Qualcomm Technologies, Inc.

QCS40X Smart Audio – SSRD System Bring up Guide

80-YB420-4 Rev. B

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	June 2019	Initial release
B	July 2019	Update for r1.0_00007.0 software

Agenda

- System Overview
- Setup Build System
- Install Tools for Windows
- Obtain Software
- Build Software
- Device Boot Configuration
- Writing Images
- Basic Operation Test
- References

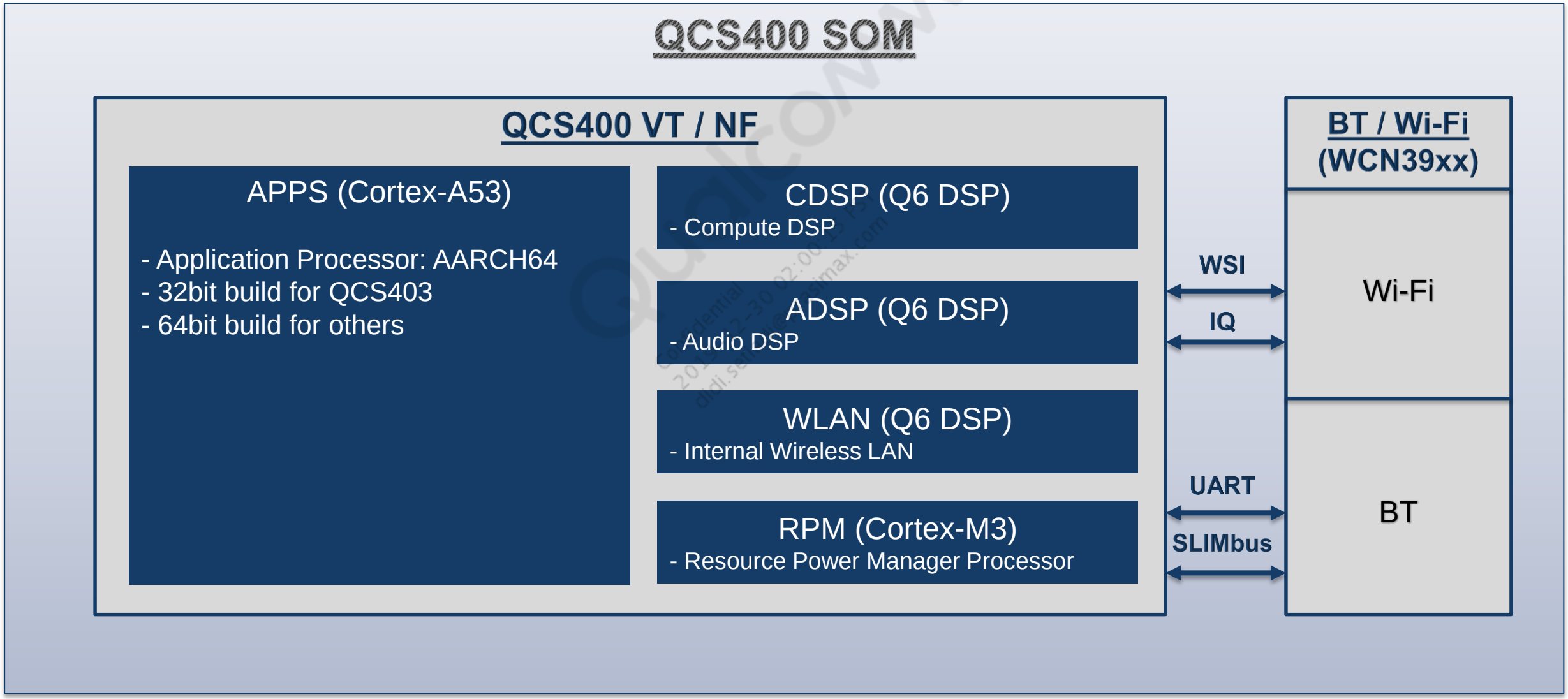
Conventions

No	Mark	Description
1	>	Prefix for commands on Windows Command Prompt
2	$	Prefix for commands on Linux Terminal
3	/ #	Prefix for commands on adb shell
4		RED Line: Line to highlight ERROR MESSAGE on terminal window
5		YELLOW Line: Line to highlight USER INPUT COMMAND on terminal window
6		ORANGE Line: Line to highlight CHECK POINT on terminal window

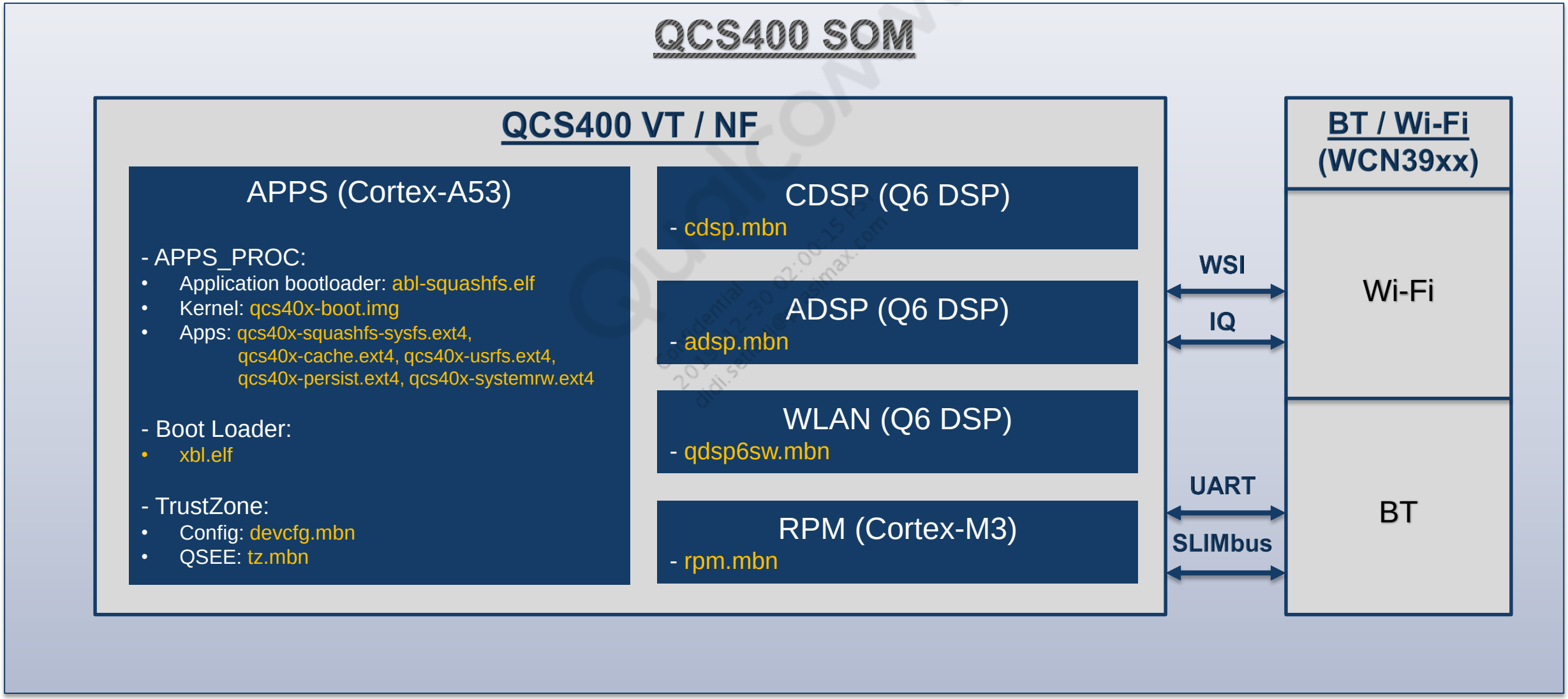
Section 1

System Overview

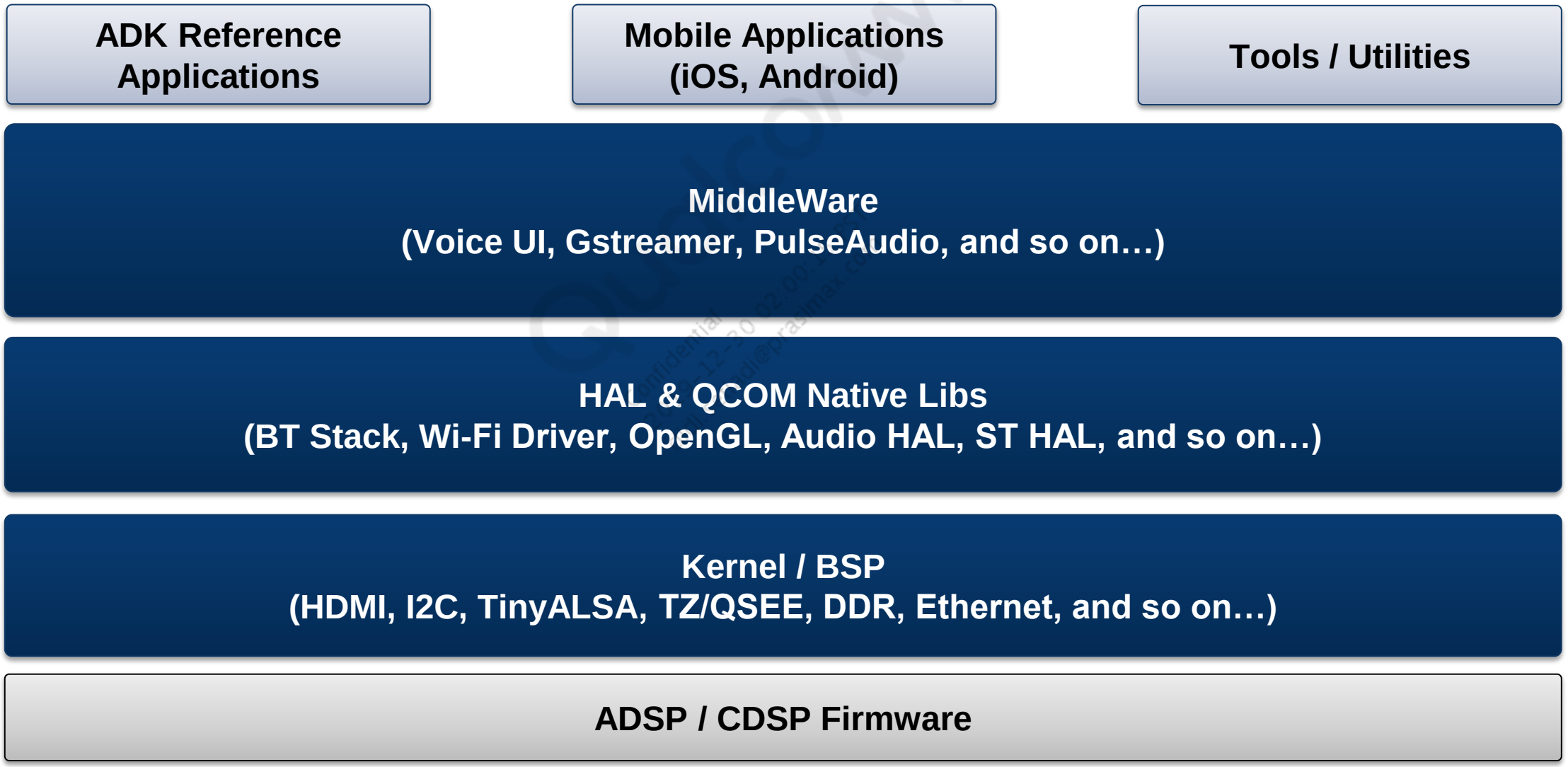
QCS400 Sub-system Overview



QCS400 Sub-system Overview – Image Files

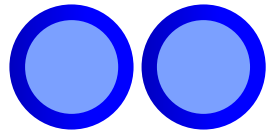


Software Overview



Section 2

Setup Build System



Section 2.1

Basic Setup

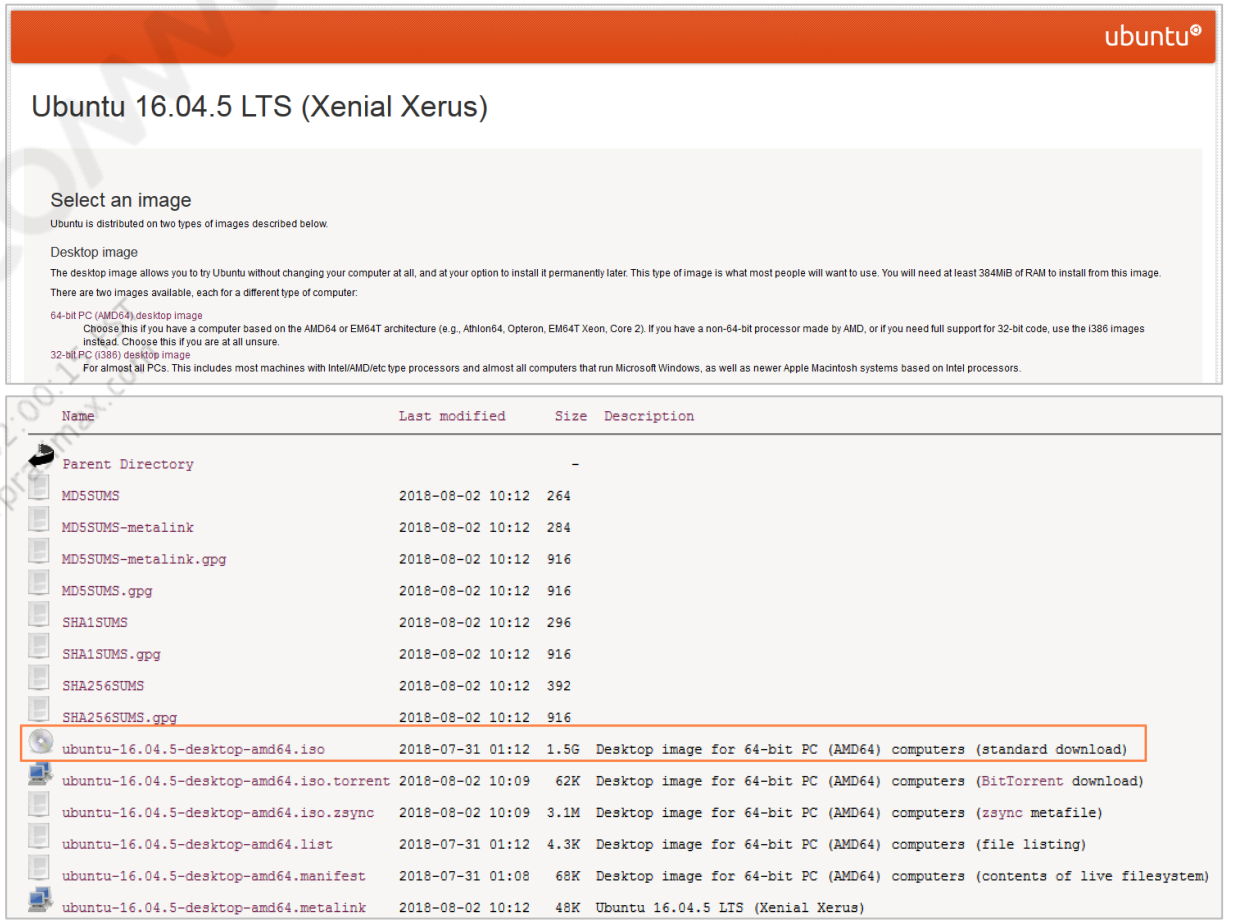
Basic Setup - Download Ubuntu

- Ubuntu 16.04 LTS for 64bit desktop PC is recommended
  - Download 'ubuntu-16.04.5-desktop-**amd64**.iso' file from: http://releases.ubuntu.com/xenial/

- Virtual Machine
  - Tested VMWare setting example:

VMWare@Workstation 14 Pro
- VMWare Version : Workstation 14.x
- Memory 8GB
- Processors = 1
- Cores per processor = 2
- HDD 200GB pre-allocated
- OS : Ubuntu 16.04 LTS (64bit build)

  - NOTE on using VMWare:
    - Unexpected build errors can occur during HLOS software build
    - You can avoid this problem by adjusting the number of build tasks
    - This will be mentioned in the Build HLOS Software chapter



Basic Setup - Installing development packages

- Update Linux System Packages
  - $ sudo apt-get update
  - $ sudo apt-get upgrade
- Setup bash
  - $ sudo ln -sf /bin/bash /bin/sh
- Install Android Tools
  - $ sudo apt-get install android-tools-adb android-tools-fastboot
- Install Build Tools
  - $ sudo apt-get install curl git git-core gawk chrpath build-essential texinfo libz-dev
  - $ sudo apt-get install gcc-arm-linux-gnueabi
  - $ sudo apt-get install gcc-multilib zlib1g:i386 libssl-dev clang llvm llvm-3.8

apt-get will choose 'zlibb1g-dev' for 'libz-dev' on ubuntu 16.04

Basic Setup - Installing Java v1.7.0 JDK

- Since Java v1.7.0 JDK is not in the Linux default repository, you need to add repository separately
  - $ sudo add-apt-repository ppa:openjdk-r/ppa
  - $ sudo apt-get update

- Install Java v1.7.0 JDK
  - $ sudo apt-get install openjdk-7-jdk

- Make sure the default Java version is 1.7.0
  - $ java -version

- If multiple Java packages installed on your Linux build system and the default Java version is not v1.7.0,
- Then you can change the default Java version to v1.7.0 as follows:
  - $ update-java-alternatives --list
  - $ sudo update-java-alternatives -s java-1.7.0-openjdk-amd64

Basic Setup - Checking Linux LLVM Toolchain

- Check the installed LLVM version: v3.8 or later is required
  - $ cd /usr/lib
  - /usr/lib$ ls | grep llvm

```
donald@ubuntu:~$ cd /usr/lib
donald@ubuntu:/usr/lib$ ls -al | grep llvm
drwxr-xr-x   8 root root   4096 Feb 28 02:28 llvm-3.8
donald@ubuntu:/usr/lib$
```

- Check the symbolic link for ‘llvm-profdata’
  - $ cd /usr/bin
  - /usr/bin$ ls -al | grep llvm-profdata

```
donald@ubuntu:~$ cd /usr/bin
donald@ubuntu:/usr/bin$ ls -al | grep llvm-profdata
lrwxrwxrwx  1 root root         33 Jul 12  2016 llvm-profdata-3.8 -> ../lib/llvm-3.8/bin/llvm-profdata
donald@ubuntu:/usr/bin$
donald@ubuntu:/usr/bin$
donald@ubuntu:/usr/bin$
donald@ubuntu:/usr/bin$
```

- If there is no the symbolic link for ‘llvm-profdata’, then create the link
  - /usr/bin$ sudo ln -s ../lib/llvm-3.8/bin/llvm-profdata
- Make sure the symbolic link for ‘llvm-profdata’
  - /usr/bin$ ls -al | grep llvm-profdata

```
donald@ubuntu:/usr/bin$ sudo ln -s ../lib/llvm-3.8/bin/llvm-profdata
[sudo] password for donald:
donald@ubuntu:/usr/bin$ ls -al | grep llvm-profdata
lrwxrwxrwx  1 root root         33 Mar 26 01:14 llvm-profdata -> ../lib/llvm-3.8/bin/llvm-profdata
lrwxrwxrwx  1 root root         33 Jul 12  2016 llvm-profdata-3.8 -> ../lib/llvm-3.8/bin/llvm-profdata
donald@ubuntu:/usr/bin$
donald@ubuntu:/usr/bin$
donald@ubuntu:/usr/bin$
donald@ubuntu:/usr/bin$
```

Basic Setup - Installing repo

- Install repo
  - $ mkdir ~/bin
  - $ PATH=~/bin:$PATH
  - $ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
  - $ chmod a+x ~/bin/repo
- Initialize repo client
  - $ git config --global user.name donald
  - $ git config --global user.email donald@domain.com
- Reference
  - https://source.android.com/setup/downloading

Use your build machine login id and password in place of doland and donald@domain.com respectively

Basic Setup - Installing samba

- Samba is required for file sharing with Windows test machine
- The following processes require Samba:
  - Writing Images using QFIL
  - Writing Images using fastboot on Windows
  - Packaging Images using QFIL

Basic Setup - Installing samba

- Install samba
  - $ sudo apt-get install samba
- Add account
  - $ sudo gedit /etc/samba/smb.conf

```
[donald]
comment = Donald Duck
path =/home/donald
valid users = donald
public = yes
writable = yes
```

- Password setup
  - $ sudo smbpasswd -a donald
- Restart samba service
  - $ sudo service smbd restart

Use your build machine login id and password in place of doland and donald@domain.com respectively

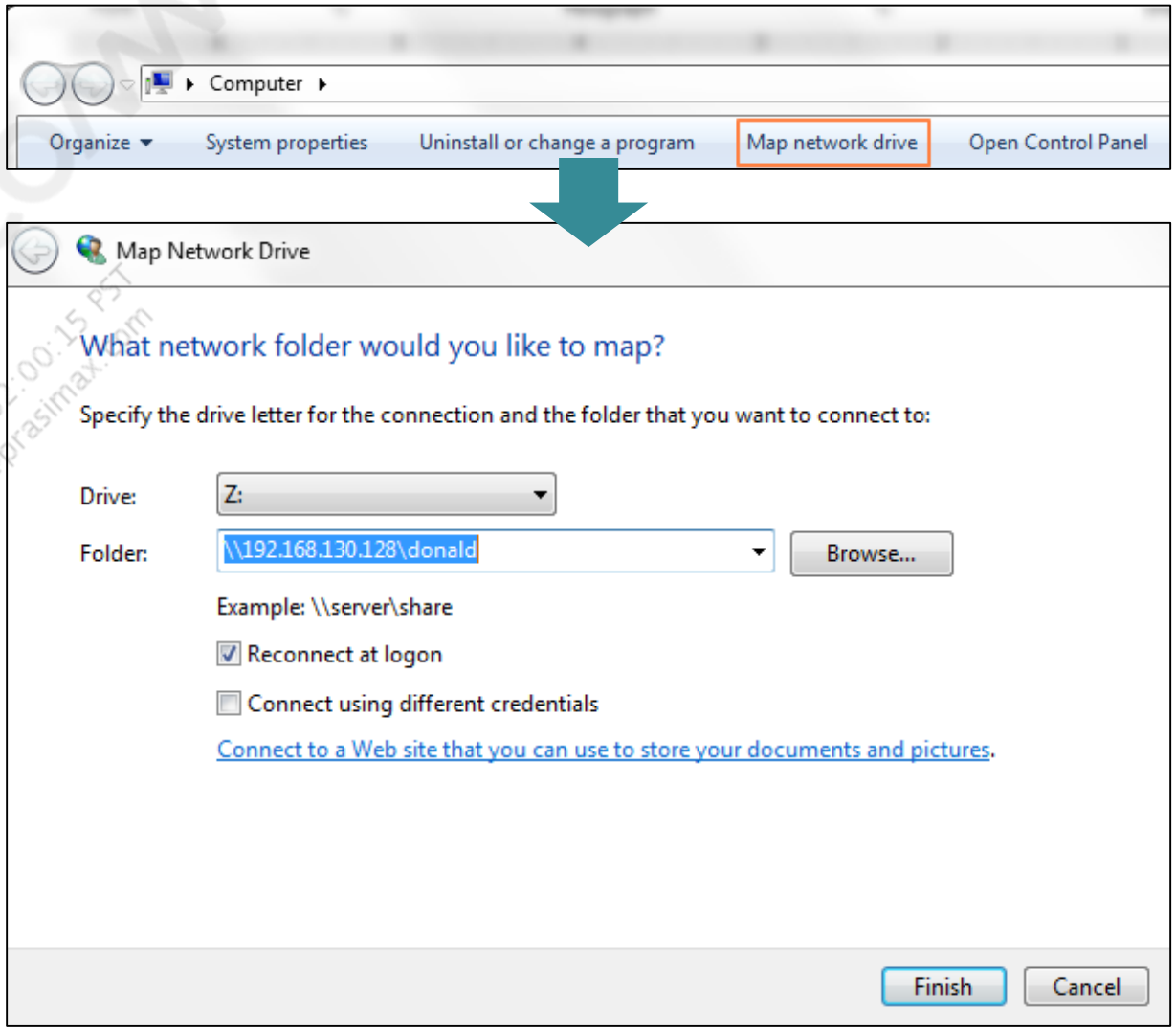
Basic Setup - Connecting to samba

- Check IP address of Linux machine

```
donald@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:ed:2b:61
          inet addr:192.168.130.128  Bcast:192.168.130.255  Mask:255.255.255.0
          inet6 addr: fe80::f6f2:c54f:f34c:49c2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:285653 errors:0 dropped:0 overruns:0 frame:0
          TX packets:87996 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:353154476 (353.1 MB)  TX bytes:5349289 (5.3 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:327 errors:0 dropped:0 overruns:0 frame:0
          TX packets:327 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27365 (27.3 KB)  TX bytes:27365 (27.3 KB)

donald@ubuntu:~$
```

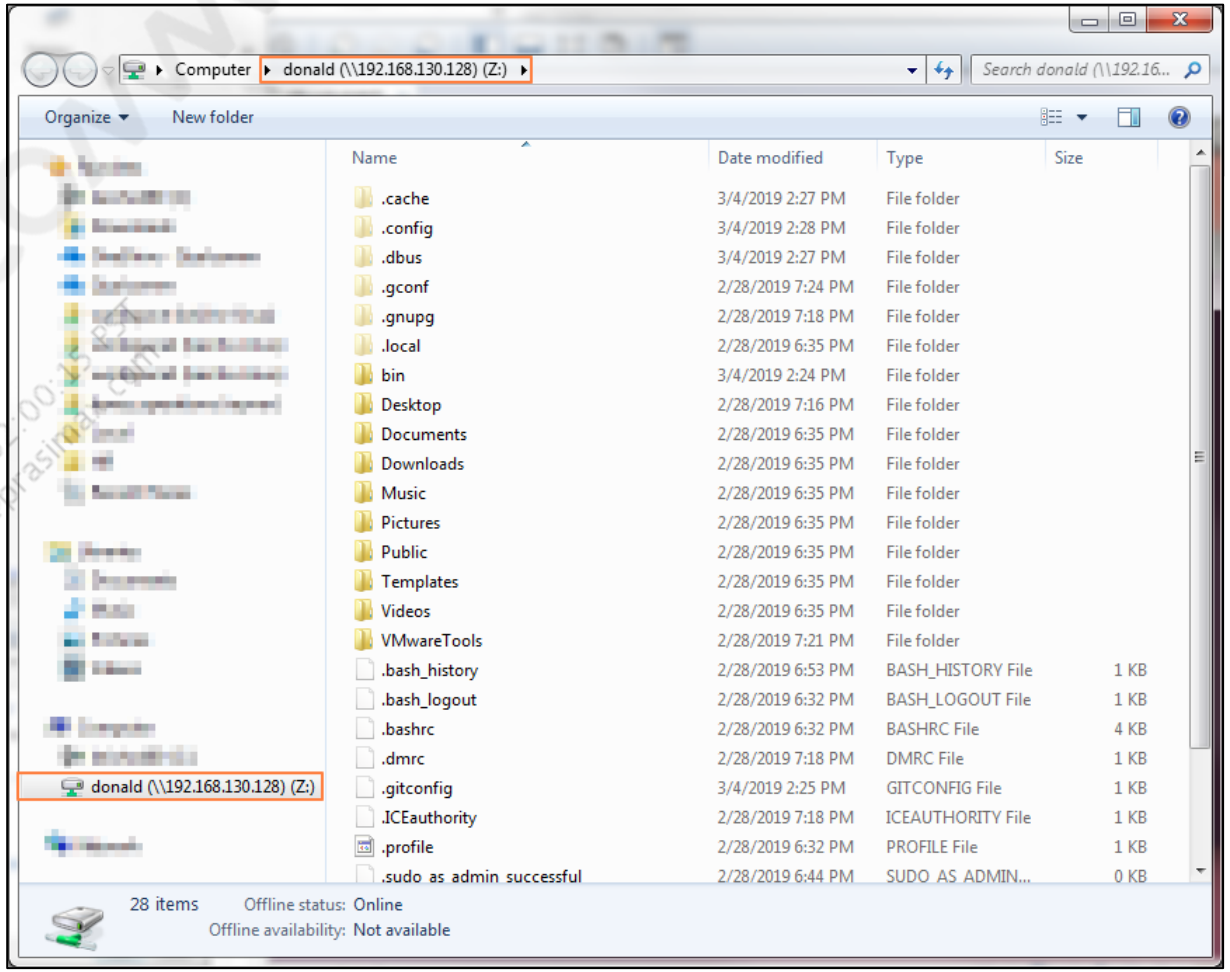
Basic Setup - Connecting to samba

- Check IP address of Linux machine
- Open “Map Network Drive” windows from Windows File Explorer
- Specify a network folder to map
  - *e.g. \\192.168.130.128\donald*
- Click on ‘Finish’ button



Basic Setup - Connecting to samba

- Check IP address of Linux machine
- Open “Map Network Drive” windows from Windows File Explorer
- Specify a network folder to map
  - *e.g.* *\\192.168.130.128\donald*
- Click on ‘Finish’ button
- Make sure connection state on Windows File Explorer
- DONE !



Basic Setup - Installing ssh

- Ubuntu Terminal requires large resource
- Build machine may crash during build process unless RAM size of the build system is big enough
- The terminal window launched from Ubuntu desktop is known to consume good amount of system resource
- Therefore, it’s recommended for light weight build machine users to rely on remote terminal which requires SSH

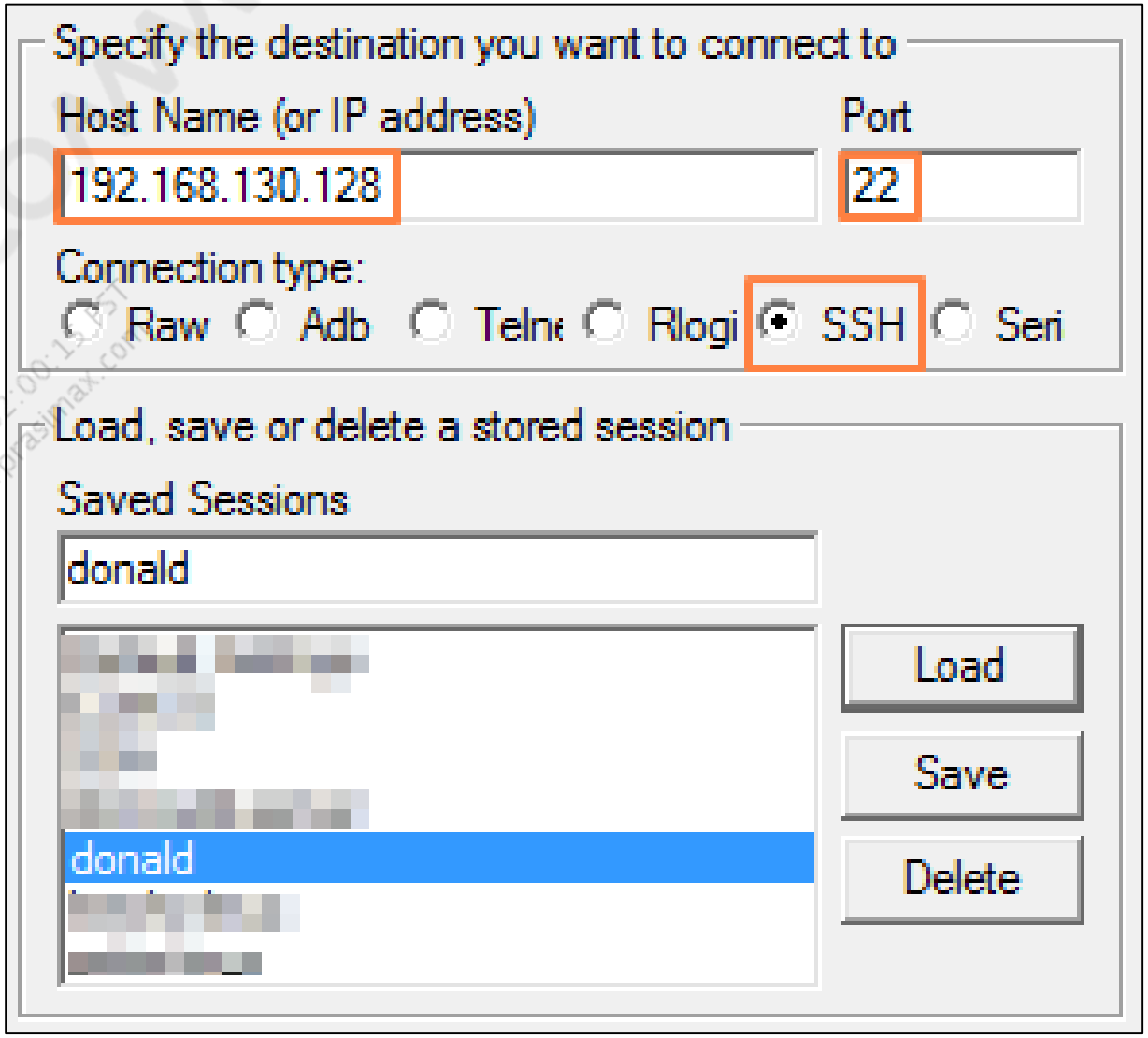
Basic Setup - Installing ssh

- Install ssh
  - $ sudo apt-get install ssh
- Check IP address of Linux machine

```
donald@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:ed:2b:61
          inet addr:192.168.130.128  Bcast:192.168.130.255  Mask:255.255.255.0
          inet6 addr: fe80::f6f2:c54f:f34c:49c2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:285653 errors:0 dropped:0 overruns:0 frame:0
          TX packets:87996 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:353154476 (353.1 MB)  TX bytes:5349289 (5.3 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:327 errors:0 dropped:0 overruns:0 frame:0
          TX packets:327 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27365 (27.3 KB)  TX bytes:27365 (27.3 KB)

donald@ubuntu:~$
```

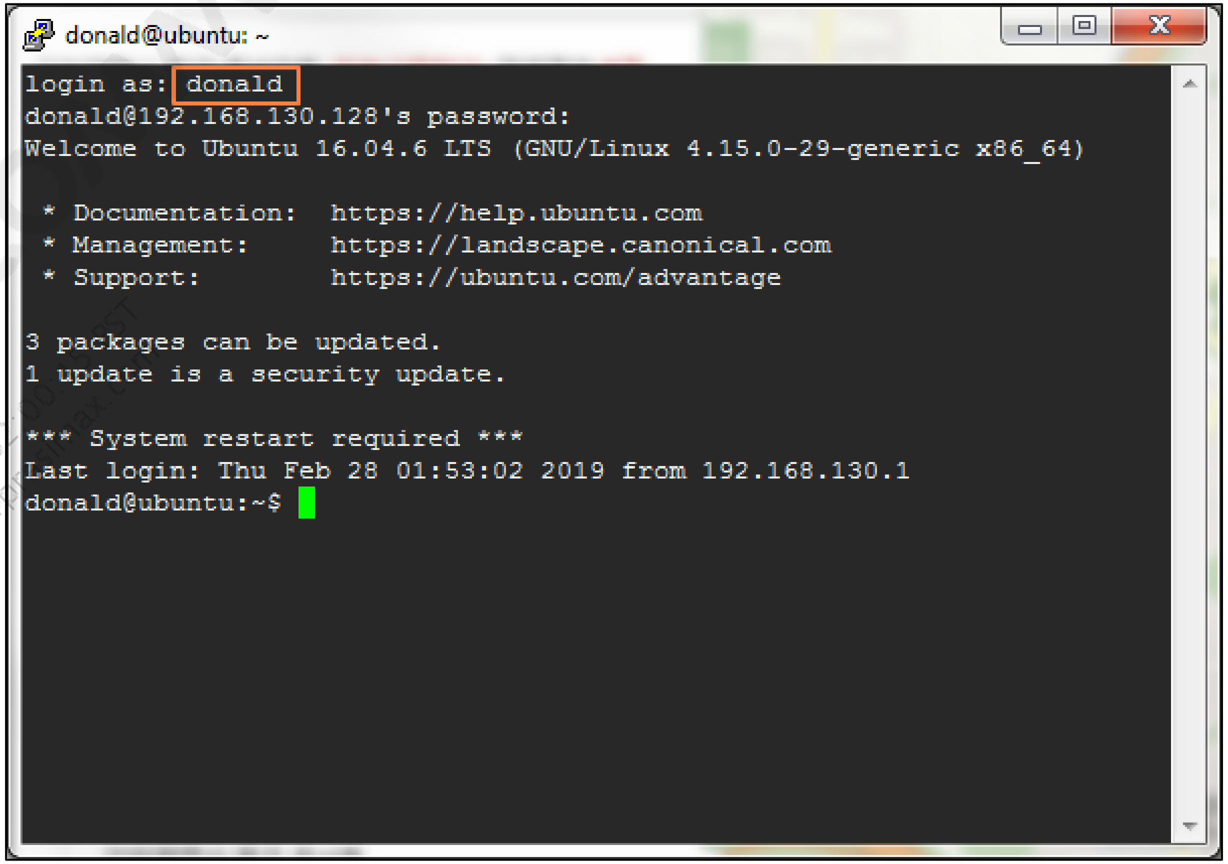
Basic Setup - Installing ssh

- Ubuntu Terminal requires large resource so user may encounter system crash during build process unless there is big enough RAM in the build system. Therefore, we are suggesting SSH which is known to use less resource
- Install ssh
  - $ sudo apt-get install ssh
- Check IP address of Linux machine
- Open your preferred terminal application and connect with following setting
  - *e.g. : [Putty]*
  - *Connection type : SSH*
  - *Host IP address : 192.168.130.128*
  - *Port : 22 (is default ssh port number)*



Basic Setup - Installing ssh

- Ubuntu Terminal requires large resource so user may encounter system crash during build process unless there is big enough RAM in the build system. Therefore, we are suggesting SSH which is known to use less resource
- Install ssh
  - $ sudo apt-get install ssh
- Check IP address of Linux machine
- Open your preferred terminal application and connect with following setting
  - *e.g. : [Putty]*
  - *Connection type : SSH*
  - *Host IP address : 192.168.130.128*
  - *Port : 22 (is default ssh port number)*
- Check login as your account
- DONE !



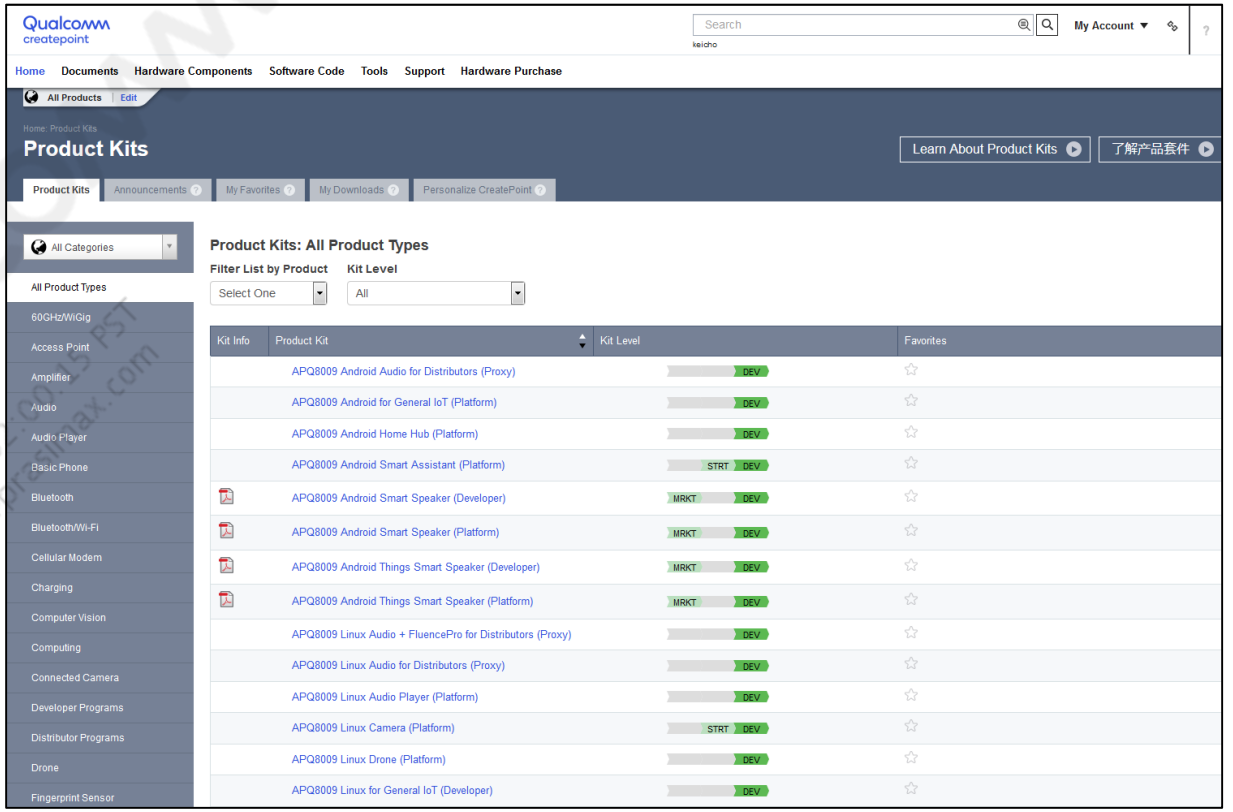
Section 2.2

Installing Toolchains

Installing Toolchains

- Required Toolchains for building non-HLOS images:
  - Snapdragon LLVM ARM Toolchain
  - Qualcomm Hexagon LLVM Toolchain
  - GCC Linaro Toolchain

- Qualcomm Toolchains are available in Qualcomm CreatePoint
  - https://createpoint.qti.qualcomm.com/dashboard/



Section 2.2.1

Snapdragon LLVM ARM Toolchain

Snapdragon LLVM ARM Toolchain

- First, check required SD(Snapdragon) LLVM version in *[80-YB405-129] QCS40x Software User Guide* document
- Below are required SD LLVM version by each SW module based on *[80-YB405-129] QCS40x Software User Guide*

SW module	SD LLVM version
RPM	3.9.2
Boot Loader	4.0.2
Trust Zone	4.0.11

QCS40x Software

User Guide

80-YB405-129 Rev. F

December 31, 2018

2.5.5 Build and compile RPM

The compiler version is Qualcomm® Snapdragon™ Mobile Platform LLVM 3.9.2.

2.5.6 Build and compile boot loader

The compiler version is Qualcomm Snapdragon Mobile Platform LLVM v4.0.2.

