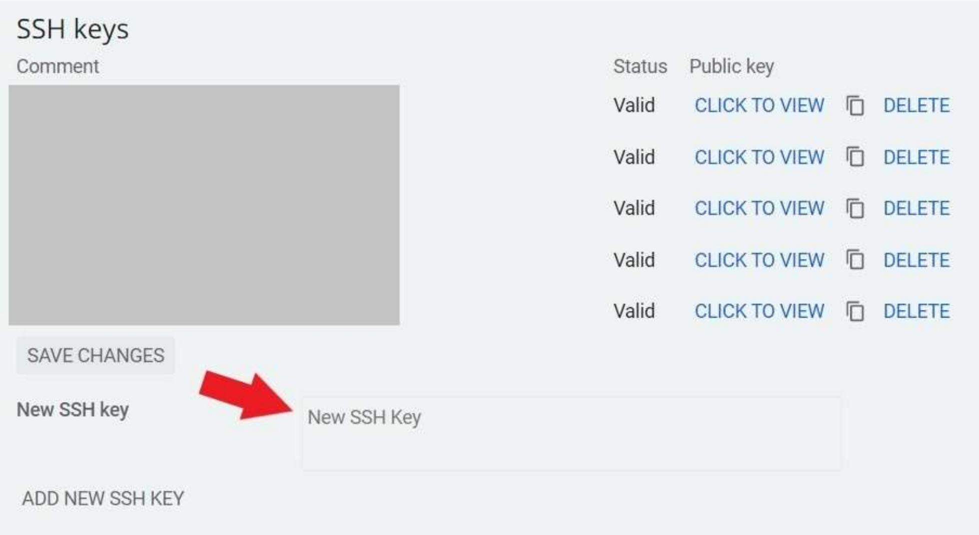
The ssh-keygen tool will prompt you with some other options like where to save the key files and if you want to set a passphrase on the private key. Just leave these fields blank and keep pressing enter in order to accept all of the default options. By accepting all of the defaults, no passphrase will be set on your private key. This is acceptable because your private key will only ever live on your development machine, and you probably don't want to have to deal with the complexities of working with ssh-agent.

Next, print the contents of your *public* key file. You will need to copy-and-paste the entire public key string into Gerrit.

```
cat ~/.ssh/id_ed25519.pub
```

Add Your SSH Public Key to Gerrit

1. In a web browser, navigate to https://androidhub.harman.com/settings/#SSHKeys
    a. Log in to Gerrit if you aren't already logged in.
2. Locate the text box which says "New SSH Key" and paste your entire SSH public key string into it.
    a. After following the steps above for generating a key pair, your public key should be the file located at `$HOME/.ssh/id_ed25519.pub` in the environment where you generated the key pair.
    b. Don't worry if the "New SSH Key" text box appears grayed-out in Gerrit. That's just how Gerrit looks. You can still click on it and paste your public key into it.
    c.



SSH keys

Comment	Status	Public key		
[redacted]	Valid	CLICK TO VIEW	⧉	DELETE
	Valid	CLICK TO VIEW	⧉	DELETE
	Valid	CLICK TO VIEW	⧉	DELETE
	Valid	CLICK TO VIEW	⧉	DELETE
	Valid	CLICK TO VIEW	⧉	DELETE

3. With your public key entered, click on "ADD NEW SSH KEY" to complete the process of adding your public key.
4. You will receive an email from the Gerrit server alerting you of the key you just added.
    a. Since you just added this key yourself, you may safely disregard this email.
5. Test i
    a

Pages / Build and Integration for Infotainment - North America Home / Programs and process

https://androidhub.harman.com/settings/#Profile

```
ssh -vv -p 29418 <your Gerrit username>@androidhub.harman.com
```

    b. If this is your first time connecting to androidhub, and you are prompted to accept any server keys when connecting, then press "y" then ENTER in order to do so.
6. You should now see a message welcoming you to Gerrit but telling you that interactive shells are disabled. This means you have SSH set up correctly.
    a. If you see "Permission denied (publickey)", then there is either a problem with your SSH key pair, with how you are invoking SSH, or with how you have SSH configured.
    b. In that case, you can either look through the SSH debug messages to try and figure out what is wrong, contact a B&I team member for support, or contact IT for support.
    c. Other things to check:
        i. Make sure you've spelled your Gerrit username correctly.
        ii. Make sure you're invoking SSH from the same terminal/environment where you set up your SSH key pair.
        iii. Double check your public key is actually listed at https://androidhub.harman.com/settings/#SSHKeys
        iv. Make sure you generated an Ed25519 key instead of an RSA key, since certain variants of SSH RSA keys are now deprecated for security reasons.

Create Your ~/.ssh/config File

This step is optional, though recommended if you use a tool such as VS Code which supports tracking hosts via the SSH config file. Your SSH config file can be used to set options for connecting to particular hosts so you don't have to remember to specify those options manually on the command line each time you use SSH. You can also assign arbitrary alias names to hosts.

Start by creating the config file:

```
touch ~/.ssh/config
```

Next, add your hosts and options. See here for a reference: https://linux.die.net/man/5/ssh_config. The following is an example for simplifying communication with our Gerrit server, androidhub.harman.com:

~/.ssh/config

```
# Options for all hosts.
# Send a keepalive packet once every 5 minutes for up to 10 hours.
Host *
    ServerAliveInterval 300
    ServerAliveCountMax 120

# Make androidhub an alias to androidhub.harman.com and add options.
# Additional alias names can also be added on the same "Host" line, space-separated.
# If you're copy-and-pasting this, remember to replace "bobsmith" with your Gerrit username.
Host androidhub
    HostName androidhub.harman.com
    Port 29418
    User bobsmith
```

With the above config, you could test your SSH connection to androidhub with just the following command:

```
ssh androidhub
```

Install the Repo Launcher

The Repo launcher is a single Python script which installs and runs the actual Repo tool within your project workspaces. Having the Repo launcher on your PATH is enough to use the actual Repo tool. Here at Harman, we have our own fork of the Repo tool. You can download the Repo launcher script from it.