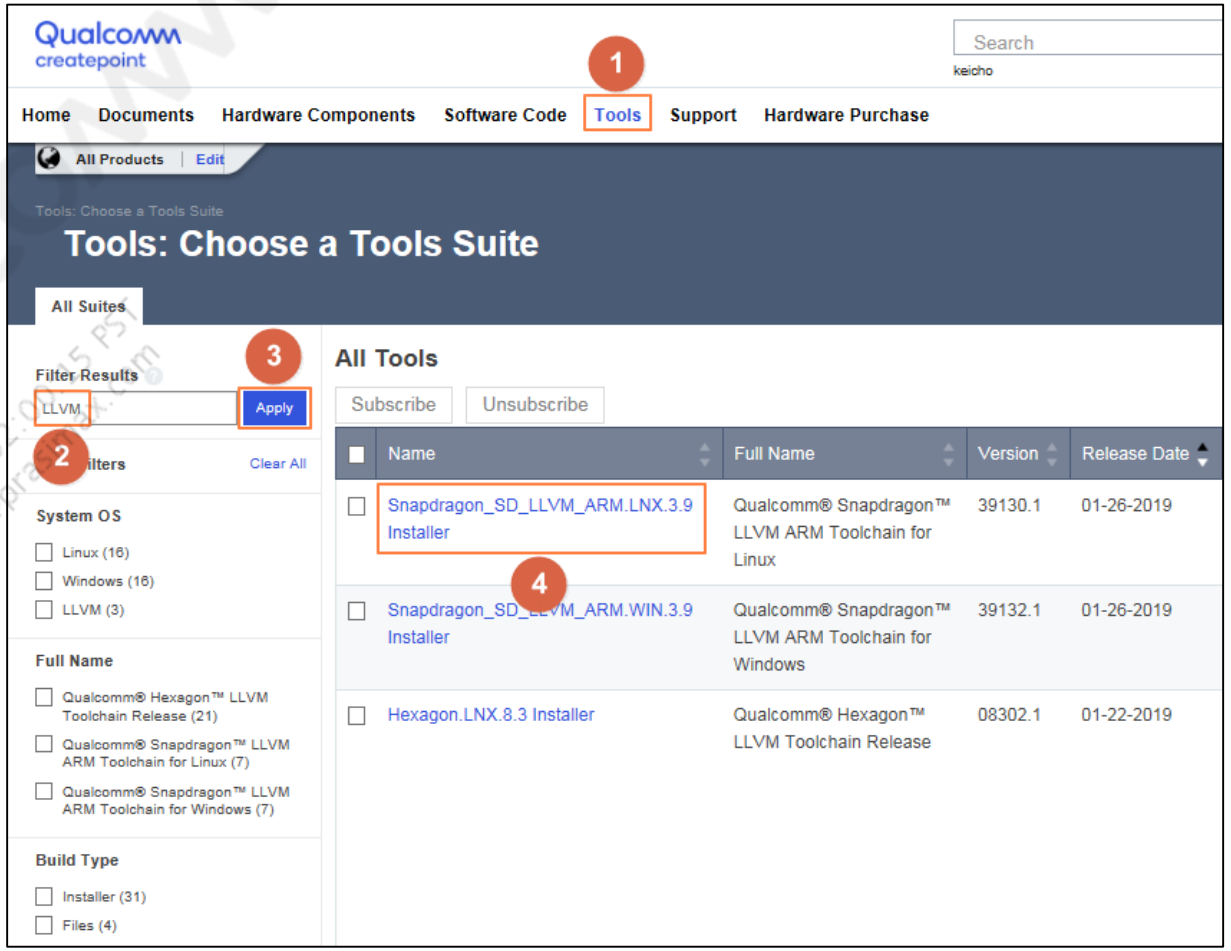
2.5.7 Build and compile TrustZone

The following are the compiler versions:

- QTI – Snapdragon LLVM v4.0.11

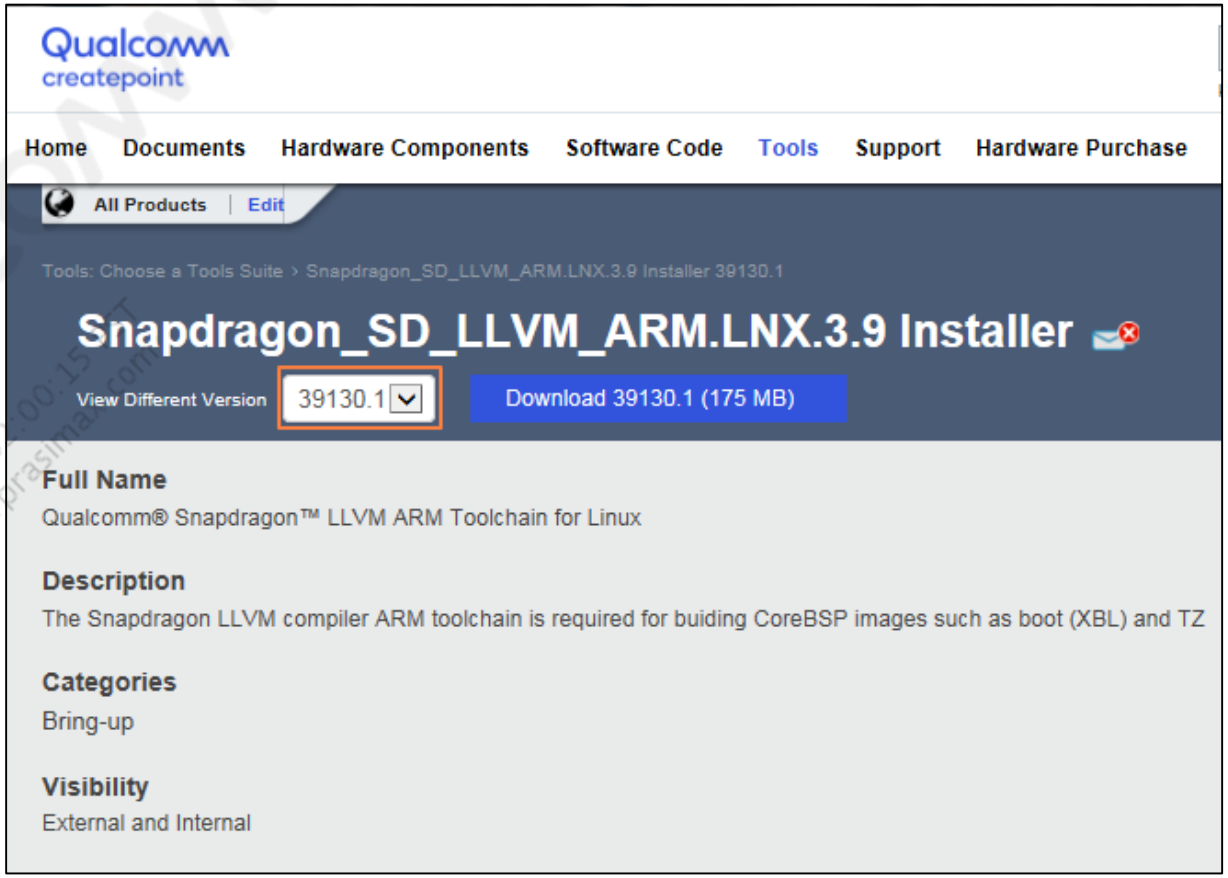
Snapdragon LLVM ARM Toolchain – Download

- To download SD LLVM,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on 'Tools' tab
- (2) Input "LLVM" into 'Filter Results'
- (3) Click on 'Apply' button
- (4) Find required SD LLVM version
  - *e.g. '3.9' for SD LLVM 3.9.2*
  - Make sure 'Snapdragon LLVM ARM Toolchain for Linux'
  - Click on the item in 'Name' column



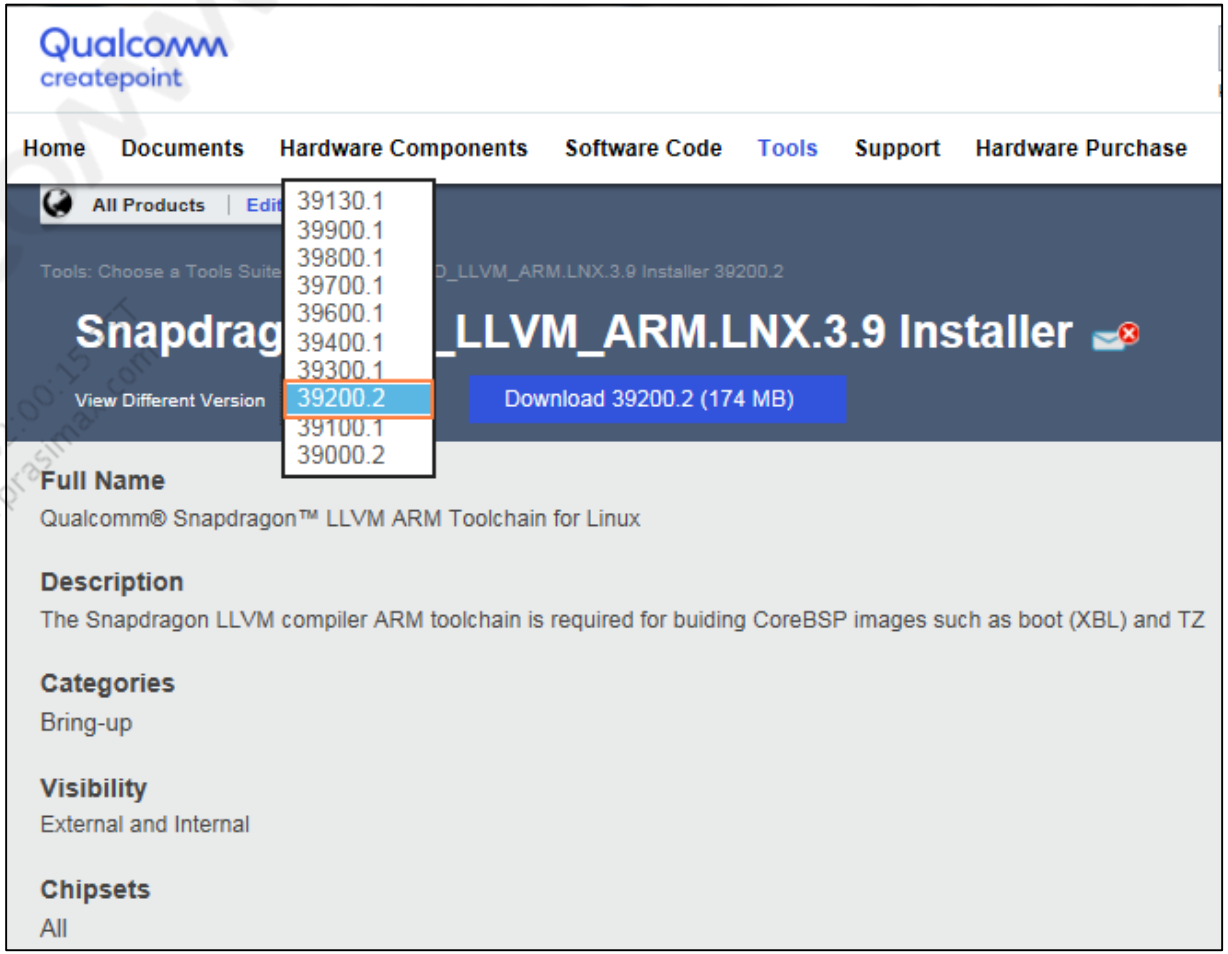
Snapdragon LLVM ARM Toolchain – Download

- To download SD LLVM,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “LLVM” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Find required SD LLVM version
  - *e.g. ‘3.9’ for SD LLVM 3.9.2*
  - Make sure ‘Snapdragon LLVM ARM Toolchain for Linux’
  - Click on the item in ‘Name’ column
- Click on ‘View Different Version’ combo box for specific version setting



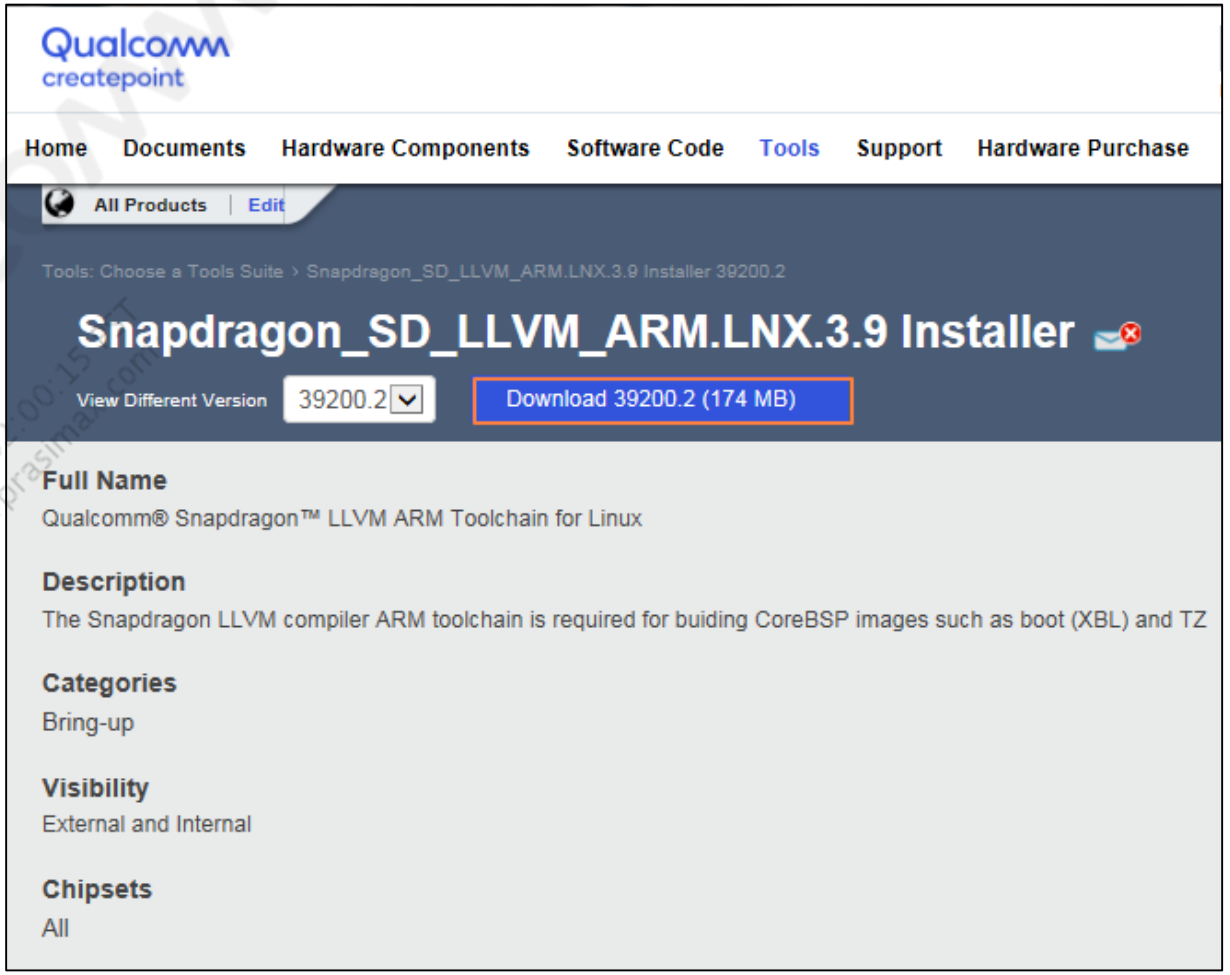
Snapdragon LLVM ARM Toolchain – Download

- To download SD LLVM,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “LLVM” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Find required SD LLVM version
  - *e.g. ‘3.9’ for SD LLVM 3.9.2*
  - Make sure ‘Snapdragon LLVM ARM Toolchain for Linux’
  - Click on the item in ‘Name’ column
- Click on ‘View Different Version’ combo box for specific version setting
- Select specific version
  - *e.g. ‘39200.2’ for SD LLVM 3.9.2*



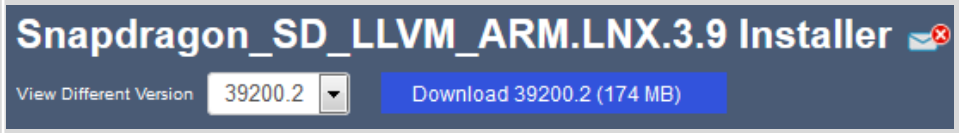
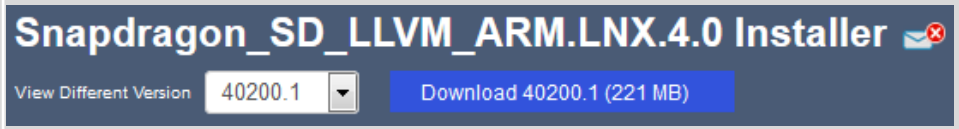
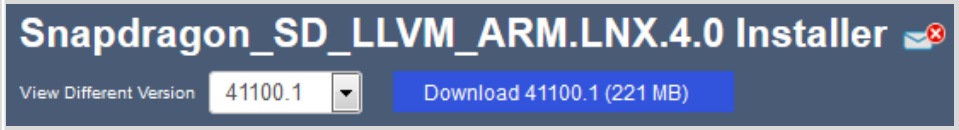
Snapdragon LLVM ARM Toolchain – Download

- To download SD LLVM,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “LLVM” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Find required SD LLVM version
  - *e.g. ‘3.9’ for SD LLVM 3.9.2*
  - Make sure ‘Snapdragon LLVM ARM Toolchain for Linux’
  - Click on the item in ‘Name’ column
- Click on ‘View Different Version’ combo box for specific version setting
- Select specific version
  - *e.g. ‘39200.2’ for SD LLVM 3.9.2*
- Click on ‘Download’ button to start downloading
  - A compressed file will be downloaded



Snapdragon LLVM ARM Toolchain – Download

- Repeat Download step for all SD LLVM Toolchains
  - [3.9.2]
  - [4.0.2]
  - [4.0.11]

Version	Capture from CreatePoint
3.9.2	
4.0.2	
4.0.11	

- Make sure that the downloaded files are in <Download_Folder>

```
snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar
snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip
snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip
```

Snapdragon LLVM ARM Toolchain – Install

- Create <SD LLVM version> folders in <Download_Folder>
  - <Download_Folder>$ mkdir 3.9.2
  - <Download_Folder>$ mkdir 4.0.2
  - <Download_Folder>$ mkdir 4.0.11

```
~/Downloads$ mkdir 3.9.2
~/Downloads$ mkdir 4.0.2
~/Downloads$ mkdir 4.0.11
~/Downloads$
~/Downloads$ ls -al

donald donald      4096 Apr 21 21:31 .
donald donald      4096 Apr 21 19:32 ..
donald donald      4096 Apr 21 21:31 3.9.2
donald donald      4096 Apr 21 21:31 4.0.11
donald donald      4096 Apr 21 21:31 4.0.2
donald donald 551893237 Dec  6  2017 Hexagon.LLVM_linux_installer_8.2.02.bin
donald donald 548279768 Mar  4 19:22 hexagon.lnx.8.2_installer_08202.303-05-19_02_56_23.zip
donald donald 551895040 Mar  4 18:56 Hexagon_LNX_8_2_Installer_08202_3.tar
donald donald 182108160 Apr 21 21:28 snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar
donald donald 230600801 Apr 21 21:29 snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip
donald donald 230590321 Apr 21 21:29 snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip
~/Downloads$
```

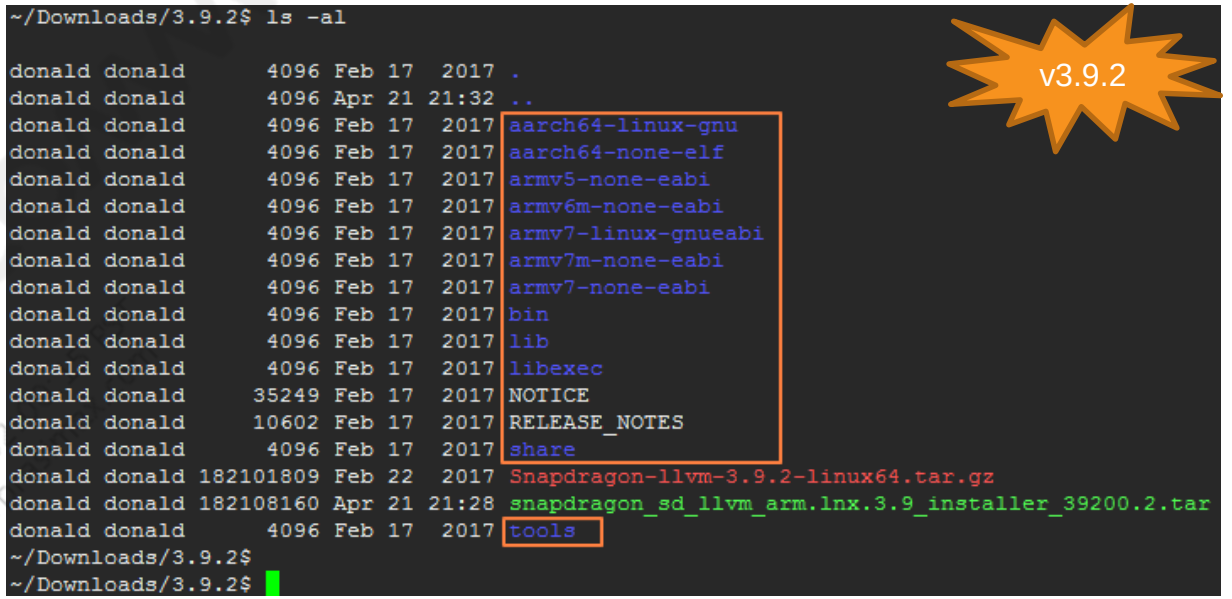
Snapdragon LLVM ARM Toolchain – Install

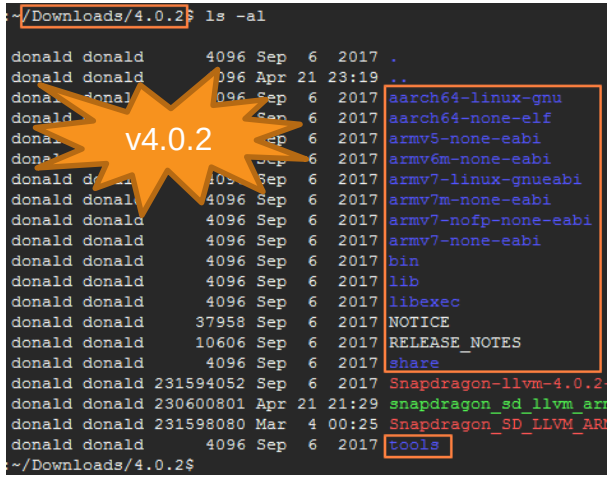
- Create <SD LLVM version> folders in <Download_Folder>
  - <Download_Folder>$ mkdir 3.9.2
  - <Download_Folder>$ mkdir 4.0.2
  - <Download_Folder>$ mkdir 4.0.11
- Move each downloaded file to each <SD LLVM version> folder
  - $ mv snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar 3.9.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip 4.0.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip 4.0.11/

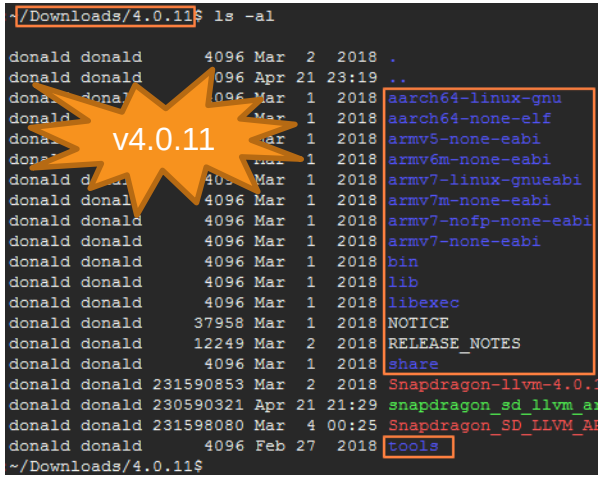
```
donald@ubuntu:~/Downloads$ mv snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar 3.9.2/
donald@ubuntu:~/Downloads$ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip 4.0.2/
donald@ubuntu:~/Downloads$ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip 4.0.11/
donald@ubuntu:~/Downloads$
donald@ubuntu:~/Downloads$ cd 3.9.2
donald@ubuntu:~/Downloads/3.9.2$ ls
snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar
donald@ubuntu:~/Downloads/3.9.2$
donald@ubuntu:~/Downloads/3.9.2$ cd ../4.0.2
donald@ubuntu:~/Downloads/4.0.2$ ls
snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip
donald@ubuntu:~/Downloads/4.0.2$
donald@ubuntu:~/Downloads/4.0.2$ cd ../4.0.11
donald@ubuntu:~/Downloads/4.0.11$ ls
snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip
donald@ubuntu:~/Downloads/4.0.11$
```

Snapdragon LLVM ARM Toolchain – Install

- Create <SD LLVM version> folders in <Download_Folder>
  - <Download_Folder>$ mkdir 3.9.2
  - <Download_Folder>$ mkdir 4.0.2
  - <Download_Folder>$ mkdir 4.0.11
- Move each downloaded file to each <SD LLVM version> folder
  - $ mv snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar 3.9.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip 4.0.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip 4.0.11/
- Uncompresse each downloaded file with following commands and make sure the result
  - <Download_Folder>$ cd <SD LLVM version>
  - <SD LLVM version>$ unzip <file_name>.zip
  - <SD LLVM version>$ tar -xvf <file_name>.tar
  - <SD LLVM version>$ tar -xzvf <file_name>.tar.gz







Snapdragon LLVM ARM Toolchain – Install

- Create <SD LLVM version> folders in <Download_Folder>
  - $ mkdir 3.9.2
  - $ mkdir 4.0.2
  - $ mkdir 4.0.11
- Move each downloaded file to each <SD LLVM version> folder
  - $ mv snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar 3.9.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip 4.0.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip 4.0.11/
- Uncompress each downloaded file with following commands and make sure the result
  - <Download_Folder>$ cd <SD LLVM version>
  - <SD LLVM version>$ unzip <file_name>.zip
  - <SD LLVM version>$ tar -xvf <file_name>.tar
  - <SD LLVM version>$ tar -xzvf <file_name>.tar.gz
- Create '/pkg/qct/software/llvm/release/arm/' folder
  - $ sudo mkdir -p /pkg/qct/software/llvm/release/arm
  - $ sudo chmod 755 /pkg/qct/software/llvm/release/arm


```
donald@ubuntu:~/Downloads$ sudo mkdir -p /pkg/qct/software/llvm/release/arm
donald@ubuntu:~/Downloads$ sudo chmod 755 /pkg/qct/software/llvm/release/arm
donald@ubuntu:~/Downloads$ ls -al /pkg/qct/software/llvm/release
total 12
drwxr-xr-x 3 root root 4096 Mar  3 23:57 .
drwxr-xr-x 3 root root 4096 Mar  3 23:57 ..
drwxr-xr-x 2 root root 4096 Apr 21 23:19 arm
donald@ubuntu:~/Downloads$
```

Snapdragon LLVM ARM Toolchain – Install

- Create <SD LLVM version> folders in <Download_Folder>
  - $ mkdir 3.9.2
  - $ mkdir 4.0.2
  - $ mkdir 4.0.11
- Move each downloaded file to each <SD LLVM version> folder
  - $ mv snapdragon_sd_llvm_arm.lnx.3.9_installer_39200.2.tar 3.9.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_40200.103-04-19_08_25_16.zip 4.0.2/
  - $ mv snapdragon_sd_llvm_arm.lnx.4.0_installer_41100.103-04-19_08_24_36.zip 4.0.11/
- Uncompress each downloaded file with following commands and make sure the result
  - <Download_Folder>$ cd <SD LLVM version>
  - <SD LLVM version>$ unzip <file_name>.zip
  - <SD LLVM version>$ tar -xvf <file_name>.tar
  - <SD LLVM version>$ tar -xzvf <file_name>.tar.gz
- Create '/pkg/qct/software/llvm/release/arm/' folder
  - $ sudo mkdir -p /pkg/qct/software/llvm/release/arm
  - $ sudo chmod 755 /pkg/qct/software/llvm/release/arm
- Move all <SD LLVM version> folders to '…/arm/' folder
  - <Download_Folder>$ sudo mv 3.9.2 /pkg/qct/software/llvm/release/arm/
  - <Download_Folder>$ sudo mv 4.0.2 /pkg/qct/software/llvm/release/arm/
  - <Download_Folder>$ sudo mv 4.0.11 /pkg/qct/software/llvm/release/arm/

```
~/Downloads$ sudo mv 3.9.2 /pkg/qct/software/llvm/release/arm/
~/Downloads$ sudo mv 4.0.2 /pkg/qct/software/llvm/release/arm/
~/Downloads$ sudo mv 4.0.11 /pkg/qct/software/llvm/release/arm/
```

Snapdragon LLVM ARM Toolchain – Install

- Make sure the result
  - $ cd /pkg/qct/software/llvm/release/arm
  - $ ls -al
- SD LLVM installation DONE !

```
:~/Downloads$ cd /pkg/qct/software/llvm/release/arm
:/pkg/qct/software/llvm/release/arm$
:/pkg/qct/software/llvm/release/arm$ ls -al

 root   root   4096 Apr 21 23:32 .
 root   root   4096 Mar  3 23:57 ..
 donald donald 4096 Feb 17  2017 3.9.2
 donald donald 4096 Mar  2  2018 4.0.11
 donald donald 4096 Sep  6  2017 4.0.2
:/pkg/qct/software/llvm/release/arm$
```

Section 2.2.2

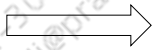
Qualcomm Hexagon LLVM Toolchain

Qualcomm Hexagon LLVM Toolchain

- First, make sure required Hexagon Tool version in *[80-YB405-129] QCS40x Software User Guide* document

- Below are required Hexagon Tool version based on *[80-YB405-129] QCS40x Software User Guide*

SW module	Hexagon Tool version
ADSP, CDSP	8.2.02



2.5.1 Build and compile ADSP

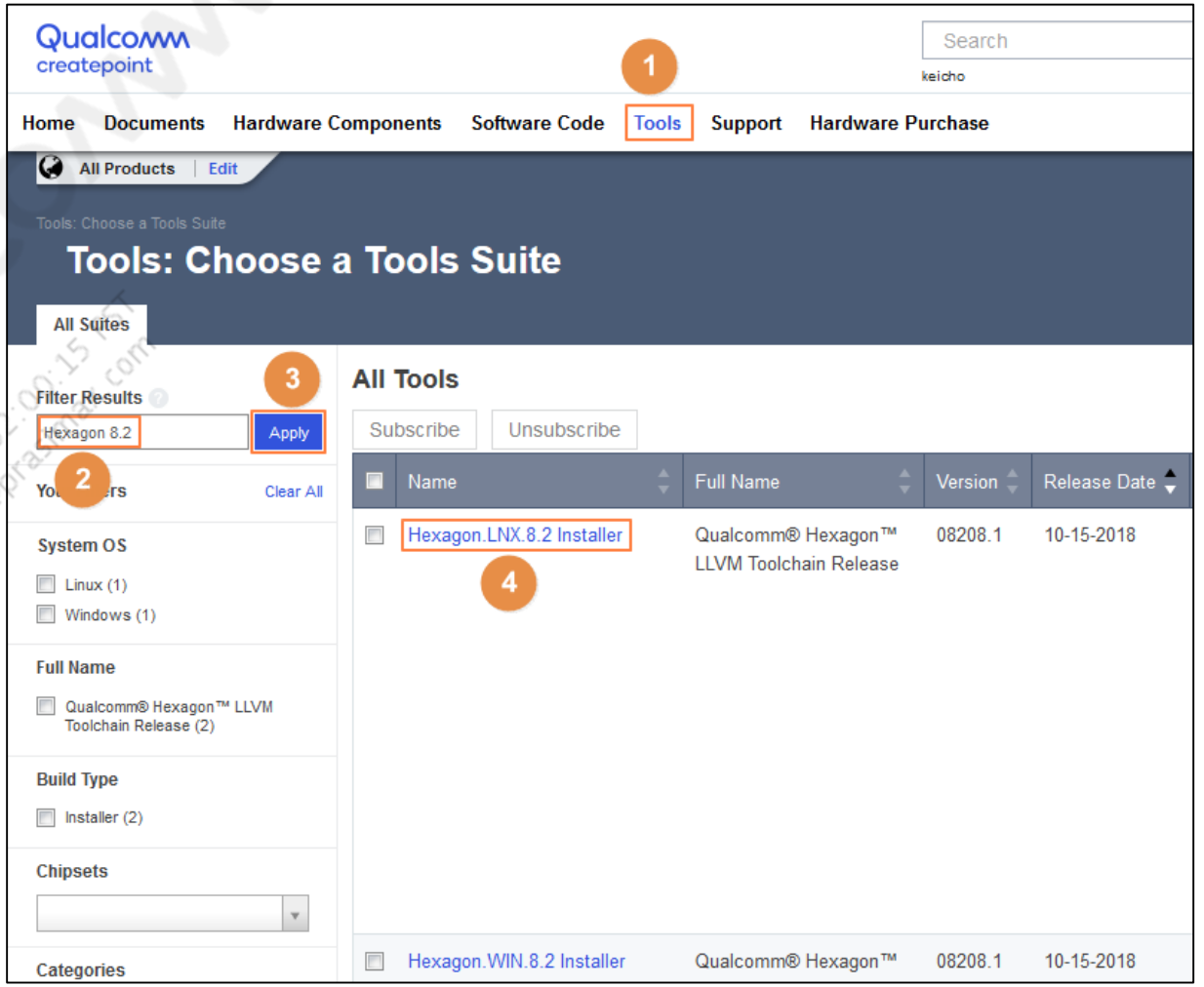
The build commands are used by QTI. The compiler version is Qualcomm® Hexagon™ v8.2.02.

2.5.2 Build and compile CDSP image

The compiler version is Hexagon v8.2.02. The following procedure is used by QTI to build the compute digital signal processor (CDSP) image.

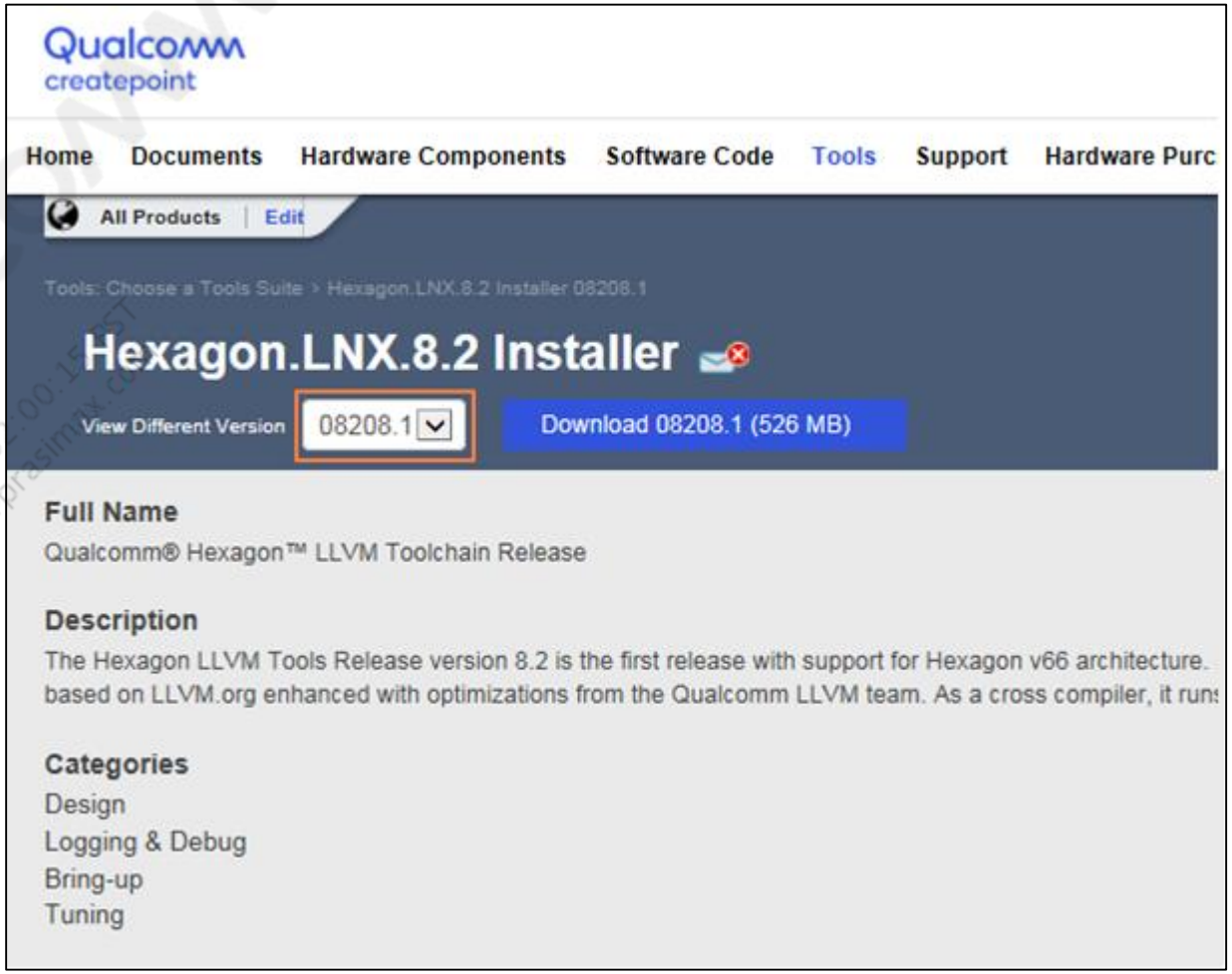
Qualcomm Hexagon LLVM Toolchain – Download

- To download Hexagon Tool,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “Hexagon 8.2” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Click on ‘Hexagon.LNX.8.2 Installer’ in ‘Name’ column



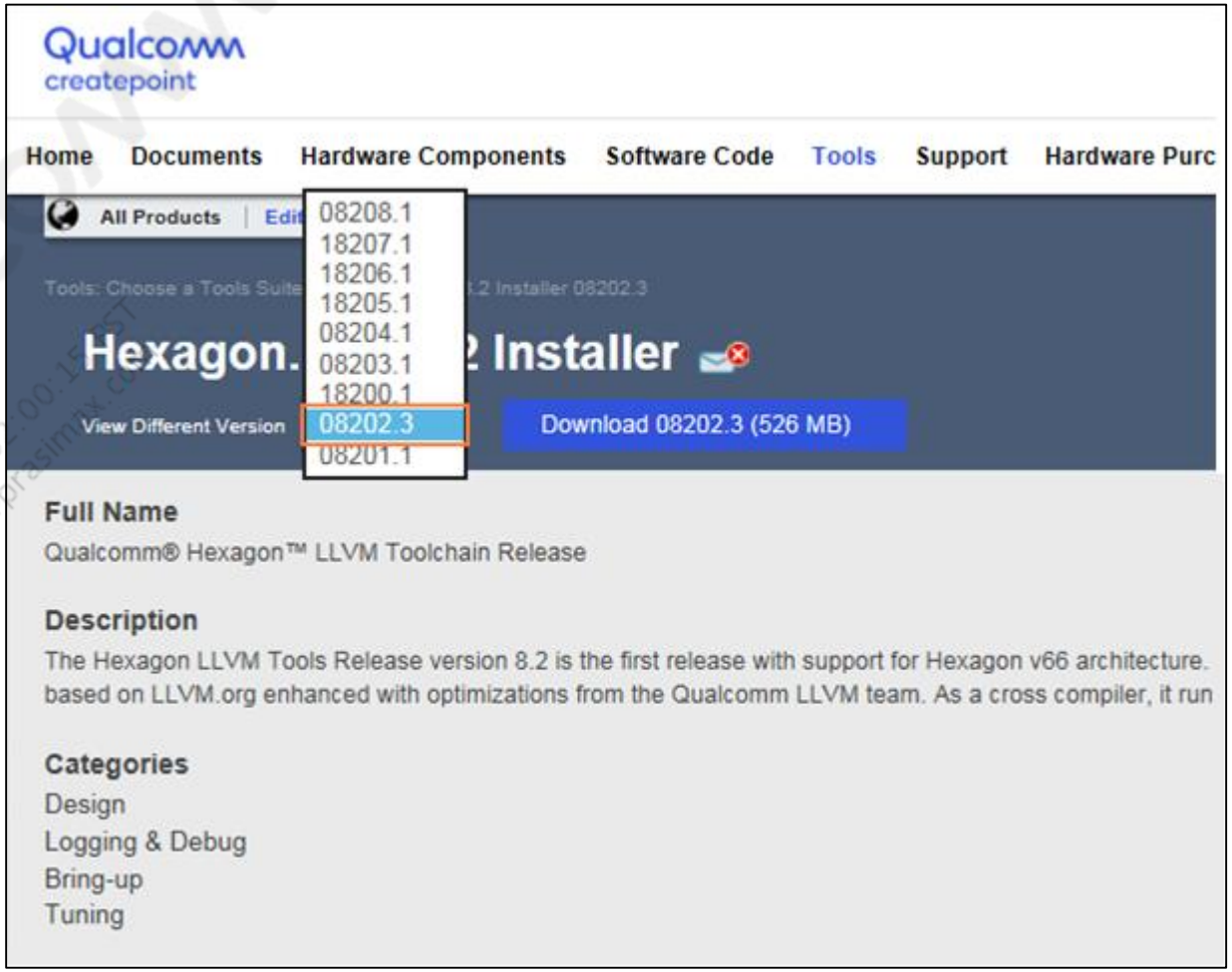
Qualcomm Hexagon LLVM Toolchain – Download

- To download Hexagon Tool,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “Hexagon 8.2” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Click on ‘Hexagon.LNX.8.2 Installer’ in ‘Name’ column
- Click on ‘View Different Version’ combo box for specific version setting



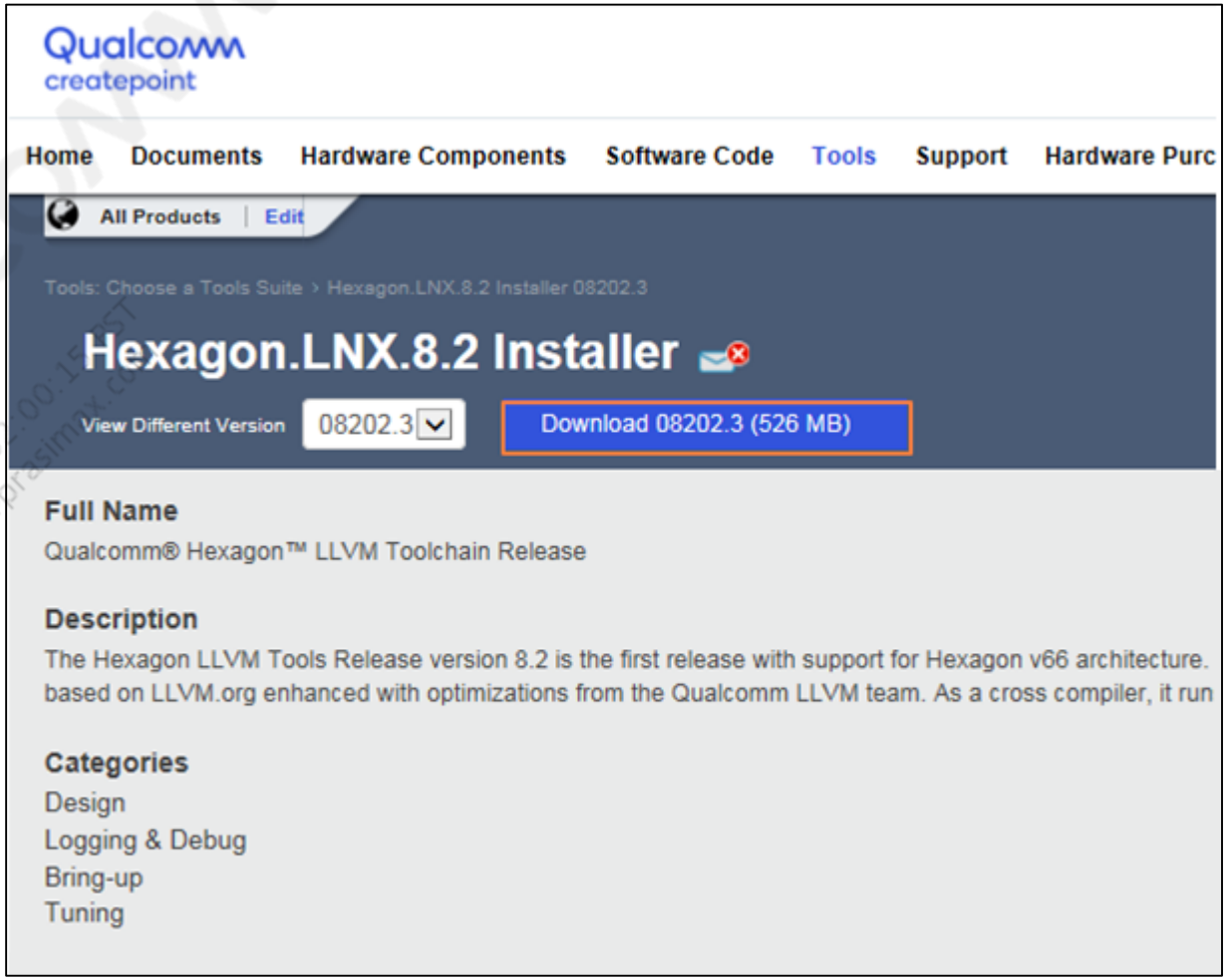
Qualcomm Hexagon LLVM Toolchain – Download

- To download Hexagon Tool,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “Hexagon 8.2” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Click on ‘Hexagon.LNX.8.2 Installer’ in ‘Name’ column
- Click on ‘View Different Version’ combo box for specific version setting
- Select specific version
  - *e.g. ‘08202.3’ for Hexagon Tool 8.2.02 version*



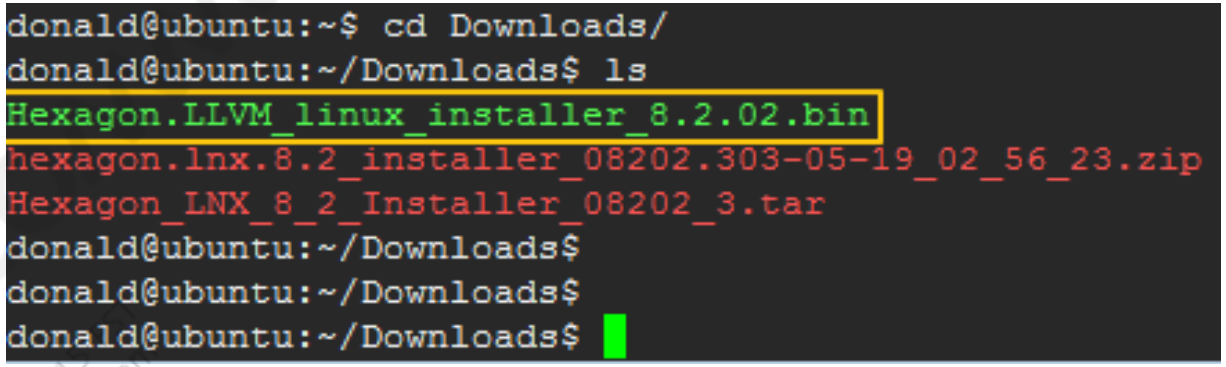
Qualcomm Hexagon LLVM Toolchain – Download

- To download Hexagon Tool,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “Hexagon 8.2” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Click on ‘Hexagon.LNX.8.2 Installer’ in ‘Name’ column
- Click on ‘View Different Version’ combo box for specific version setting
- Select specific version
  - *e.g. ‘08202.3’ for Hexagon Tool 8.2.02 version*
- Click on ‘Download’ button to start downloading
  - A compressed file will be downloaded



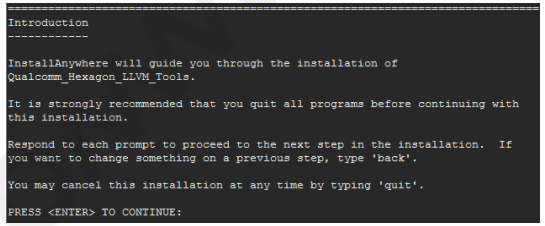
Qualcomm Hexagon LLVM Toolchain – Install

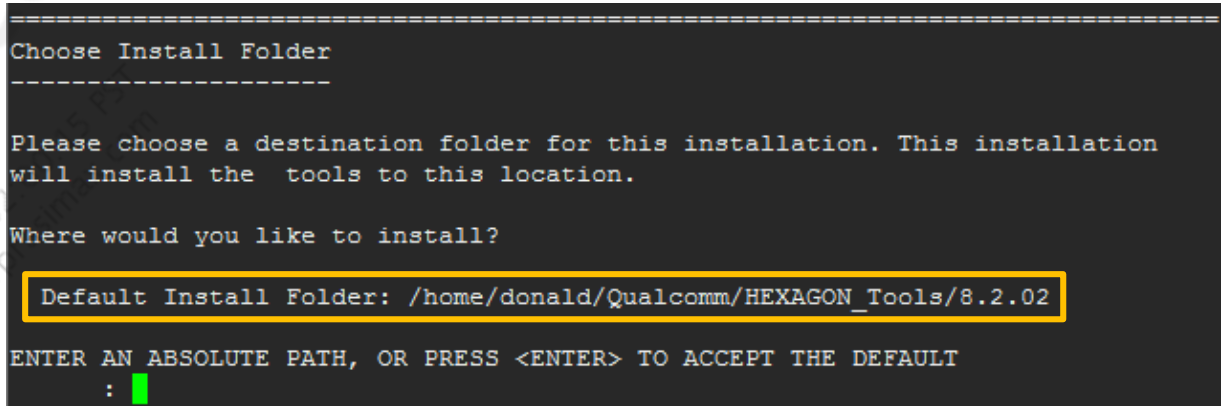
- Uncompress with following commands until .bin file appears
  - $ cd <download_folder>
  - $ unzip hexagon.lnx.8.2_installer_08202.303-05-19_02_56_23.zip
  - $ tar -xvf Hexagon_LNX_8_2_Installer_08202_3.tar

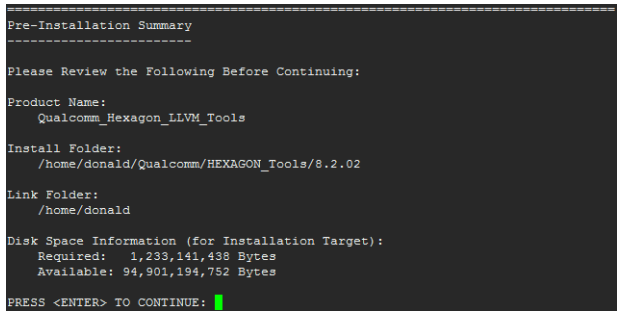


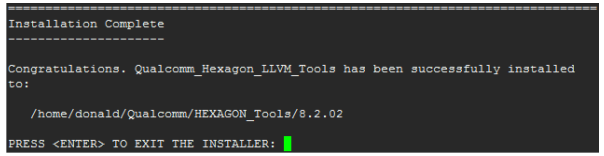
Qualcomm Hexagon LLVM Toolchain – Install

- Uncompress with following commands until .bin file appears
  - $ cd <download_folder>
  - $ unzip hexagon.lnx.8.2_installer_08202.303-05-19_02_56_23.zip
  - $ tar -xvf Hexagon_LNX_8_2_Installer_08202_3.tar
- Start to install with following commands
  - $ chmod +x Hexagon.LLVM_linux_installer_8.2.02.bin
  - $ ./Hexagon.LLVM_linux_installer_8.2.02.bin
- Use recommended settings unless you know what you are doing
- Check the 'Install Folder', the default install folder is current user's home folder
  - /home/donald/Qualcomm/HEXAGON_Tools/8.2.02









Qualcomm Hexagon LLVM Toolchain – Install

- Uncompress with following commands until .bin file appears
  - $ cd <download_folder>
  - $ unzip hexagon.lnx.8.2_installer_08202.303-05-19_02_56_23.zip
  - $ tar -xvf Hexagon_LNX_8_2_Installer_08202_3.tar
- Start to install with following commands
  - $ chmod +x Hexagon.LLVM_linux_installer_8.2.02.bin
  - $ ./Hexagon.LLVM_linux_installer_8.2.02.bin
- Use recommended settings unless you know what you are doing
- Check the 'Install Folder', the default install folder is current user's home folder
  - /home/donald/Qualcomm/HEXAGON_Tools/8.2.02
- Add following environment variables to build ADSP and CDSP
  - $ export HEXAGON_ROOT=~/Qualcomm/HEXAGON_Tools
  - $ export HEXAGON_RTOS_RELEASE=8.2.02
  - $ export HEXAGON_Q6VERSION=v66
  - $ export HEXAGON_IMAGE_ENTRY=0x8AC00000
- HEXAGON_ROOT depends on where the tools are installed
- Hexagon Tool installation DONE !


```
donald@ubuntu:~/Downloads$ export HEXAGON_ROOT=~/Qualcomm/HEXAGON_Tools
donald@ubuntu:~/Downloads$ export HEXAGON_RTOS_RELEASE=8.2.02
donald@ubuntu:~/Downloads$ export HEXAGON_Q6VERSION=v66
donald@ubuntu:~/Downloads$ export HEXAGON_IMAGE_ENTRY=0x8AC00000
donald@ubuntu:~/Downloads$
donald@ubuntu:~/Downloads$ env | grep HEXA
HEXAGON_Q6VERSION=v66
HEXAGON_ROOT=/home/donald/Qualcomm/HEXAGON_Tools
HEXAGON_IMAGE_ENTRY=0x8AC00000
HEXAGON_RTOS_RELEASE=8.2.02
donald@ubuntu:~/Downloads$
```

Section 2.2.3

GCC Linaro Toolchain

GCC Linaro Toolchain

- All the required GCC Linaro Toolchains

SW module	Linaro Toolchain version	Download Site	File Name
Boot Loader	4.8-2014.02	https://releases.linaro.org/archive/14.02/components/toolchain/binaries/	gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
Trust Zone	4.9-2014.07	https://releases.linaro.org/archive/14.07/components/toolchain/binaries/	gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2

GCC Linaro Toolchain – Download

- Download Linaro Toolchain files for v4.8 and v4.9 with following ‘wget’ commands on <Download_Folder>
  - <Download_Folder>$ wget https://releases.linaro.org/archive/14.02/components/toolchain/binaries/gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
  - <Download_Folder>$ wget https://releases.linaro.org/archive/14.07/components/toolchain/binaries/gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2

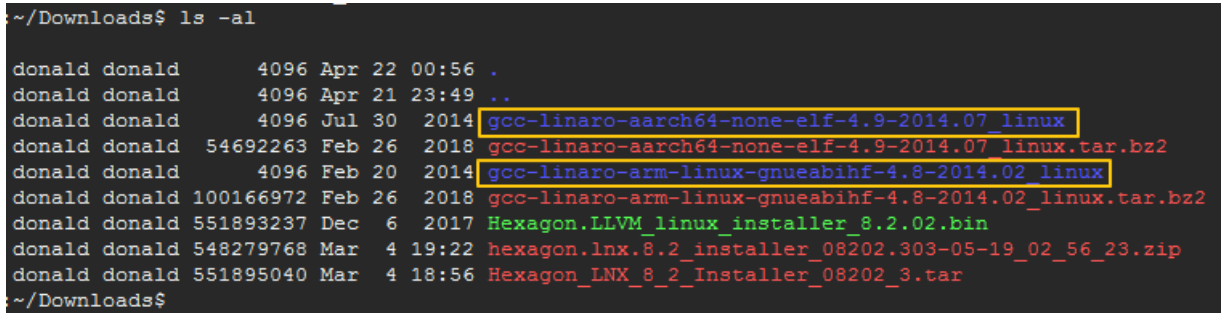
Version	Download Command
4.8	$ wget https://releases.linaro.org/archive/14.02/components/toolchain/binaries/gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
4.9	$ wget https://releases.linaro.org/archive/14.07/components/toolchain/binaries/gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2

```
~/Downloads$ ls -al

donald donald      4096 Apr 22 00:44 .
donald donald      4096 Apr 21 23:49 ..
donald donald  54692263 Feb 26  2018 gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2
donald donald 100166972 Feb 26  2018 gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
donald donald 551893237 Dec  6  2017 Hexagon.LLVM_linux_installer_8.2.02.bin
donald donald 548279768 Mar  4 19:22 hexagon.lnx.8.2_installer_08202.303-05-19_02_56_23.zip
donald donald 551895040 Mar  4 18:56 Hexagon_LNX_8_2_Installer_08202_3.tar
~/Downloads$
```

GCC Linaro Toolchain – Install

- Uncompress the downloaded file for each downloaded file
  - <Download_Folder>$ tar -xvjf gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
  - <Download_Folder>$ tar -xvjf gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2

Version	Install Commands
4.8	$ tar -xvjf gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
4.9	$ tar -xvjf gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2



GCC Linaro Toolchain – Install

- Uncompress the downloaded file for each downloaded file
  - <Download_Folder>$ tar -xvjf gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
  - <Download_Folder>$ tar -xvjf gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2
- Create following folders for each Toolchain
  - $ sudo mkdir -p /prj/llvm-arm/home/common/build_tools
  - $ sudo chmod 755 /prj/llvm-arm/home/common/build_tools
  - $ sudo mkdir -p /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf
  - $ sudo chmod 755 /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf

Version	Install Commands

4.8	$ tar -xvjf gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
	$ sudo mkdir -p /prj/llvm-arm/home/common/build_tools
	$ sudo chmod 755 /prj/llvm-arm/home/common/build_tools

4.9	$ tar -xvjf gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2
	$ sudo mkdir -p /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf
	$ sudo chmod 755 /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf


```
donald@ubuntu:~/Downloads$ ls -al /prj/llvm-arm/home/common/
total 12
drwxr-xr-x 3 root root 4096 Apr 22 01:01 .
drwxr-xr-x 3 root root 4096 Mar  4 20:22 ..
drwxr-xr-x 2 root root 4096 Apr 22 01:01 build_tools
donald@ubuntu:~/Downloads$
donald@ubuntu:~/Downloads$ ls -al /pkg/qct/software/arm/linaro-toolchain/
total 12
drwxr-xr-x 3 root root 4096 Apr 22 01:02 .
drwxr-xr-x 3 root root 4096 Mar  4 20:28 ..
drwxr-xr-x 2 root root 4096 Apr 22 01:02 aarch64-none-elf
donald@ubuntu:~/Downloads$
```

GCC Linaro Toolchain – Install

- Uncompress the downloaded file for each downloaded file
  - <Download_Folder>$ tar -xvjf gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
  - <Download_Folder>$ tar -xvjf gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2
- Create following folders for each Toolchain
  - $ sudo mkdir -p /prj/llvm-arm/home/common/build_tools
  - $ sudo chmod 755 /prj/llvm-arm/home/common/build_tools
  - $ sudo mkdir -p /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf
  - $ sudo chmod 755 /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf
- Move the uncompressed folders to each Toolchain install target folder
- NOTE: For the v4.9, the target folder name should be changed to '4.9-2014.07'
  - <Download_Folder>$ sudo mv gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux /prj/llvm-arm/home/common/build_tools/
  - <Download_Folder>$ sudo mv gcc-linaro-aarch64-none-elf-4.9-2014.07_linux /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf/4.9-2014.07
- Linaro Toolchain installation DONE !