1. Create a personal `bin` directory in your home directory if you don't already have one.
    a.
    ```
    mkdir ~/bin
    ```
2. Download the Repo launcher script into your `~/bin` directory.
    a.
    ```
    # For existing and legacy programs use:
    git archive --remote=ssh://androidhub.harman.com:29418/harman/repo.git harman-stable repo | tar -x -C ~/bin

    # For future programs use:
    git archive --remote=ssh://androidhub.harman.com:29418/harman/repo.git harman-latest repo | tar -x -C ~/bin
    ```
3. Ensure the Repo launcher is executable.
    a.
    ```
    chmod a+x ~/bin/repo
    ```

4. Ensure your personal `~/bin` directory is set to appear at or very close to the start of your PATH.
    a [illegible]
    b

5. Verify the Repo launcher is available on your PATH.
    a.
    ```
    which repo
    ```

Running Repo Init and Sync

Repo Init

The Repo init command will initialize the current directory you're invoking it from as the root directory of your project workspace. This means Repo will create a hidden `.repo` directory in the current working directory. The `.repo` directory contains the actual Repo tool, local copies of the manifest files, and the actual `.git` directories for each of the projects specified in the manifest files. It's important to know all of this because if you ever have to delete or recreate your `.repo` directory, then you will lose any work which you haven't committed and pushed to Gerrit yet.

Unless specified otherwise, invoking the Repo init command for doing development work generally follows this template:

```
repo init \
  -u ssh://androidhub.harman.com:29418/android/manifest \     # 1
  -b <BRANCH> \                                               # 2
  -m <MANIFEST> \                                             # 3
  --repo-url=ssh://androidhub.harman.com:29418/harman/repo \  # 4
  --repo-branch=harman-stable \                               # 5
  --no-repo-verify                                            # 6
```

Here's an explanation of the options given:

1. The URL of the Git repo containing the Repo manifest files to use.
2. The project branch or Git revision of the manifest repo to work with.
3. The initial manifest file to use. Think of this as the root or top-most manifest file of your project.
4. The Repo repository location. The location of the Harman fork of the Repo tool is given here.
5. The Repo branch or Git revision to download. The `harman-stable` branch is given here.
6. The Harman fork of Repo won't verify against Google's verification keys, so we simply skip verification.

⚠ You don't need to create multiple workspaces for each of the different project branches you work on. Instead, you only need to create one workspace and then run the same Repo init command again but replace the branch argument with the different branch you want to checkout in order to checkout that branch in the same workspace. Be careful to commit and push, or stash any work you've done on the currently checked out branch first. Otherwise, you will lose work.

⚠ Once the actual Repo tool is installed in your workspace, you can run commands such as `repo <COMMAND> --help` or `repo help <COMMAND>` to see what options and arguments those Repo commands take.

Repo Sync

The Repo sync command will clone, or update and rebase one or more projects (individual repos) within the workspace. Running Repo sync without specifying any specific projects will sync all projects within the workspace. If a project is being synced for the first time, then this command is equivalent to `git clone`. If a project has previously been synced, and is being synced again, then this command is equivalent to `git remote update && git rebase origin/<BRANCH>`.

Unless specified otherwise, invoking the Repo sync command generally looks like the following:

```
repo sync \
  -c \        # 1
  -j8 \       # 2
  --no-tags   # 3
```

Here's an explanation of the options given:

1. Fetch only the current branch specified in the Repo init command.
2. The number of sync jobs to run in parallel.
3. Don't fetch Git tags.

❗ Since syncing rebases existing projects, any committed and un-pushed changes will be permanently discarded. Make sure you push any work to Gerrit prior to syncing in order to ensure you don't lose your work. If you find yourself thinking you need to sync in order to get a teammate or someone else's code while you are still working on your changes, consider using normal Git commands to fetch the code you need from Gerrit instead.

⚠ If any merge conflicts occur while syncing, you will need to run normal Git commands such as `git rebase` in order to work with and resolve the conflicts.

Initialize and synchronize the repository

Make sure you have 100GB of free disk space, case-sensitive (Linux default).

```
rm -rf .repo/repo (one time)
repo init -u ssh://androidhub.harman.com:29418/android/manifest -b fca-my20-mainline -m dev.xml --no-repo-verify --repo-url=ssh://androidhub.harman.co

repo sync
```

1. Create local development/feature/bug-fix branch in specific git repository where you want to modify/add/delete source-code :
   Branch Naming Convention : <User-Story/Feature-ID/Bug-Fix-ID>-<short-description>
   e.g US118-info3-target

   4a. Preferred way to create a local branch:

   using **Repo Tool**

   ```
   repo start <Branch name> <"Project List" repos where changes will be made>
   ```

   Examples

   ```
   repo start US118-fcar1-target vendor/harman/apps/FCAHomeScreen
   repo start US118-fcar1-target --all


   #BranchName: name the branch in regards to what feature or bug you are working on
   #ProjectList: "'.' is a useful shorthand for the project in the current working directory." https://source.android.com/setup/develop/repo
   repo start elvis1234567 .
   ```

   › 4b. Alternative NOT PREFERRED ways to create a local branch...

   using Git Checkout

   ```
   cd to respective repos
   git checkout -b <branch name>
   ```

   Example

   ```
   git checkout -b US118-fcar1-target
   ```

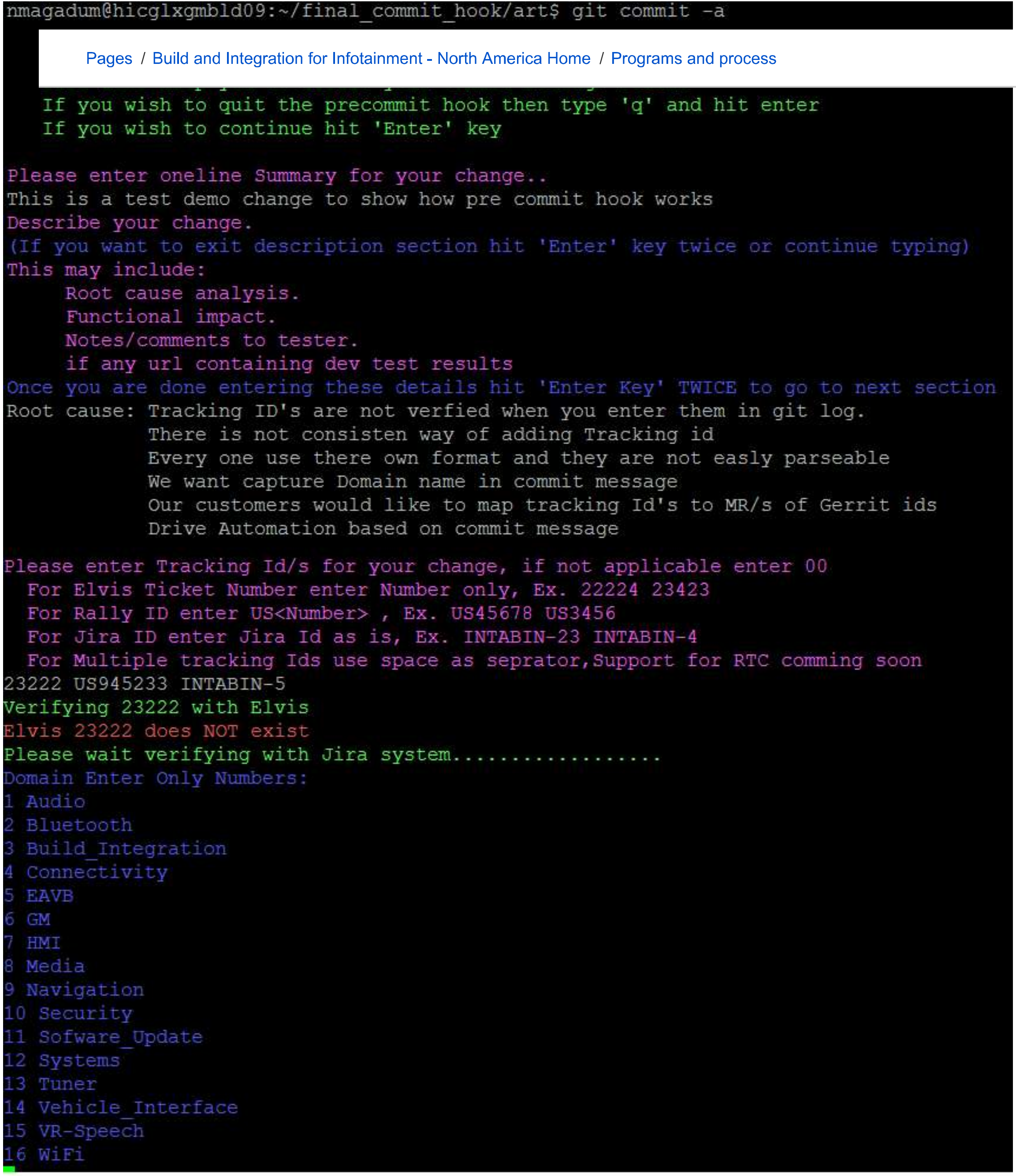
   ✓ ReSync before starting new work. When you start working on new change you need to repo sync to latest target baseline, create new branch and then start making changes.