Version	Install Commands

4.8	$ tar -xvjf gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux.tar.bz2
	$ sudo mkdir -p /prj/llvm-arm/home/common/build_tools
	$ sudo chmod 755 /prj/llvm-arm/home/common/build_tools
	$ sudo mv gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux /prj/llvm-arm/home/common/build_tools/

4.9	$ tar -xvjf gcc-linaro-aarch64-none-elf-4.9-2014.07_linux.tar.bz2
	$ sudo mkdir -p /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf
	$ sudo chmod 755 /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf
	$ sudo mv gcc-linaro-aarch64-none-elf-4.9-2014.07_linux /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf/4.9-2014.07

```
donald@ubuntu:~/Downloads$ ls -al /pkg/qct/software/arm/linaro-toolchain/aarch64-none-elf/
total 12
drwxr-xr-x 3 root   root   4096 Apr 22 01:24 .
drwxr-xr-x 3 root   root   4096 Apr 22 01:02 ..
drwxr-xr-x 7 donald donald 4096 Jul 30  2014 4.9-2014.07
donald@ubuntu:~/Downloads$ ls -al /prj/llvm-arm/home/common/build_tools/
total 12
drwxr-xr-x 3 root   root   4096 Apr 22 01:23 .
drwxr-xr-x 3 root   root   4096 Apr 22 01:01 ..
drwxr-xr-x 7 donald donald 4096 Feb 20  2014 gcc-linaro-arm-linux-gnueabihf-4.8-2014.02_linux
donald@ubuntu:~/Downloads$
```

Section 3

Installing Tools for Windows

Installing Tools for Windows

- In this chapter, only basic required tools will be explained for Windows
  - Qualcomm Product Support Tool (QPST)
  - Qualcomm Smart Audio Platform Configuration Tool
  - Android SDK Tools (fastboot, adb, and so on)

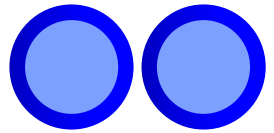
- For more details, refer to *[80-YB405-129] QCS40x Software User Guide* document

Installing Tools for Windows

- Here are required tools information

Tool / Software	Version	Source / Vendor	Main function
Qualcomm Product Support Tool (QPST)	✓ 2.7.477 or later	QTI	✓ OS: Windows ✓ Download firmware images ✓ QFIL
Qualcomm SAP Configuration Tool	✓ 1.0.4 or later	QTI	✓ OS: Windows, Linux ✓ Configure numerous parameters for Linux Smart Speaker
Android SDK Tools	✓ r10 or higher ✓ ADB 1.0.39 or later	Android open-source project	✓ OS: Windows, Linux ✓ Host USB driver ✓ adb ✓ Fastboot

- For more details for QPST and QFIL, refer to:
  - *[80-V1400-3] QUALCOMM PRODUCT SUPPORT TOOL (QPST) 2.7 USER GUIDE*
  - *[80-NN120-1] QUALCOMM FLASH IMAGE LOADER (QFIL) USER GUIDE*

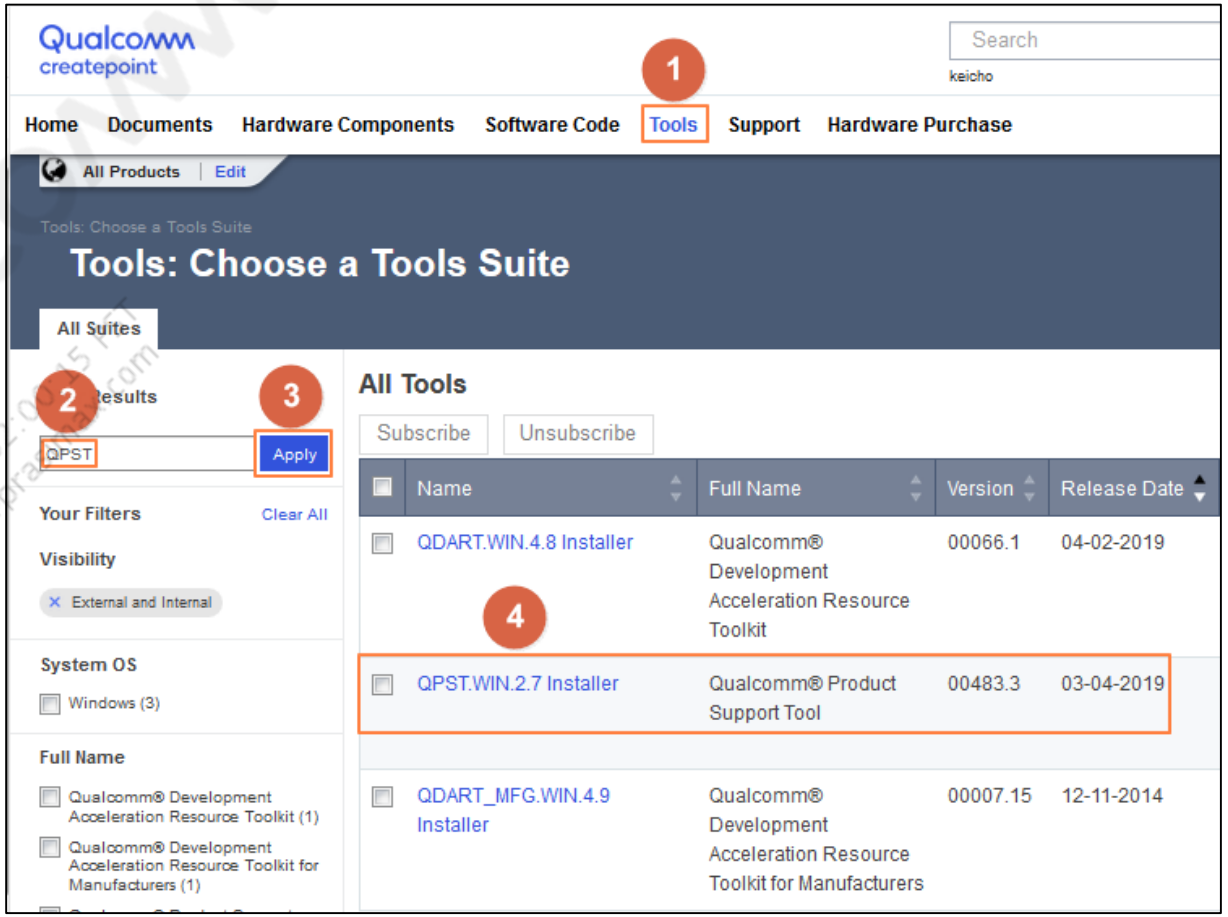


Section 3.1

Installing QPST

Installing QPST

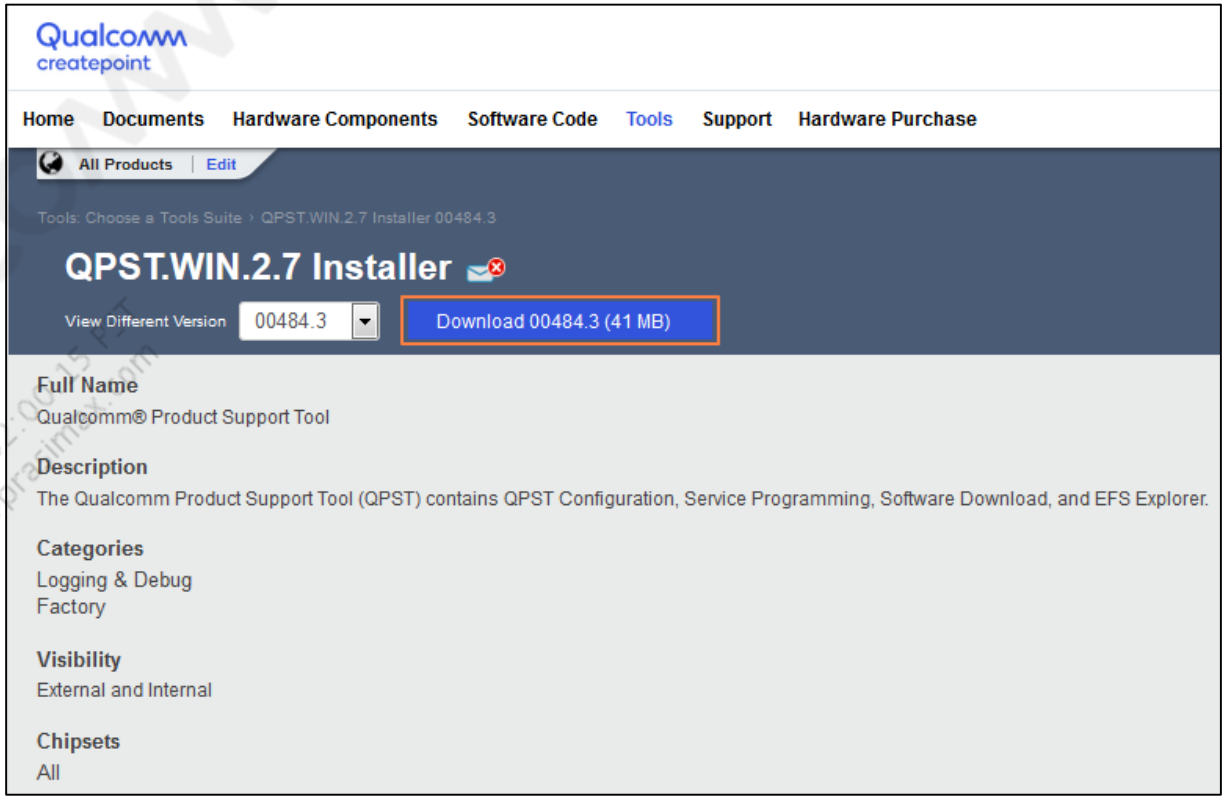
- To download QPST,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on 'Tools' tab
- (2) Input "QPST" into 'Filter Results'
- (3) Click on 'Apply' button
- (4) Click on 'QPST.WIN.2.7 Installer' item



Installing QPST

- To download QPST,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “QPST” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Click on ‘QPST.WIN.2.7 Installer’ item
- Click on ‘Download’ button to start downloading
  - A compressed file will be downloaded
- Unzip and run setup file (QPST.<version>.exe) to start installation and follow the guidelines for a successful installation

- DONE!

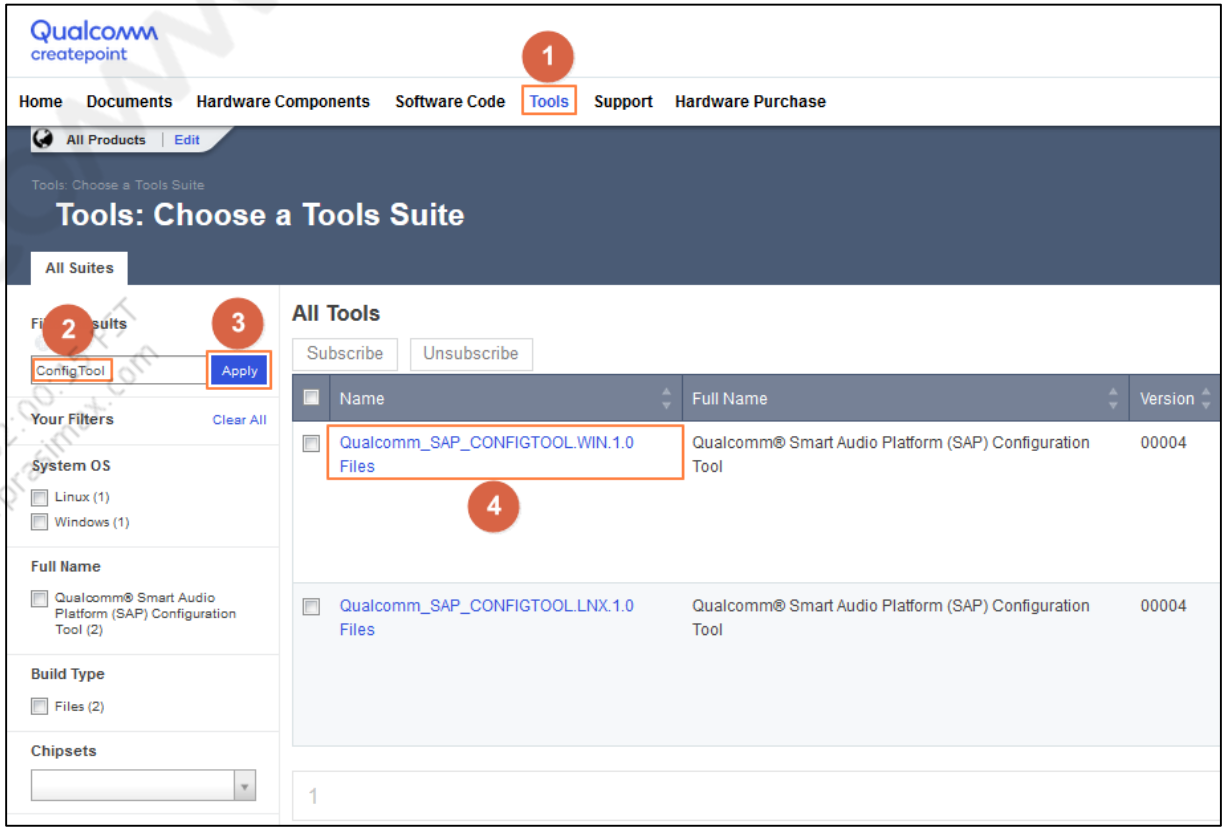


Section 3.2

Installing Qualcomm Smart Audio Platform Configuration Tool

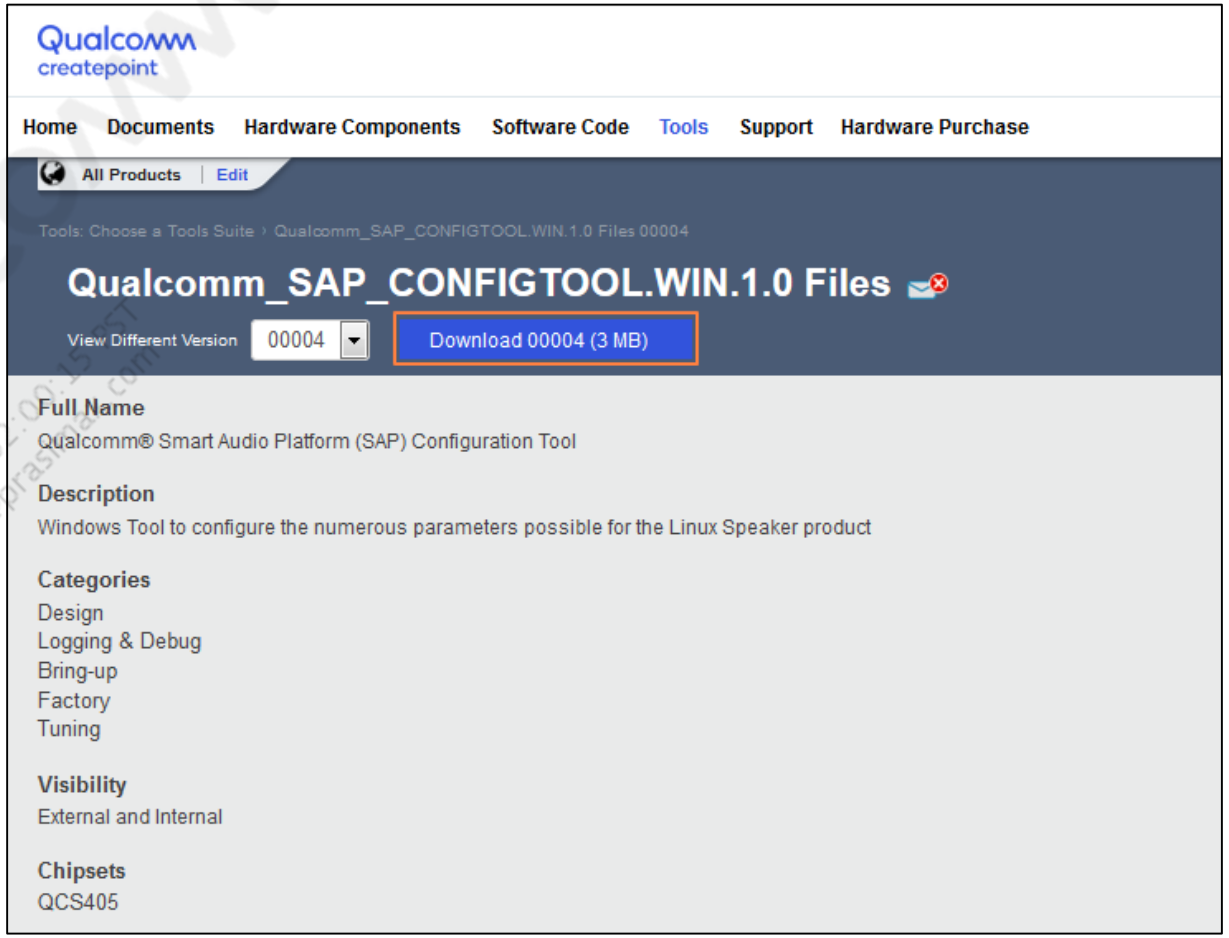
Installing Qualcomm Smart Audio Platform Configuration Tool

- To download SAP Configuration Tool,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on 'Tools' tab
- (2) Input "ConfigTool" into 'Filter Results'
- (3) Click on 'Apply' button
- (4) Click on 'Qualcomm_SAP_CONFIGTOOL.WIN.1.0 Files' item

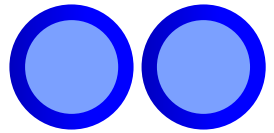


Installing Qualcomm Smart Audio Platform Configuration Tool

- To download SAP Configuration Tool,
  - Go : https://createpoint.qti.qualcomm.com/dashboard/
- (1) Click on ‘Tools’ tab
- (2) Input “ConfigTool” into ‘Filter Results’
- (3) Click on ‘Apply’ button
- (4) Click on ‘Qualcomm_SAP_CONFIGTOOL.WIN.1.0 Files’ item
- Click on ‘Download’ button to start downloading
  - A compressed file will be downloaded
- Unzip and run setup file (QualcommDeviceConfigTool-<version>.exe) to start installation and follow the guidelines for a successful installation
- DONE!



- NOTE: There is also SAP Configuration Tool for Linux



Section 3.3

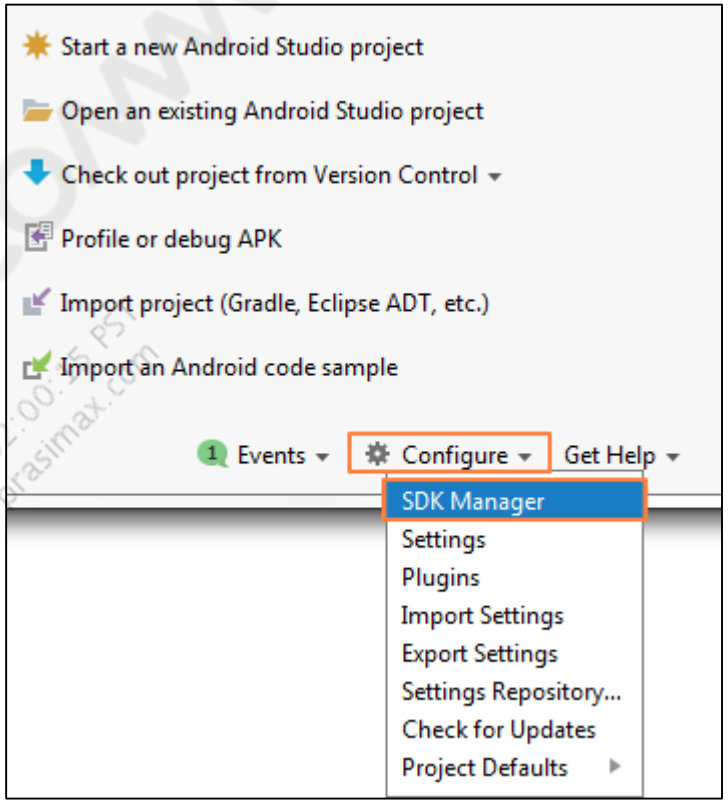
Installing Android SDK Tools for Windows

Installing Android SDK Tools for Windows

- Android Studio installation is recommended to use Android SDK Tools
- To download Android Studio, refer to:
  - https://developer.android.com/studio
  - https://developer.android.com/studio/install

Installing Android SDK Tools for Windows

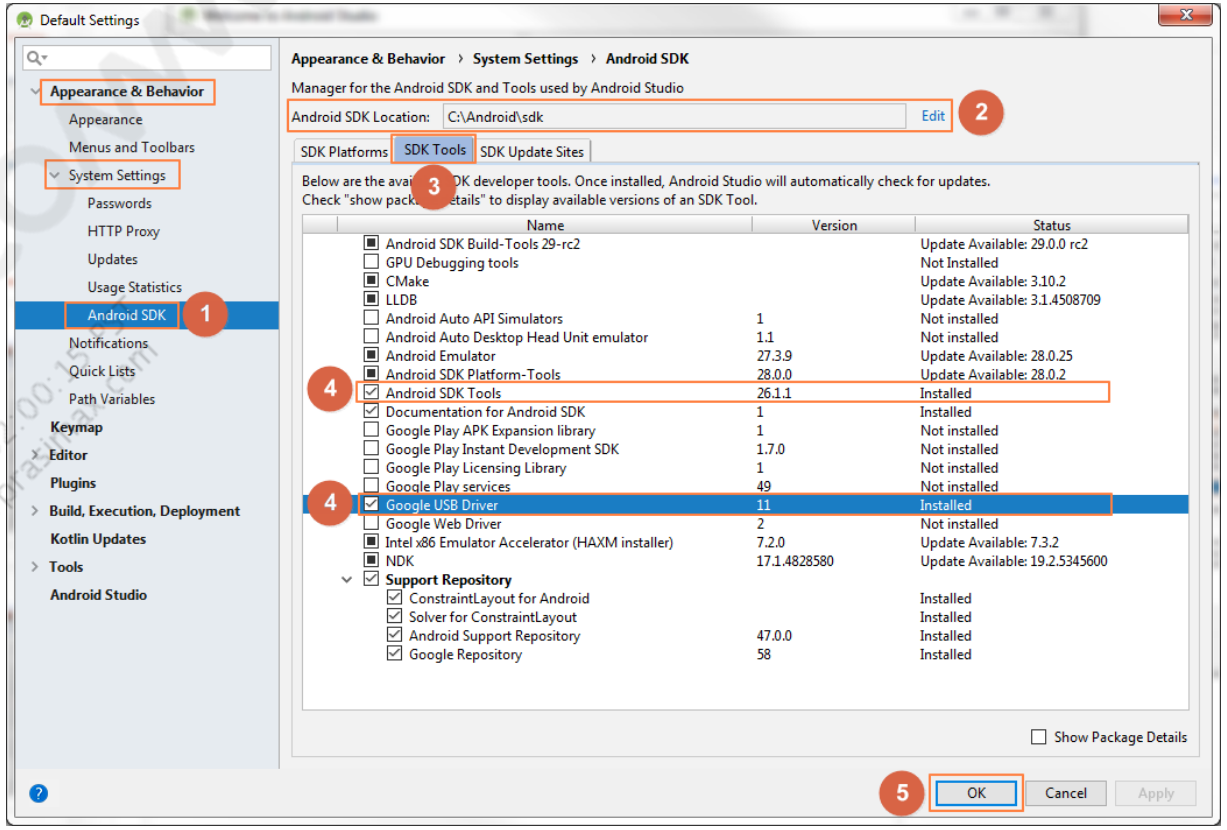
- When Android Studio is launched, ‘Welcome to Android Studio’ window appears
- Click on ‘Configure’ > ‘SDK Manager’ menu



Installing Android SDK Tools for Windows

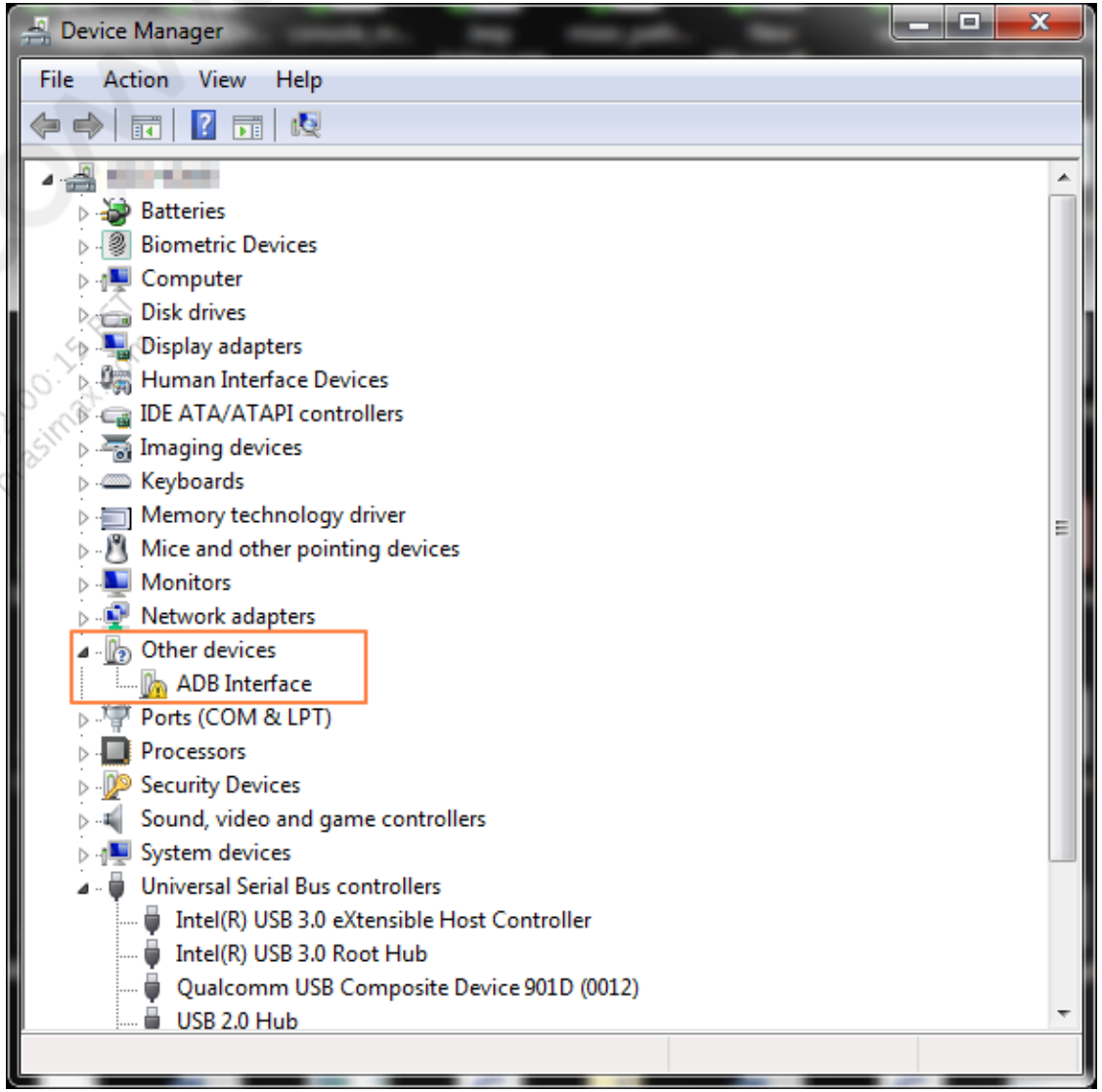
- When Android Studio is launched, 'Welcome to Android Studio' window appears
- Click on 'Configure' > 'SDK Manager' menu
- (1) On the left tree control, Select 'Appearance & Behavior > System Settings > Android SDK'
- (2) Select 'Android SDK Location' with Edit button
- (3) Click on 'SDK Tools' tab
- (4) Check on 'Android SDK Tools' and 'Google USB Driver' to install adb, fastboot and usb driver
- (5) Click on 'OK' button to install the required tools

- DONE !



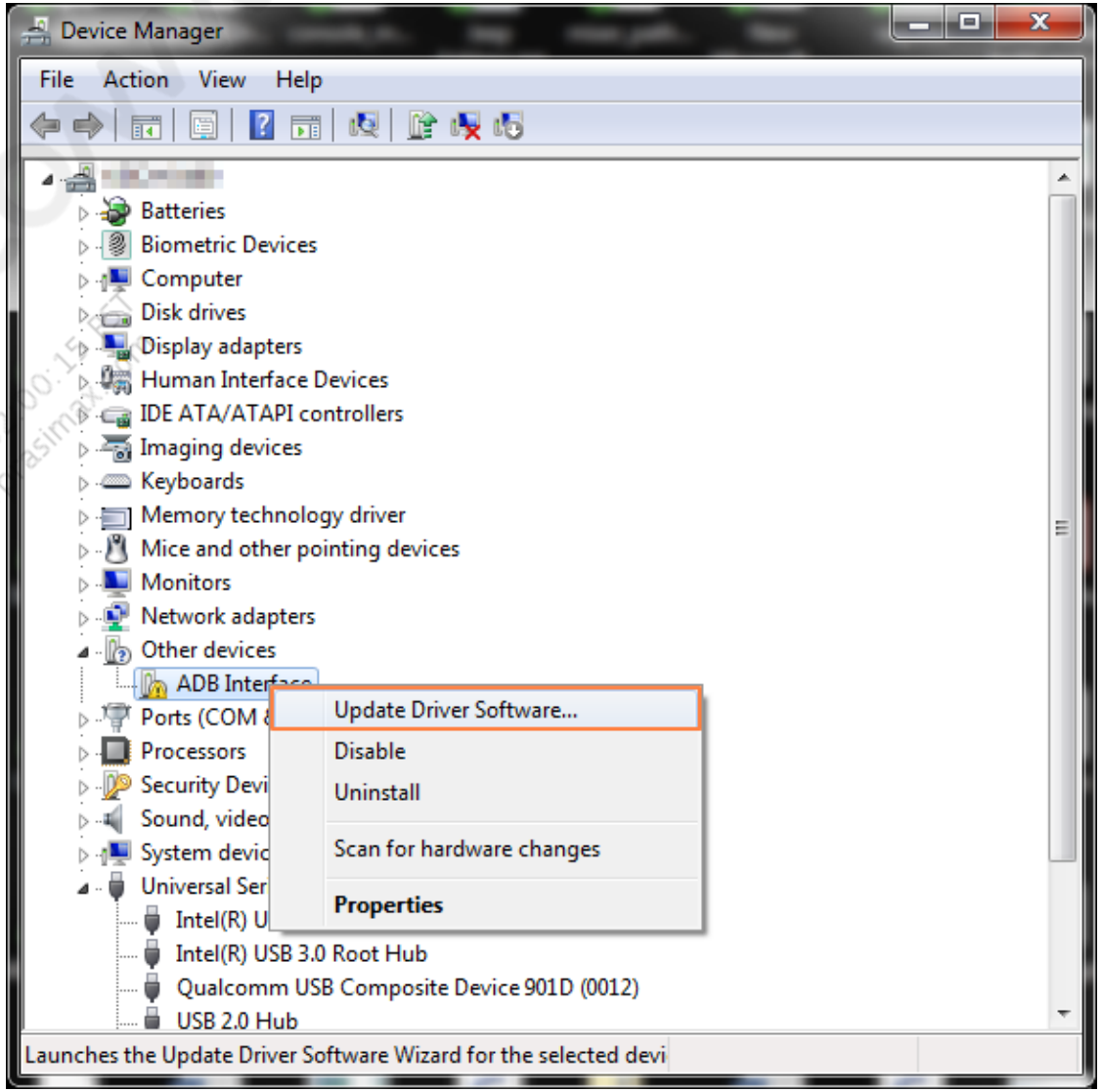
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows ‘Device Manager’ to install USB Driver for adb interface



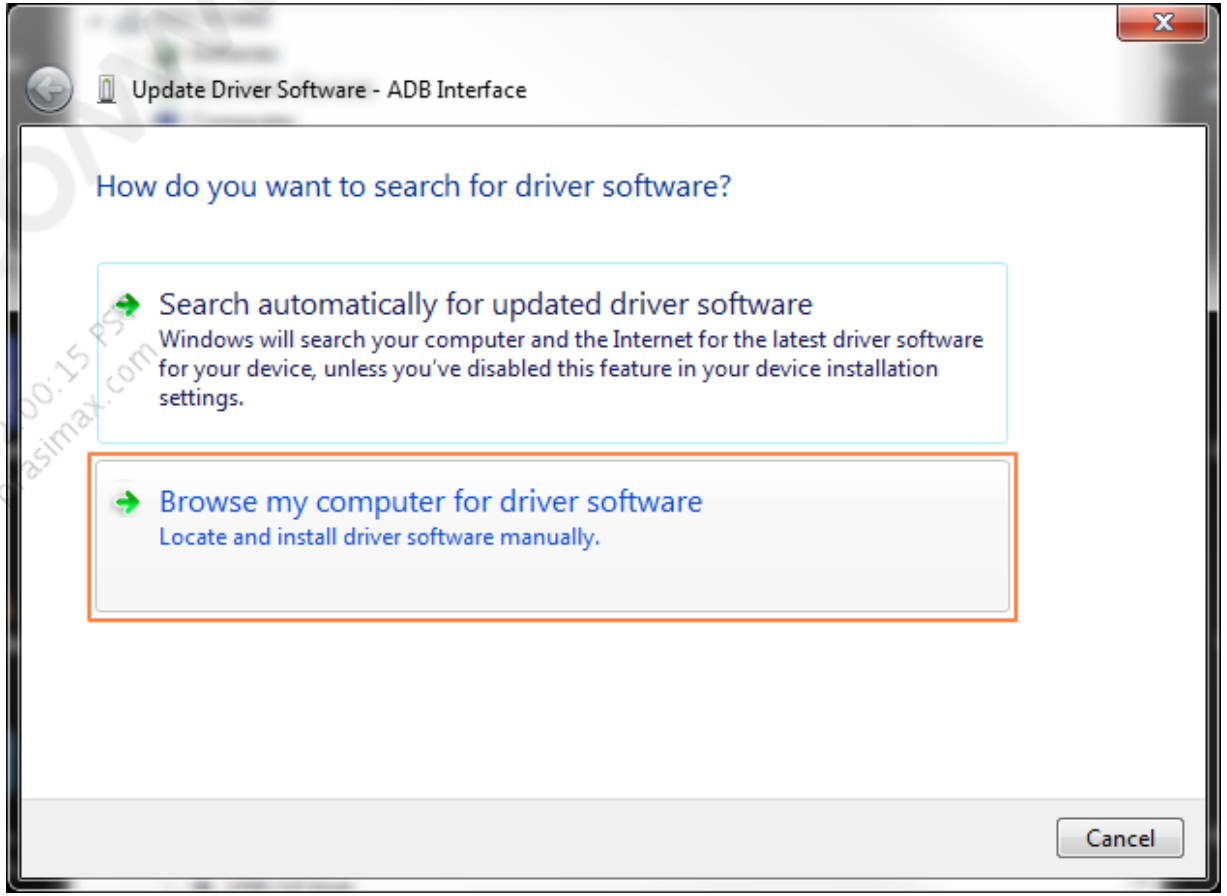
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'



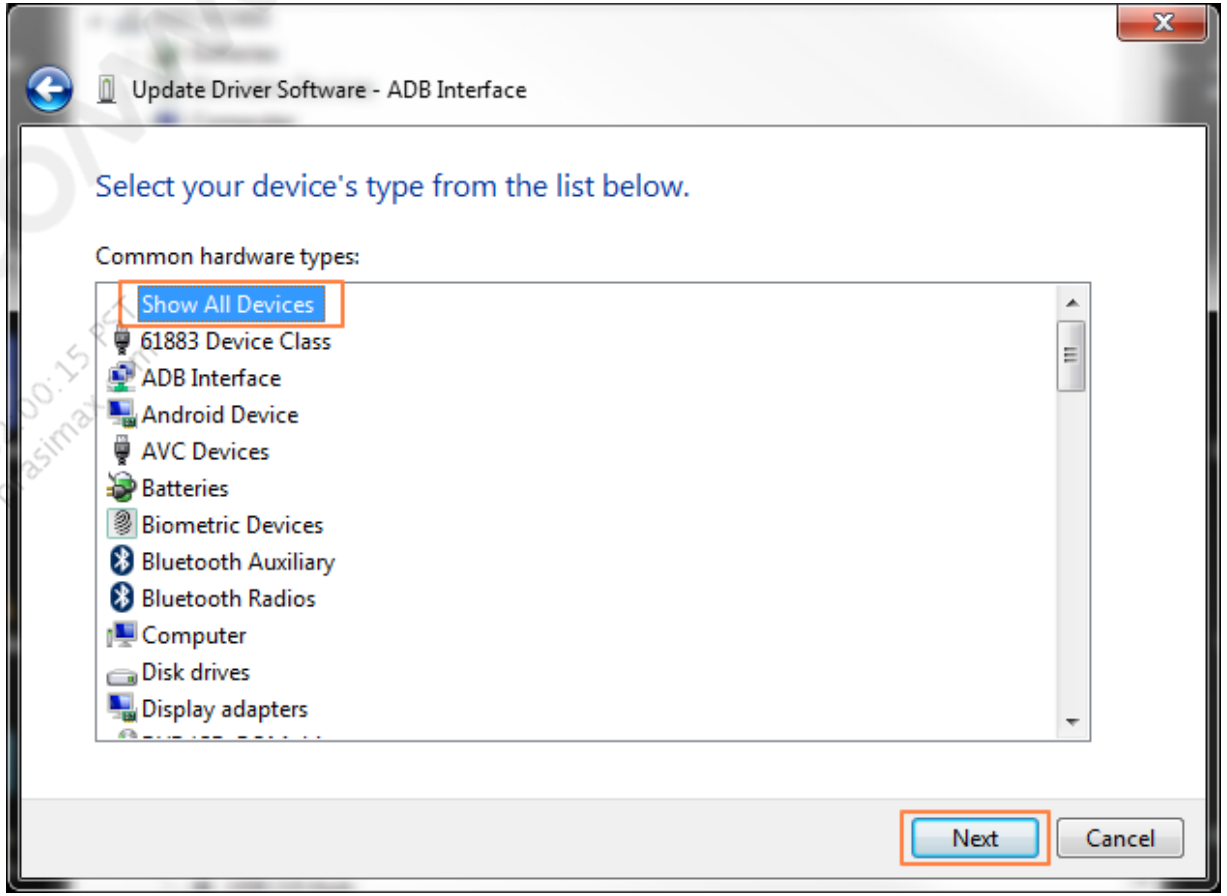
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'



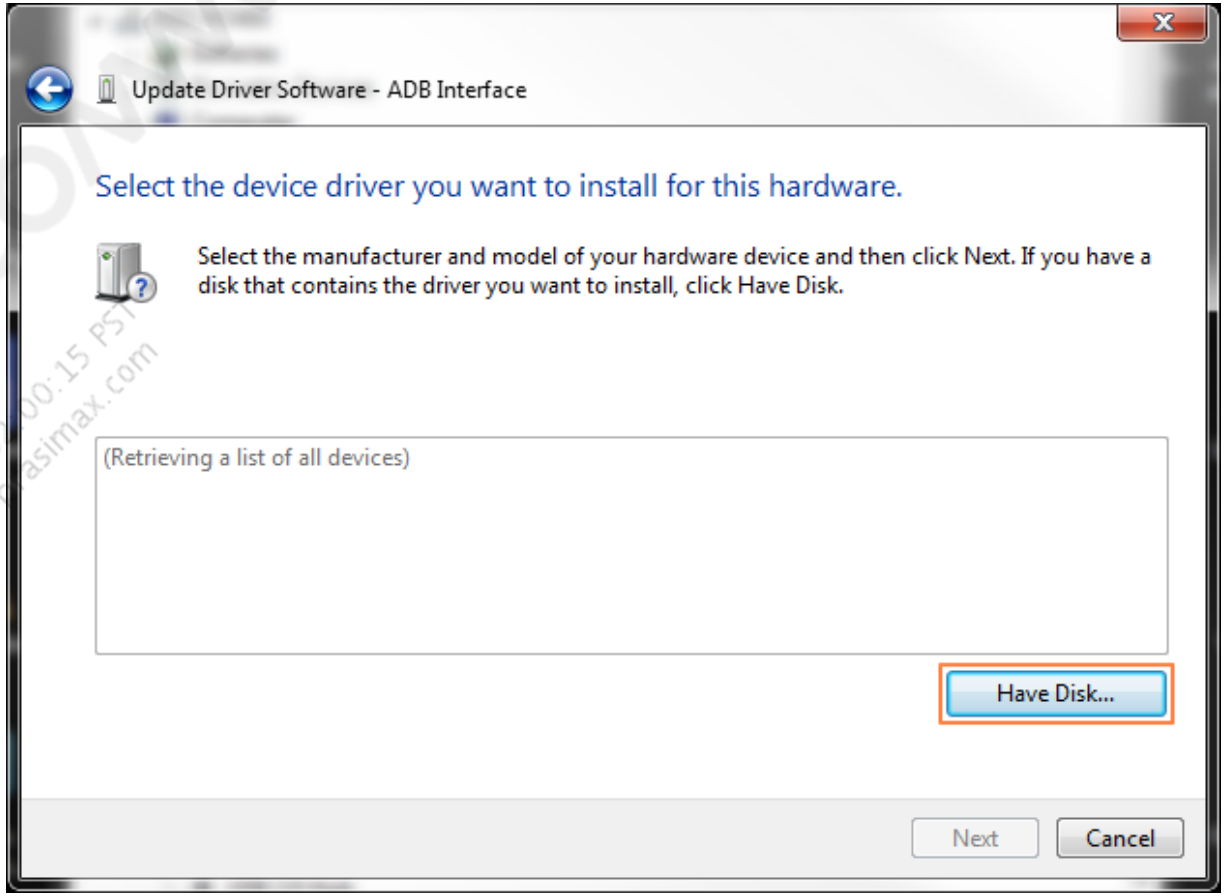
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'
- Select 'Show All Devices' and Click on 'Next' button