   ⚠ Please make sure you are working on latest target branch not your local Branch.

   ❗ If you make changes on top of your other changes which is not yet in baseline then you will see merge conflict as parent commit for your change does not exist in the basline

2. Make your code changes
3. git add . in repos you have might have added new files.

4. Training video for effective committing your changes.



```
nmagadum@hicglxgmbld09:~/final_commit_hook/art$ git commit -a

   If you wish to quit the precommit hook then type 'q' and hit enter
   If you wish to continue hit 'Enter' key

Please enter oneline Summary for your change..
This is a test demo change to show how pre commit hook works
Describe your change.
(If you want to exit description section hit 'Enter' key twice or continue typing)
This may include:
     Root cause analysis.
     Functional impact.
     Notes/comments to tester.
     if any url containing dev test results
Once you are done entering these details hit 'Enter Key' TWICE to go to next section
Root cause: Tracking ID's are not verfied when you enter them in git log.
            There is not consisten way of adding Tracking id
            Every one use there own format and they are not easly parseable
            We want capture Domain name in commit message
            Our customers would like to map tracking Id's to MR/s of Gerrit ids
            Drive Automation based on commit message

Please enter Tracking Id/s for your change, if not applicable enter 00
  For Elvis Ticket Number enter Number only, Ex. 22224 23423
  For Rally ID enter US<Number> , Ex. US45678 US3456
  For Jira ID enter Jira Id as is, Ex. INTABIN-23 INTABIN-4
  For Multiple tracking Ids use space as seprator,Support for RTC comming soon
23222 US945233 INTABIN-5
Verifying 23222 with Elvis
Elvis 23222 does NOT exist
Please wait verifying with Jira system.................
Domain Enter Only Numbers:
1 Audio
2 Bluetooth
3 Build_Integration
4 Connectivity
5 EAVB
6 GM
7 HMI
8 Media
9 Navigation
10 Security
11 Sofware_Update
12 Systems
13 Tuner
14 Vehicle_Interface
15 VR-Speech
16 WiFi
```

5. Submitting your changes.
   Sqaush your commits (please study this very very important)
   git rebase -i HEAD~N
   N is number of commits you would like to squash.

1.Using repo upload command (preferred way for Gerrit migrated changes) **(few developer facing issues with repo upload,we are investigating it, if you are facing any challenge with repo upload then use git push as below)**

```
    repo upload -t (works only if you have used repo start to create local branch, your local branch will be used as default topic branch in Gerrit)

 You can also use git push to push your changes for Git lab but for Gerrit recommended way is repo upload,

 once all repos and migrated it will be mandatory to use repo upload -t command for submitting changes.

    cd to your modified repo and then
    git push origin <Your local branch>:refs/for/fca-my20-mainline/<Topic Branch>  ( Please note remote name harman in is no more supported "origin" i

Above commands generate Gerrit ID's (MR's) in Gerrit and provide you url/s for your change.

https://androidhub.harman.com/#/c/<New MR Number>
```

Typical Usage

```
# go
croot
cd vendor/fca/apps/myApp
# create local branch.  Using "." to tell repo to use the current directory...
repo start myBranchName .
# make changes
touch myNewFile
# stage changes
git add myNewFile
# commit changes to git.  allowing it to help you format your commit message...
git commit -a
# create a Gerrit Merge Request
repo upload -t
```

I'll try to get back to this tomorrow...

Process Description

10. Dependency Management between Gerrit Changes.

There are two type of Dependents

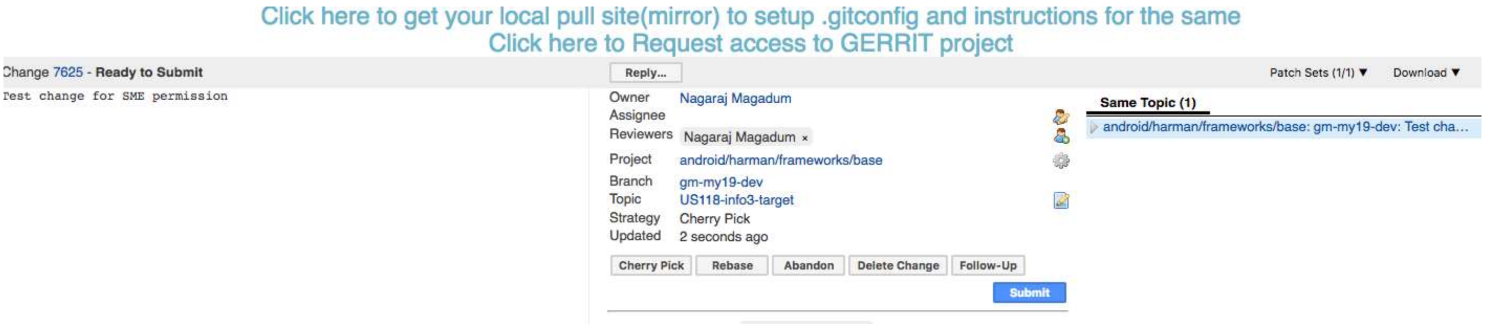
1.Changes needs to go together (Circular dependency's) and

2.Change/s depends on another change/s (one way dependency)

When you pushed your changes using "repo upload -t" option topic branch is already created in Gerrit , if your change is in multiple repos then it already take care.

if you did not use -t option or you used git push to push your changes then you need to add "Topic" name manually into the gerrit "Topic" section as shown below.

If your change depends on other developer change then you make sure both changes have same topic name.



Click here to get your local pull site(mirror) to setup .gitconfig and instructions for the same
Click here to Request access to GERRIT project

Change 7625 - Ready to Submit

Test change for SME permission

Owner Nagaraj Magadum
Assignee
Reviewers Nagaraj Magadum ×
Project android/harman/frameworks/base
Branch gm-my19-dev
Topic US118-info3-target
Strategy Cherry Pick
Updated 2 seconds ago

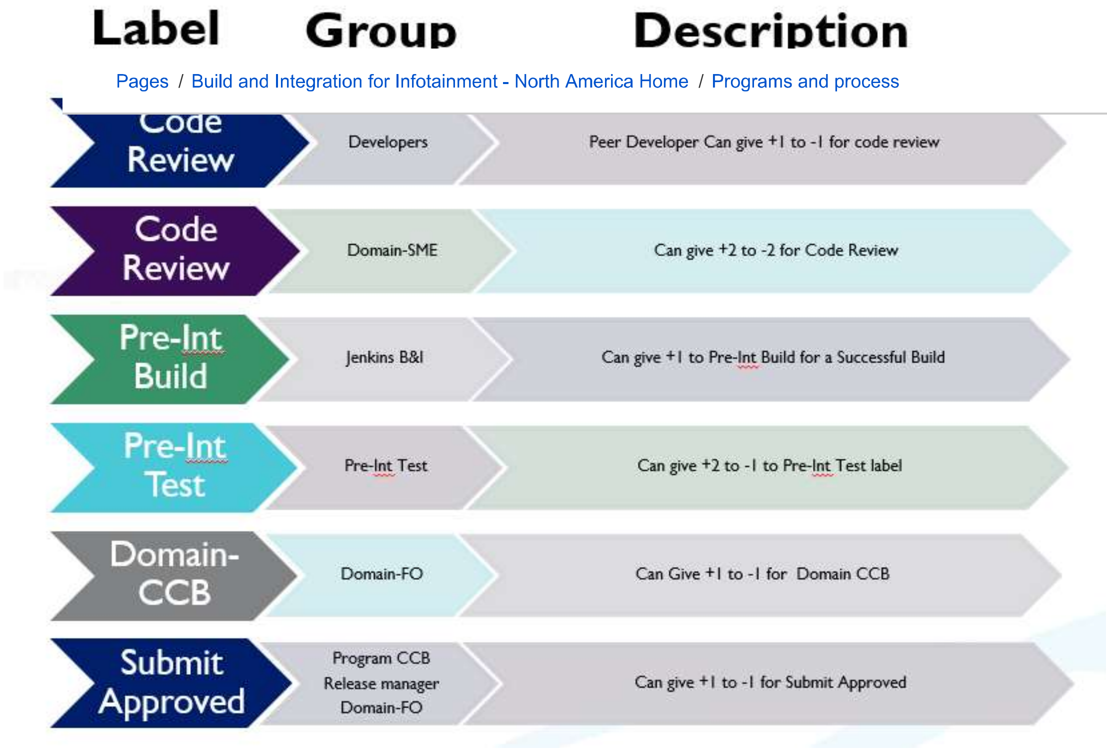
Same Topic (1)
android/harman/frameworks/base: gm-my19-dev: Test cha...

11. Now review process starts, please see this link for overall flow, if for any reason you want hold the MR then set Code-Review -1 and add a reason.

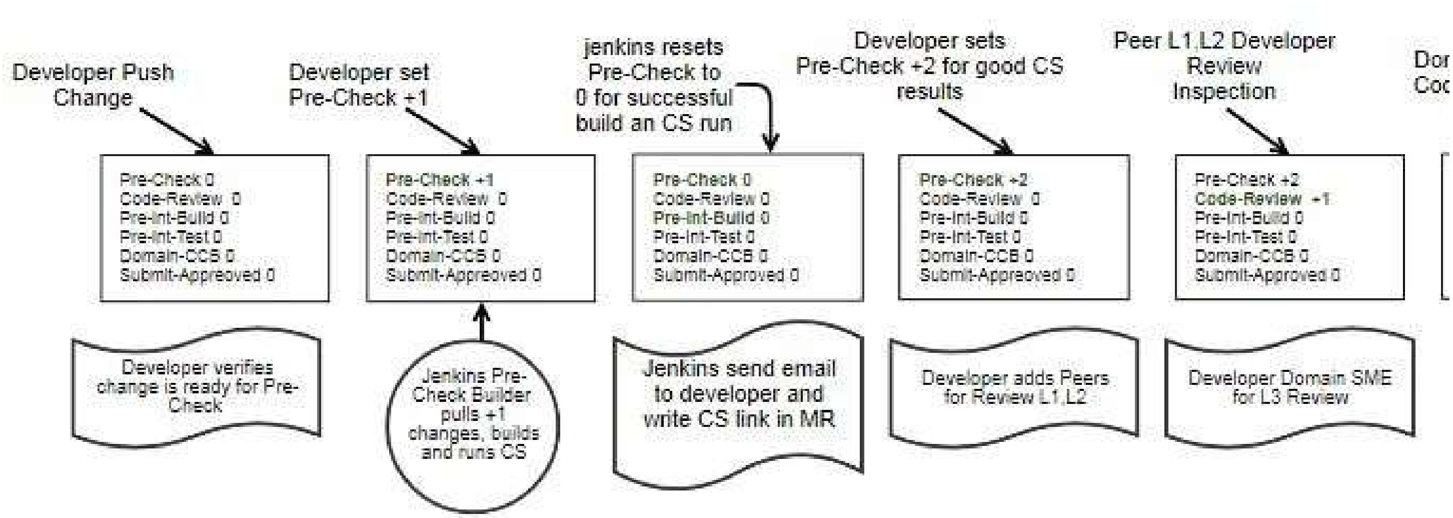
11.1 List of Domain SME (Who can do Code-Review +2) list is here for respective Domain repos.