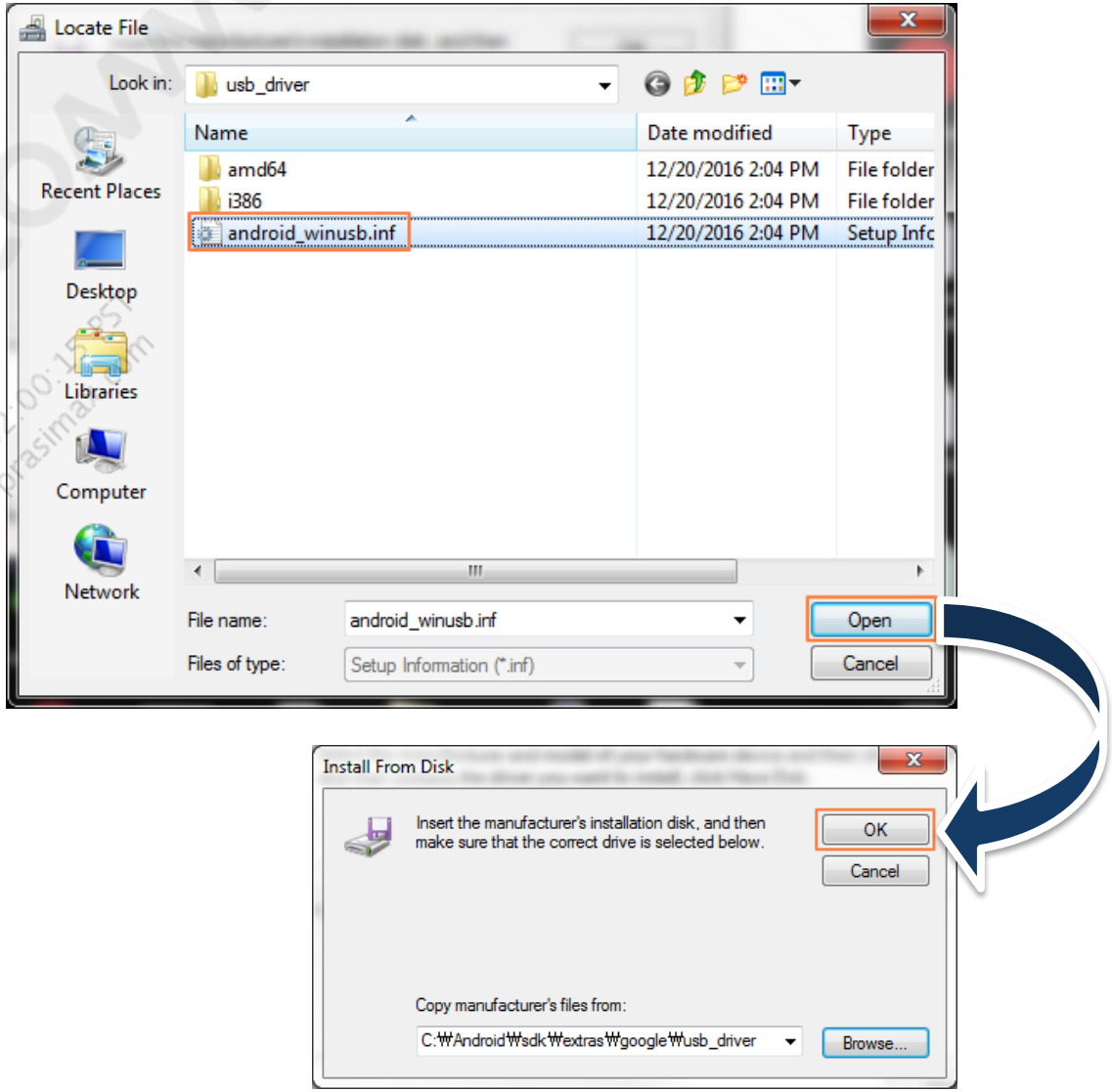
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'
- Select 'Show All Devices' and Click on 'Next' button
- Click on 'Have Disk…' button



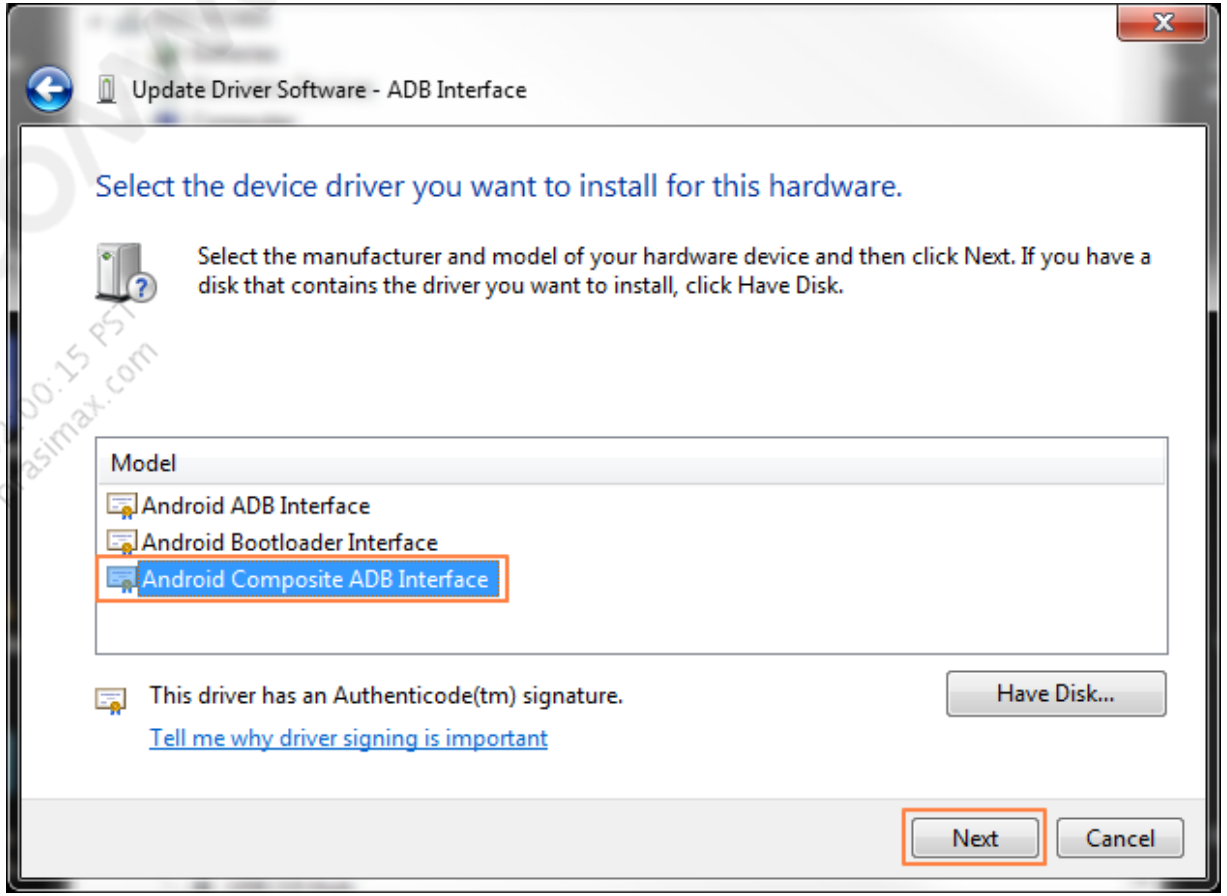
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'
- Select 'Show All Devices' and Click on 'Next' button
- Click on 'Have Disk…' button
- Select 'android_winusb.inf' file on '<Android SDK Location>\extras\google\usb_driver\' directory and then click on 'Open' button
- Click on 'OK' button



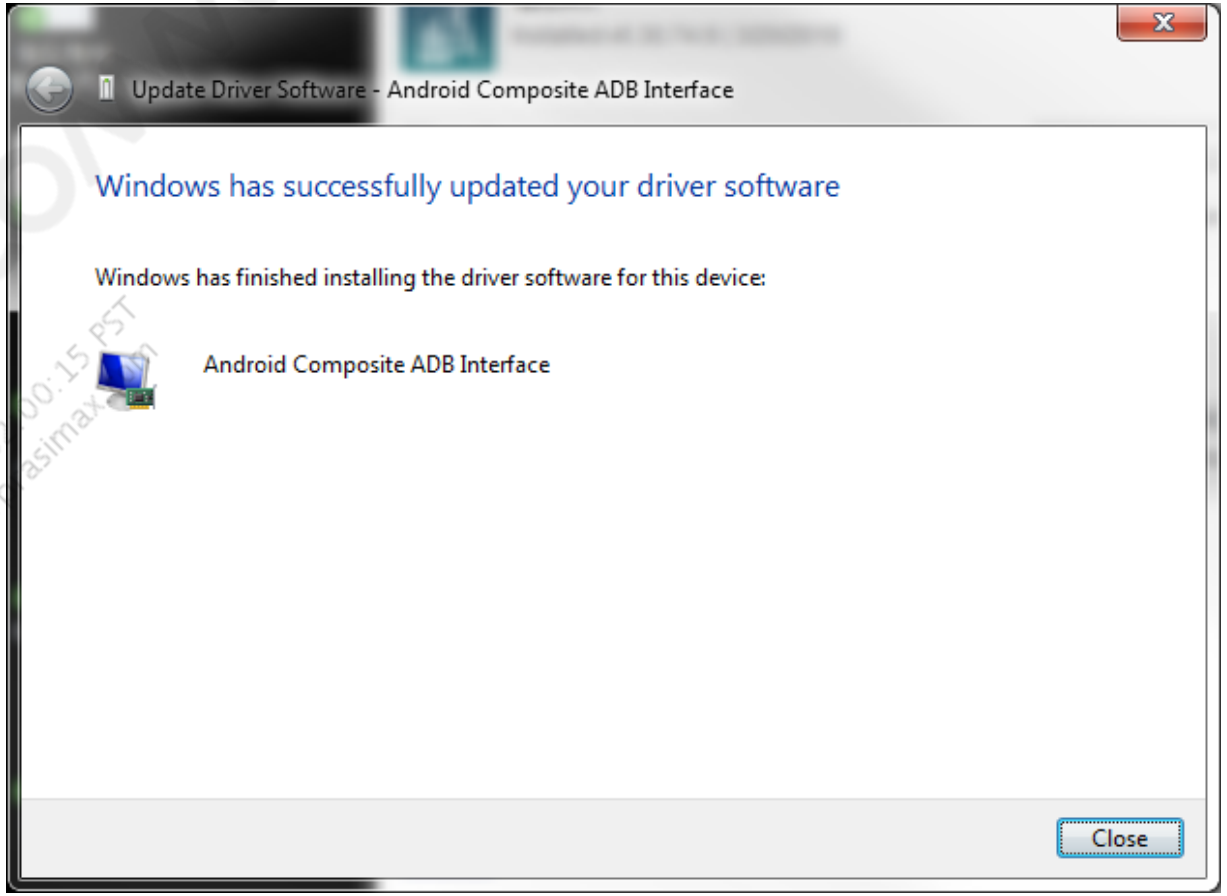
Installing Android SDK Tools for Windows – Install USB Driver (adb)

- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'
- Select 'Show All Devices' and Click on 'Next' button
- Click on 'Have Disk…' button
- Select 'android_winusb.inf' file on '<Android SDK Location>\extras\google\usb_driver\' directory and then click on 'Open' button
- Click on 'OK' button
- Select 'Android Composite ADB Interface' and Click on 'Next'



Installing Android SDK Tools for Windows – Install USB Driver (adb)

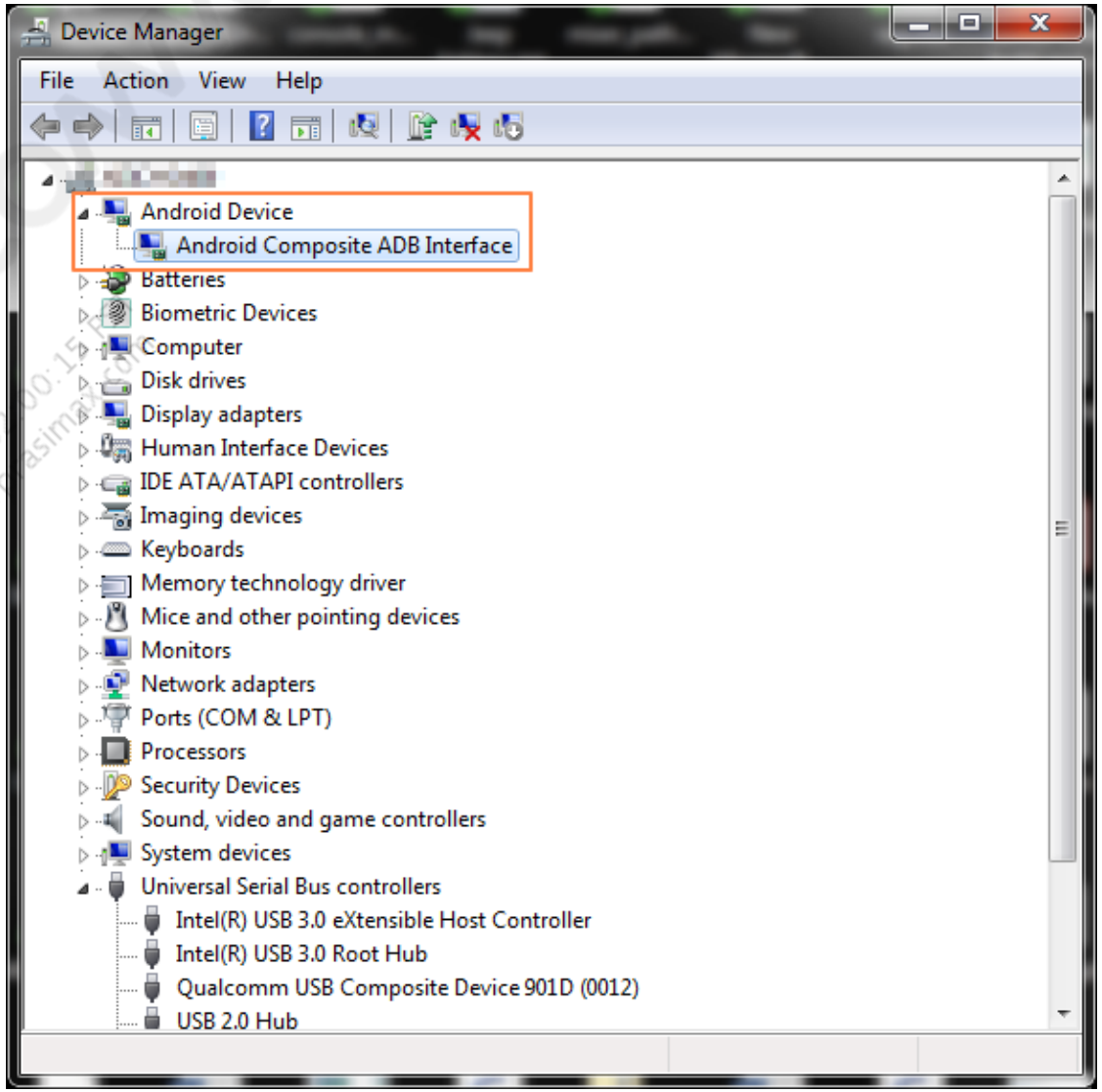
- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'
- Select 'Show All Devices' and Click on 'Next' button
- Click on 'Have Disk…' button
- Select 'android_winusb.inf' file on '<Android SDK Location>\extras\google\usb_driver\' directory and then click on 'Open' button
- Click on 'OK' button
- Select 'Android Composite ADB Interface' and Click on 'Next'
- Make sure the installation success message



Installing Android SDK Tools for Windows – Install USB Driver (adb)

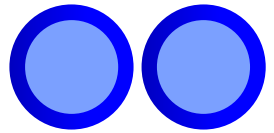
- After the device in plugged in, open Windows 'Device Manager' to install USB Driver for adb interface
- Right-click on 'ADB Interface' and select 'Update Driver Software…'
- Click on 'Browse my computer for driver software'
- Select 'Show All Devices' and Click on 'Next' button
- Click on 'Have Disk…' button
- Select 'android_winusb.inf' file on '<Android SDK Location>\extras\google\usb_driver\' directory and then click on 'Open' button
- Click on 'OK' button
- Select 'Android Composite ADB Interface' and Click on 'Next'
- Make sure the installation success message
- Make sure the ADB Interface on 'Device Manager' window

- DONE !



Section 4

Obtain Software



Section 4.1

Software Components

Software Components

- QCS40x software consists of following categories
  - HLOS software
  - non-HLOS software

	Category	SW Modules
QCS40x software	HLOS	
	non-HLOS	

Software Components

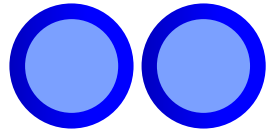
- QCS40x software consists of following categories
  - HLOS software
  - non-HLOS software
- HLOS software includes the following SW module
  - APPS_PROC

	Category	SW Modules
QCS40x software	HLOS	APPS_PROC
	non-HLOS	

Software Components

- QCS40x software consists of following categories
  - HLOS software
  - non-HLOS software
- HLOS software includes the following SW module
  - APPS_PROC
- non-HLOS software includes the following SW modules
  - COMMON : Build environment to generate final images
  - BTFM_PROC : Bluetooth / FM SoC firmware and NVM files
  - ADSP_PROC : Audio DSP
  - CDSP_PROC : Computational DSP
  - RPM_PROC : Resource Power Management
  - WLAN_PROC : Wi-Fi firmware image and board profile
  - BOOT_IMAGES : Boot loader
  - TRUSTZONE_IMAGES : For more details, refer to ‘Qualcomm® Secure Execution Environment (QSEE)’.

	Category	SW Modules
QCS40x software	HLOS	APPS_PROC
	non-HLOS	COMMON
		BTFM_PROC
		ADSP_PROC
		CDSP_PROC
		RPM_PROC
		WLAN_PROC
		BOOT_IMAGES
		TRUSTZONE_IMAGES

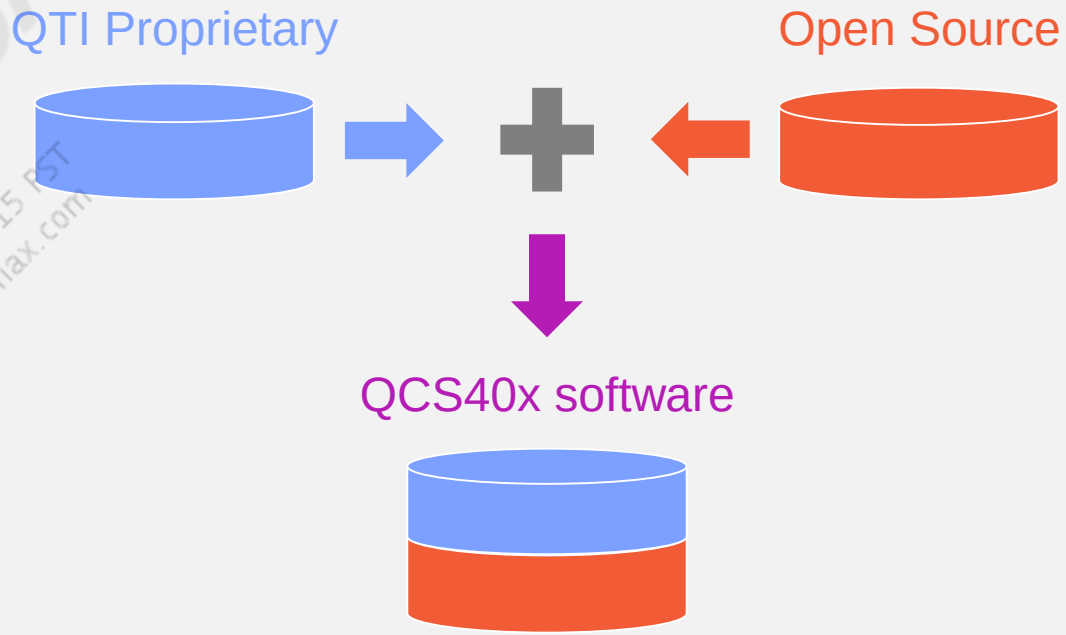


Section 4.2

Release Packages

Release Packages

- QCS40x software release consists of the following two packages
  - QTI Proprietary Software
    - Qualcomm ChipCode™ portal
    - https://chipcode.qti.qualcomm.com
    - Download with WEB browser
  - Open Source Software
    - Code Aurora Foundation (CAF)
    - https://www.codeaurora.org
    - Sync to source server with ‘repo’ command



Release Packages

- QCS40x software release consists of the following two packages
  - QTI Proprietary Software
    - ChipCode portal
    - https://chipcode.qti.qualcomm.com
    - Download with WEB browser
  - Open Source Software
    - Code Aurora Foundation (CAF)
    - https://www.codeaurora.org
    - Sync to source server with ‘repo’ command
- Detail components for each release package

Release Package	Category	Components
ChipCode	HLOS	❑ Proprietary source and firmware images for the applications processor HLOS
	non-HLOS	❑ Proprietary source and firmware images for all the non applications processors (non-apps) ❑ An umbrella package built from a combined set of integrated individual component releases
CAF	HLOS	❑ Open source for the applications processor HLOS
	non-HLOS	-

Release Packages

- QCS40x software release consists of the following two packages
  - QTI Proprietary Software
    - ChipCode portal
    - https://chipcode.qti.qualcomm.com
    - Download with WEB browser
  - Open Source Software
    - Code Aurora Foundation (CAF)
    - https://www.codeaurora.org
    - Sync to source server with ‘repo’ command
- Detail components for each release package
- Release package for each SW Module

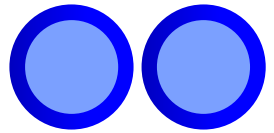
	Category	SW Modules	Release Package
QCS40x software	HLOS	APPS_PROC	ChipCode
			CAF
	non-HLOS	COMMON	ChipCode
		BTFM_PROC	
		ADSP_PROC	
		CDSP_PROC	
		RPM_PROC	
		WLAN_PROC	
		BOOT_IMAGES	
		TRUSTZONE_IMAGES	

Release Packages

- There are three types of packages as below
  - ap_standard_oem: for system software developer
  - hlos_dev: for HLOS software developer
  - test_device: for tester (demo, board bring-up, factory test, etc)

- Each package consists as shown in the table right
  - Source: Build is required
  - Binary: Build is NOT required

SW Modules	ap_standard_oem	hlos_dev	test_device
APPS_PROC	Source	Source	Source & Library
COMMON	Source	Source	Source
BTFM_PROC	Binary	Binary	Binary
ADSP_PROC	Source	Binary	Binary
CDSP_PROC	Source	Binary	Binary
RPM_PROC	Source	Binary	Binary
WLAN_PROC	Binary	Binary	Binary
BOOT_IMAGES	Source	Binary	Binary
TRUSTZONE_IMAGES	Source	Binary	Binary



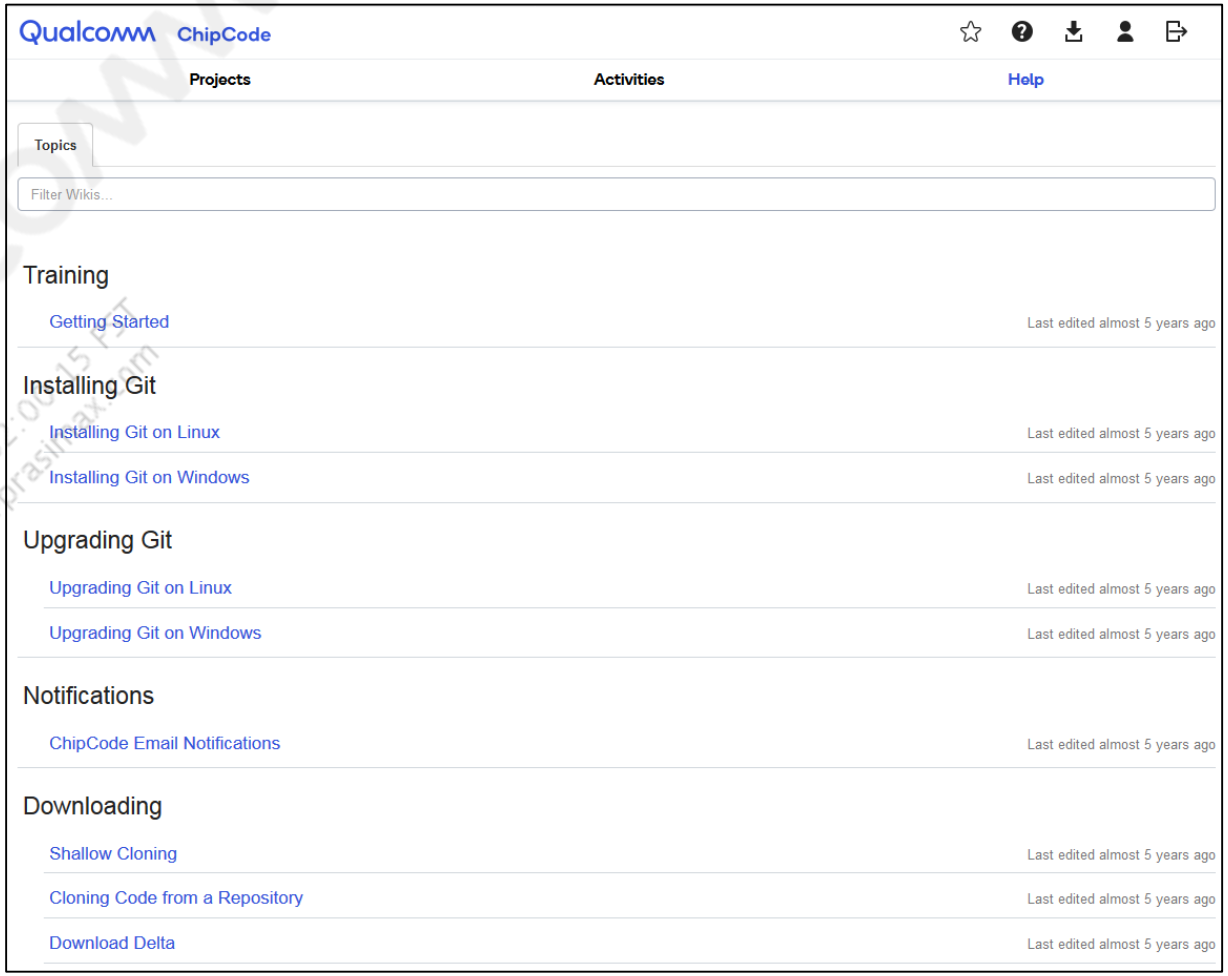
Section 4.3

QTI Proprietary Software

QTI Proprietary Software

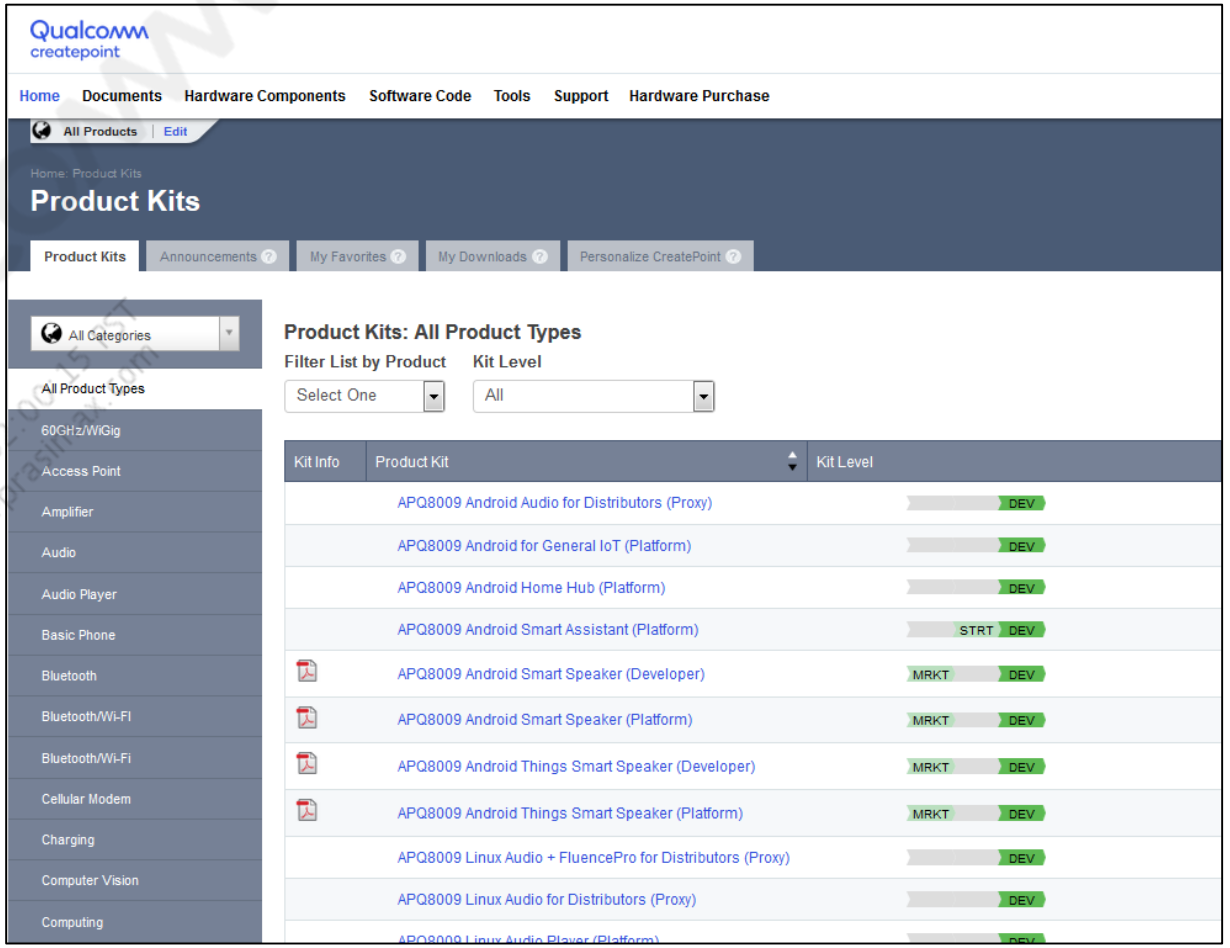
- If user wants to access ChipCode for the first time,
  - Refer to https://chipcode.qti.qualcomm.com/helpki

- NOTE: This document will explain how to obtain software based on getting r1.0_00007.0



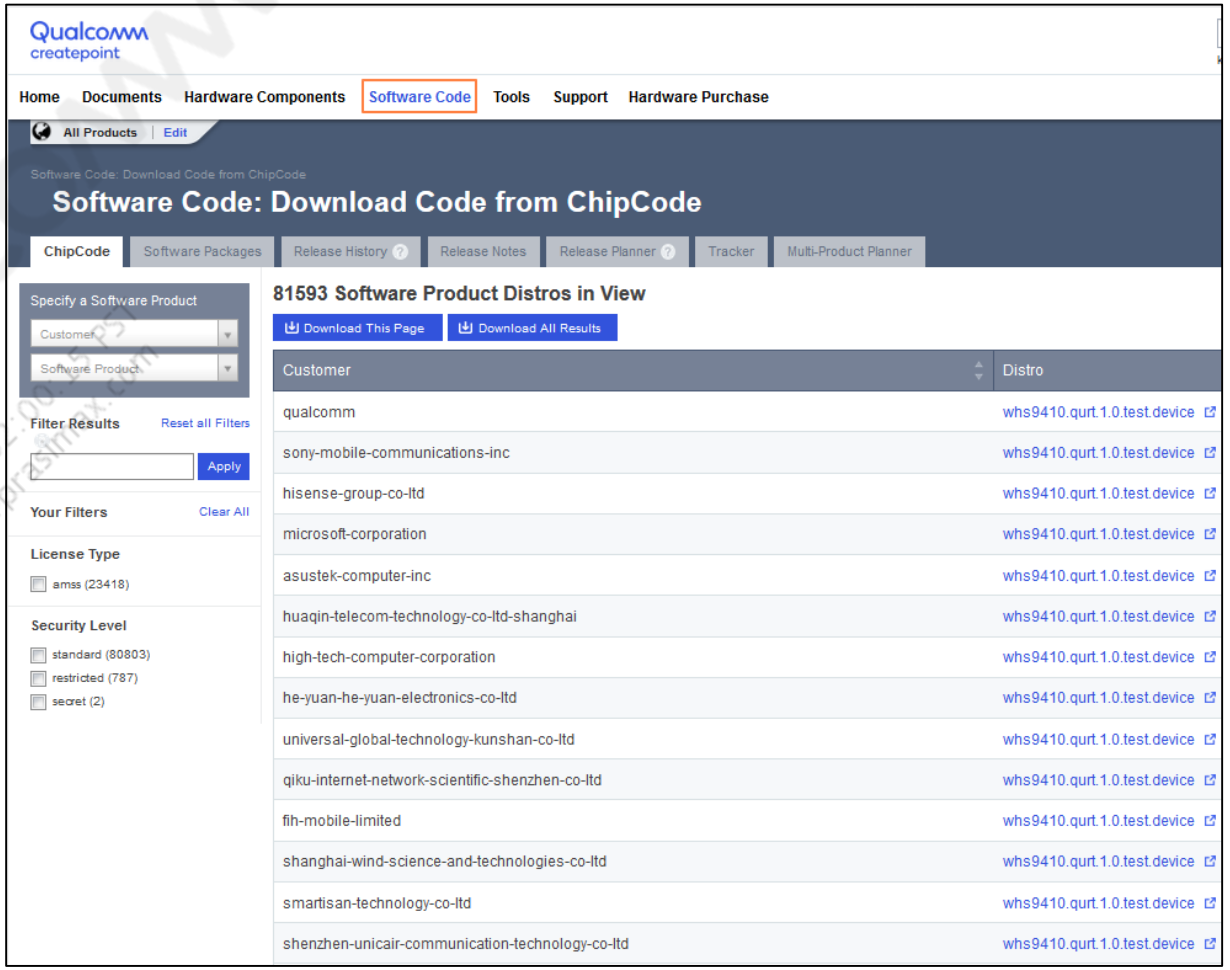
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/



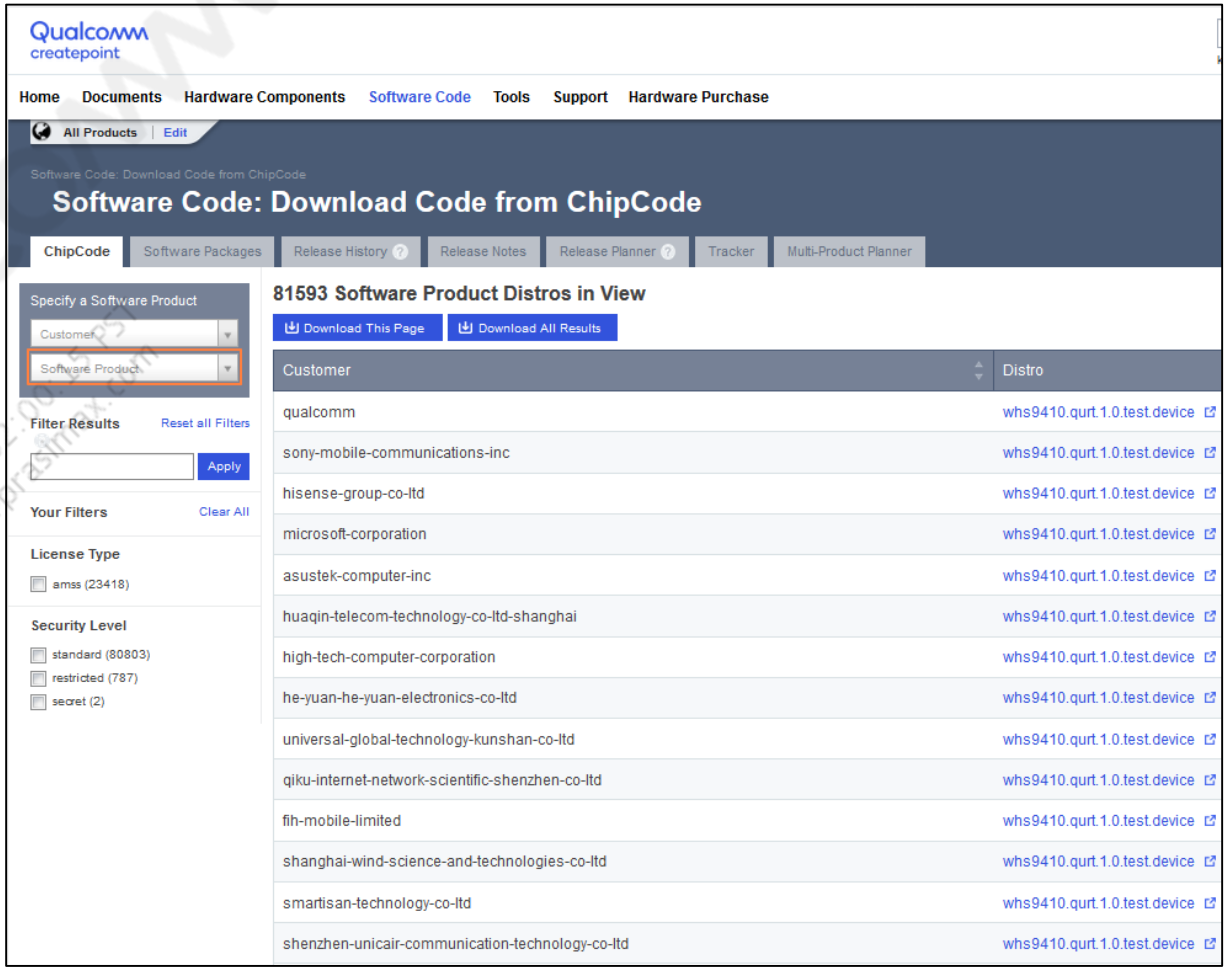
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab



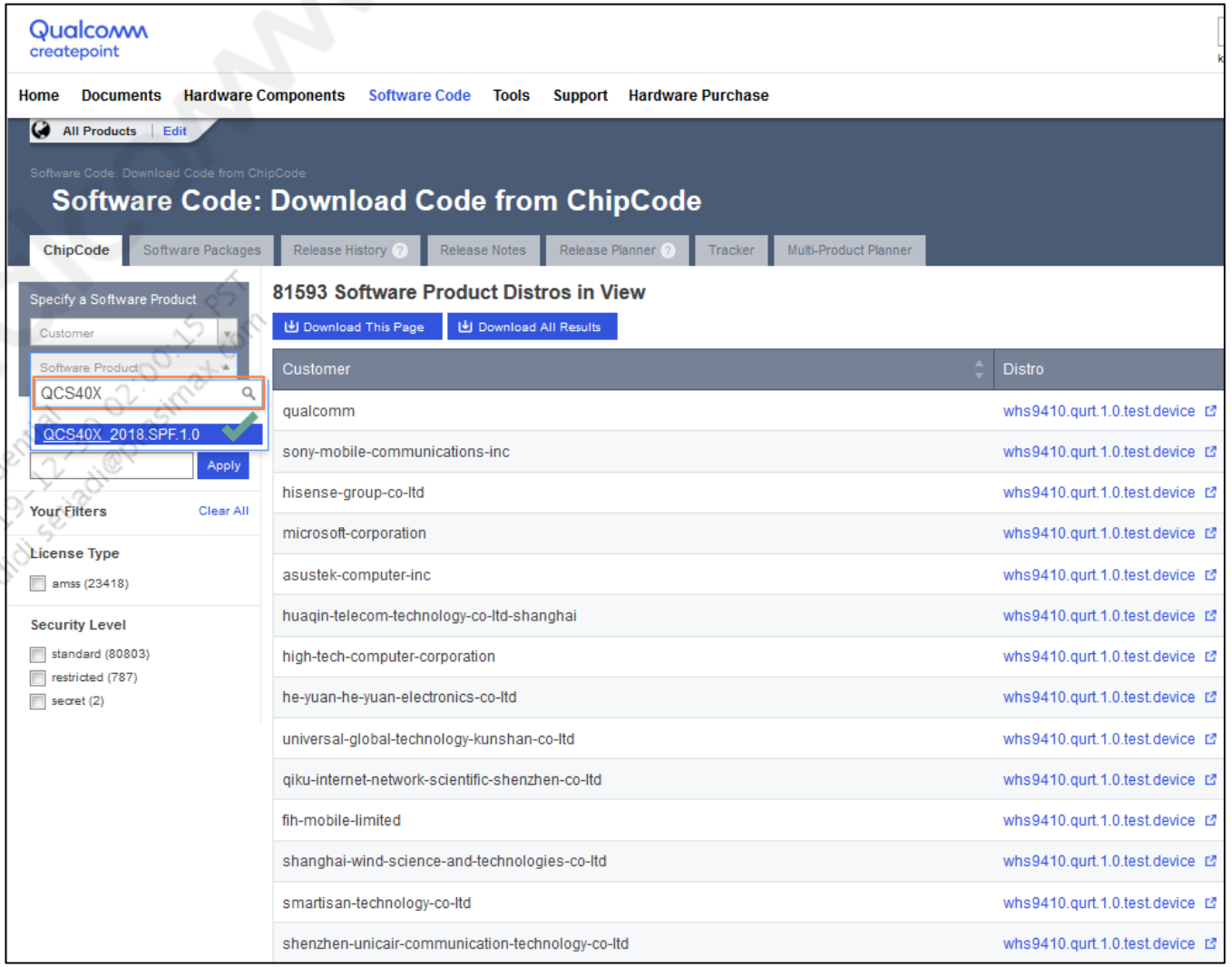
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box



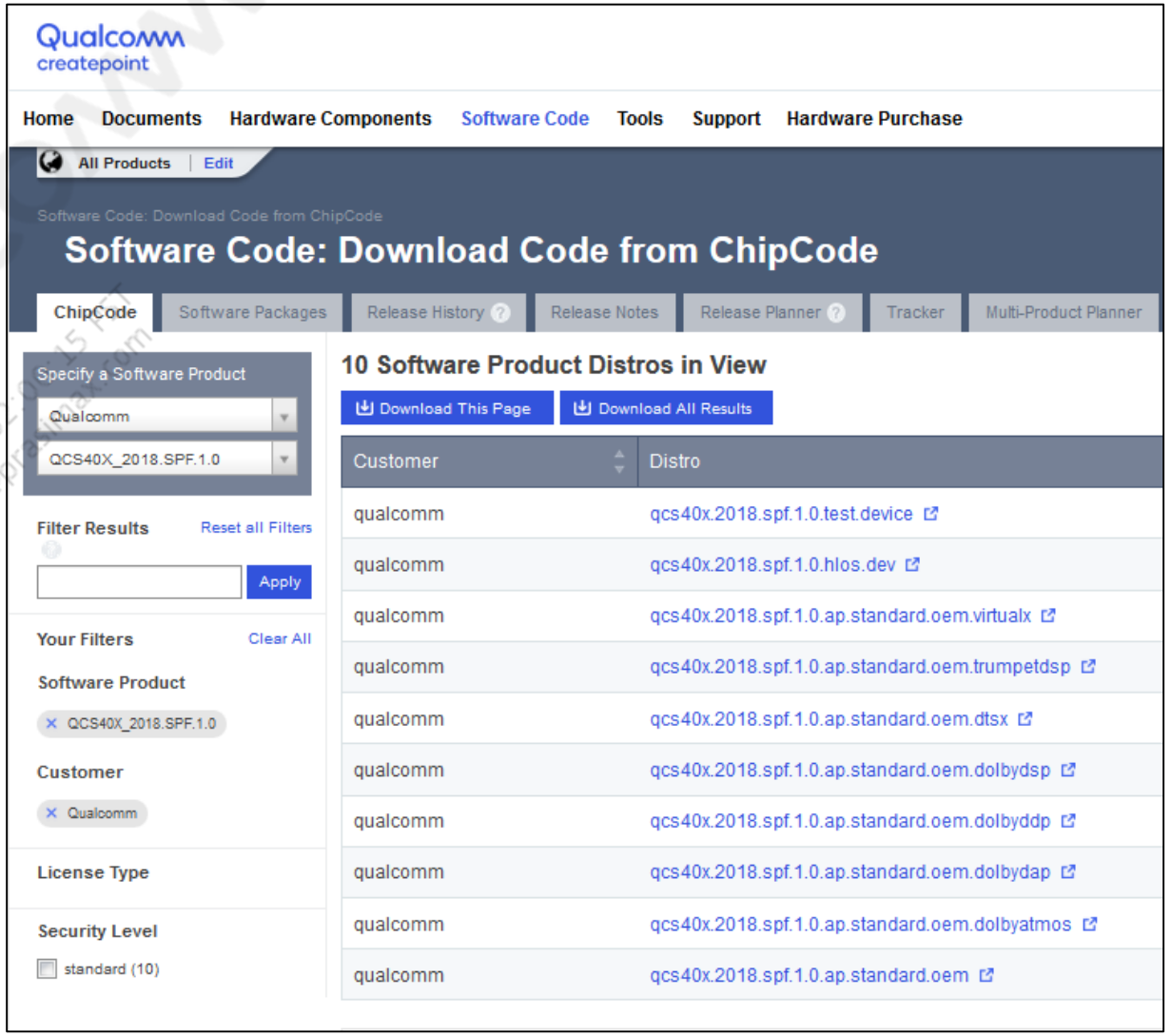
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item



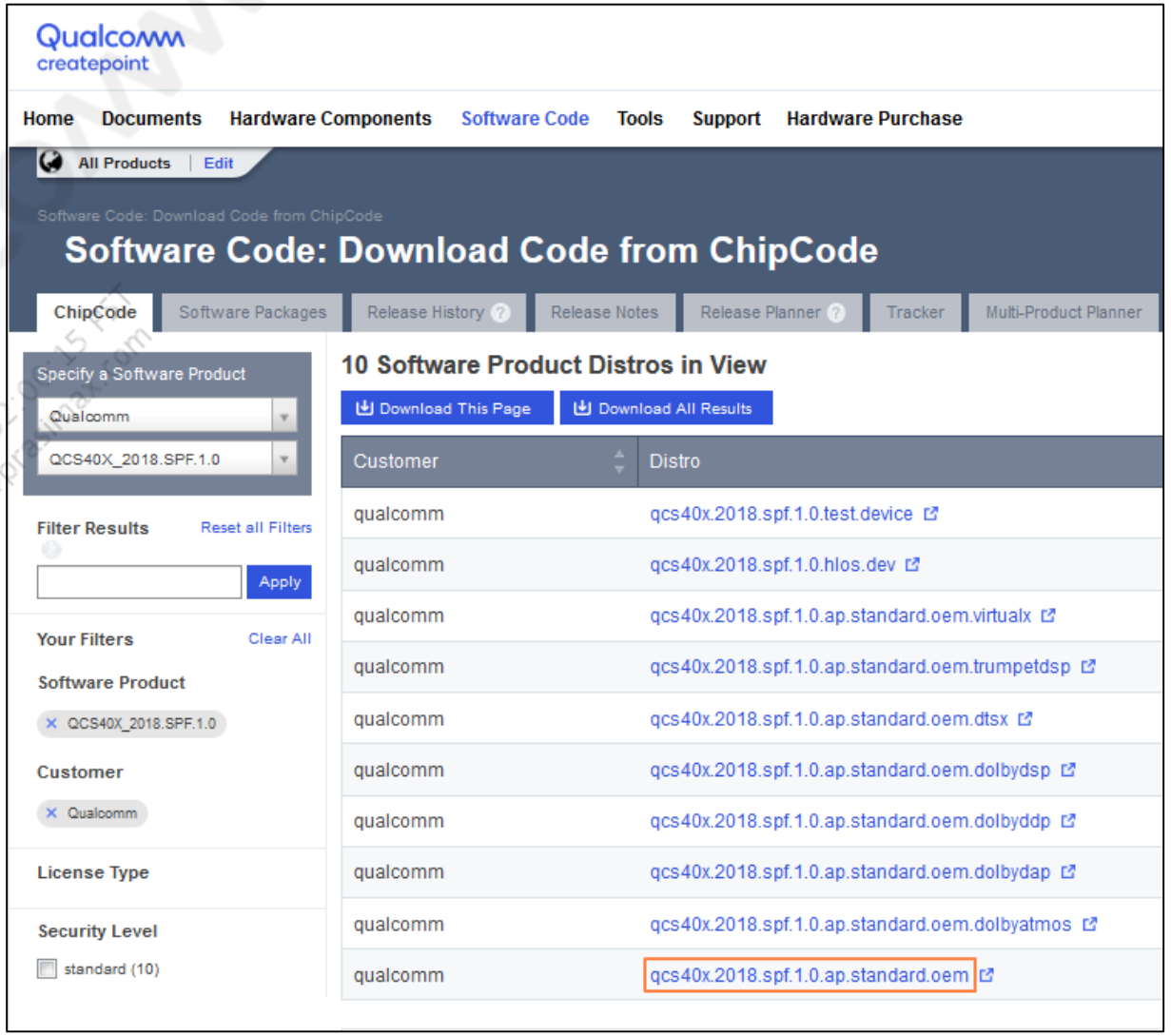
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item
- All available QCS40x software packages will be shown



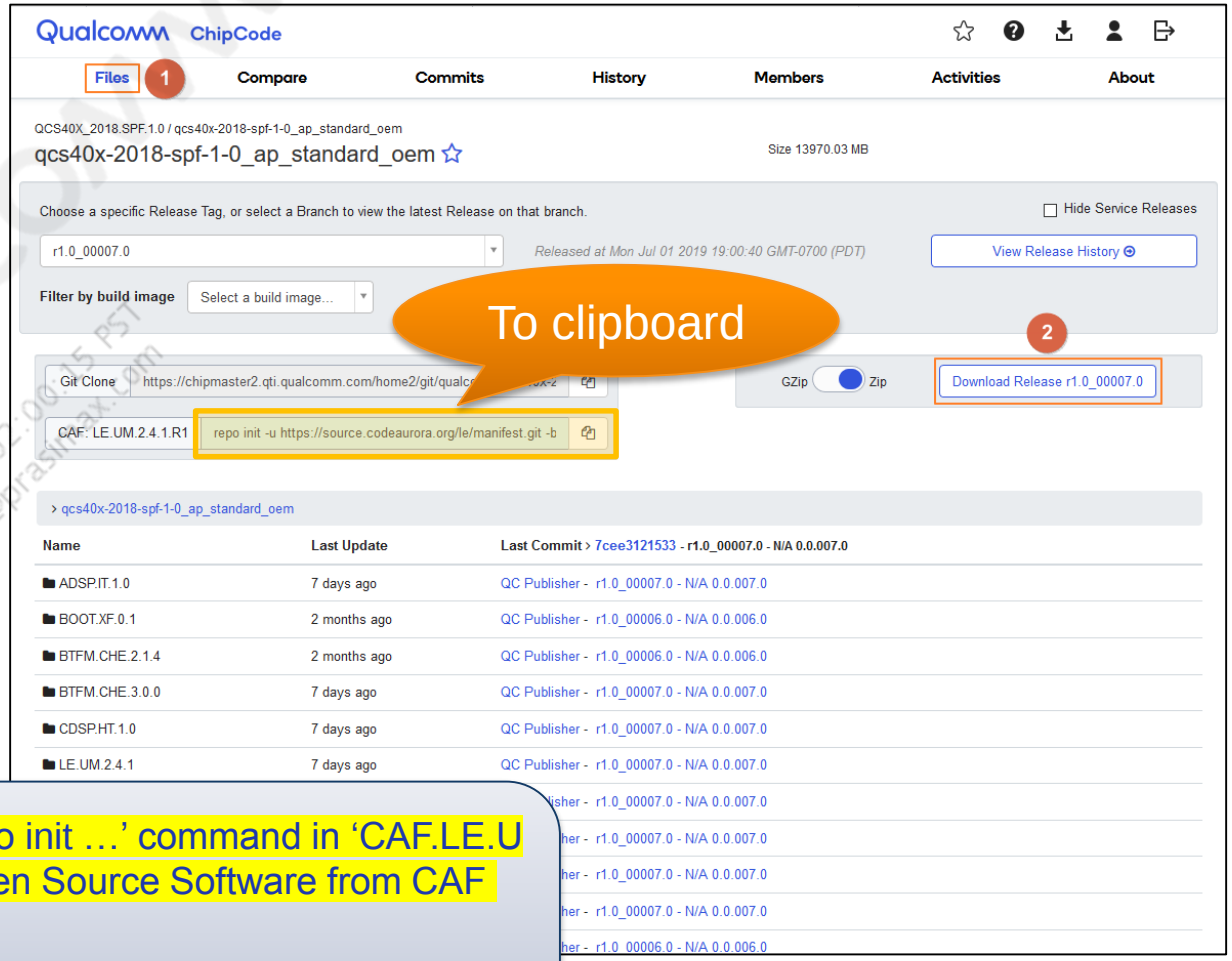
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item
- All available QCS40x software packages will be shown
- Click on ‘qcs40x.2018.spf.1.0.ap.standard.oem’ item



QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item
- All available QCS40x software packages will be shown
- Click on ‘qcs40x.2018.spf.1.0.ap.standard.oem’ item
- Click on ‘Files’ tab (1)
- Click on ‘Download Release <version>’ button (2)

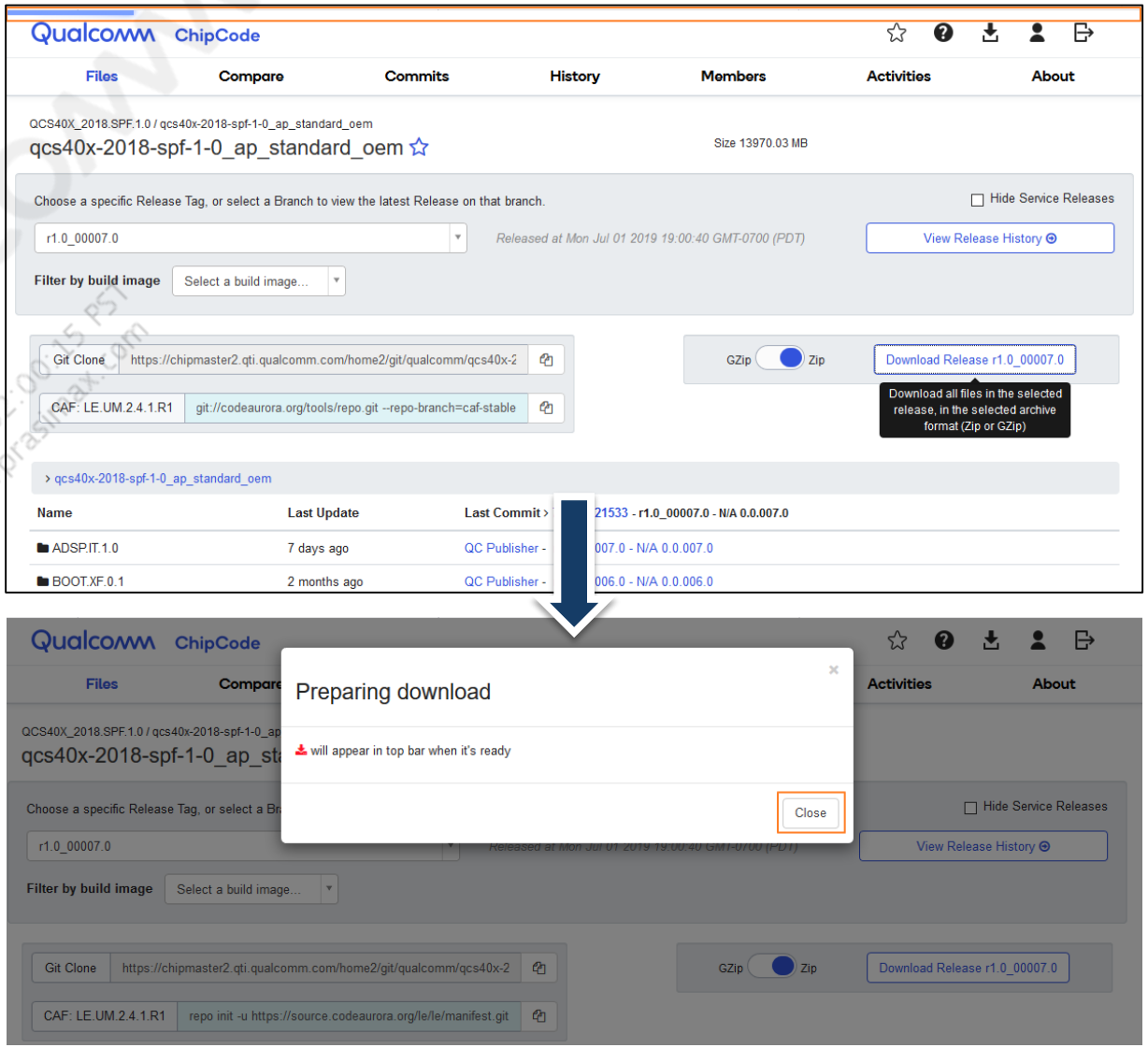


NOTE : Copy and backup ‘repo init …’ command in ‘CAF.LE.UM.2.4.1.R1’ edit box to get Open Source Software from CAF

e.g. for r1.0_00007.0>
repo init -u https://source.codeaurora.org/le/le/manifest.git -b release -m LE.UM.2.4.1.r1-09900-qcs405.0.xml --repo-url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable

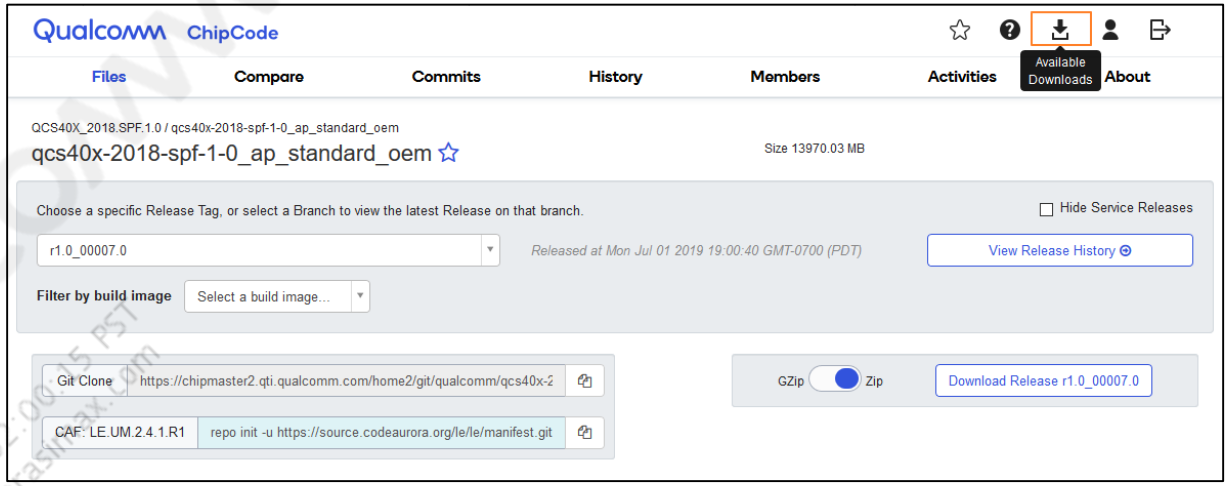
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item
- All available QCS40x software packages will be shown
- Click on ‘qcs40x.2018.spf.1.0.ap.standard.oem’ item
- Click on ‘Files’ tab
- Click on ‘Download Release <version>’ button
- Wait for the progress at the top until ‘Preparing download’ window appears
- Click on ‘Close’ button on ‘Preparing download’ window



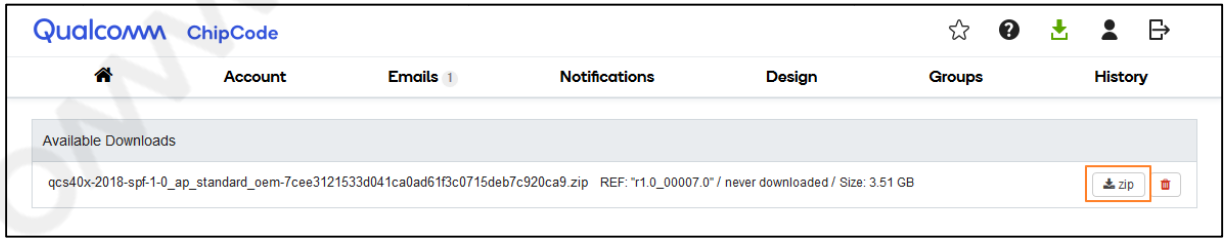
QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item
- All available QCS40x software packages will be shown
- Click on ‘qcs40x.2018.spf.1.0.ap.standard.oem’ item
- Click on ‘Files’ tab
- Click on ‘Download Release <version>’ button
- Wait for the progress at the top until ‘Preparing download’ window appears
- Click on ‘Close’ button on ‘Preparing download’ window
- Click on ‘Available Downloads’ button



QTI Proprietary Software

- To download QTI proprietary software,
  - Go: https://createpoint.qti.qualcomm.com/dashboard/
- Click on ‘Software Code’ tab
- Click on ‘Software Product’ combo box
- Input ‘QCS40X’ into the search box
- Click on ‘QCS40X_2018.SPF.1.0’ item
- All available QCS40x software packages will be shown
- Click on ‘qcs40x.2018.spf.1.0.ap.standard.oem’ item
- Click on ‘Files’ tab
- Click on ‘Download Release <version>’ button
- Wait for the progress at the top until ‘Preparing download’ window appears
- Click on ‘Close’ button on ‘Preparing download’ window
- Click on ‘Available Downloads’ button
- Click on ‘zip’ button to start download – DONE !

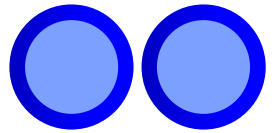


QTI Proprietary Software – Links

- Below are the direct links for each package
  - Replace the <Company ID> with own Company ID of CreatePoint
  - For example for ‘qualcomm’: https://chipcode.qti.qualcomm.com/projects/qualcomm/qcs40x-2018-spf-1-0_ap_standard_oem

- SSRD & SBRD without 3rd party codices
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_test_device
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_hlos_dev
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem ← superset of above two

- SBRD with 3rd party codices (Dolby Atmos/DTS:X)
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_test_device
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_hlos_dev
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem ← superset of above two
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_virtualx
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_trumpetdsp
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_dtsx
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_dolbydsp
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_dolbyddp
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_dolbydap
  - https://chipcode.qti.qualcomm.com/projects/<Company ID>/qcs40x-2018-spf-1-0_ap_standard_oem_dolbyatmos



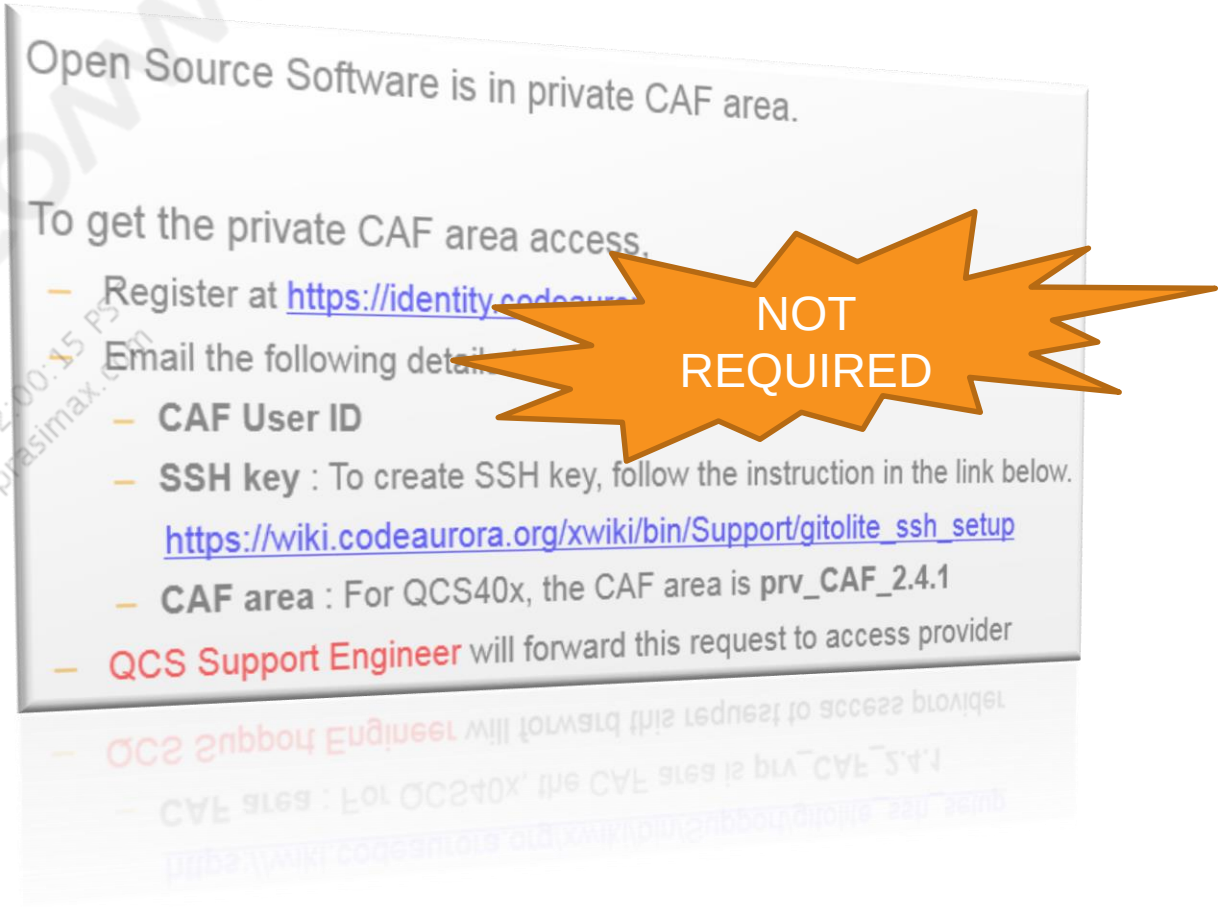
Section 4.4

Open Source Software

Open Source Software

- Starting with r1.0_00007.0 software, Open Source Software moves to public CAF area
- So, approval process is not required for downloading the Open Source Software

- If the organization is not registered yet, contact the QTI regional sales team



Open Source Software

- To get Open Source Software, run commands on the right side in your Linux Build Machine
  - ‘repo init …’ command is copied from QTI Proprietary Source downloading step
  - <work_root> is base folder to build QCS403 or QCS405 software
- DONE !

Example commands for *<r1.0_00007.0>* version

<work_root> $ mkdir CAF

<work_root> $ cd CAF

<work_root>/CAF $ repo init -u https://source.codeaurora.org/le/le/manifest.git -b release -m LE.UM.2.4.1.r1-09900-qcs405.0.xml --repo-url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable

<work_root>/CAF $ repo sync

Section 4.5

Merge / Reconstruct Software

Merge / Reconstruct Software

- The following steps are required before attempting to build software images
  - Merge the software downloaded from CAF and ChipCode
  - Reconstruct folder structure

Merge software downloaded from CAF and ChipCode

COPY from CAF source to ChipCode/APPS_PROC

Goal of Reconstruct folder structure

<QCS40x_WS>/contents.xml
<QCS40x_WS>/common
<QCS40x_WS>/apps_proc
<QCS40x_WS>/adsp_proc
<QCS40x_WS>/boot_images
<QCS40x_WS>/btfm_proc
<QCS40x_WS>/cdsp_proc
<QCS40x_WS>/rpm_proc
<QCS40x_WS>/trustzone_images
<QCS40x_WS>/wlan_proc

Section 4.5.1

Merge Software

Merge Software

- 1. Copy QTI proprietary software .zip file to <work_root> folder and unzip the .zip file
  - Make sure ‘qcs40x-2018-spf-1-0_ap_standard_oem.git’ folder

Merge CAF and ChipCode commands

<work_root>$ unzip *qcs40x-2018-spf-1-0_ap_standard_oem-….zip*

<work_root>$ ls
qcs40x-2018-spf-1-0_ap_standard_oem.git

Merge Software

- 1. Copy QTI proprietary software .zip file to <work_root> folder and unzip the .zip file
  - Make sure ‘qcs40x-2018-spf-1-0_ap_standard_oem.git’ folder
- 2. Rename ‘qcs40x-2018-spf-1-0_ap_standard_oem.git’ to ‘chipcode_root’

Merge CAF and ChipCode commands

<work_root>$ unzip *qcs40x-2018-spf-1-0_ap_standard_oem-….zip*

<work_root>$ ls
qcs40x-2018-spf-1-0_ap_standard_oem.git

<work_root>$ mv qcs40x-2018-spf-1-0_ap_standard_oem.git chipcode_root

Merge Software

- 1. Copy QTI proprietary software .zip file to <work_root> folder and unzip the .zip file
  - Make sure ‘qcs40x-2018-spf-1-0_ap_standard_oem.git’ folder
- 2. Rename ‘qcs40x-2018-spf-1-0_ap_standard_oem.git’ to ‘chipcode_root’
- 3. Make sure ‘CAF’ folder is in <work_root> folder and copy CAF code to ChipCode/APPS_PROC
- DONE !

Merge CAF and ChipCode commands

```
<work_root>$ unzip qcs40x-2018-spf-1-0_ap_standard_oem-….zip

<work_root>$ ls
qcs40x-2018-spf-1-0_ap_standard_oem.git

<work_root>$ mv qcs40x-2018-spf-1-0_ap_standard_oem.git chipcode_root

<work_root>$ ls
CAF

<work_root>$ cp -Rf CAF/* chipcode_root/LE.UM.2.4.1/apps_proc/
```

Section 4.5.2

Reconstruct Software

Reconstruct Software

- QTI Proprietary Software contains both QCS403 and QCS405 software
- The table on the right side shows the corresponding chipset for each folder in QTI Proprietary Software

- NOTE: Starting with r1.0_00007.0 software, **TZ.XF.5.1.2** folder is used for TrustZone instead of TZ.XF.5.1 folder

Folder	SW Module	Chipset
LE.UM.2.4.1	APPS_PROC	Both
QCS405.LE.1.0.1	COMMON	QCS403
BTFM.CHE.2.1.4	BTFM_PROC	QCS403
ADSP.IT.1.0	ADSP_PROC	Both
CDSP.HT.1.0	CDSP_PROC	Both
RPM.BF.1.9	RPM_PROC	Both
WLAN.HL.3.1	WLAN_PROC	Both
BOOT.XF.0.1	BOOT_IMAGES	Both
TZ.XF.5.1.2	TRUSTZONE_IMAGES	Both
QCS405.LE.1.0	COMMON	QCS405
BTFM.CHE.3.0.0	BTFM_PROC	QCS405

Reconstruct Software

- QTI Proprietary Software contains both QCS403 and QCS405 software
- The table on the right side shows the corresponding chipset for each folder in QTI Proprietary Software

- If you need to build both QCS403 and QCS405 on the same build machine, the following modules must be physically separated before build starts
  - APPS_PROC
  - ADSP_PROC
- If you don’t want to separate the source code, you should do ‘build clean’ before build starts for the two modules above

Folder	SW Module	Chipset
LE.UM.2.4.1	APPS_PROC	Both
QCS405.LE.1.0.1	COMMON	QCS403
BTFM.CHE.2.1.4	BTFM_PROC	QCS403
ADSP.IT.1.0	ADSP_PROC	Both
CDSP.HT.1.0	CDSP_PROC	Both
RPM.BF.1.9	RPM_PROC	Both
WLAN.HL.3.1	WLAN_PROC	Both
BOOT.XF.0.1	BOOT_IMAGES	Both
TZ.XF.5.1.2	TRUSTZONE_IMAGES	Both
QCS405.LE.1.0	COMMON	QCS405
BTFM.CHE.3.0.0	BTFM_PROC	QCS405

Reconstruct Software – QCS403 reconstruct for only QCS403 user

- If you build only QCS403, follow the steps in below table

Step	Commands
1. Create work space folder	<work_root>$ mkdir QCS403_WS <work_root>$ cd QCS403_WS
2. Copy COMMON - <QCS403_WS> is '<work_root>/QCS403' folder	<QCS403_WS>$ cp ../chipcode_root/QCS405.LE.1.0.1/contents.xml ./contents.xml <QCS403_WS>$ cp -Rf ../chipcode_root/QCS405.LE.1.0.1/common/ common/
3. Create symbolic links for others	<QCS403_WS>$ ln -s ../chipcode_root/LE.UM.2.4.1/apps_proc <QCS403_WS>$ ln -s ../chipcode_root/ADSP.IT.1.0/adsp_proc <QCS403_WS>$ ln -s ../chipcode_root/BTFM.CHE.2.1.4/btfm_proc <QCS403_WS>$ ln -s ../chipcode_root/BOOT.XF.0.1/boot_images <QCS403_WS>$ ln -s ../chipcode_root/CDSP.HT.1.0/cdsp_proc <QCS403_WS>$ ln -s ../chipcode_root/RPM.BF.1.9/rpm_proc <QCS403_WS>$ ln -s ../chipcode_root/TZ.XF.5.1.2/trustzone_images <QCS403_WS>$ ln -s ../chipcode_root/WLAN.HL.3.1/wlan_proc

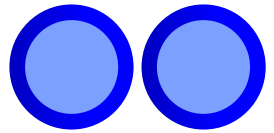
Reconstruct Software – QCS403 reconstruct for both QCS403 and QCS405 user

- If you build both QCS403 and QCS405, follow the steps in below table

Step	Commands
1. Create work space folder	<work_root>$ mkdir QCS403_WS <work_root>$ cd QCS403_WS
2. Copy COMMON, APPS_PROC and ADSP_PROC - <QCS403_WS> is '<work_root>/QCS403' folder	<QCS403_WS>$ cp ../chipcode_root/QCS405.LE.1.0.1/contents.xml ./contents.xml <QCS403_WS>$ cp -Rf ../chipcode_root/QCS405.LE.1.0.1/common/ common/ <QCS403_WS>$ cp -Rf ../chipcode_root/LE.UM.2.4.1/apps_proc/ apps_proc/ <QCS403_WS>$ cp -Rf ../chipcode_root/ADSP.IT.1.0/adsp_proc/ adsp_proc/
3. Create symbolic links for others	<QCS403_WS>$ ln -s ../chipcode_root/BTFM.CHE.2.1.4/btfm_proc <QCS403_WS>$ ln -s ../chipcode_root/BOOT.XF.0.1/boot_images <QCS403_WS>$ ln -s ../chipcode_root/CDSP.HT.1.0/cdsp_proc <QCS403_WS>$ ln -s ../chipcode_root/RPM.BF.1.9/rpm_proc <QCS403_WS>$ ln -s ../chipcode_root/TZ.XF.5.1.2/trustzone_images <QCS403_WS>$ ln -s ../chipcode_root/WLAN.HL.3.1/wlan_proc

Reconstruct Software

- Make sure the reconstruction result in <QCS40x_WS> folder
- The figure on the right shows the results for QCS403_WS
- DONE !



Section 4.6

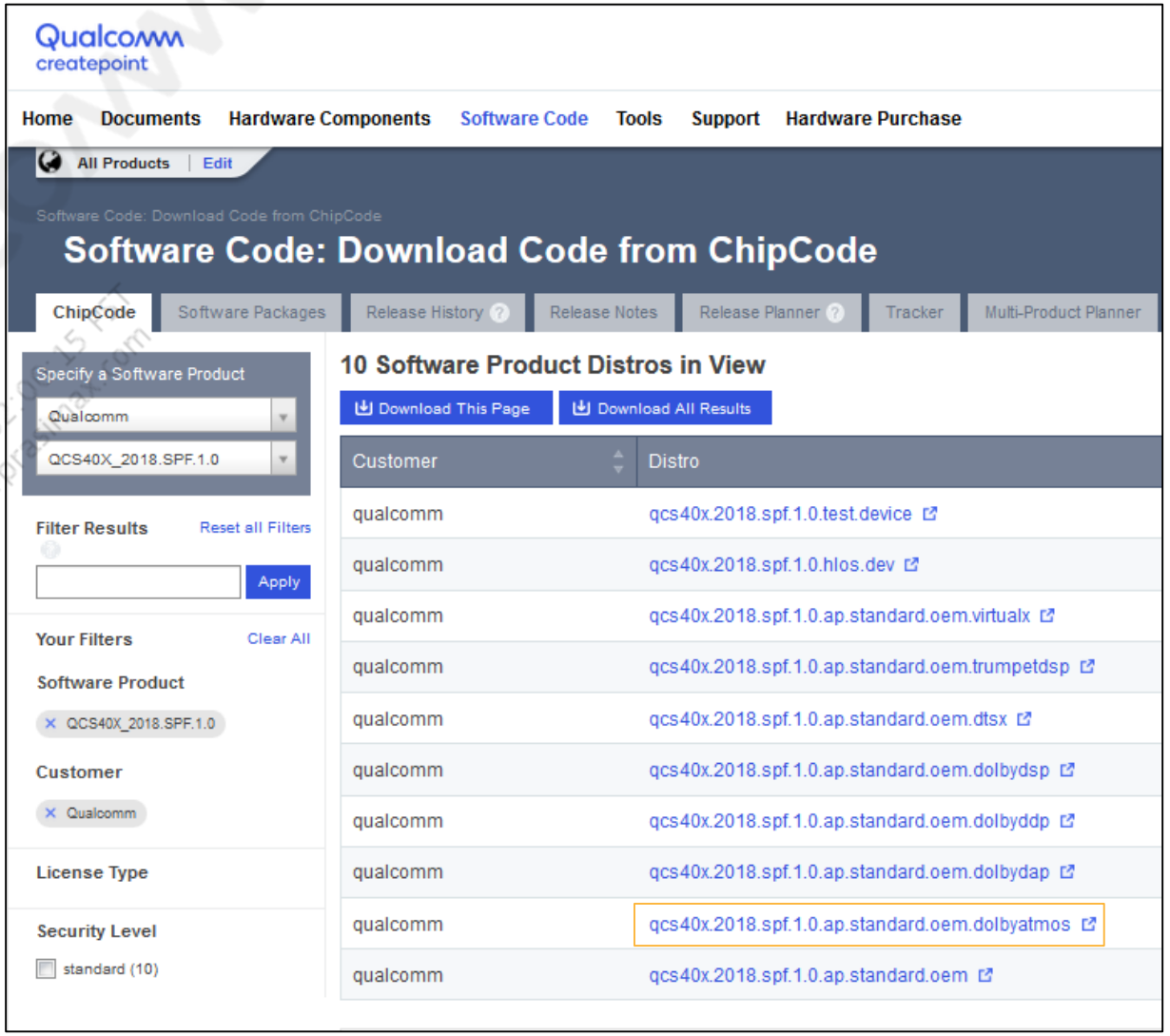
3rd Party Software Integration

3rd Party Software Integration

- 3rd Party Software consists:
  - Dolby
  - DTS
  - VirtualX
- The default QTI Proprietary Software does not contain 3rd Party Software
- In order to include the 3rd Party Software in the built image, separate library integration is required
- 3rd Party Software is provided as a Library through ChipCode portal

3rd Party Software Integration

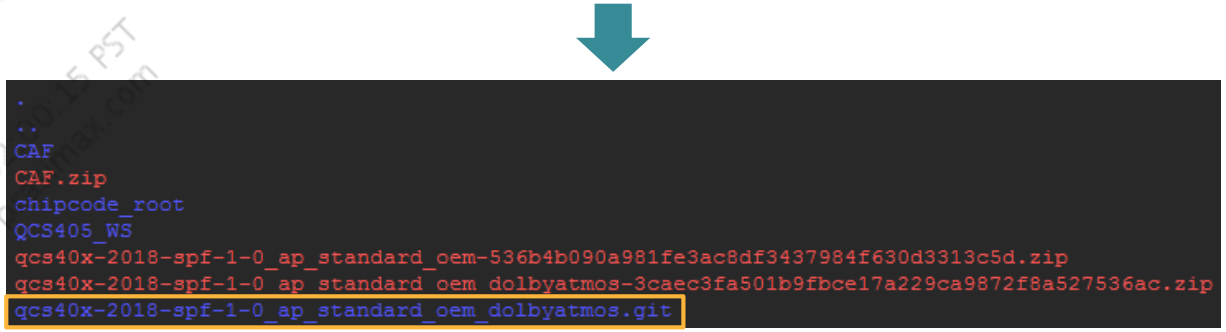
- To download 3rd Party Software, refer to QTI Proprietary Software downloading step first
- In this chapter, DolbyAtmos will be explained as an example
- Download ‘qcs40x.2018.spf.1.0.ap.standard.oem.dolbyatmos’ package



3rd Party Software Integration

- To download 3rd Party Software, refer to QTI Proprietary Software downloading step first
- In this chapter, DolbyAtmos will be explained as an example
- Download ‘qcs40x.2018.spf.1.0.ap.standard.oem.dolbyatmos’ package
- Move downloaded .zip file to <work_root> folder and unzip the .zip file
  - Make sure ‘qcs40x-2018-spf-1-0_ap_standard_oem_dolbyatmos.git’ folder on the <work_root> folder

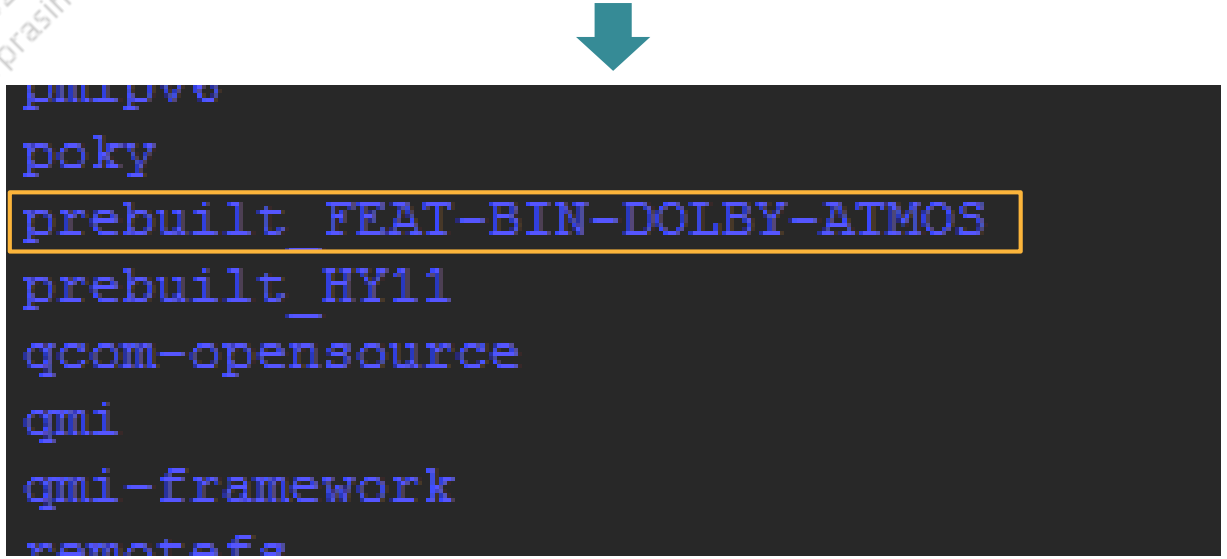
```
<work_root>$ unzip qcs40x-2018-spf-1-0_ap_standard_oem_dolbyatmos-xxx.zip

<work_root>$ ls -al
```



3rd Party Software Integration

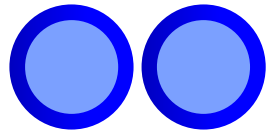
- To download 3rd Party Software, refer to QTI Proprietary Software downloading step first
- In this chapter, DolbyAtmos will be explained as an example
- Download ‘qcs40x.2018.spf.1.0.ap.standard.oem.dolbyatmos’ package
- Move downloaded .zip file to <work_root> folder and unzip the .zip file
  - Make sure ‘qcs40x-2018-spf-1-0_ap_standard_oem_dolbyatmos.git’ folder on the <work_root> folder
- Copy ‘prebuilt-FEAT-BIN-DOLBY_ATMOS’ folder into <QCS40x_WS>/apps_proc/
  - Make sure ‘prebuild-FEAT-BIN-DOLBY_ATMOS’ folder on the <QCS40x_WS>/apps_proc folder
- Integration DONE !
- Now DolbyAtmos will be included in the built image

```
<work_root>$ cp -nr qcs40x-2018-spf-1-0_ap_standard_oem_dolbyatmos.git/LE.UM.2.4.1/apps_proc/prebuilt_FEAT-BIN-DOLBY-ATMOS QCS405_WS/apps_proc/prebuilt_FEAT-BIN-DOLBY-ATMOS

<work_root>$ cd QCS405_WS/apps_proc

<work_root>/QCS405_WS/apps_proc $ ls -al
```



Section 5

Build Software



Section 5.1

Build HLOS Software

Build HLOS Software – QCS403

	HLOS software image build commands for QCS403
Build setup	<QCS403_WS>$ cd apps_proc/poky <QCS403_WS>/apps_proc/poky$ export SHELL=/bin/bash <QCS403_WS>/apps_proc/poky$ source build/conf/set_bb_env.sh <QCS403_WS>/apps_proc/poky/build$ list-build-commands
Build all (debug)	<QCS403_WS>/apps_proc/poky/build$ build-qcs403-som2-qsap-image
Build all (perf)	<QCS403_WS>/apps_proc/poky/build$ build-qcs403-som2-qsap-perf-image
Clean all	<QCS403_WS>/apps_proc/poky/build$ buildclean all
Build log	NOTE: Tasks Summary: Attempted 5907 tasks of which 5262 didn't need to be rerun and all succeeded. NOTE: Writing buildhistory Summary: There were 21 WARNING messages shown. /home/donald/r1.0_00006.0/chipcode_root/LE.UM.2.4.1/apps_proc/poky/build
Build log file	✓ There are multiple build logs in the folder below <QCS403_WS>/apps_proc/poky/build/tmp-glibc/work/qcs403_som2-oe-linux-gnueabi/machine-image/1.0-r0/temp/
Build result	✓ All result files are <QCS403_WS>/apps_proc/poky/build/tmp-glibc/deploy/images/<Build_Variant>/ ✓ <Build_Variant> is 'qcs403-som2-qsap' or 'qcs403-som2-qsap-perf' <QCS403_WS>/apps_proc/poky/build/tmp-glibc/deploy/images/<Build_Variant>/abl-squashfs.elf <QCS403_WS>/apps_proc/poky/build/tmp-glibc/deploy/images/<Build_Variant>/qcs40x-boot.img <QCS403_WS>/apps_proc/poky/build/tmp-glibc/deploy/images/<Build_Variant>/qcs40x-squashfs-sysfs.ubi

Build HLOS Software - Trouble Shooting

Build HLOS Software – Trouble shooting

- [ERROR]: Often unexpected build error or Linux reboot may occur due to the physical performance of your build machine (Memory, CPU)

- [SOLUTION]: You can adjust the number of build threads by modifying the file below and then try to build again
  - <QCS40x_WS>/apps_proc/poky/build/conf/local.conf
    - BB_NUMBER_THREADS ?= " 20“
    - PARALLEL_MAKE ?= "-j 20“
  - 20 is default value of the build thread count
  - Change the value from 20 to 8 or 4

  - **NOTE #1**: KEEP the question mark and space before the count value
    - BB_NUMBER_THREADS ?= " 20“

  - **NOTE #2**: This unexpected error can cause either object or archive files broken which is extremely hard to catch, therefore **‘buildclean all’** first and then try to build again

Error message example

ERROR: Worker process (21511) exited unexpectedly (-9), shutting down...