12. Gerrit score explained here.

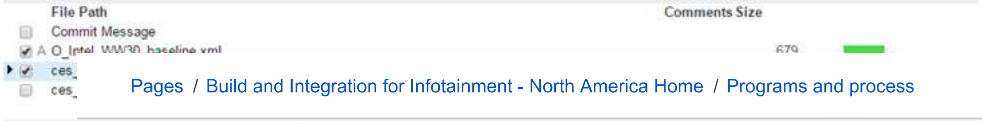
Pages / Build and Integration for Infotainment - North America Home / Programs and process



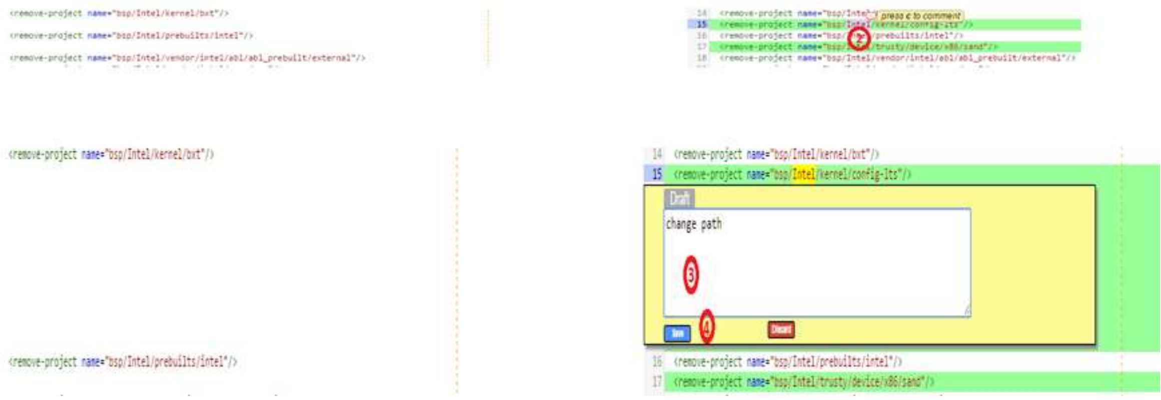
Label	Group	Description
Code Review	Developers	Peer Developer Can give +1 to -1 for code review
Code Review	Domain-SME	Can give +2 to -2 for Code Review
Pre-Int Build	Jenkins B&I	Can give +1 to Pre-Int Build for a Successful Build
Pre-Int Test	Pre-Int Test	Can give +2 to -1 to Pre-Int Test label
Domain-CCB	Domain-FO	Can Give +1 to -1 for Domain CCB
Submit Approved	Program CCB Release manager Domain-FO	Can give +1 to -1 for Submit Approved


Developer Push Change
Developer set Pre-Check +1
jenkins resets Pre-Check to 0 for successful build an CS run
Developer sets Pre-Check +2 for good CS results
Peer L1,L2 Developer Review Inspection
Dor Coc
Pre-Check 0 Code-Review 0 Pre-Int-Build 0 Pre-Int-Test 0 Domain-CCB 0 Submit-Approeved 0
Pre-Check +1 Code-Review 0 Pre-Int-Build 0 Pre-Int-Test 0 Domain-CCB 0 Submit-Appreoved 0
Pre-Check 0 Code-Review 0 Pre-Int-Build 0 Pre-Int-Test 0 Domain-CCB 0 Submit-Appreoved 0
Pre-Check +2 Code-Review 0 Pre-Int-Build 0 Pre-Int-Test 0 Domain-CCB 0 Submit-Appreoved 0
Pre-Check +2 Code-Review +1 Pre-Int-Build 0 Pre-Int-Test 0 Domain-CCB 0 Submit-Appreoved 0
Developer verifies change is ready for Pre-Check
Jenkins Pre-Check Builder pulls +1 changes, builds and runs CS
Jenkins send email to developer and write CS link in MR
Developer adds Peers for Review L1,L2
Developer Domain SME for L3 Review

12. Reviewing changes.



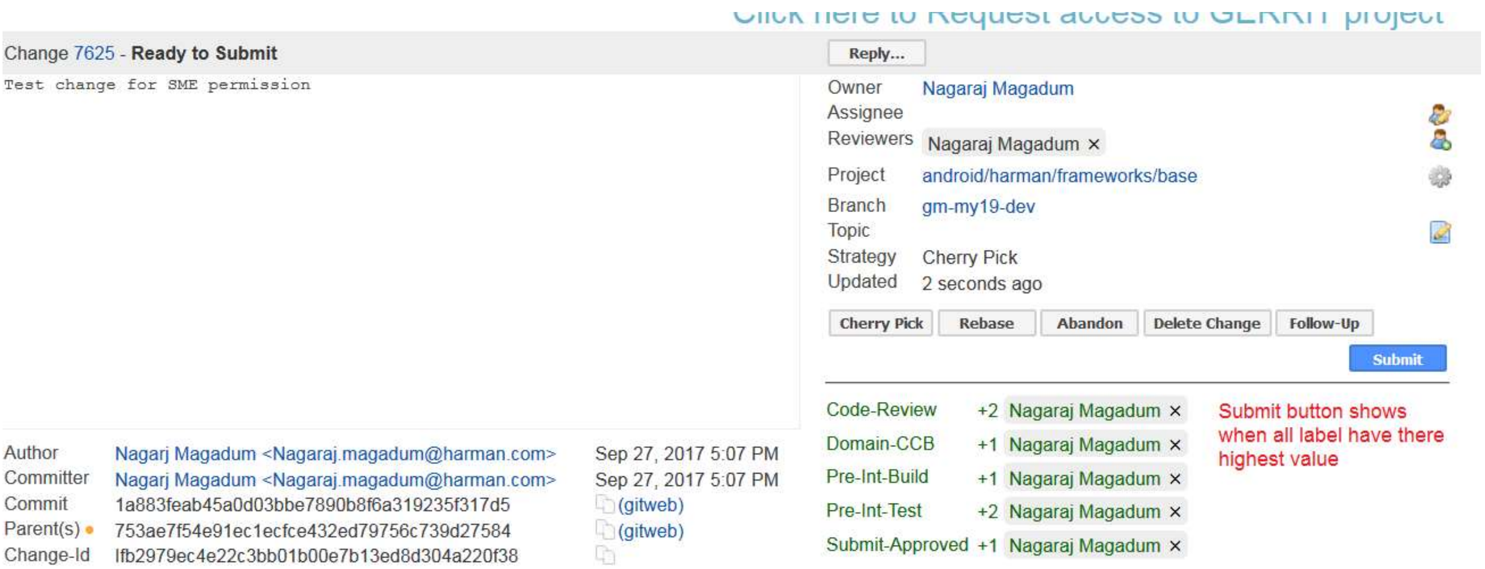
	File Path	Comments	Size
□	Commit Message		
☑	A O_Intel_WW30_baseline.xml		679
☑	ces_		
□	ces_		



```
<remove-project name="bsp/Intel/kernel/bxt"/>
<remove-project name="bsp/Intel/prebuilts/intel"/>
<remove-project name="bsp/Intel/vendor/intel/abl/abl_prebuilt/external"/>
```

```
14 <remove-project name="bsp/Intel/kernel/bxt"/>
15 <remove-project name="bsp/Intel/kernel/config-lts"/>
16 <remove-project name="bsp/Intel/prebuilts/intel"/>
17 <remove-project name="bsp/Intel/trusty/device/x86/sand"/>
18 <remove-project name="bsp/Intel/vendor/intel/abl/abl_prebuilt/external"/>
```

13. Developer submit the code, Yes you the developer submit the code.



Click here to Request access to GERRIT project

Change 7625 - **Ready to Submit**

Test change for SME permission

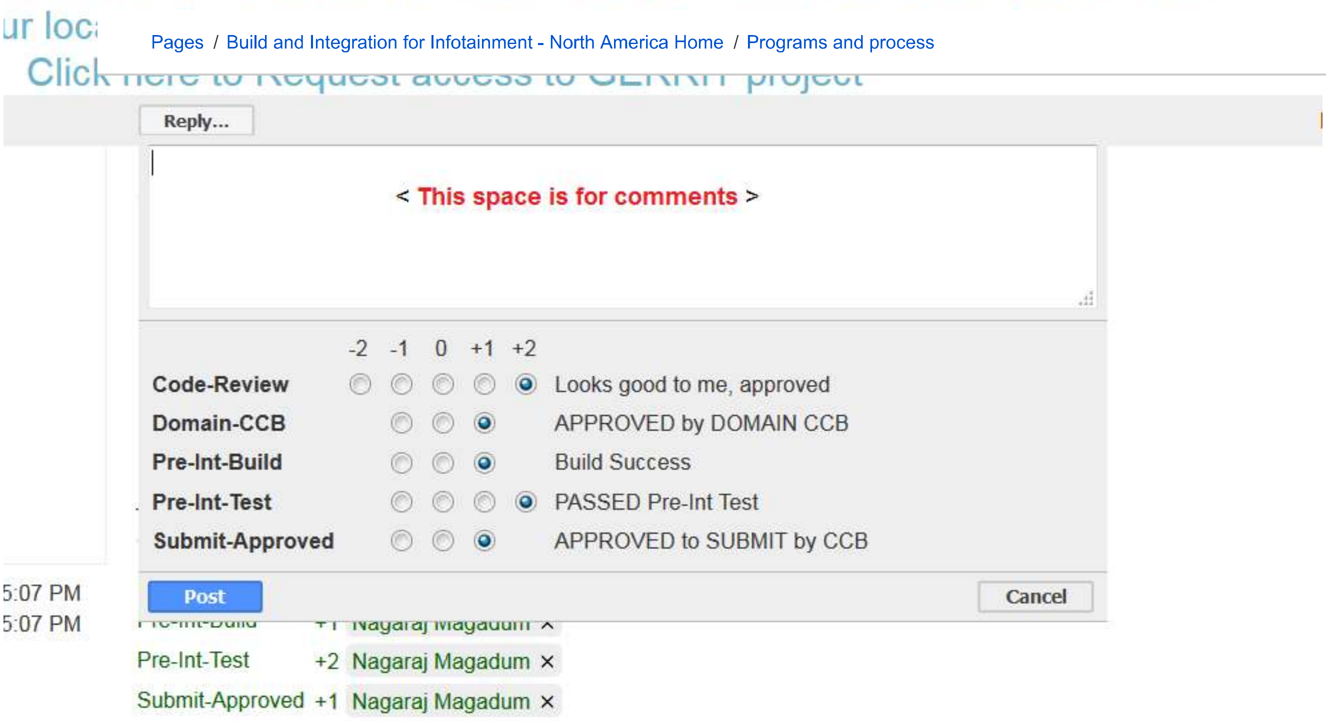
Owner	Nagaraj Magadum
Assignee	
Reviewers	Nagaraj Magadum ×
Project	android/harman/frameworks/base
Branch	gm-my19-dev
Topic	
Strategy	Cherry Pick
Updated	2 seconds ago

Author	Nagarj Magadum <Nagaraj.magadum@harman.com>	Sep 27, 2017 5:07 PM	
Committer	Nagarj Magadum <Nagaraj.magadum@harman.com>	Sep 27, 2017 5:07 PM	
Commit	1a883feab45a0d03bbe7890b8f6a319235f317d5	(gitweb)	
Parent(s)	753ae7f54e91ec1ecfce432ed79756c739d27584	(gitweb)	
Change-Id	Ifb2979ec4e22c3bb01b00e7b13ed8d304a220f38		

Code-Review	+2	Nagaraj Magadum ×
Domain-CCB	+1	Nagaraj Magadum ×
Pre-Int-Build	+1	Nagaraj Magadum ×
Pre-Int-Test	+2	Nagaraj Magadum ×
Submit-Approved	+1	Nagaraj Magadum ×

Submit button shows when all label have there highest value

Pages / Build and Integration for Infotainment - North America Home / Programs and process