<QCS40x_WS>/apps_proc/poky/build/conf/local.conf

```
#
# Parallelism Options
#
# These two options control how much parallelism BitBake should use. The first
# option determines how many tasks bitbake should run in parallel:
#
BB_NUMBER_THREADS ?= " 20“
#
# The second option controls how many processes make should run in parallel when
# running compile tasks:
#
PARALLEL_MAKE ?= "-j 20“
#
# For a quadcore, BB_NUMBER_THREADS = "4", PARALLEL_MAKE = "-j 4" would
# be appropriate for example.
```

Build HLOS Software – Trouble shooting

- [ERROR]: ‘opensslv.h’ file not found error

- [SOLUTION]: Install ‘libssl-dev’ package
  - $ sudo apt-get install libssl-dev

- If you failed to install ‘libssl-dev’ package, resolve as follow
  - $ apt-cache policy libssl-dev
  - $ sudo apt-get install libssl1.0.0=1.0.2g-1ubuntu4.1
  - $ sudo apt-get install libssl-dev

Error message example

```
/poky/build/tmp-glibc/work-shared/qcs405-som1/kernel-source/scripts/sign-file.c:25:30:
fatal error: openssl/opensslv.h: No such file or directory
```

Install libssl-dev

```
$ sudo apt-get install libssl-dev
…
E: Unable to correct problems, you have held broken packages.

$ apt-cache policy libssl-dev
libssl-dev:
  Installed: (none)
  Candidate: 1.0.2g-1ubuntu4.1
  Version table:
     1.0.2g-1ubuntu4.1 500
        500 http://mirror.uoregon.edu/ubuntu xenial-updates/main amd64 Packages
        500 http://mirror.uoregon.edu/ubuntu xenial-security/main amd64 Packages
     1.0.2g-1ubuntu4 500
        500 http://mirror.uoregon.edu/ubuntu xenial/main amd64 Packages

$ sudo apt-get install libssl1.0.0=1.0.2g-1ubuntu4.1
$ sudo apt-get install libssl-dev
```

Build HLOS Software – Trouble shooting

- [ERROR]: ‘bouncycastle’ recipe build error

- [SOLUTION]: Install Java v1.7.0 JDK
  - $ sudo add-apt-repository ppa:openjdk-r/ppa
  - $ sudo apt-get update
  - $ sudo apt-get install openjdk-7-jdk
  - $ update-java-alternatives --list
  - $ sudo update-java-alternatives -s java-1.7.0-openjdk-amd64
  - $ java -version

Error message example

ERROR: Task (/home/donald/work_root/chipcode_root/LE.UM.2.4.1/apps_proc/poky/meta-qti-bsp/recipes-devtools/bouncycastle/bouncycastle_git.bb:do_compile) failed with exit code '1'

Install Java v1.7

```
$ sudo add-apt-repository ppa:openjdk-r/ppa
gpg: key 86F44E2A: public key "Launchpad OpenJDK builds (all archs)" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK

$ sudo apt-get update
$ sudo apt-get install openjdk-7-jdk
$ update-java-alternatives --list
java-1.7.0-openjdk-amd64      1071      /usr/lib/jvm/java-1.7.0-openjdk-amd64
java-1.8.0-openjdk-amd64      1081      /usr/lib/jvm/java-1.8.0-openjdk-amd64

$ sudo update-java-alternatives -s java-1.7.0-openjdk-amd64

$ java -version
java version "1.7.0_95"
OpenJDK Runtime Environment (IcedTea 2.6.4) (7u95-2.6.4-3)
OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)
```

Build HLOS Software – Trouble shooting

- [ERROR]: MD5 and SHA256 Checksum mismatch! error

- [REASON]:
  - Some open source packages will be downloaded in build time
    - e.g. gstreamer, iotivity, AVS SDK, and so on…
  - The downloaded file's expected checksum is saved into the related recipe file
  - If the downloaded file from open source package is changed, then the checksum in recipe file must be changed too
  - If the open source package changes after the software release, this problem will occur

- [SOLUTION]: Modify recipe file
  - The error log guide us the solution. So modify recipe file according to error log's guide first
  - NOTE: Maybe this modification will be reflected in the next SW release

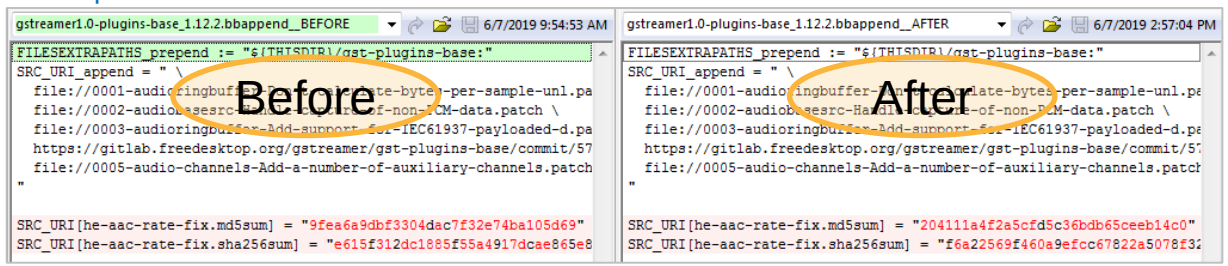
Error message example

a63d757390b80d44f8c2.patch;name=he-aac-rate-fix'. **Checksum mismatch!**
…
If this change is expected (e.g. you have upgraded to a new version without updating the checksum s) then you can use these lines within the recipe:
SRC_URI[he-aac-rate-fix.**md5**sum] = "**204111a4f2a5cfd5c36bdb65ceeb14c0**"
SRC_URI[he-aac-rate-fix.**sha256**sum] = "**f6a22569f460a9efcc67822a5078f32cdbf89cd76e58026b85be185398ca798f**"
…
ERROR: Logfile of failure stored in: /local2/mnt/workspace/QCS40x/r1.0_00006.4/**QCS403_WS/apps_proc/poky/build/tmp-glibc/work/armv7ahf-neon-oe-linux-gnueabi/gstreamer1.0-plugins-base/1.12.2-r0/temp/log.do_fetch.24678**
…

Modify recipe(gstreamer1.0-plugins-base_1.12.2.bbappend) file

1. Check build log (log.do_fetch.24678) file and find the exact recipe file for error
- In this case '**<QCS403_WS>/apps_proc/poky/meta-qti-adk/recipes-multimedia/gstreamer/gstreamer1.0-plugins-base_1.12.2.bbappend**' is the **recipe** file for gstreamer

2. Replace MD5 and SHA256 checksum values like below



Build HLOS Software - Tips

Build HLOS Software – Tips: How to enable apps debug log in perf image

- In perf image, apps debug log is disabled by default
- If you want to enable apps debug log, follow the steps below:
  - ✓ Modify '/apps_proc/kernel/msm-4.14/arch/**arm**/configs/vendor/qcs405-perf_defconfig' file
    - Following three configurations are required to enable apps debug log

```
CONFIG_SERIAL_MSM=y
CONFIG_SERIAL_MSM_CONSOLE=y
CONFIG_SERIAL_MSM_HS=y
```

    - If the configuration exists, change value to **y**
    - If the configuration does not exist, add it

  - ✓ Rebuild Kernel

```
<QCS403_WS>/apps_proc/poky/build$ MACHINE=qcs403-som2 DISTRO=qsap VARIANT=perf rebake virtual/kernel
```

  - ✓ Build all apps image

```
<QCS403_WS>/apps_proc/poky/build$ build-qcs403-som2-qsap-perf-image
```

Build HLOS Software – Tips: How to enable Login Function in perf image

- In perf image, Login Function is disabled by default
- If you want to enable Login Function, follow the steps below:
  - ✓ Modify '/apps_proc/poky/meta-qti-bsp/conf/qcs40x.inc' file

BEFORE	SERIAL_CONSOLE ?= ${@['', '115200 ttyMSM0'][d.getVar('VARIANT', True) == '' or 'debug')]}
AFTER	SERIAL_CONSOLE ?= "ttyMSM0"

  - ✓ Rebuild Kernel

```
<QCS403_WS>/apps_proc/poky/build$ MACHINE=qcs403-som2 DISTRO=qsap VARIANT=perf rebake virtual/kernel
```

  - ✓ Build all apps image

```
<QCS403_WS>/apps_proc/poky/build$ build-qcs403-som2-qsap-perf-image
```

Build HLOS Software – Tips: How to make ubifs image (1/2)

- Starting with r1.0_00007.0 software, ubifs images(abl.elf, qcs40x-sysfs.ubi) do not generated by default
- If you want to build ubifs images, follow the steps below:
  - ✓ Modify '/apps_proc/poky/meta-qti-bsp/recipes-kernel/edk2/edk2_git.bb' file

*** Remove following two lines**

```
NAND_SQUASHFS_SUPPORT = ${@bb.utils.contains('DISTRO_FEATURES', 'nand-squashfs', '1', '0', d)}
EXTRA_OEMAKE_append = " 'NAND_SQUASHFS_SUPPORT=${NAND_SQUASHFS_SUPPORT}'"
```

  - ✓ Modify '/apps_proc/poky/meta-qti-bsp-prop/recipes-products/images/qcs40x/qcs40x-qti-image.inc' file

*** Replace 'qcs40x-sysfs.squash' with 'qcs40x-sysfs.ubifs'**

*** Replace 'create_squash_ubi_img()' function with:**

```
create_squash_ubi_img() {
create_squashfs_ubinize_config
mkfs.ubifs -r ${USR_IMAGE_ROOTFS} -o ${OUTPUT_FILE_USR_UBIFS} ${MKUBIFS_ARGS}
mkfs.ubifs -r ${IMAGE_ROOTFS} -o ${DEPLOY_DIR_IMAGE}/qcs40x-sysfs.ubifs ${MKUBIFS_ARGS}
ubinize -o ${OUTPUT_FILE_FINAL_SQUASHFS_UBI} ${UBINIZE_ARGS} ${UBINIZE_SQUASHFS_CFG}
chmod 644 ${OUTPUT_FILE_FINAL_SQUASHFS_UBI}
}
```

*** Modify 'SYSTEM_SQUASHFS_VOLUME_SIZE' value to fit the generated image size**

- If you build after reflecting the above fix, then the following error will occur:

ubinize: error!: error in section "sysfs_volume": size of the image file "…/apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/qcs40x-sysfs.ubifs" is **122150912**, which is larger than volume size **90725120**

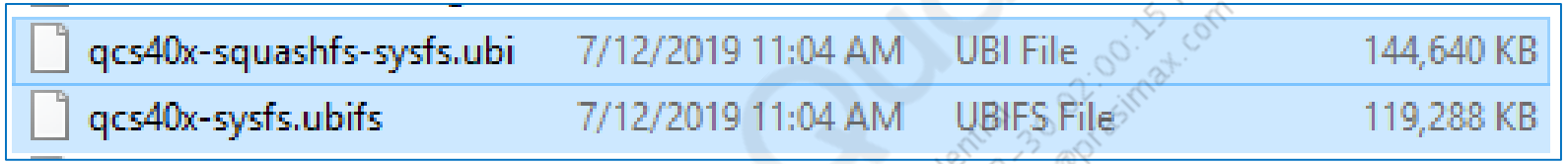
- SYSTEM_SQUASHFS_VOLUME_SIZE is a value that limits the size of the sysfs. Therefore, the value should be modified to more than image size
- SYSTEM_SQUASHFS_VOLUME_SIZE value should be multiple of LEB, LEB is 253952 bytes
- For example, SYSTEM_SQUASHFS_VOLUME_SIZE = "146784256"

Build HLOS Software – Tips: How to make ubifs image (2/2)

- Build all apps image

Debug	<QCS403_WS>/apps_proc/poky/build$ build-qcs403-som2-qsap-image
Perf	<QCS403_WS>/apps_proc/poky/build$ build-qcs403-som2-qsap-perf-image

- Check the build result in '/apps_proc/poky/build/tmp-glibc/deploy/images/<Build_Variant>/' folder



- Note that, the final image file name (qcs40x_squashfs_sysfs.ubi) include squashfs, but it is made up of ubifs

Section 5.2

Build non-HLOS software

Build non-HLOS software

- Based on ‘ap_standard_oem’ package:

- Following modules must be built before to generate final image
  - ADSP_PROC
  - CDSP_PROC
  - RPM_PROC
  - BOOT_IMAGES
  - TRUSTZONE_IMAGES

- Following two modules are provided as binary, so build is not required
  - WLAN_PROC
  - BTFM_PROC

Section 5.2.1

Build ADSP

Build ADSP

	ADSP build commands and prerequisite for QCS403
Prerequisite	✓ Hexagon Tool v8.2.02 installation
Build setup	✓ HEXAGON_ROOT depends on where the tools are installed <QCS403_WS>$ cd adsp_proc <QCS403_WS>/adsp_proc$ export HEXAGON_ROOT=~/Qualcomm/HEXAGON_Tools <QCS403_WS>/adsp_proc$ export HEXAGON_RTOS_RELEASE=8.2.02 <QCS403_WS>/adsp_proc$ export HEXAGON_Q6VERSION=v66 <QCS403_WS>/adsp_proc$ export HEXAGON_IMAGE_ENTRY=0x8AC00000 <QCS403_WS>/adsp_proc$ env | grep HEXA
Build	<QCS403_WS>/adsp_proc$ python ./build/build.py -c qcs405 -o all -f ADSP,USES_VT_AUDIO
Clean	<QCS403_WS>/adsp_proc$ python ./build/build.py -c qcs405 -o clean -f ADSP,USES_VT_AUDIO
Build log	scons: done reading SConscript files. scons: Building targets ... scons: `.' is up to date. scons: done building targets. Compilation SUCCESS
Build log file	<QCS403_WS>/adsp_proc/build/ms/build-log.txt
Build result file	<QCS403_WS>/adsp_proc/build/ms/adsp.bin

Section 5.2.2

Build CDSP

Build CDSP

	CDSP build commands and prerequisite for QCS40x
Prerequisite	✓ Hexagon Tool v8.2.02 installation
Build setup	✓ HEXAGON_ROOT depends on where the tools are installed <QCS40x_WS>$ cd cdsp_proc <QCS40x_WS>/cdsp_proc$ export HEXAGON_ROOT=~/Qualcomm/HEXAGON_Tools <QCS40x_WS>/cdsp_proc$ export HEXAGON_RTOS_RELEASE=8.2.02 <QCS40x_WS>/cdsp_proc$ export HEXAGON_Q6VERSION=v66 <QCS40x_WS>/cdsp_proc$ export HEXAGON_IMAGE_ENTRY=0x8AC00000 <QCS40x_WS>/cdsp_proc$ env | grep HEXA
Build	<QCS40x_WS>/cdsp_proc$ python ./build/build.py -c qcs405 -o all -f CDSP
Clean	<QCS40x_WS>/cdsp_proc$ python ./build/build.py -c qcs405 -o clean -f CDSP
Build log	scons: done reading SConscript files. scons: Building targets ... scons: `.' is up to date. scons: done building targets. Compilation SUCCESS
Build log file	<QCS40x_WS>/cdsp_proc/build/ms/build-log.txt
Build result file	<QCS40x_WS>/cdsp_proc/build/ms/cdsp.bin

Section 5.2.3

Build RPM

Build RPM

	RPM build commands and prerequisite for QCS40x
Prerequisite	✓ SD LLVM Toolchain v3.9.2
Build setup	<QCS40x_WS>$ cd rpm_proc/build
Build	<QCS40x_WS>/rpm_proc/build$ ./build_405.sh
Clean	<QCS40x_WS>/rpm_proc/build$ ./build_405.sh -c
Build log	scons: done building targets. ========== SCons build summary ========== ** Build time... Build start : Thu Mar 28 14:48:31 2019 Build end : Thu Mar 28 14:52:10 2019 Elapsed time : 0:03:40 Cleaning up any old build artifacts Copying build artifacts to ../core/bsp/rpm/build/405/ Build started: 2019-03-28 14:34:33.741437 Build ended: 2019-03-28 14:52:10.906696 Elapsed time: 0:17:37.165259
Build log file	<QCS40x_WS>/rpm_proc/build/build-log.txt
Build result file	<QCS40x_WS>/rpm_proc/build/ms/bin/AAAAANAZR/rpm.mbn

Section 5.2.4

Build Bootloader

Build Bootloader – Check Configuration Data Table

- In the following case, you should use hardcoded CDT:
  - Target device does not have a storage for CDT
  - The storage for CDT does not have CDT information
- SBRD with QCS405 based SOM stores CDT into eMMC
- SSRD with QCS403 based SOM stores CDT into EPROM
- For more details about using hardcoded CDT, refer to *[80-YB420-3] Bootloader* training slide

Build Bootloader

	Bootloader build commands and prerequisite for QCS40x
Prerequisite	✓ SD LLVM Toolchain v4.0.2 ✓ GCC Linaro Toolchain v4.8
Build setup	<QCS40x_WS>$ cd boot_images/QcomPkg/Qcs405Pkg
Build	<QCS40x_WS>/boot_images/QcomPkg/Qcs405Pkg$ python ../buildit.py --variant LA -r RELEASE -t Qcs405Pkg,QcomToolsPkg
Clean	<QCS40x_WS>/boot_images/QcomPkg/Qcs405Pkg$ python ../buildit.py --variant LA -r RELEASE -t Qcs405Pkg,QcomToolsPkg -b clean
Build log	******* [buildit.py] Completed building Target: QcomToolsPkg Subtarget: ---- Variant: -- Single: -- Release: RELEASE [buildit.py] Successfully built following flavors: ## Target: Qcs405Pkg Subtarget: ---- Varaint: LA Single: -- release: RELEASE Target: QcomToolsPkg Subtarget: ---- Variant: -- Single: -- release: RELEASE [buildit.py] Now exiting...
Build log file	<QCS40x_WS>/boot_images/QcomPkg/Qcs405Pkg/LA/build_Core.log <QCS40x_WS>/boot_images/QcomPkg/Qcs405Pkg/LA/build_Loader.log
Build result	<QCS40x_WS>/boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/xbl_nand.elf <QCS40x_WS>/boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/pmic.elf

Section 5.2.5

Build TrustZone

Build TrustZone – Modify ‘build_config_deploy.xml’

- Before TZ build, the build configuration should be corrected
- File : <QCS40x_WS>/trustzone_images/build/ms/build_config_deploy.xml
  - Line #51
  - Parameter name: “PYTHON_PATH”
  - Default value: /pkg/qct/software/python/2.7/bin
- Modify with your python2.7 binary installed folder path
  - Command to check python2.7 binary installed path

    $ whereis python2.7
- For example, if the installed folder is /usr/bin

BEFORE	AFTER
/pkg/qct/software/python/2.7/bin	/usr/bin

<QCS40x_WS>\trustzone_images\build\ms\build_config_deploy.xml

```
<build>
 <branch aliases="tz51,5.1,tz.xf.5.1" default_target="qcs405 … >
  <chipset build_asic="405" build_id="OAPAANAA“… >
   <environment>
…
…
      <var description="Bin directory …" … name="PYTHON_PATH" value="/usr/bin"/>
…
…
```

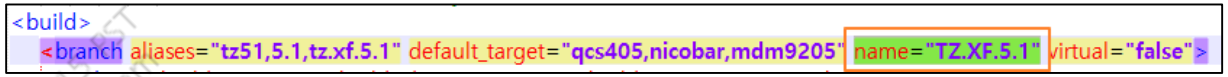
```
intermediate="false" name="CRMPERL" value="/usr/bin"/>
 name="LLVMCLANG" value="/pkg/qct/software/llvm/release/arm/4.
" intermediate="false" name="CCBIN" value="/pkg/qct/software/l
ct/software/llvm/release/arm/4.0.11/aarch64-none-elf/libc"/>
E" value="/pkg/asw/compilers/arm/ADS1.2"/>
LPATH" value="/pkg/qct/software/llvm/release/arm/4.0.11/tools/
" value="/pkg/qct/software/bullseyecoverage/BullseyeCoverage-8
" intermediate="false" name="PYTHON_PATH" value="/usr/bin"/>
ediate="false" name="ARMTOOLS" value="ARMCT6"/>
```

Build TrustZone – Check build branch name in ‘build_config_deploy.xml’

- TZ build command requires the build branch name argument
- The build branch name argument can be found in ‘build_config_deploy.xml’ file
- File : <QCS40x_WS>/trustzone_images/build/ms/build_config_deploy.xml
  - <Build>
    - <Branch … name=“build_branch_name” … >
- e.g. For the r1.0_00007.0 software, the build branch name is **TZ.XF.5.1**

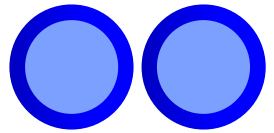
<QCS40x_WS>\trustzone_images\build\ms\build_config_deploy.xml

```
<build>
   <branch aliases="tz51,5.1,tz.xf.5.1" default_target="qcs405,nicobar,mdm9205"
name="TZ.XF.5.1" virtual="false">
…
…
```



Build TrustZone

	TZ build commands and prerequisite for QCS40x
Prerequisite	✓ SD LLVM Toolchain v4.0.11 ✓ GCC Linaro Toolchain v4.9
Build setup	<QCS40x_WS>$ cd trustzone_images/build/ms
Build for TZ.XF.5.1.2	<QCS40x_WS>/trustzone_images/build/ms$ python build_all.py -b TZ.XF.5.1 CHIPSET=qcs405 config=build_config_deploy.xml --recompile ✓ TZ.XF.5.1 is the build branch name
Clean for TZ.XF.5.1.2	<QCS40x_WS>/trustzone_images/build/ms$ python build_all.py -b TZ.XF.5.1 CHIPSET=qcs405 config=build_config_deploy.xml --clean ✓ TZ.XF.5.1 is the build branch name
Build log	Start Time = Thu Mar 28 15:28:13 KST 2019 - End Time = Thu Mar 28 15:28:59 KST 2019 Elapsed Time = 46 seconds WARNING:build_all:Extra artifacts: devcfg.mbn, gpsample.mbn, seccamdemo.mbn, seccamdemo64.mbn, secure_ui_sample.mbn, secure_ui_sample64.mbn, secureindicator.mbn, smplap32.mbn, smplap64.mbn, smplcert.mbn, smplserv.mbn, teetest.mbn INFO:build_all:Build completed successfully INFO:build_all:Total elapsed time: 1 minutes, 33 seconds
Build log file	<QCS40x_WS>/trustzone_images/build/ms/build-log.txt
Build result	<QCS40x_WS>/trustzone_images/build/ms/bin/OAPAANAA/devcfg.mbn <QCS40x_WS>/trustzone_images/build/ms/bin/OAPAANAA/tz.mbn



Section 5.3

Generate Final Image

Generate Final Image

- Before ‘Generate Final Image’, the apps image location should be modified according to APPS_PROC built variant
- The apps image location information is in ‘<QCS40x>/contents.xml‘ file
- Below is the apps image location for each build variant

Build Variant	**APPS Image Location** <apps_image_base>: apps_proc/poky/build/tmp-glibc/deploy/images	**<QCS40x>/contents.xml**
QCS403 (debug)	<apps_image_base>/qcs403-som2-qsap/	Line 94: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/</file_path> Line 110: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/</file_path> Line 114: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/</file_path> Line 118: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/</file_path>
QCS403 (perf)	<apps_image_base>/qcs403-som2-qsap-perf/	Line 94: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap-perf/</file_path> Line 110: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap-perf/</file_path> Line 114: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap-perf/</file_path> Line 118: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap-perf/</file_path>
QCS405 (debug)	<apps_image_base>/qcs405-som1-qsap/	Line 127: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 131: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 135: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 139: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 143: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 147: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 151: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 155: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</file_path> Line 719: <sparse_dir>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap/</sparse_dir>
QCS405 (perf)	<apps_image_base>/qcs405-som1-qsap-perf/	Line 127: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 131: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 135: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 139: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 143: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 147: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 151: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 155: <file_path>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</file_path> Line 719: <sparse_dir>apps_proc/poky/build/tmp-glibc/deploy/images/qcs405-som1-qsap-perf/</sparse_dir>

Generate Final Image

	Generate Final Image commands for QCS40x
Setup	<QCS40x_WS>$ cd common/build
Generate	<QCS40x_WS>/common/build$ python build.py
Build log	[22:40:43] - update_meta.py: Validating File paths in contents.xml [22:40:43] - ========== [22:40:43] - update_meta.py: Total number of file paths found in contents.xml = 93 [22:40:43] - update_meta.py: file path validation passed. [22:40:43] - ========== CHECK COMPLETE ========== [22:40:43] - update_common_info.py:========== UPDATE COMMON INFO COMPLETE==========
Build log file	<QCS403_WS>/common/build/update_common.log

Generate Final Image

- The following errors may be logged in the QCS405 build log
- This is because we are currently using unsigned images so this error can be ignored for now

Build errors example

…
[31mERROR: Following validations failed for the image:
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/smplap32.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/smplap64.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/securemm.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/cmnlib.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/cmnlib64.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/cmnlib.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/cmnlib64.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/trustzone_images/build/ms/bin/OAPAANAA/keymaster64.mbn is not encrypted
Image /local2/mnt/workspace/QCS40x/r1.0_00007.0/QCS405_WS/wlan_proc/build/ms/bin/QCAHNALAVDL/qdsp6sw.mbn is not encrypted
…

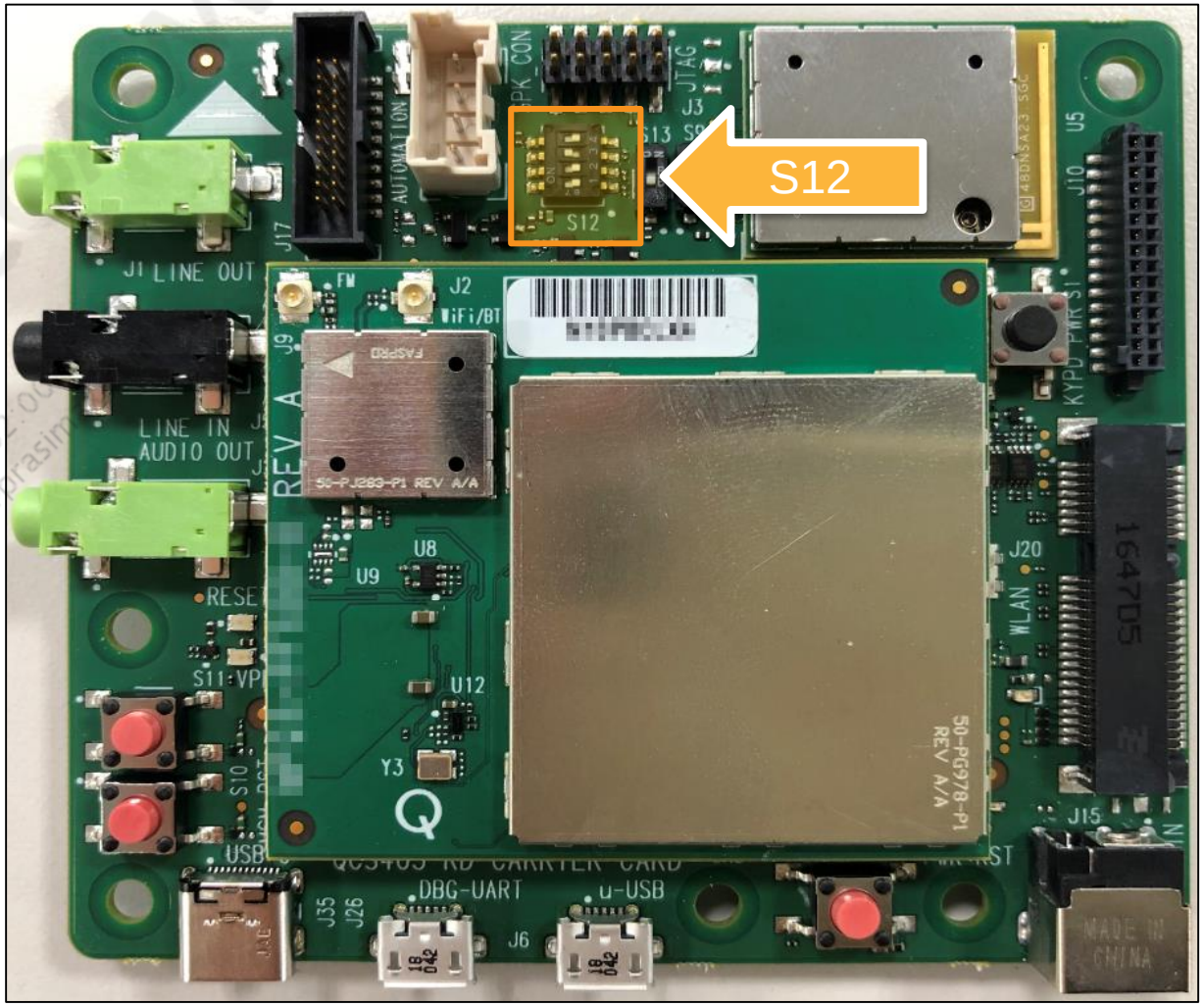
Section 6

Device Boot Configuration for SSRD

Device Boot Configuration for SSRD

- QCS403 has the following boot options using S12 circuit

Boot option	GPIO_45 (PIN #4)	GPIO_49 (PIN #3)	GPIO_57 (PIN #2)	GPIO_56 (PIN #1)
eMMC > SD > HS USB	0	0	0	0
SD > eMMC > EDL	0	0	0	1
eMMC > EDL	0	0	1	0
HS USB	0	0	1	1
NAND > EDL (default)	**0**	**1**	**0**	**0**
Forced USB BOOT	**1**	**x**	**x**	**x**

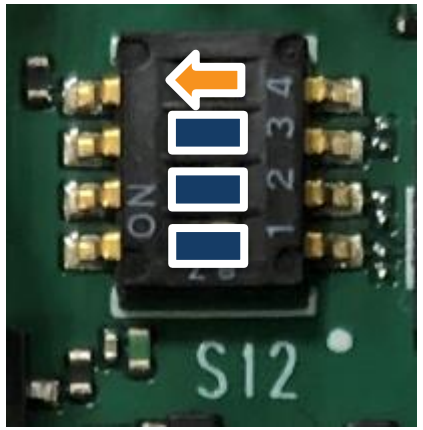


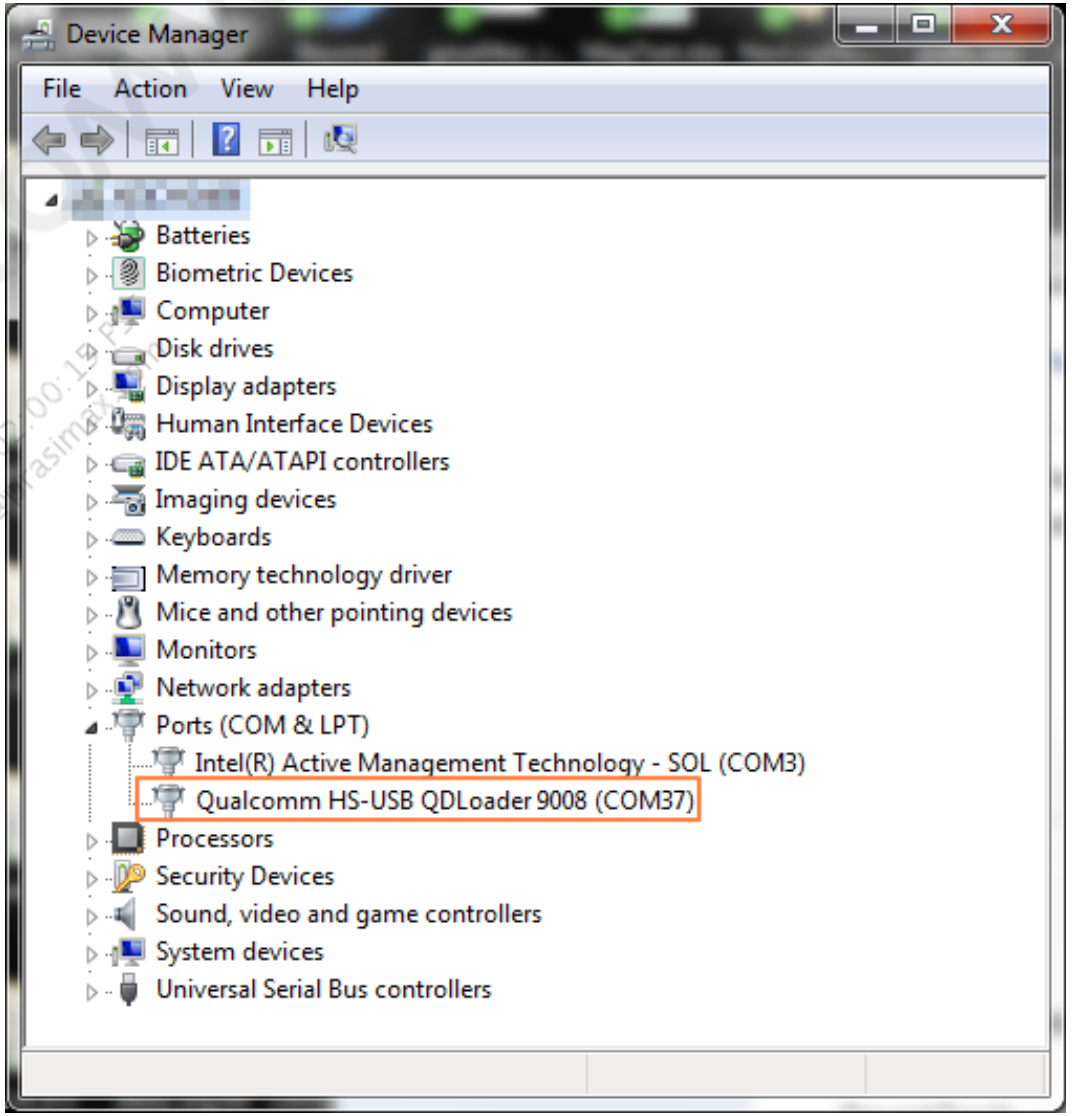
- Below is the pin assignment of GPIOs for S12

Pin #	GPIO
1	GPIO_56 / BOOT_CONFIG[1]
2	GPIO_57 / FAST_BOOT_CONFIG[2]
3	**GPIO_49 / FAST_BOOT_CONFIG[3]**
4	**GPIO_45 / Forced USB BOOT**

Device Boot Configuration for SSRD – EDL mode

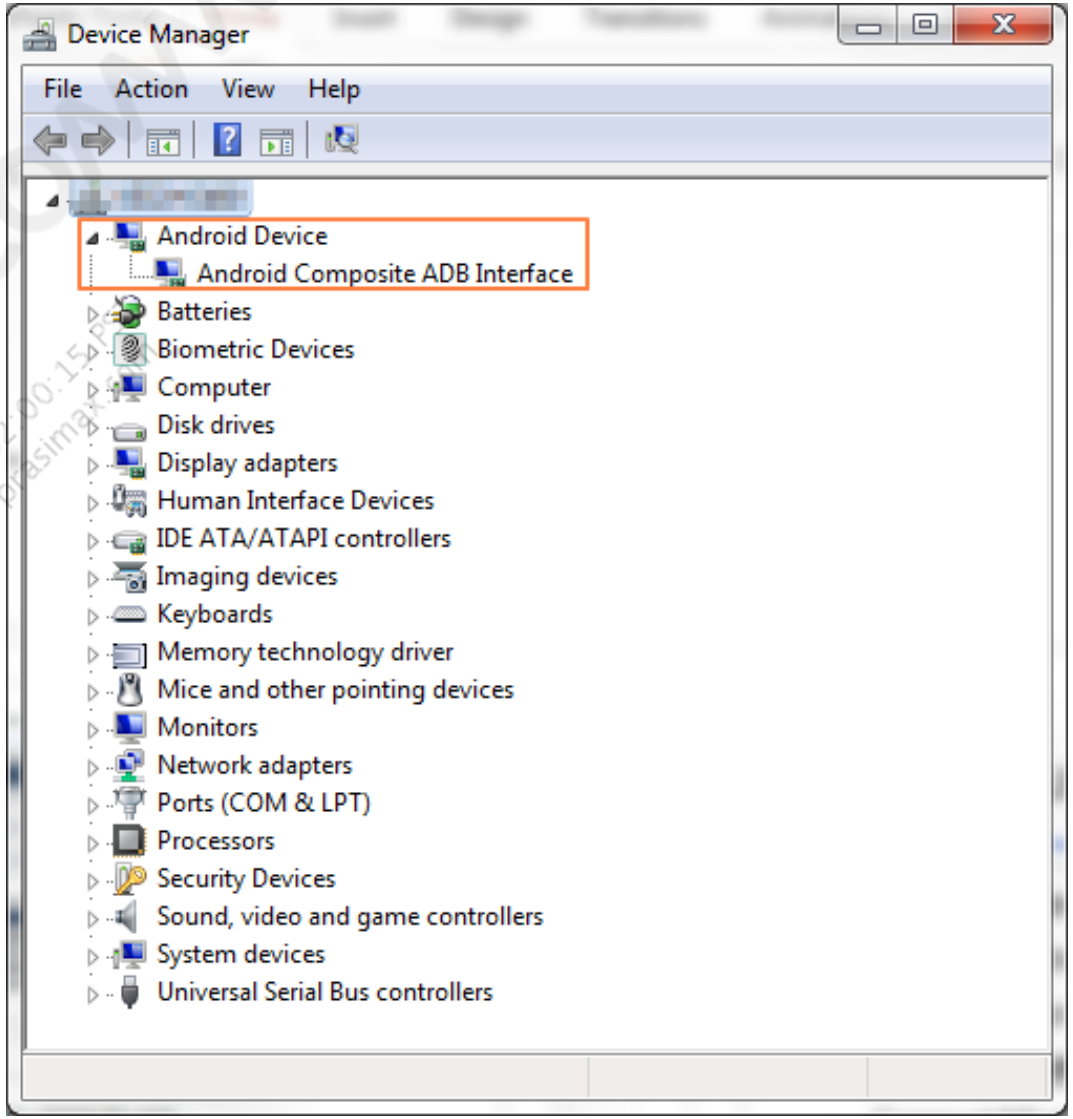
- To configure device to EDL mode, follow the instruction below
  1) Remove Main Power from target board
  2) Turn ON **PIN#4** in S12, Don't care other PINs
  3) Connect Micro USB Cable with Host PC
  4) Restore Main Power to target board
  5) Check Device Manager > QDLoader 9008 port is enumerated





Device Boot Configuration for SSRD – Normal boot mode

- To configure device to operation mode, follow the instruction below
  1) Remove Main Power from target board
  2) Turn ON **PIN#3** and Turn OFF **other PINs** in S12
  3) Connect Micro USB Cable with Host PC
  4) Restore Main Power to target board
  5) Check Device Manager > ADB Interface is enumerated



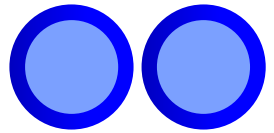
Section 7

Writing Images

Writing Images

- In this chapter, only basic procedures will be explained
  - Writing Images using:
    - Qualcomm Flash Image Loader (QFIL) which is included in QPST
    - Fastboot
  - Packaging Images using QFIL

- For more details, refer to *[80-YB405-129] QCS40x Software User Guide* document

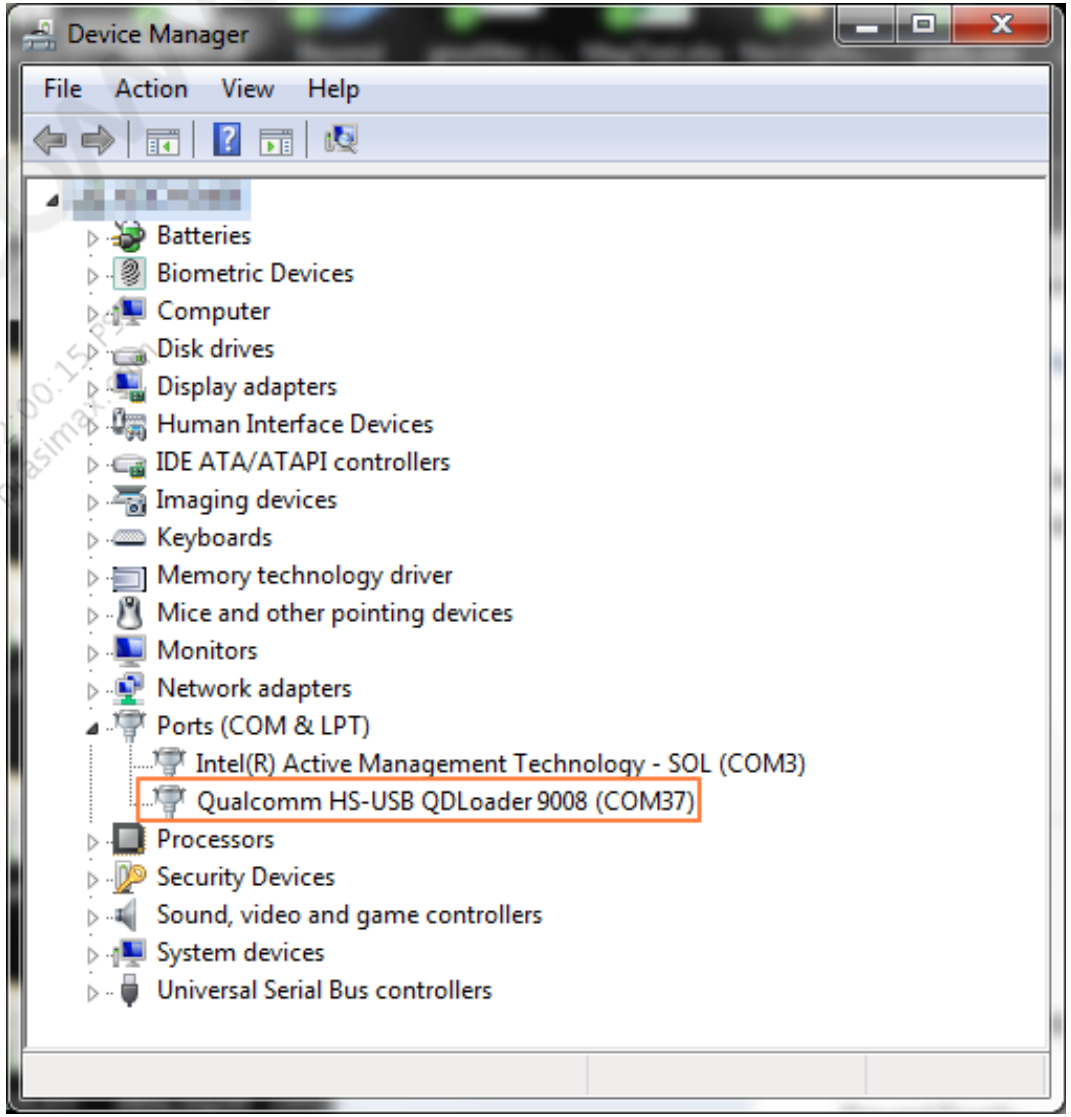


Section 7.1

Writing Images using QFIL

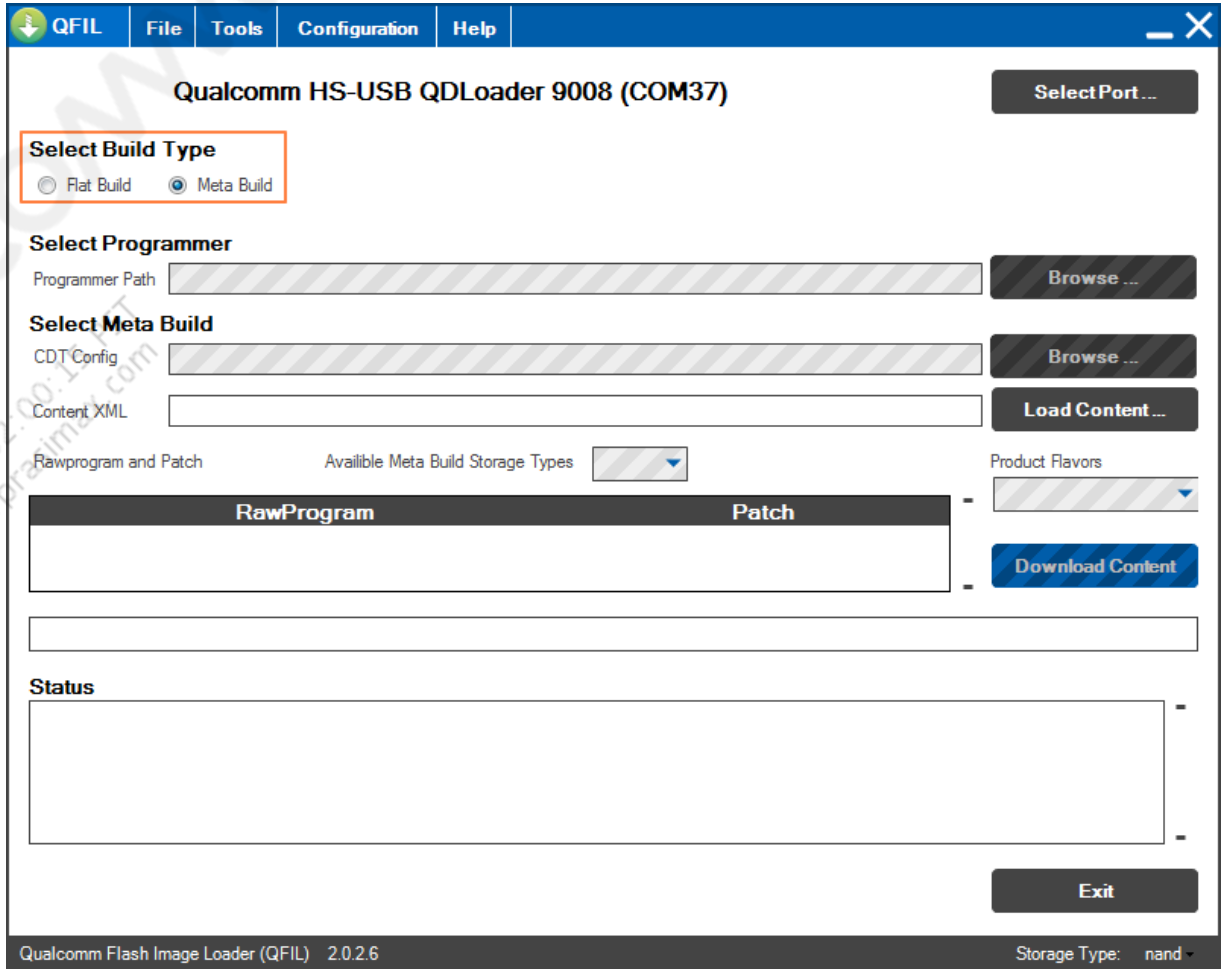
Writing Images using QFIL

- To write images using QFIL, the device must be configured to EDL mode
- Refer to:
  - Device Boot Configuration for SSRD - EDL mode



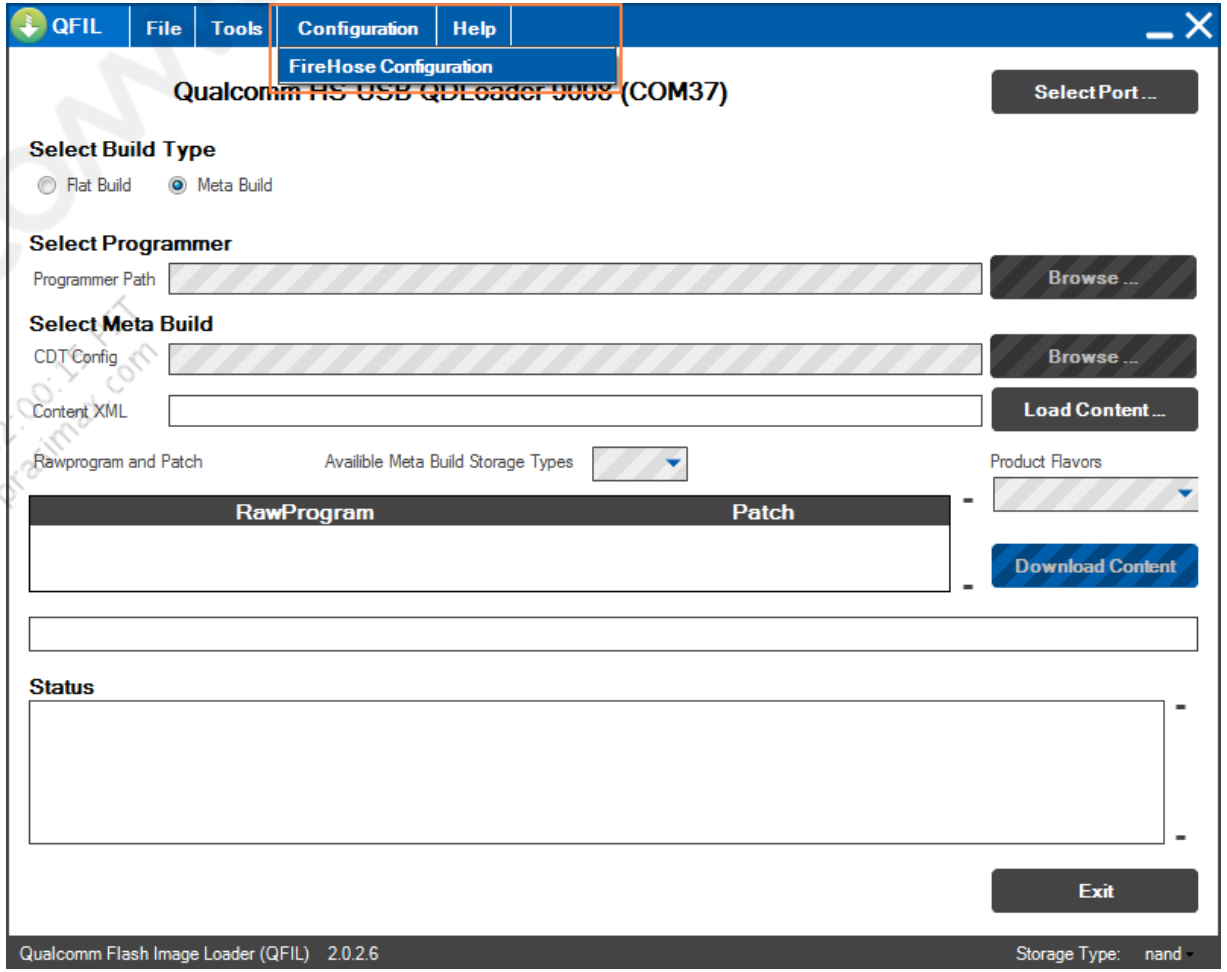
Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’



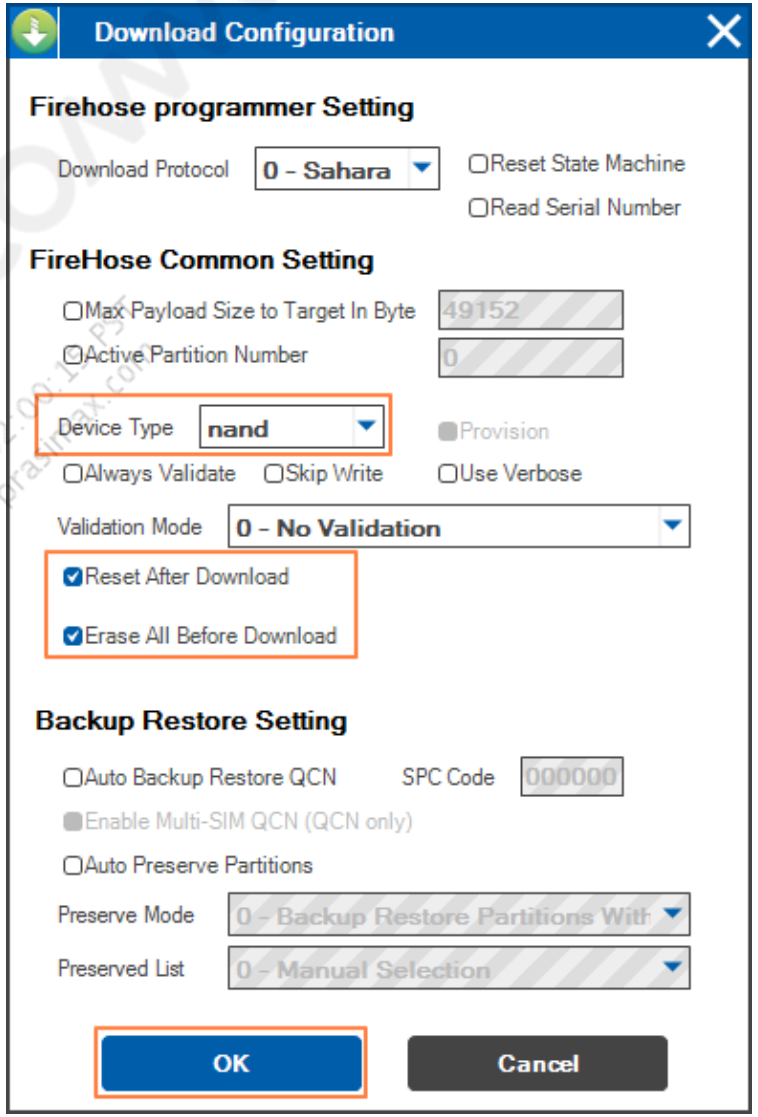
Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)