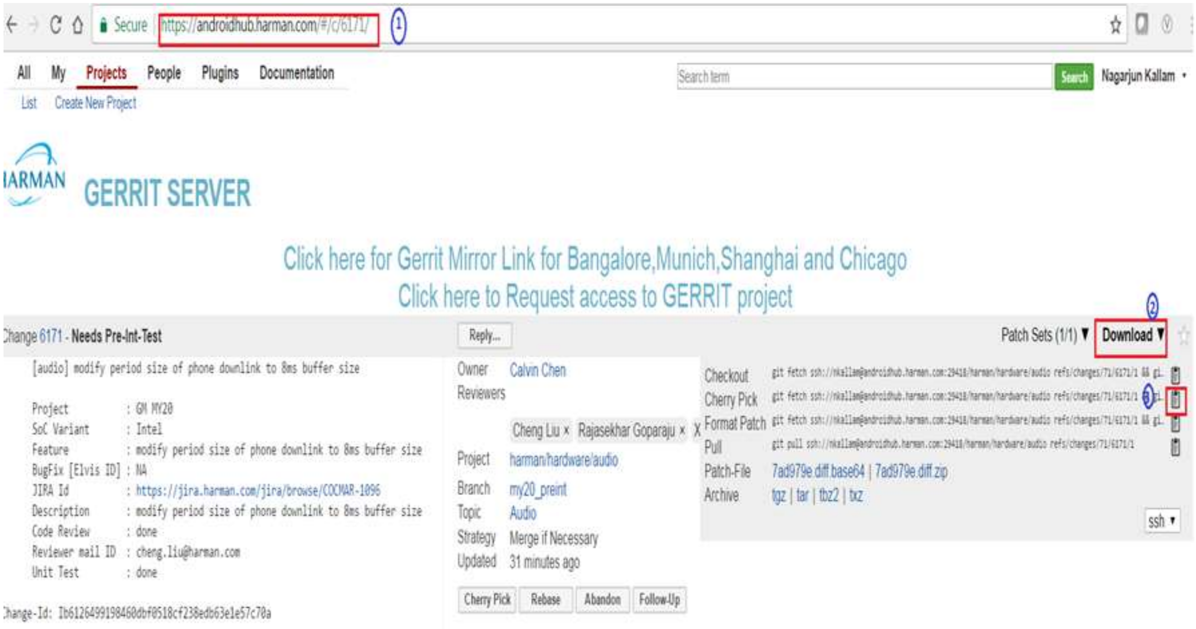

ur loc
Click here to Request access to GERRIT project
Reply...
< This space is for comments >
Post
Cancel
5:07 PM
5:07 PM

	-2	-1	0	+1	+2	
Code-Review	○	○	○	○	●	Looks good to me, approved
Domain-CCB		○	○	●		APPROVED by DOMAIN CCB
Pre-Int-Build		○	○	●		Build Success
Pre-Int-Test		○	○	○	●	PASSED Pre-Int Test
Submit-Approved		○	○	●		APPROVED to SUBMIT by CCB

Pre-Int-Build	+1	Nagaraj Magadum ×
Pre-Int-Test	+2	Nagaraj Magadum ×
Submit-Approved	+1	Nagaraj Magadum ×

14. Searching changes in Gerrit.

https://review.openstack.org/Documentation/user-search.html



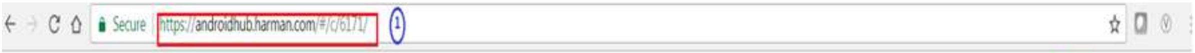
Change 6171 - Needs Pre-Int-Test

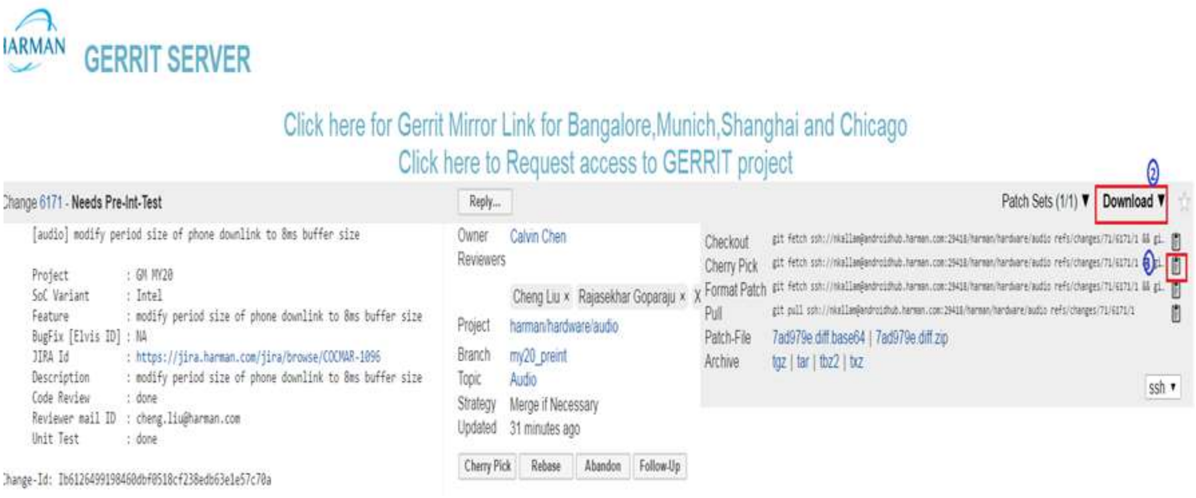
```
[audio] modify period size of phone downlink to 8ms buffer size

Project          : GM MY20
SoC Variant      : Intel
Feature          : modify period size of phone downlink to 8ms buffer size
BugFix [Elvis ID] : NA
JIRA Id          : https://jira.harman.com/jira/browse/COCMAR-1096
Description      : modify period size of phone downlink to 8ms buffer size
Code Review      : done
Reviewer mail ID : cheng.liu@harman.com
Unit Test        : done

Change-Id: Ib6126499198460dbf0518cf238edb63e1e57c70a
```

Owner	Calvin Chen
Reviewers	Cheng Liu × Rajasekhar Goparaju ×
Project	harman/hardware/audio
Branch	my20_preint
Topic	Audio
Strategy	Merge if Necessary
Updated	31 minutes ago

Checkout	git fetch ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1 && gi...
Cherry Pick	git fetch ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1 && gi...
Format Patch	git fetch ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1 && gi...
Pull	git pull ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1
Patch-File	7ad979e.diff.base64 | 7ad979e.diff.zip
Archive	tgz | tar | tbz2 | txz

15.Cherry picking changes from Gerrit.




HARMAN
GERRIT SERVER
Click here for Gerrit Mirror Link for Bangalore,Munich,Shanghai and Chicago
Click here to Request access to GERRIT project
Reply...
Patch Sets (1/1) ▼
Download ▼
2
3
Owner Calvin Chen
Reviewers Cheng Liu × Rajasekhar Goparaju × X
Cherry Pick
Rebase
Abandon
Follow-Up
ssh ▼