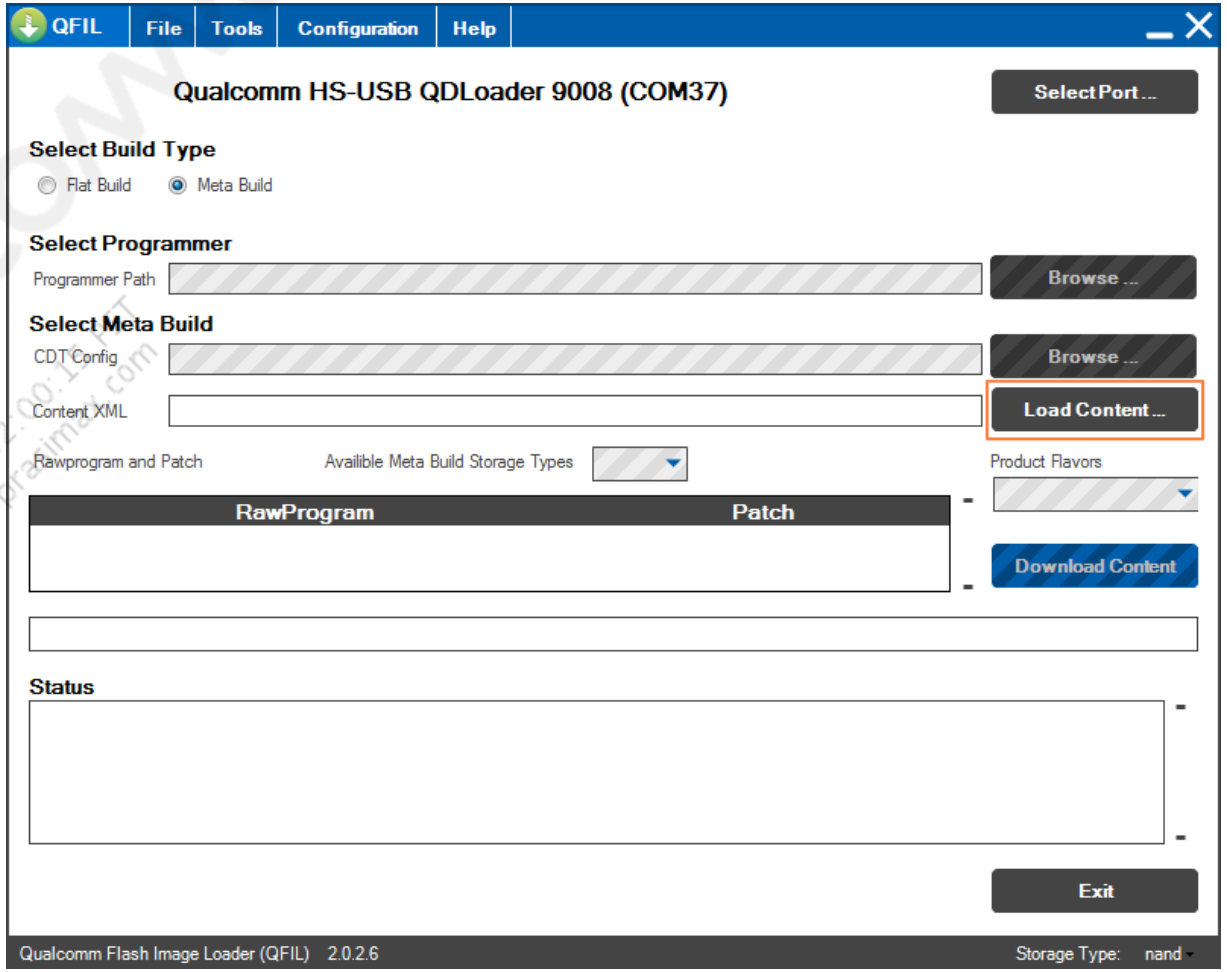
Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON



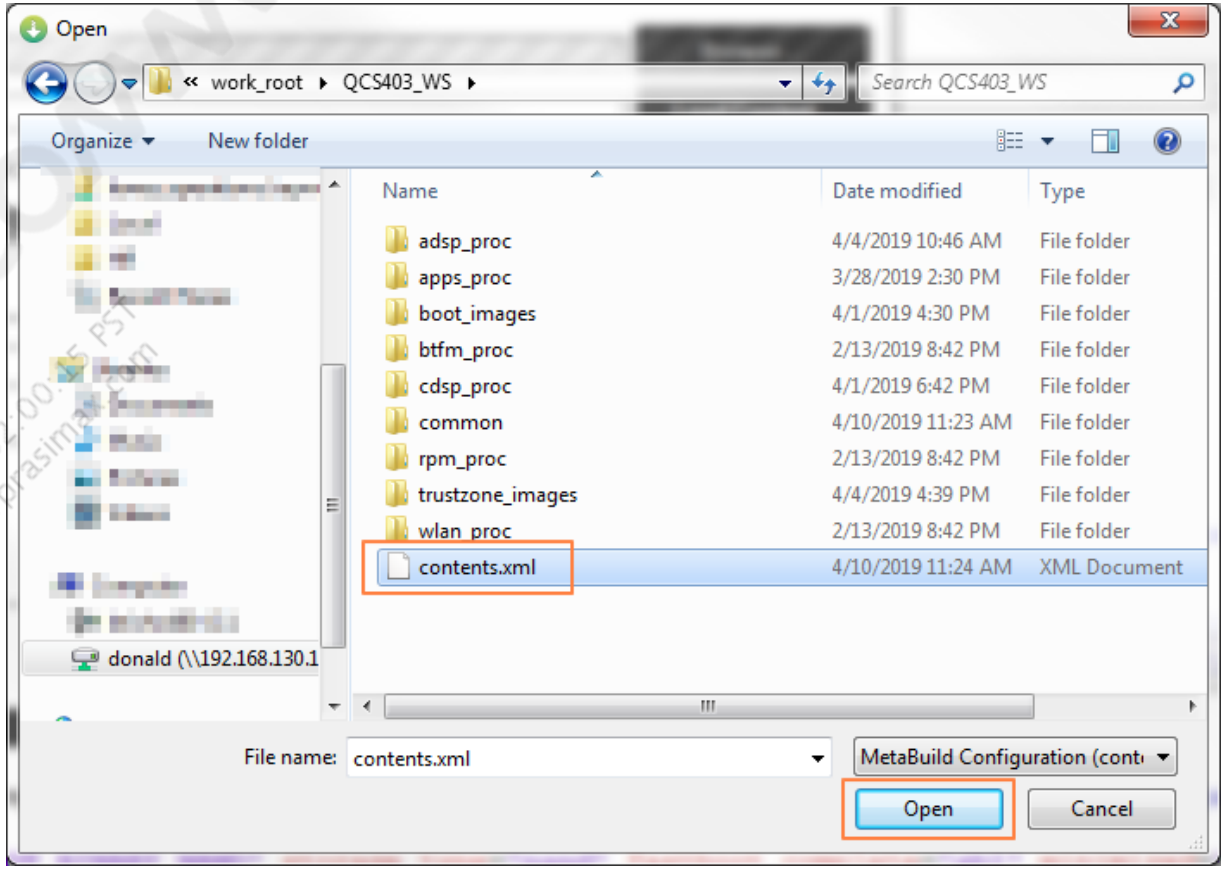
Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load Content…’ to load contents.xml file



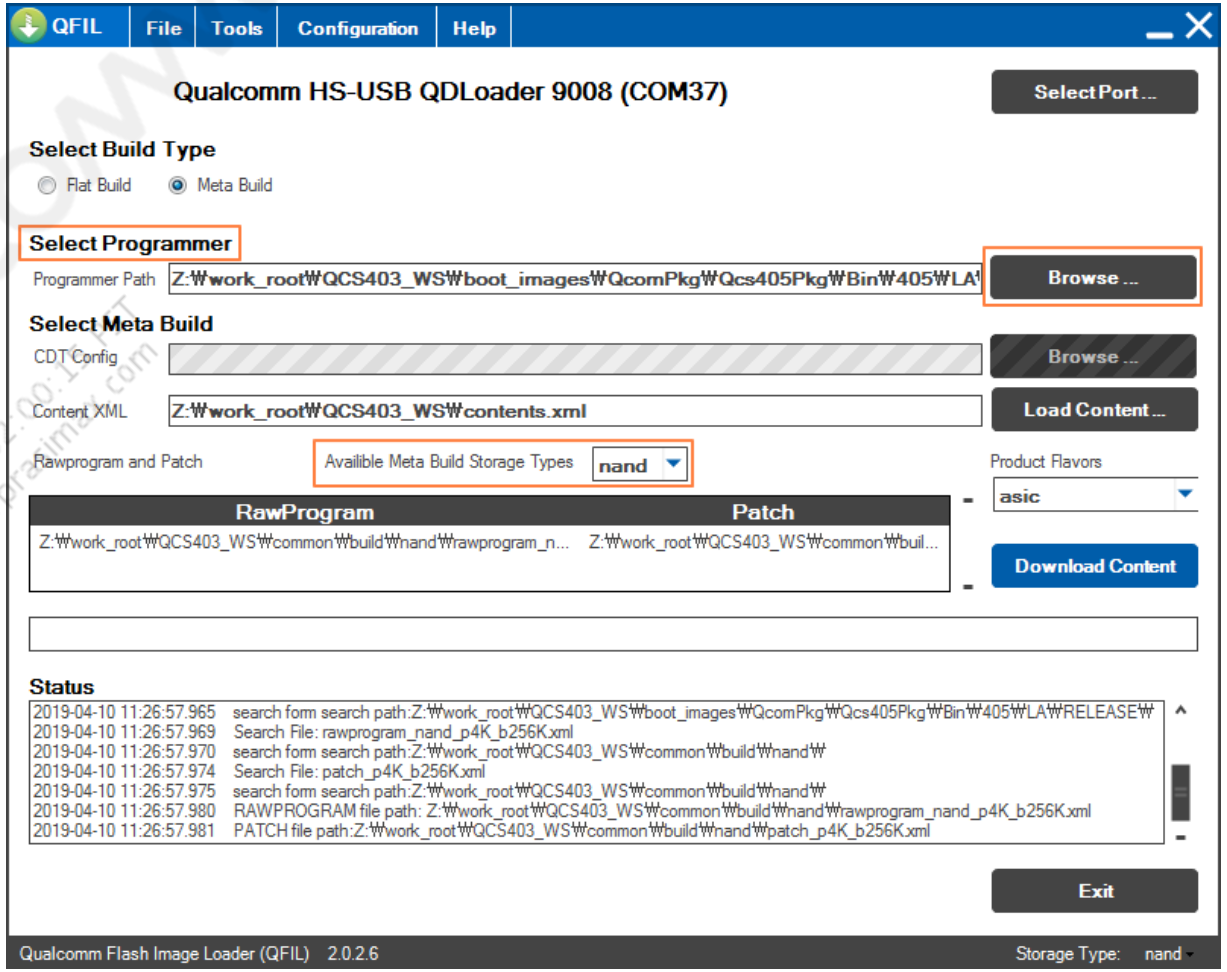
Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load Content…’ to load contents.xml file
- Select <QCS403_WS>/contents.xml file and Click on ‘Open’



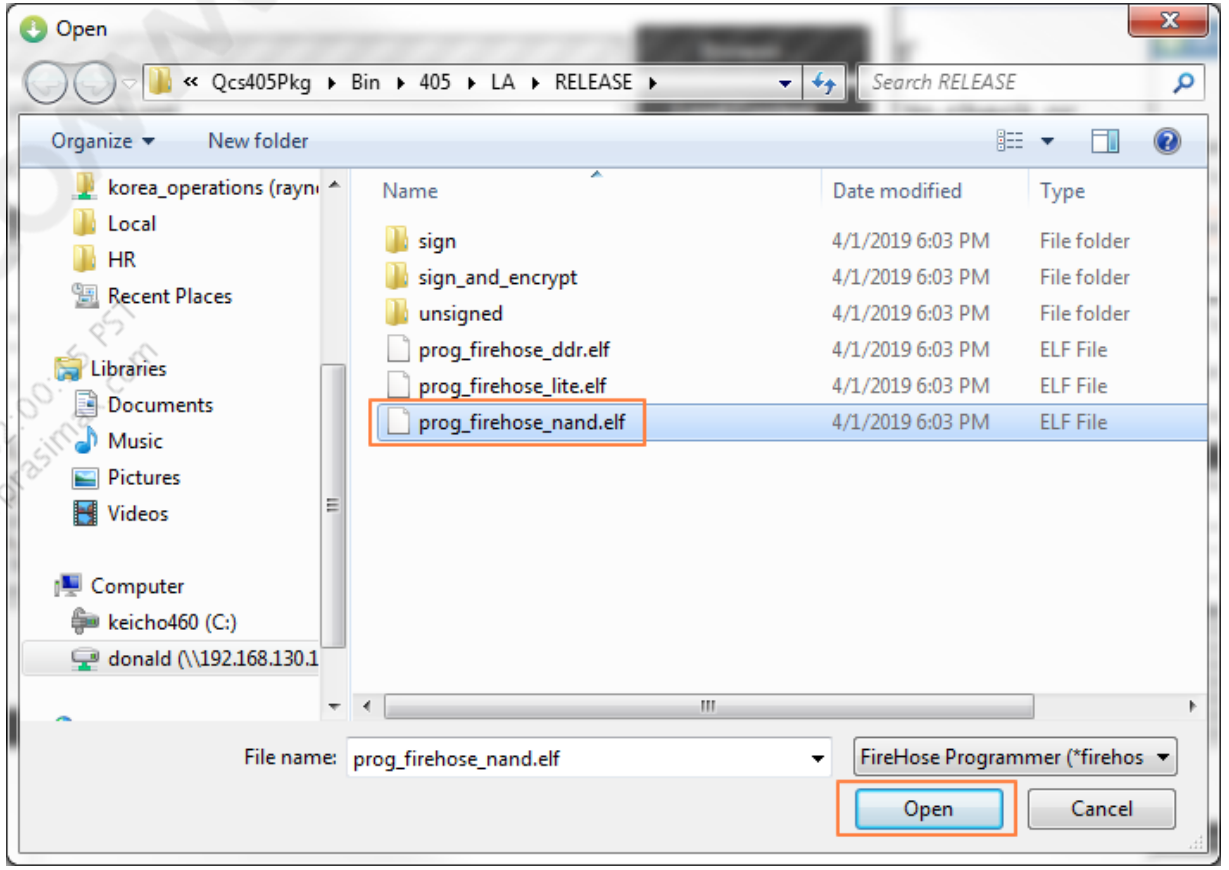
Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load Content…’ to load contents.xml file
- Select <QCS403_WS>/contents.xml file and Click on ‘Open’
- Make sure ‘Available Meta Build Storage Types’ is ‘nand’
- Click on ‘Select Programmer > Browse…’ button



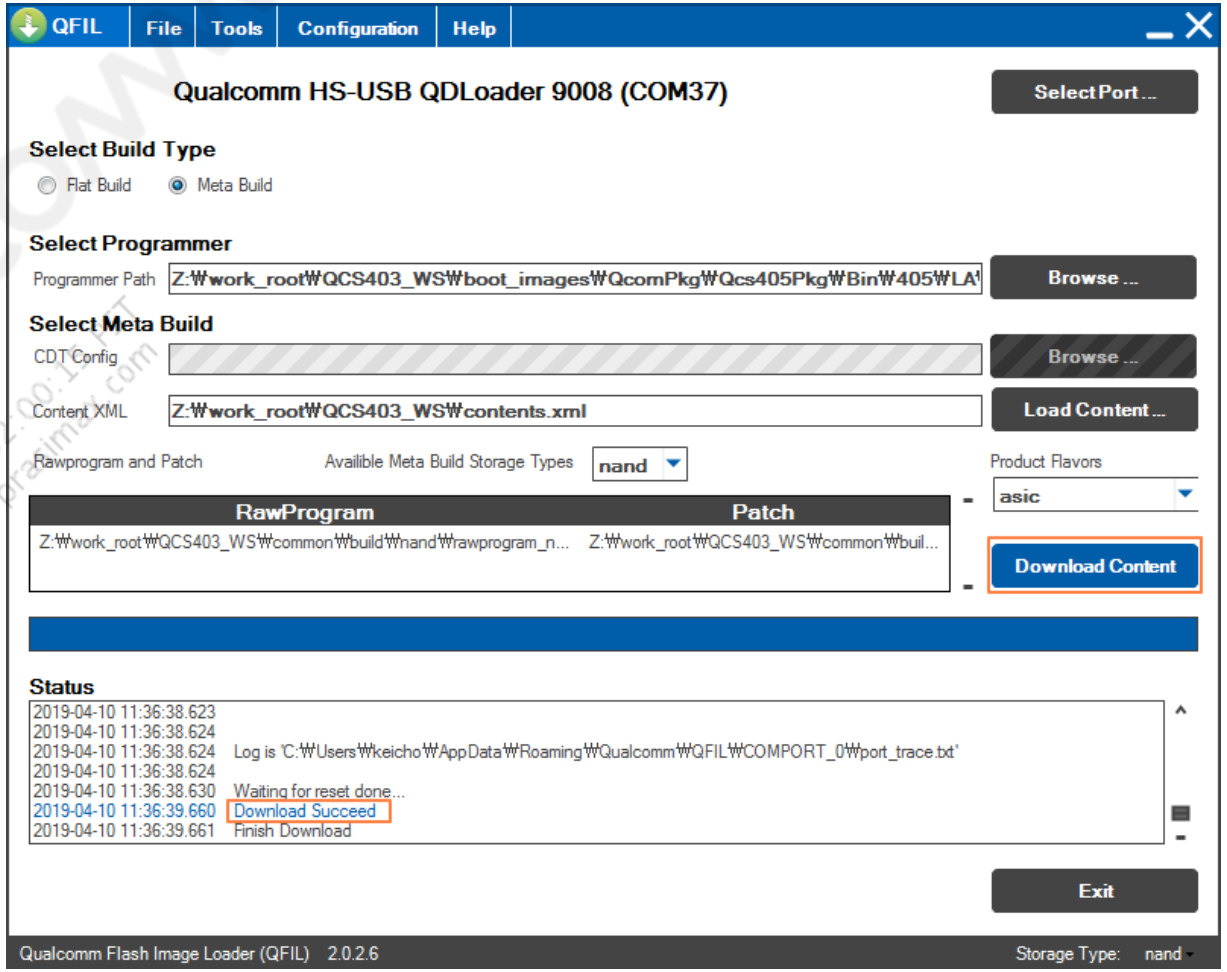
Writing Images using QFIL

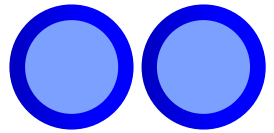
- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load Content…’ to load contents.xml file
- Select <QCS403_WS>/contents.xml file and Click on ‘Open’
- Make sure ‘Available Meta Build Storage Types’ is ‘nand’
- Click on ‘Select Programmer > Browse…’ button
- Select ‘prog_firehose_nand.elf’ file and Click on ‘Open’



Writing Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load Content…’ to load contents.xml file
- Select <QCS403_WS>/contents.xml file and Click on ‘Open’
- Make sure ‘Available Meta Build Storage Types’ is ‘nand’
- Click on ‘Select Programmer > Browse…’ button
- Select ‘prog_firehose_nand.elf’ file and Click on ‘Open’
- Click on ‘Download Content’ button
- Make sure “Download Succeed” message
- DONE !



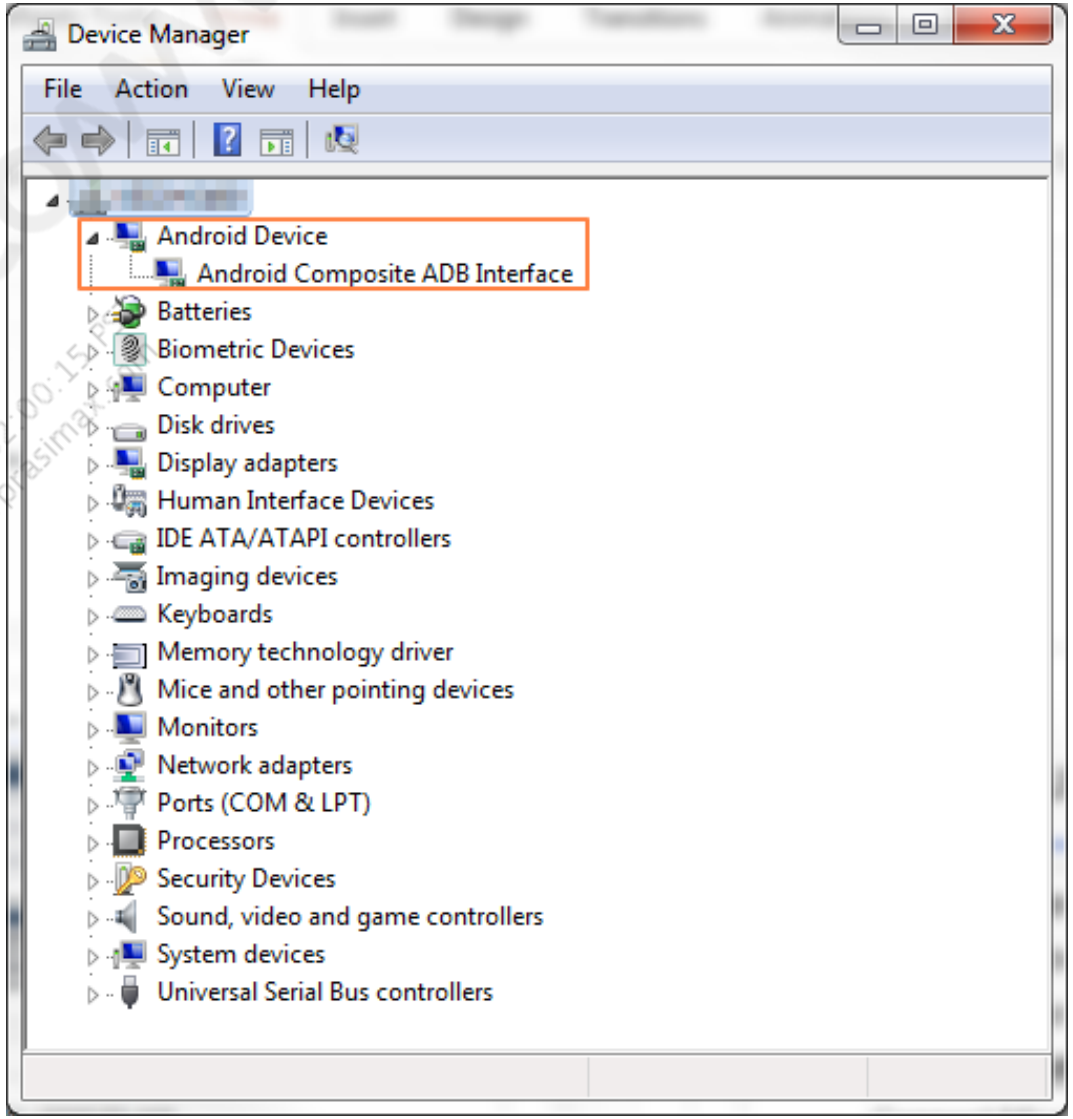


Section 7.2

Writing Images using fastboot

Writing Images using fastboot – Windows

- To write images using fastboot, the device must be configured to normal boot mode
- Refer to:
  - Device Boot Configuration for SSRD – Normal boot mode

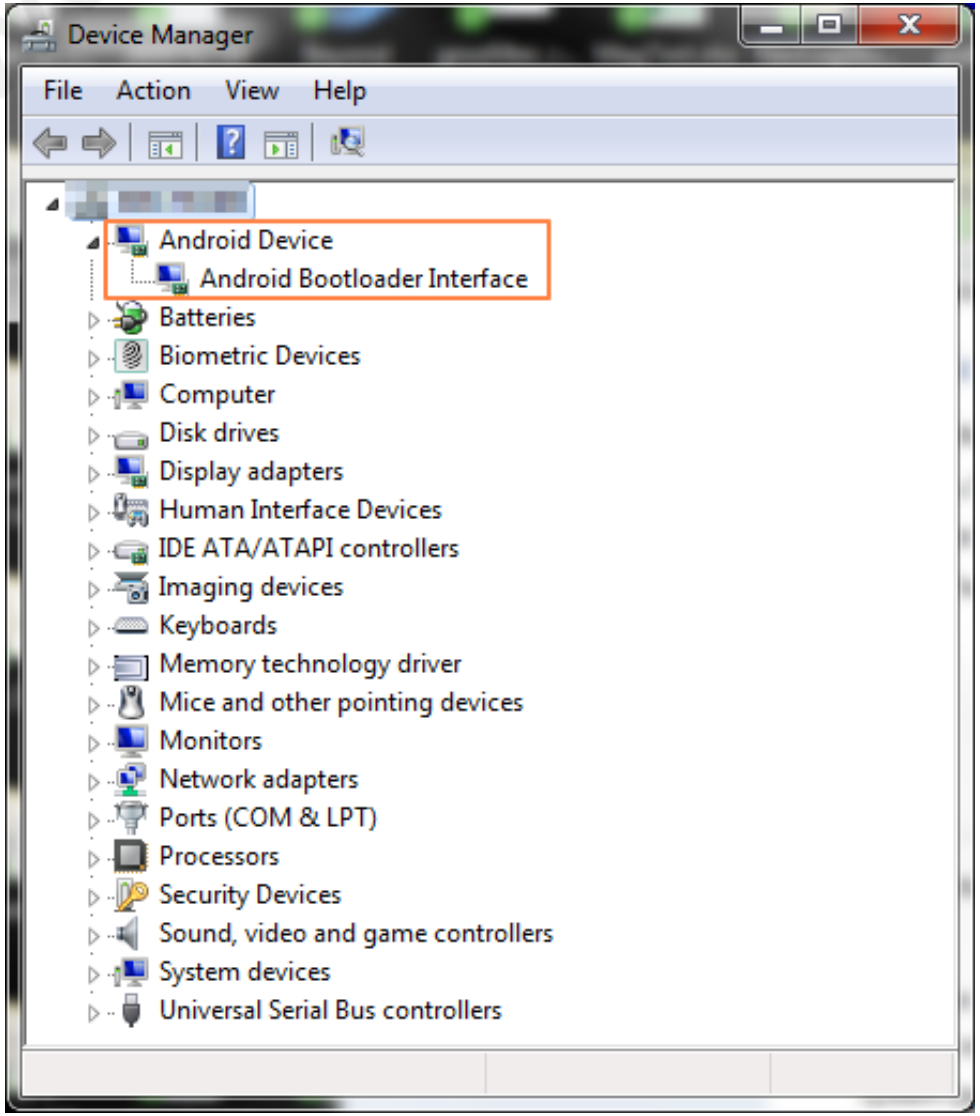


Writing Images using fastboot – Windows

- To write images using fastboot, the device must be configured to normal boot mode
- Refer to:
  - Device Boot Configuration for SSRD – Normal boot mode

- After the device boots into adb mode, reboot device from adb mode to fastboot mode

Command	Description
> adb devices	Check adb interface
> adb reboot bootloader	Switch to fastboot mode
> fastboot devices	Check fastboot interface



Writing Images using fastboot – Windows

	Commands for Writing Images with python script
Prerequisite	✓ Python 2.7
Setup	> pushd <QCS403_WS>/common/build
Run writing script	<QCS403_WS>/common/build> python fastboot_complete.py --st=nand
Log	… == fastboot_complete.py: Processing input arguments == … fastboot.exe flash abl Z:\work_root\QCS403_WS\apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/abl-squashfs.elf fastboot.exe flash boot Z:\work_root\QCS403_WS\apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/qcs40x-boot.img fastboot.exe flash logfs Z:\work_root\QCS403_WS\boot_images/QcomPkg/Tools/binaries/logfs_ufs_8mb.bin fastboot.exe flash modem Z:\work_root\QCS403_WS\common/build/nand/NON-HLOS.ubi fastboot.exe flash pmic Z:\work_root\QCS403_WS\boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/pmic.elf fastboot.exe flash rpm Z:\work_root\QCS403_WS\rpm_proc/build/ms/bin/AAAAANAZR/rpm.mbn fastboot.exe flash sbl Z:\work_root\QCS403_WS\boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/xbl_nand.elf fastboot.exe flash system Z:\work_root\QCS403_WS\apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/qcs40x-squashfs-sysfs.ubi fastboot.exe flash tz Z:\work_root\QCS403_WS\trustzone_images/build/ms/bin/OAPAANAA/tz.mbn fastboot.exe flash tz_devcfg Z:\work_root\QCS403_WS\trustzone_images/build/ms/bin/OAPAANAA/devcfg.mbn … == FLASHING SUCCESSFUL! == fastboot_complete.py: Loading complete.

Writing Images using fastboot – Linux

- After the device boots into adb mode, reboot device from adb mode to fastboot mode

Command	Description
$ lsusb	Check connected USB devices
$ sudo adb start-server	Start adb server with sudo permission
$ adb devices	Check adb interface
$ adb reboot bootloader	Switch to fastboot mode
$ sudo fastboot devices	Check fastboot interface

- If 'no permissions' error has occurred on adb or fastboot then USE 'sudo' command
- OR, add rules for the two USB devices to '51-android.rules' file
  - File path: /etc/udev/rules.d/51-android.rules
  - Add rules and Save the file:

```
# For QUALCOMM USB device
SUBSYSTEM="usb",ATTR{idVendor}=="05c6",MODE="0666",GROUP="plugdev"
# For ANDROID USB device
SUBSYSTEM="usb",ATTR{idVendor}=="18d1",MODE="0666",GROUP="plugdev"
```

  - Restart service: $ sudo service udev restart

```
donald@ubuntu:~$ lsusb
Bus 001 Device 002: ID 05c6:901d Qualcomm, Inc.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
donald@ubuntu:~$
donald@ubuntu:~$ sudo adb start-server
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
donald@ubuntu:~$
donald@ubuntu:~$ adb devices
List of devices attached
8bbd0412        device

donald@ubuntu:~$
donald@ubuntu:~$ adb reboot bootloader
donald@ubuntu:~$ fastboot devices
no permissions  fastboot
donald@ubuntu:~$ sudo fastboot devices
8bbd0412        fastboot
donald@ubuntu:~$
```

Writing Images using fastboot – Linux

	Commands for Writing Images with python script
Prerequisite	✓ Python 2.7
Setup	<QCS403_WS>$ cd common/build
Run writing script	<QCS403_WS>/common/build$ sudo python fastboot_complete.py --st=nand
Log	… ========== fastboot_complete.py: Processing input arguments ========== … fastboot flash abl /home/donald/work_root/QCS403_WS/apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/abl-squashfs.elf fastboot flash boot /home/donald/work_root/QCS403_WS/apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/qcs40x-boot.img fastboot flash logfs /home/donald/work_root/QCS403_WS/boot_images/QcomPkg/Tools/binaries/logfs_ufs_8mb.bin fastboot flash modem /home/donald/work_root/QCS403_WS/common/build/nand/NON-HLOS.ubi fastboot flash pmic /home/donald/work_root/QCS403_WS/boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/pmic.elf fastboot flash rpm /home/donald/work_root/QCS403_WS/rpm_proc/build/ms/bin/AAAAANAZR/rpm.mbn fastboot flash sbl /home/donald/work_root/QCS403_WS/boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/xbl_nand.elf fastboot flash system /home/donald/work_root/QCS403_WS/apps_proc/poky/build/tmp-glibc/deploy/images/qcs403-som2-qsap/qcs40x-squashfs-sysfs.ubi fastboot flash tz /home/donald/work_root/QCS403_WS/trustzone_images/build/ms/bin/OAPAANAA/tz.mbn fastboot flash tz_devcfg /home/donald/work_root/QCS403_WS/trustzone_images/build/ms/bin/OAPAANAA/devcfg.mbn … ========== FLASHING SUCCESSFUL! ========== fastboot_complete.py: Loading complete.

Writing Images using fastboot

- Each binary can also be flashed using the following fastboot command

SW Module	Fastboot Command	Image File Location
APPS_PROC	> fastboot flash abl abl-squashfs.elf > fastboot flash boot qcs40x-boot.img > fastboot flash system qcs40x-squashfs-sysfs.ubi	<QCS403_WS>/apps_proc/poky/build/tmp-glibc/deploy/images/<build_variant>/ e.g. <build_variant> is qcs403-som2-qsap or qcs403-som2-qsap-perf
RPM_PROC	> fastboot flash rpm rpm.mbn	<QCS403_WS>/rpm_proc/build/ms/bin/AAAAANAZR/
ADSP_PROC CDSP_PROC WLAN_PROC BTFM_PROC	> fastboot flash modem NON-HLOS.ubi	<QCS403_WS>/common/build/nand/
BOOT_IMAGES	> fastboot flash sbl xbl_nand.elf > fastboot flash pmic pmic.elf	<QCS403_WS>/boot_images/QcomPkg/Qcs405Pkg/Bin/405/LA/RELEASE/
	> fastboot flash logfs logfs_ufs_8mb.bin	<QCS403_WS>/boot_images/QcomPkg/Tools/binaries/
TRUSTZONE	> fastboot flash tz tz.mbn > fastboot flash devcfg devcfg.mbn	<QCS403_WS>/trustzone_images/build/ms/bin/OAPAANAA/

Section 7.3

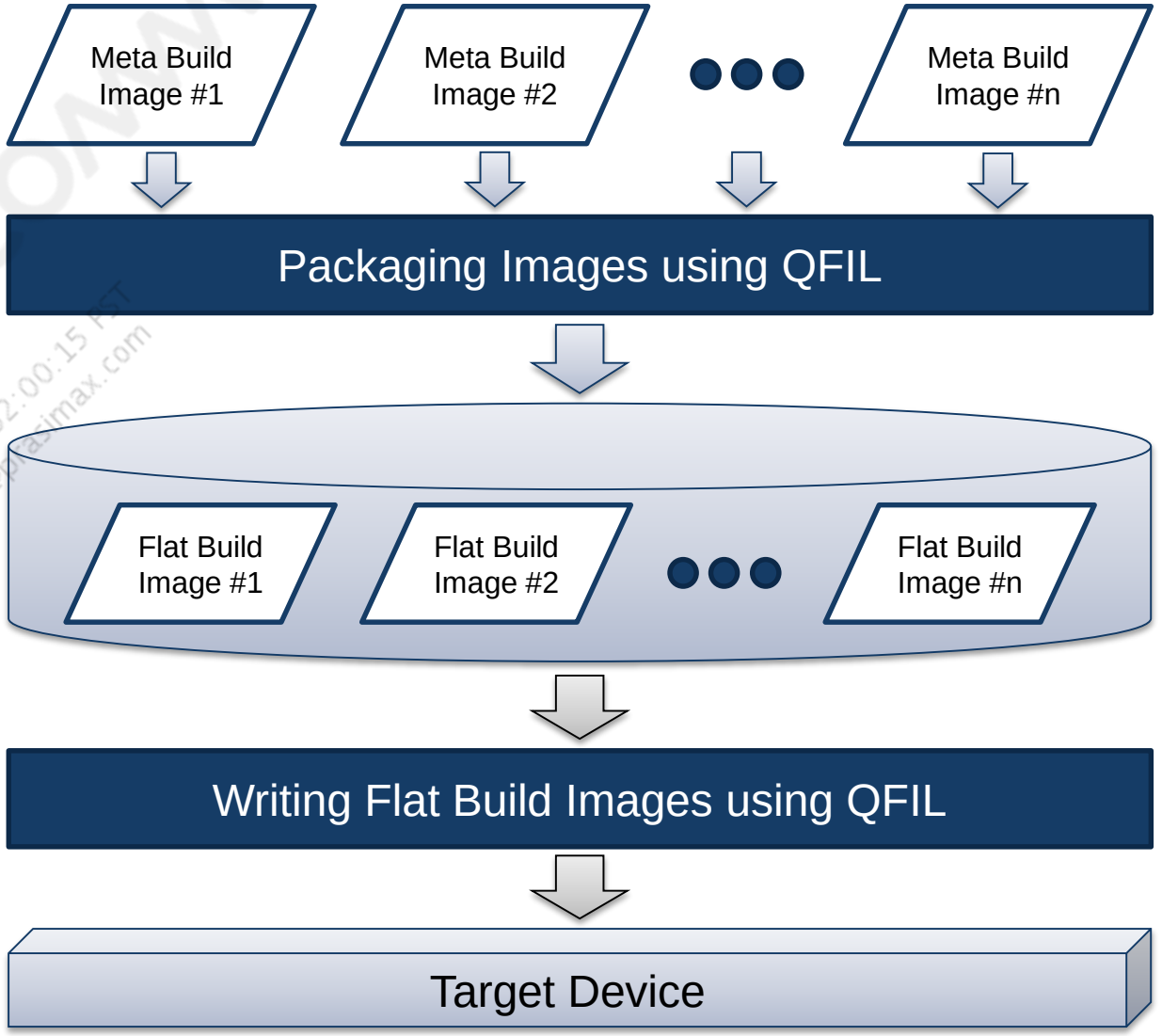
Packaging Images

Packaging Images

- Once build process is completed, images are located in multiple folders
- Packaging Images is the process of collecting these images in one folder
- This makes it easier to deploy to a machine other than the build machine

- In this chapter, following process will be explained
  - How to Package Images using QFIL
  - How to write the result of Packaging Images using QFIL

- In QFIL, Packaging Images process is called ‘Flat Meta Build’
- The result of Packaging Images is also called ‘Flat Build Images’

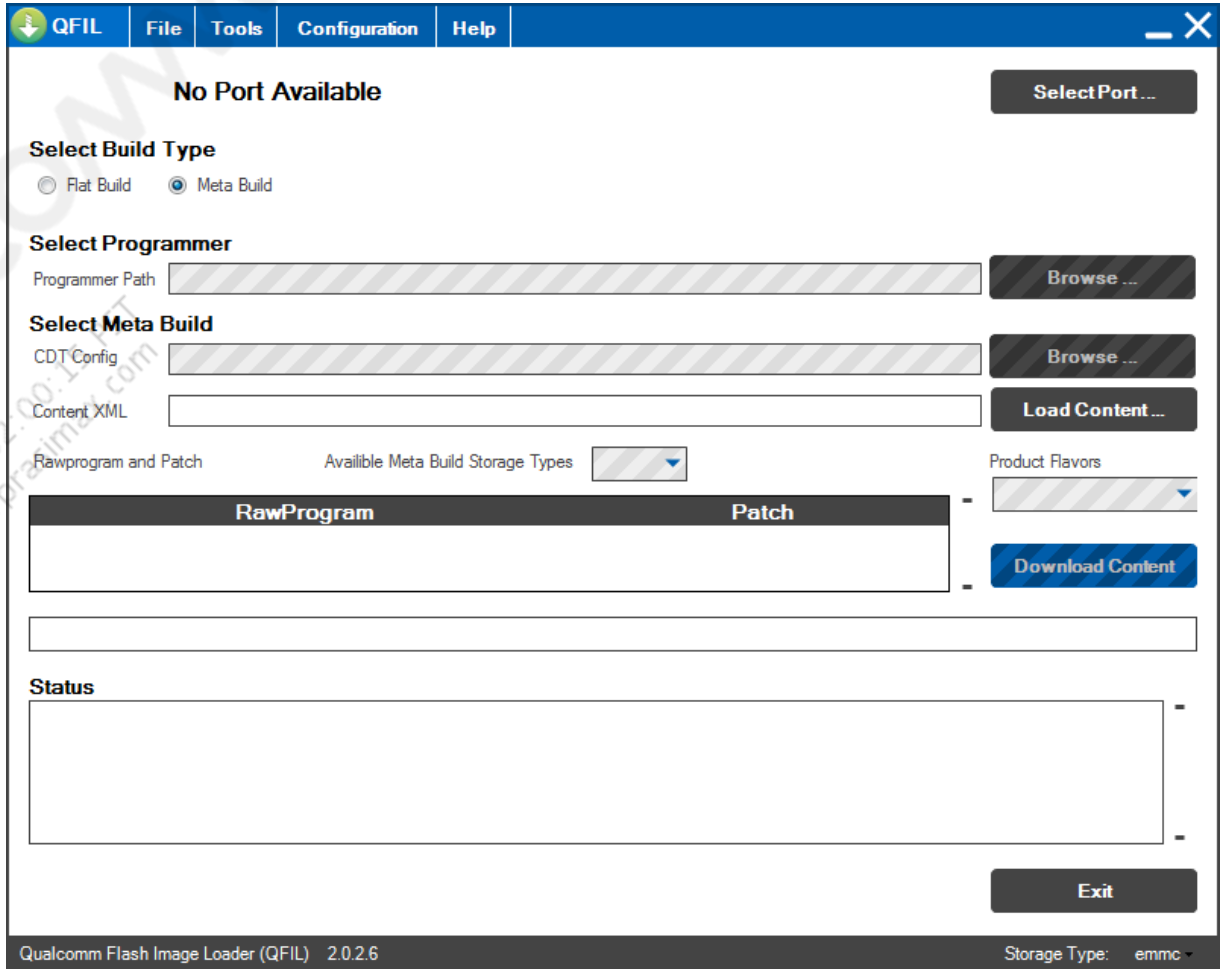


Section 7.3.1

Packaging Images using QFIL

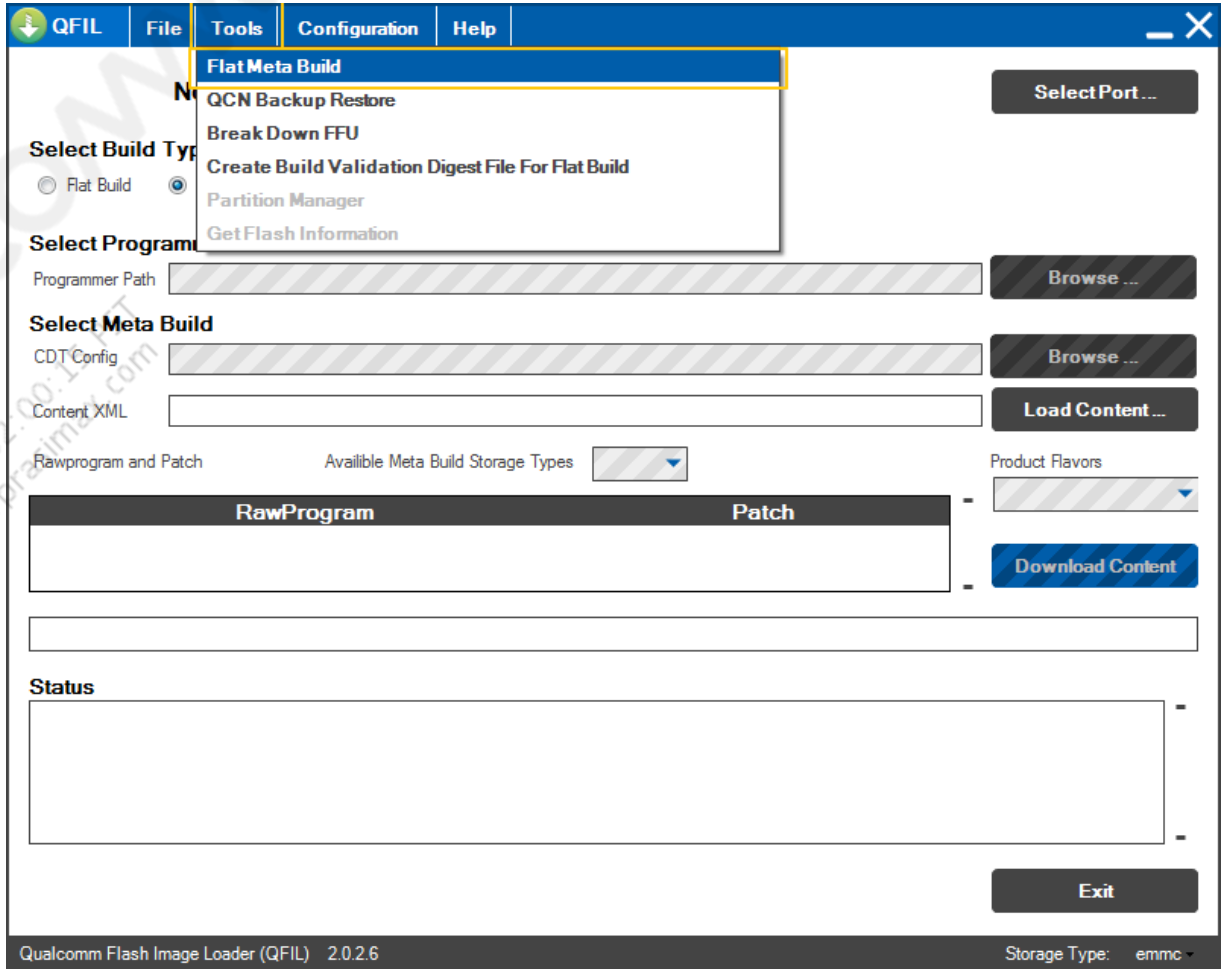
Packaging Images using QFIL

- Launch QFIL



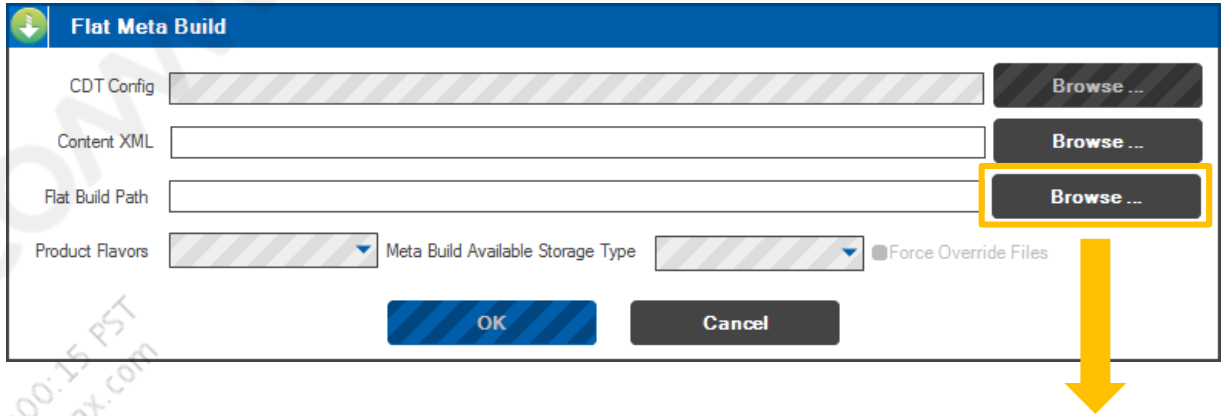
Packaging Images using QFIL

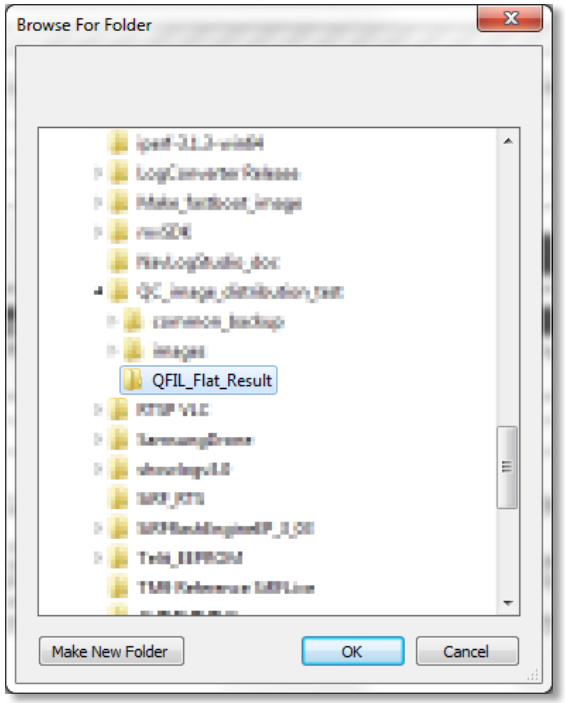
- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu