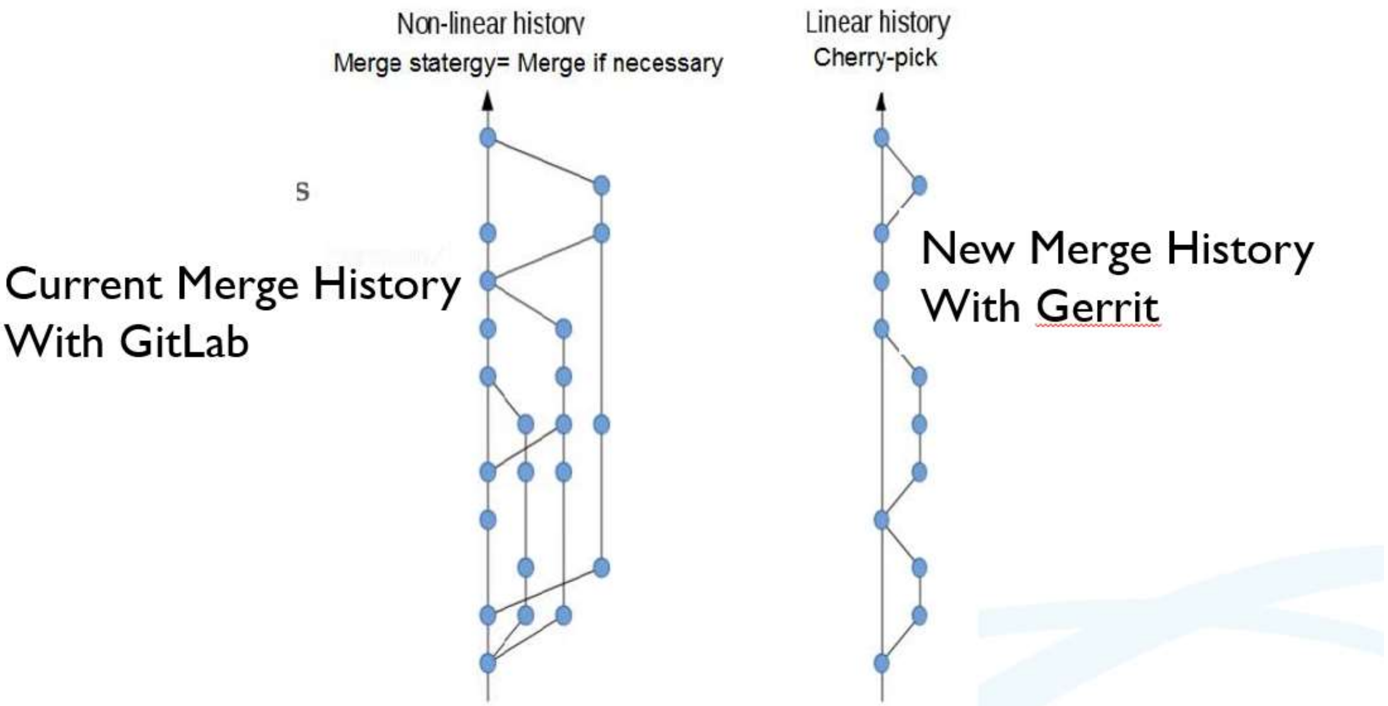
Change 6171 - Needs Pre-Int-Test

```
[audio] modify period size of phone downlink to 8ms buffer size

Project          : GM MY20
SoC Variant      : Intel
Feature          : modify period size of phone downlink to 8ms buffer size
BugFix [Elvis ID] : NA
JIRA Id          : https://jira.harman.com/jira/browse/COCMAR-1096
Description      : modify period size of phone downlink to 8ms buffer size
Code Review      : done
Reviewer mail ID : cheng.liu@harman.com
Unit Test        : done

Change-Id: Ib6126499198460dbf0518cf238edb63e1e57c70a
```

Project	harman/hardware/audio
Branch	my20_preint
Topic	Audio
Strategy	Merge if Necessary
Updated	31 minutes ago

```
Checkout      git fetch ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1 && gi...
Cherry Pick   git fetch ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1 && gi...
Format Patch  git fetch ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1 && gi...
Pull          git pull ssh://nkallam@androidhub.harman.com:29418/harman/hardware/audio refs/changes/71/6171/1
Patch-File    7ad979e.diff.base64 | 7ad979e.diff.zip
Archive       tgz | tar | tbz2 | txz
```

16. In Gerrit we are moving onto new merge strategy "cherry-pick"



Pages / Build and Integration for Infotainment - North America Home / Programs and process

git repo mainline ssh green_line

13 Comments

Stayton, Garrett

I'm guessing there is a prerequisite of having the repo tool installed on your Linux environment. Would be beneficial to have a section that covers this, before using the repo command (Essentially what this post did:

https://confluence.harman.com/confluence/display/CoCs/Android+Config+Management+%28CM%29+for+N.CAR)

Unknown User (nmagadum)

Thanks for feedback, training site updated.

Unknown User (aczajka)

when i try to cherry pick changes it asks me for a password. when i use the same password i use for gerrit it says its unauthorized and exits. ive talked to other people and they dont get a request for a password. i have no problem submitting gerrits for review. what is causing this?

Unknown User (nmagadum)

Possibly your download protocol on gerrit is http instead of ssh,

upate it and try

below download section there is drop down to select the protocol

Unknown User (aczajka)

that worked, thank you Nagaraj!

Unknown User (brdavis)

Support for RTC comming soon

ETA?

I'm not sure what to prefix my FPI for cFPI id's with for FCA R1 Feature Development... for now i guess i will manually change the final output from **ELVIS**=123456 to **RTC**=123456
Looking forward to this working correctly for my use case.

Unknown User (brdavis)

@ Unknown User (nmagadum)

I'm having trouble with **repo upload -t**

Permission denied (publickey).

My ssh seems to be setup correctly, I can successfully **repo sync**.

How do I check if I have write permissions to a project or not?

Stayton, Garrett

Its a common thing for upload to not work, use this in your repo:

```
git push origin <Your local branch>:refs/for/fca-my20-mainline/<Topic Branch>
```

Unknown User (brdavis)

Thanks

@Unknown User (nmagadum) also reached out to me via jabber.

this command worked fine, which confirms my ssh permissions and .gitconfig are configured correctly.

```
git push origin <Your local branch>:refs/for/fca-my20-mainline/<Topic Branch>
```

repo upload -t is preferred and **git push** will eventually be deprecated

@Unknown User (nmagadum) will debug in an environment like mine and get back to me:

Ubuntu 16.04.4

Python 2.7.12

Until then I will use the git push command instead.

McGladdery, Jacob

In regards to

Permission denied (publickey).

I was getting this error while attempting to run `repo init`. It turns out even though I had the public SSH key for my AWS instance added to androidhub.harman.com, that same key was not being propagated to the androidhub mirror sites. I had to login to androidhub-fh.harman.com and androidhub-cg.harman.com separately, delete my existing SSH keys (which were all old in my case), and add my current AWS SSH key manually. Once I had done that on both mirrors, I was able to run `repo init` and it worked instantly.

Arias, Franco

Hi get 2 details during getting the code:

1. during the repo sync error : `rror: Exited sync due to fetch errors` / solve including the -f in the command to force and lunch again
2. I include the .repo in ~, but during the long copy the partition almos full, so i stop the sync and copy to the correct ones (work or workspace) and restart the sync

Unknown User (nmagadum)

repo sync -c should sync the changes unless you have issue with you workspace.

repo sync -c -f will continue to sync even if any sync fails for any one of the repo.

Arias, Franco

ok, Thanks !