Packaging Images using QFIL

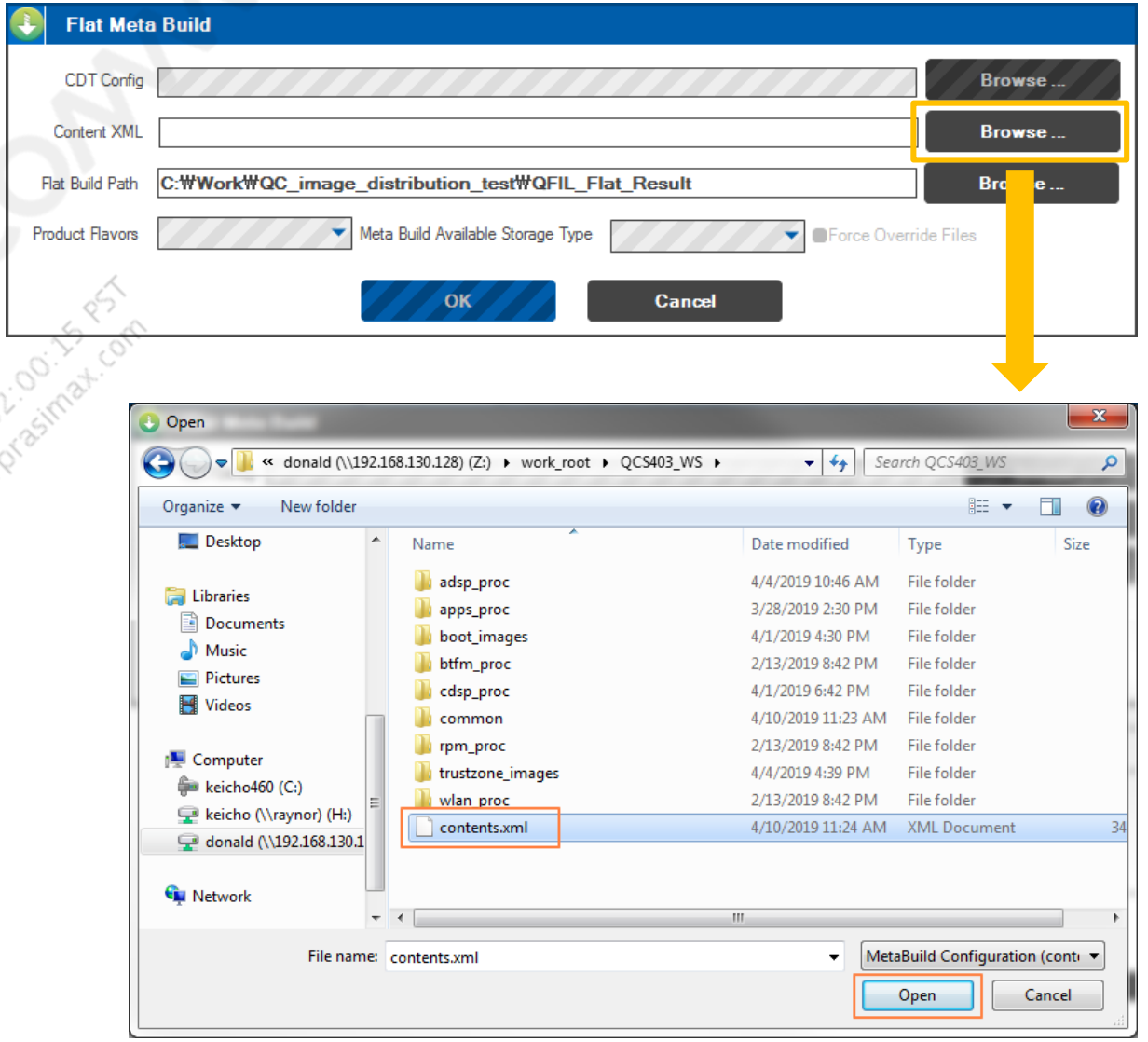
- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>





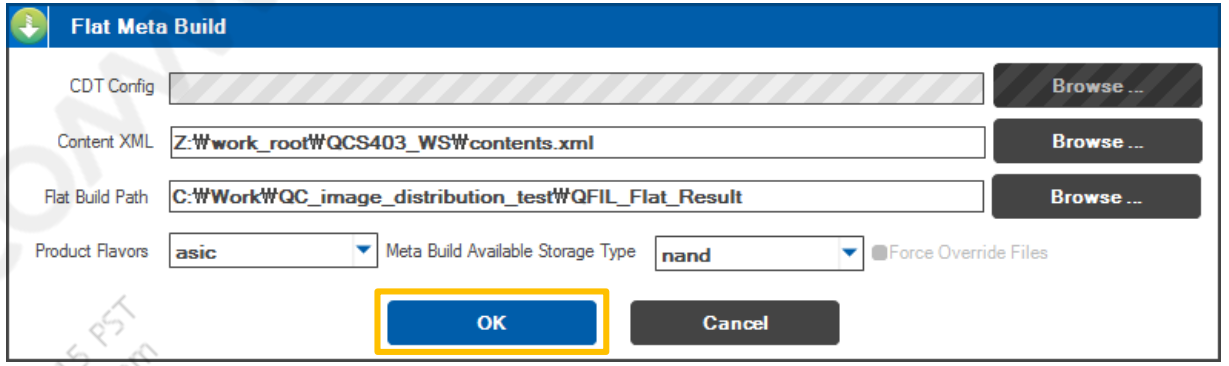
Packaging Images using QFIL

- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>
- Click on ‘Content XML > Browse…’ button
- Open ‘<QCS403_WS>/contents.xml’ file



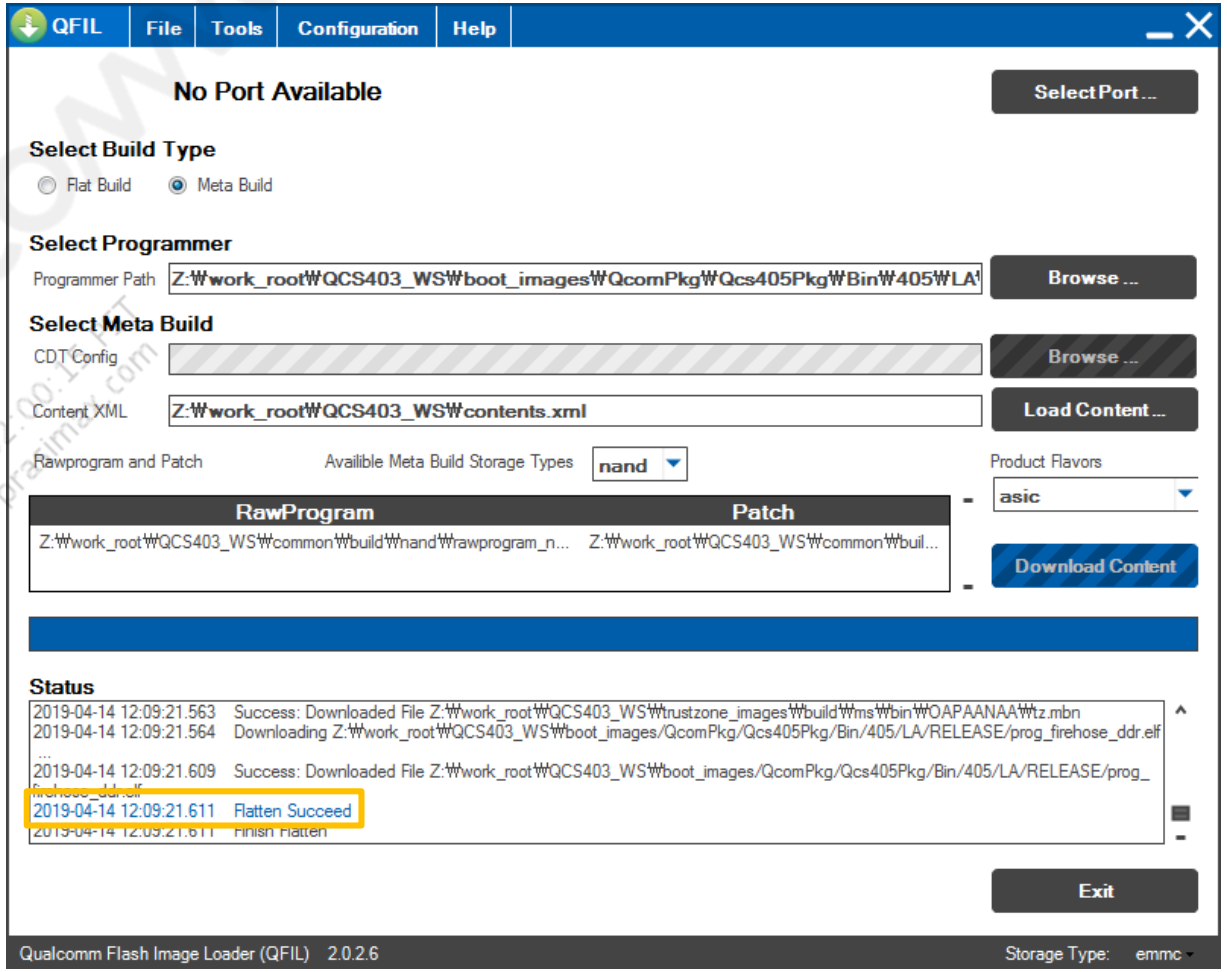
Packaging Images using QFIL

- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>
- Click on ‘Content XML > Browse…’ button
- Open ‘<QCS403_WS>/contents.xml’ file
- Click on ‘OK’ button to start Flat Meta Build



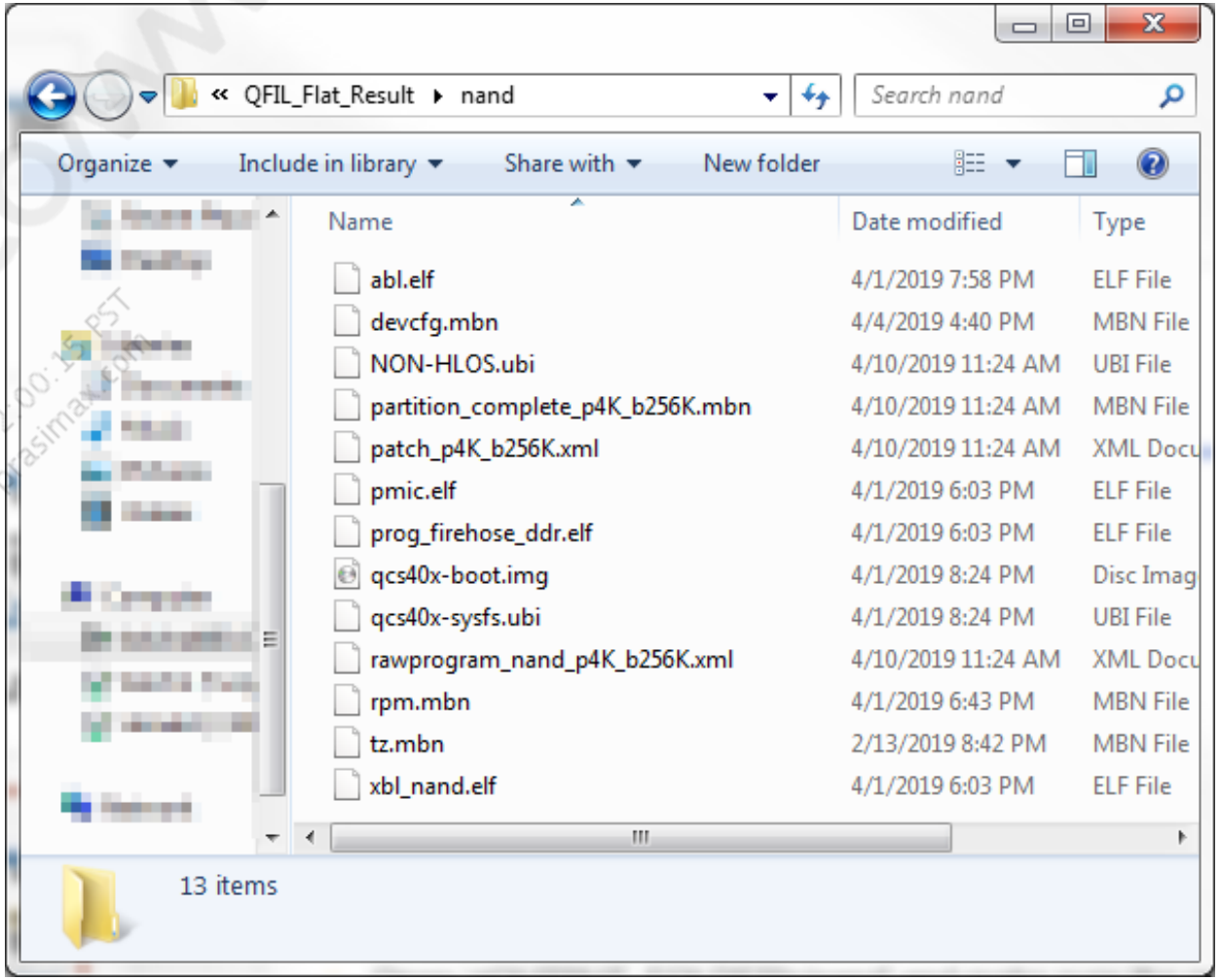
Packaging Images using QFIL

- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>
- Click on ‘Content XML > Browse…’ button
- Open ‘<QCS403_WS>/contents.xml’ file
- Click on ‘OK’ button to start Flat Meta Build
- Make sure “Flatten Succeed” message



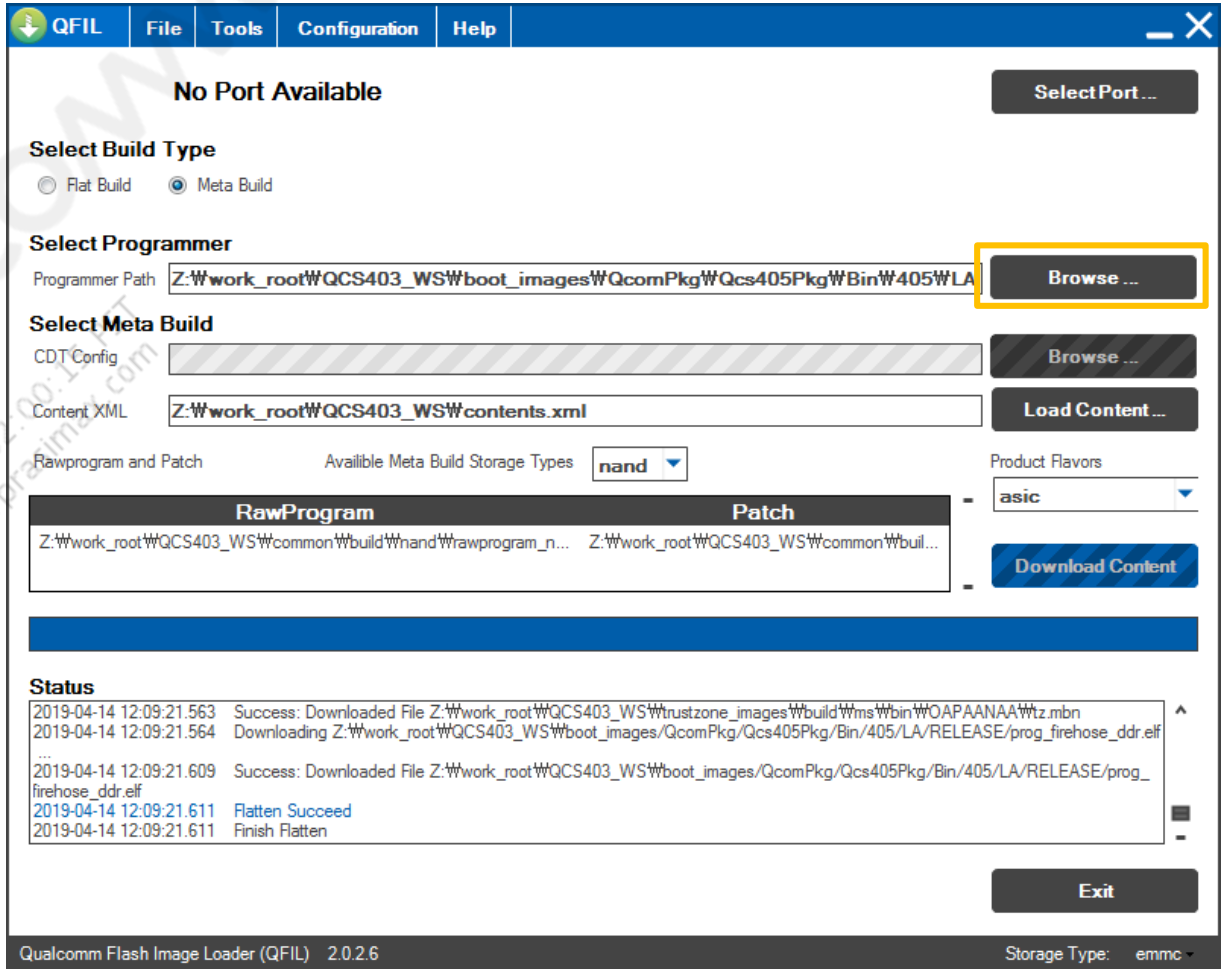
Packaging Images using QFIL

- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>
- Click on ‘Content XML > Browse…’ button
- Open ‘<QCS403_WS>/contents.xml’ file
- Click on ‘OK’ button to start Flat Meta Build
- Make sure “Flatten Succeed” message
- Open ‘<OUTPUT_FOLDER>/nand’ and make sure the result



Packaging Images using QFIL

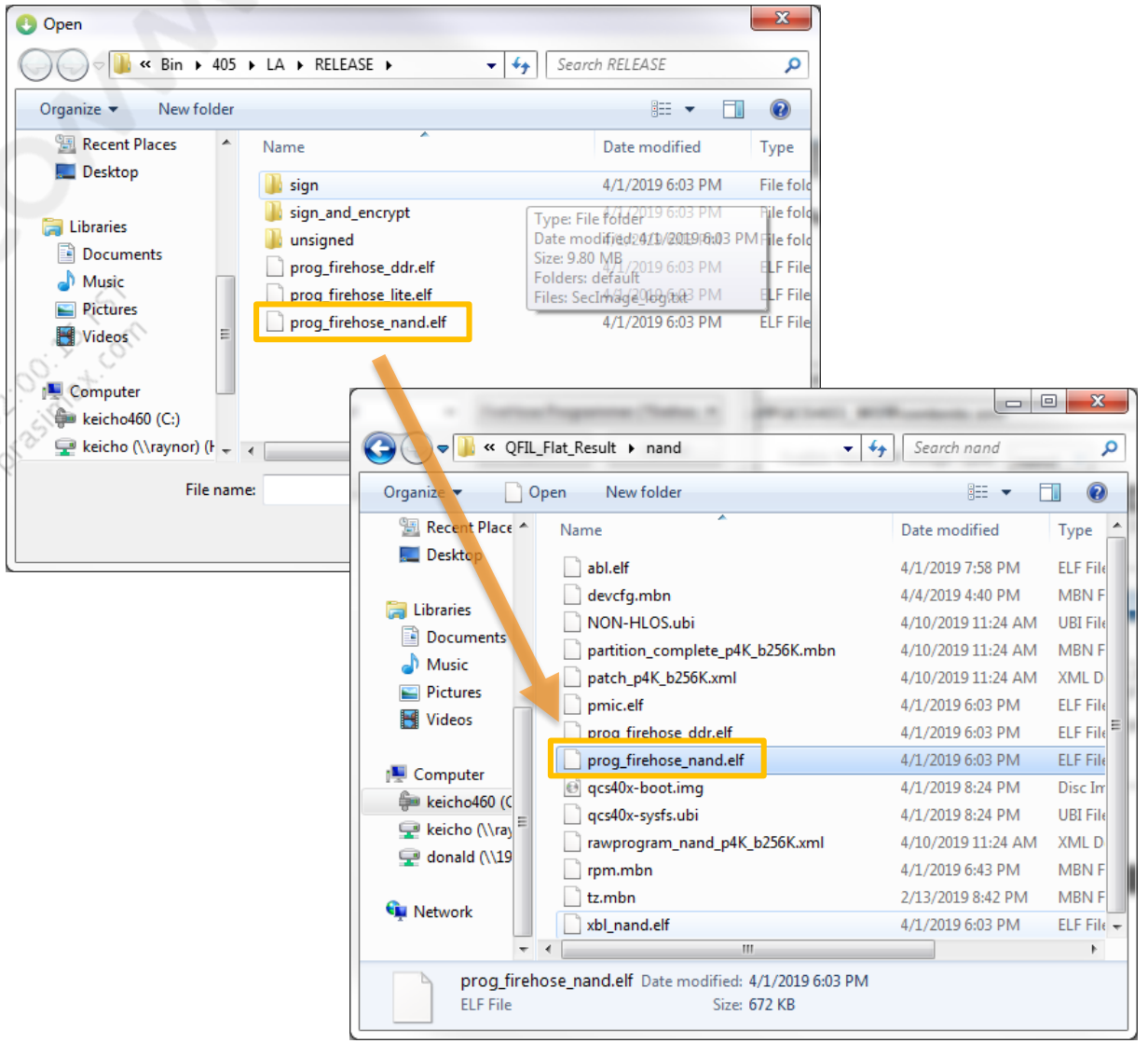
- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>
- Click on ‘Content XML > Browse…’ button
- Open ‘<QCS403_WS>/contents.xml’ file
- Click on ‘OK’ button to start Flat Meta Build
- Make sure “Flatten Succeed” message
- Open ‘<OUTPUT_FOLDER>/nand’ and make sure the result
- To manually copy additional Programmer file for nand:
  - Click on ‘Programmer Path > Browse…’ button
  - Then following folder will be opened:
    - <QCS403_WS>\boot_images\QcomPkg\Qcs405Pkg\Bin\405\LA\RELEASE



Packaging Images using QFIL

- Launch QFIL
- Click on ‘Tools > Flat Meta Build’ menu
- Click on ‘Flat Build Path > Browse…’ button
- Select <OUTPUT_FOLDER>
- Click on ‘Content XML > Browse…’ button
- Open ‘<QCS403_WS>/contents.xml’ file
- Click on ‘OK’ button to start Flat Meta Build
- Make sure “Flatten Succeed” message
- Open ‘<OUTPUT_FOLDER>/nand’ and make sure the result
- To manually copy additional Programmer file for nand:
  - Click on ‘Programmer Path > Browse…’ button
  - Then following folder will be opened:
    - <QCS403_WS>\boot_images\QcomPkg\Qcs405Pkg\Bin\405\LA\RELEASE
  - Copy ‘prog_firehose_nand.elf’ file to <OUTPUT_FOLDER>

- DONE !

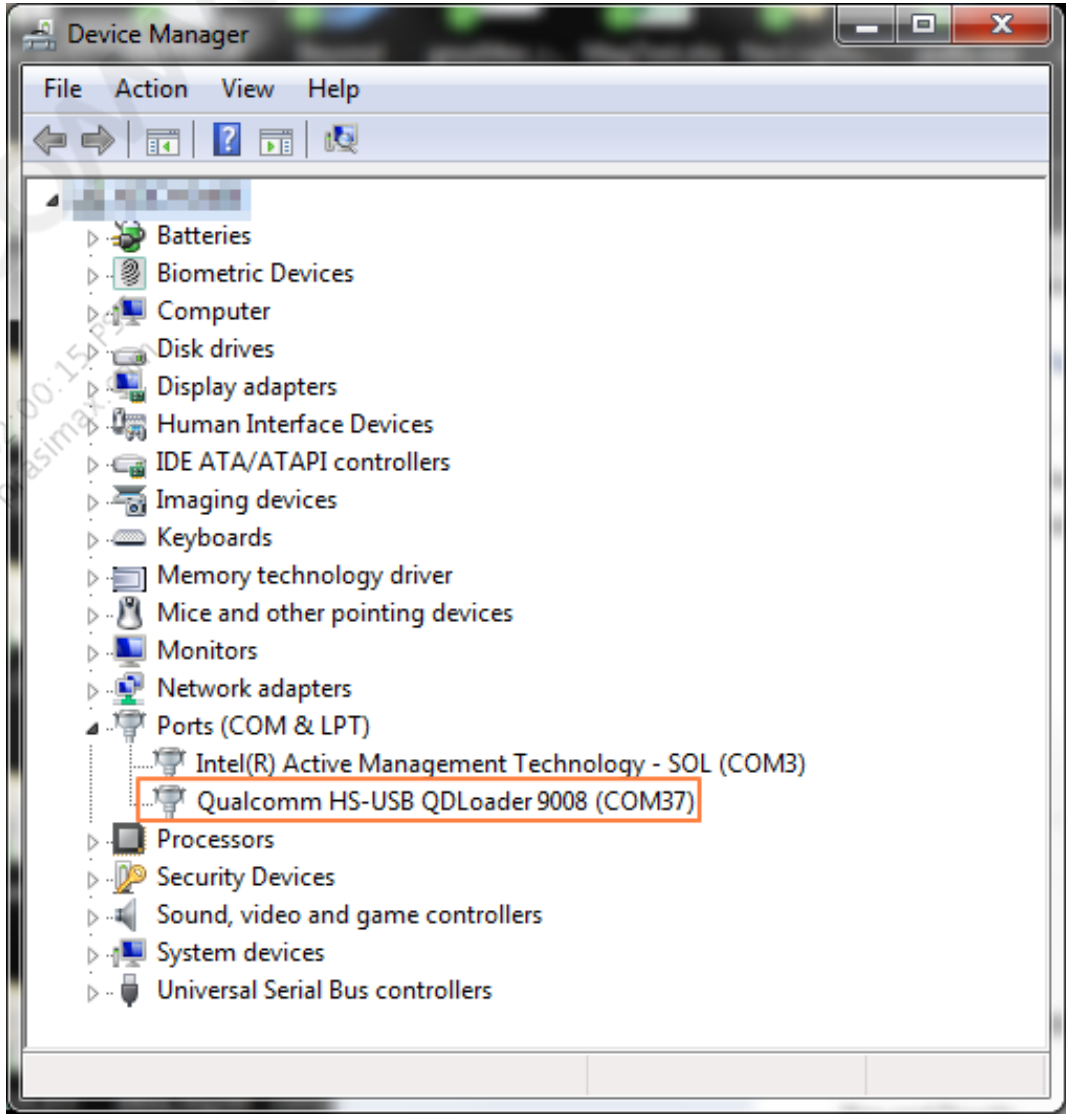


Section 7.3.2

Writing Flat Build Images using QFIL

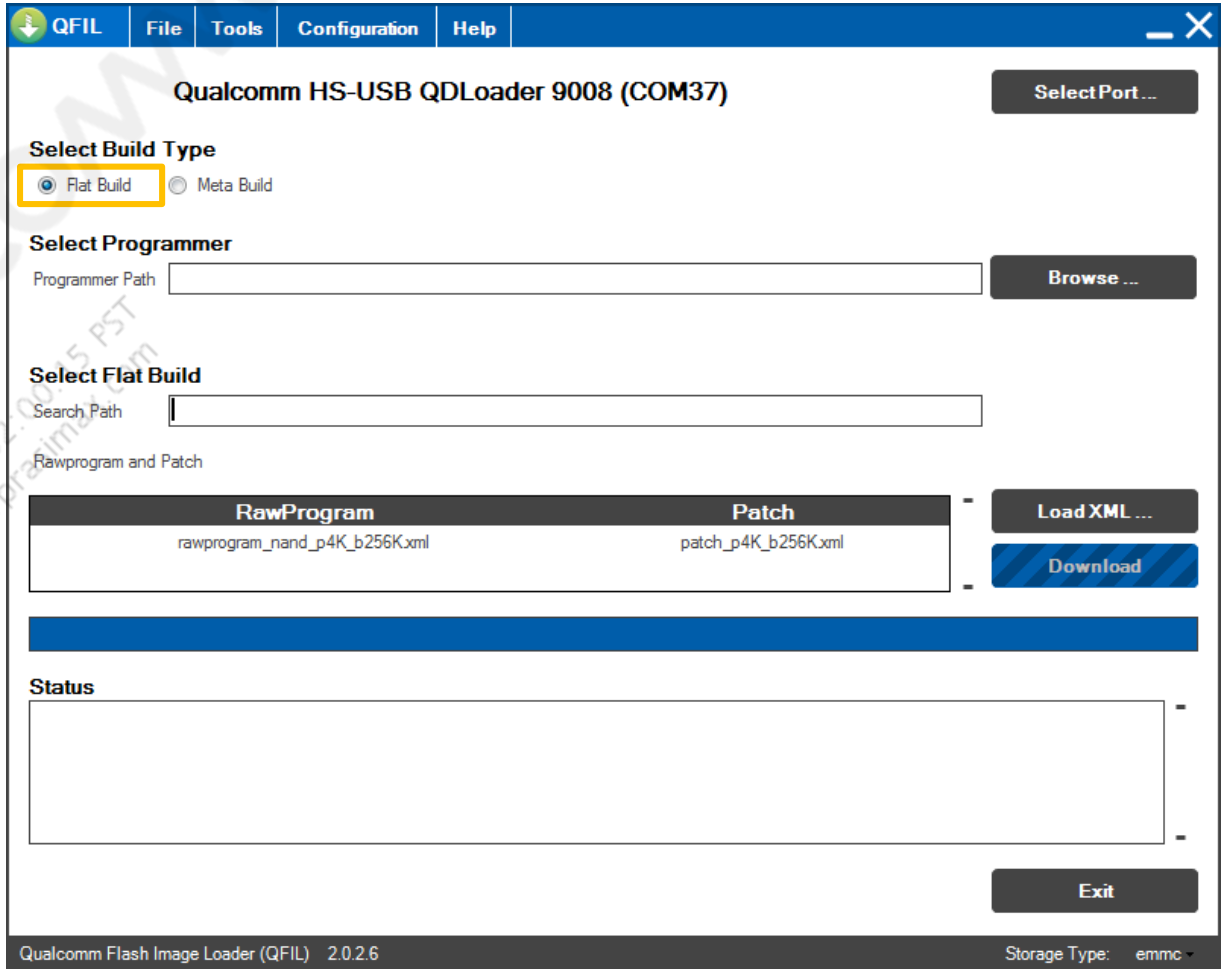
Writing Flat Build Images using QFIL

- To write images using QFIL, the device must be configured to EDL mode
- Refer to:
  - Device Boot Configuration for SSRD - EDL mode



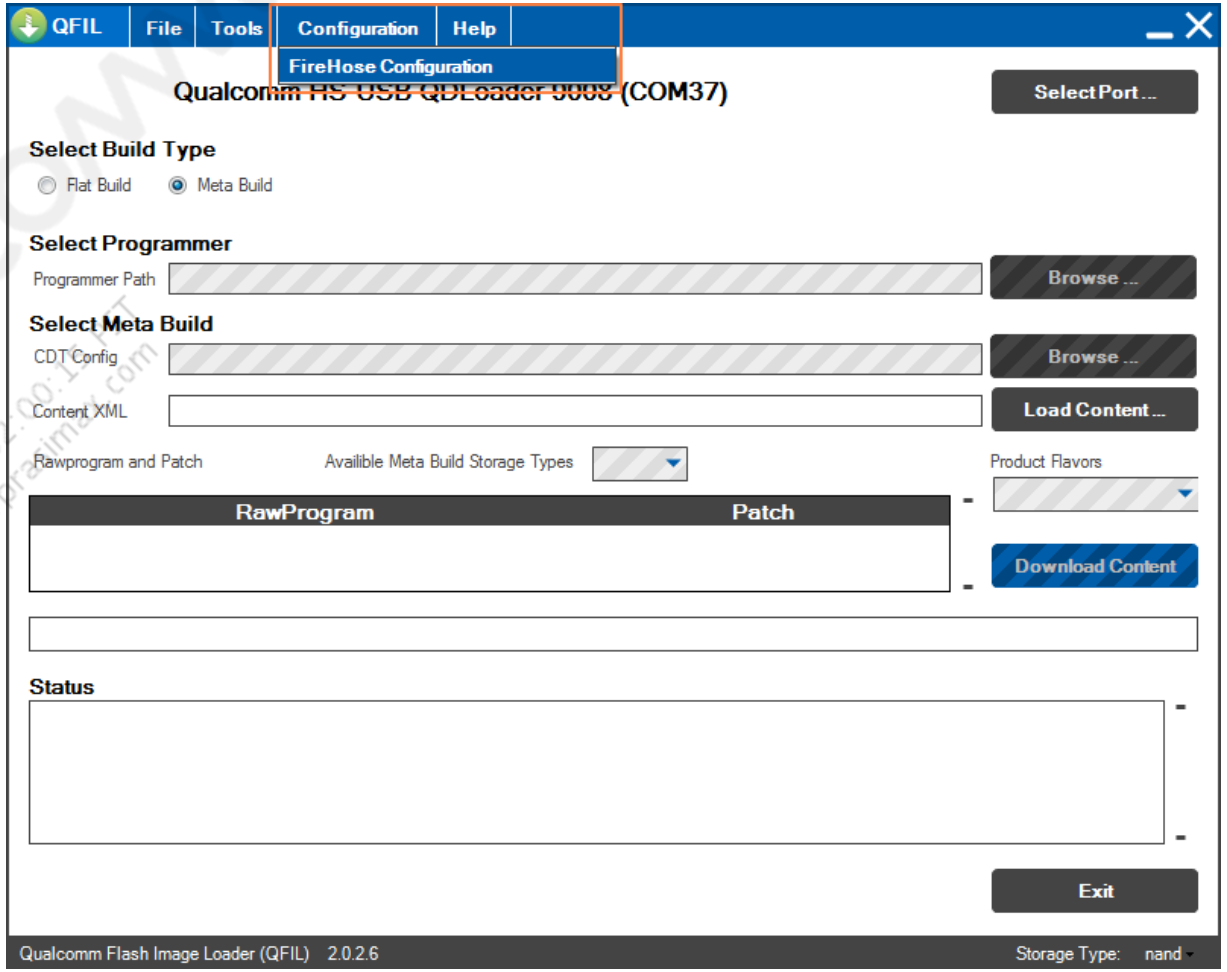
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Flat Build’



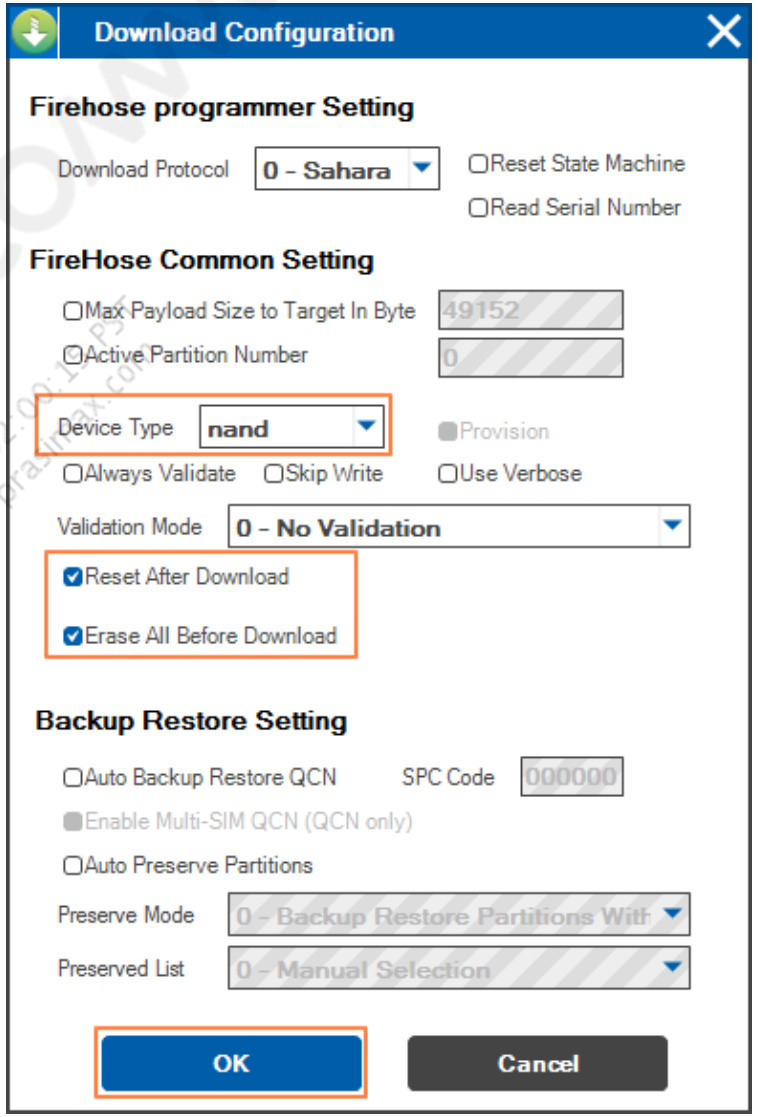
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)



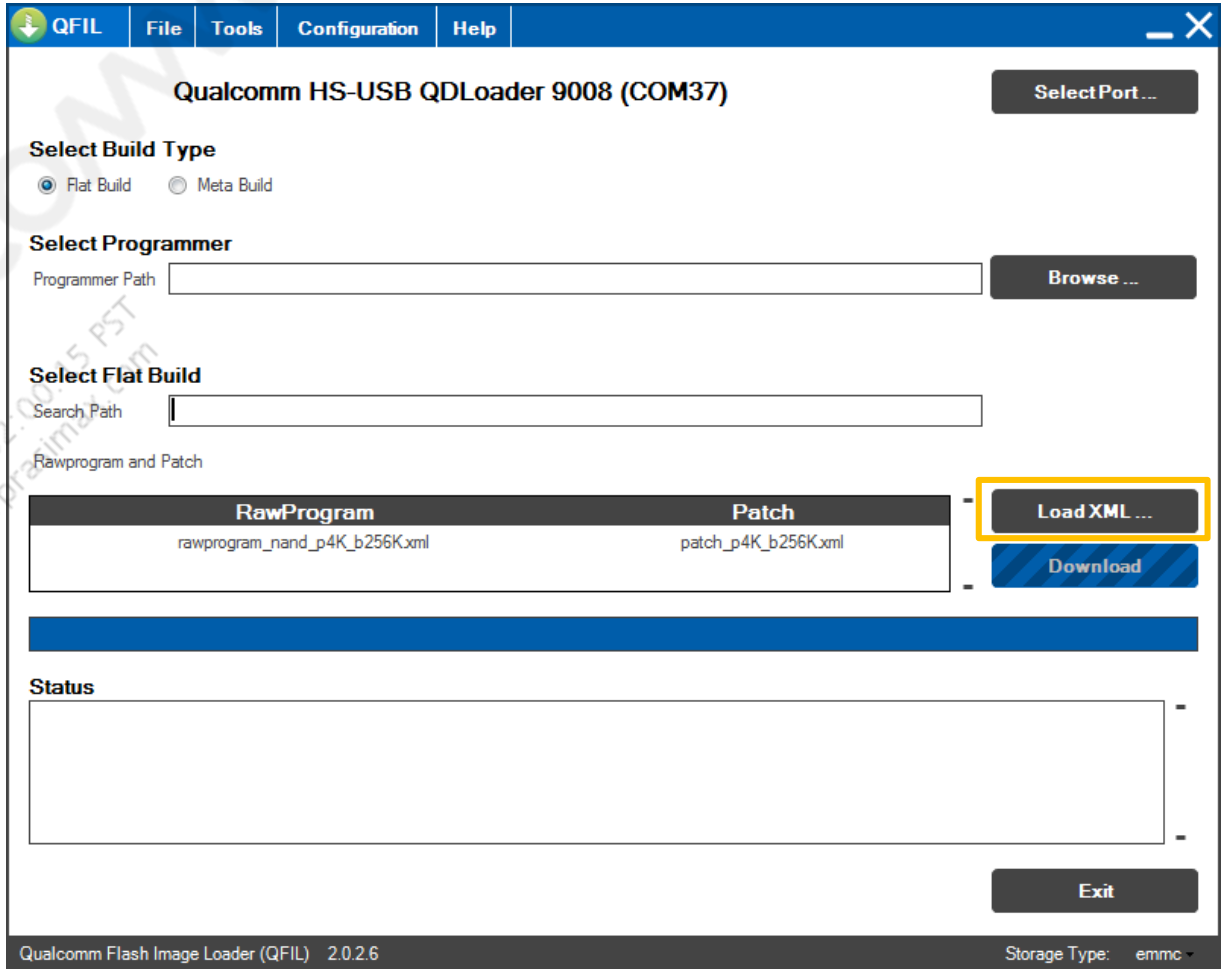
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON



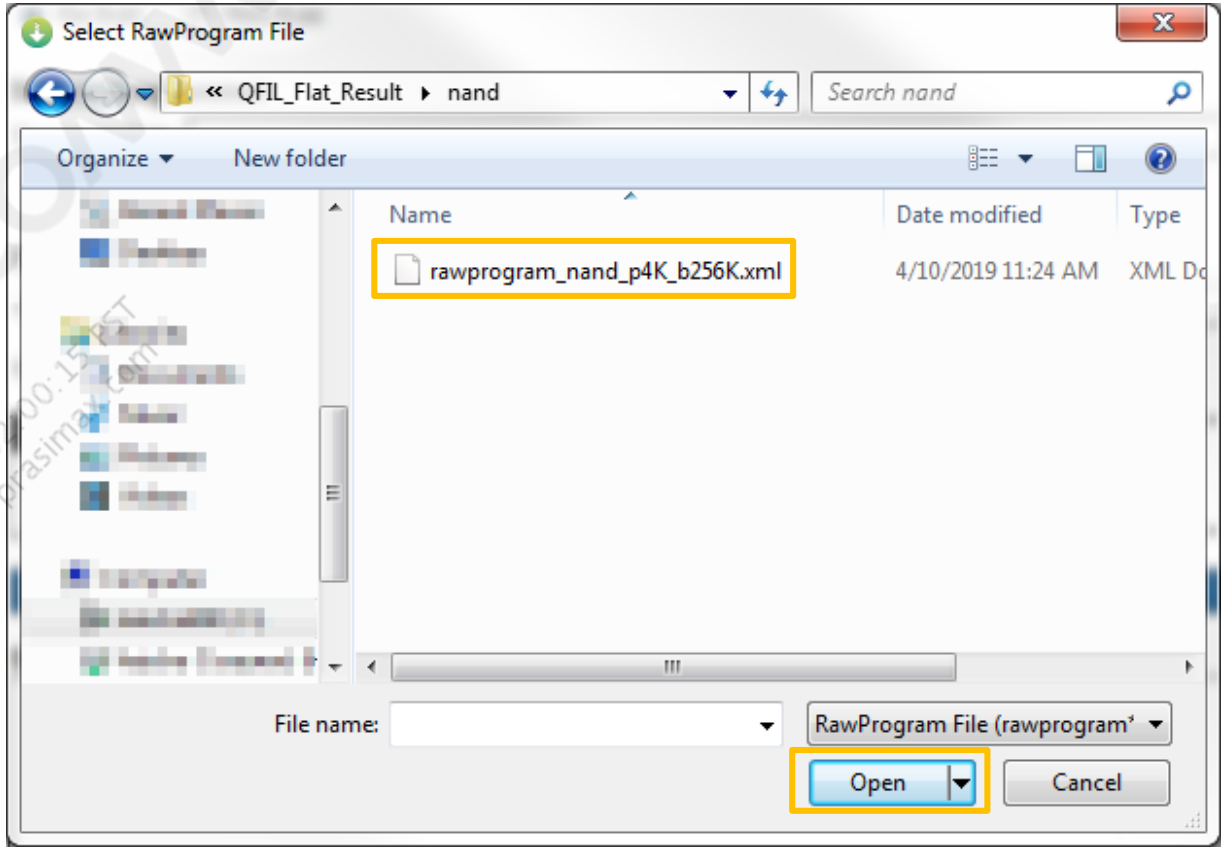
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load XML…’ button



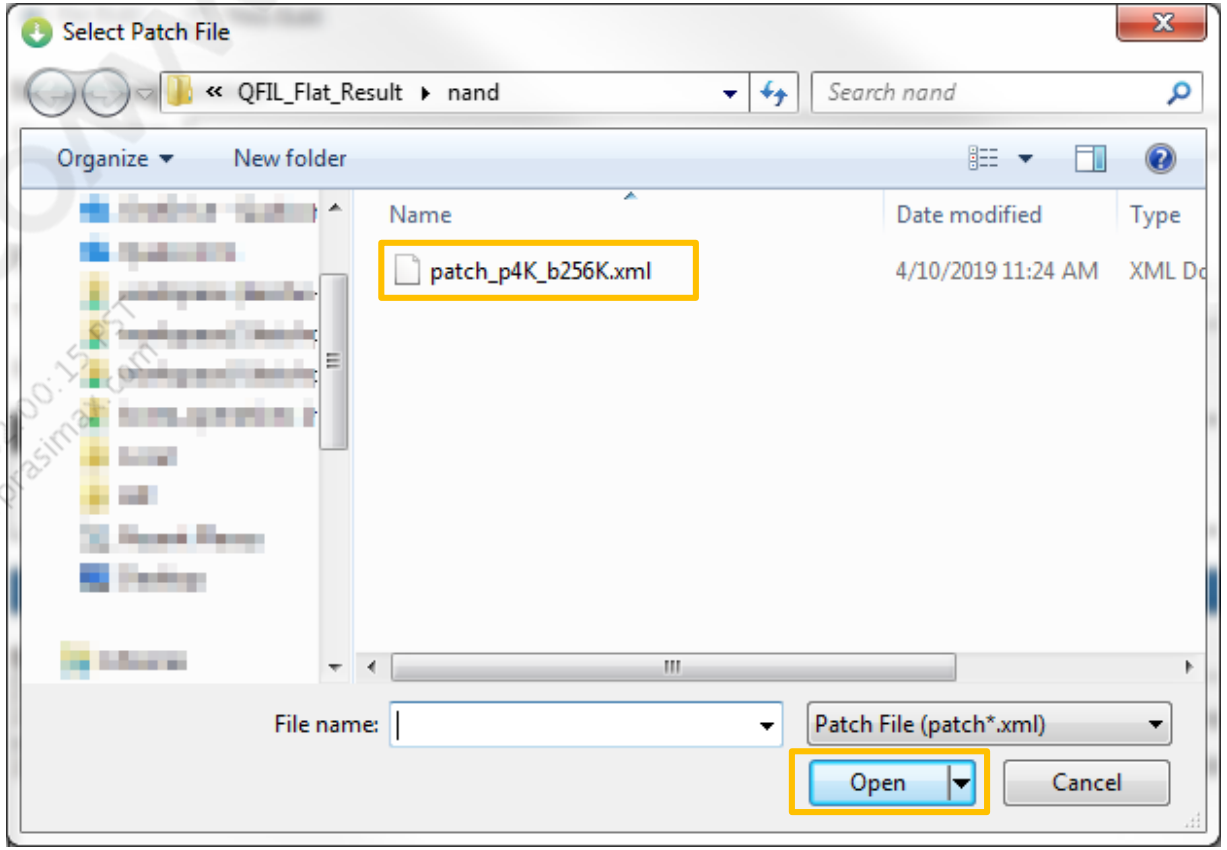
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load XML…’ button
- Open ‘rawprogram_nand’ xml file



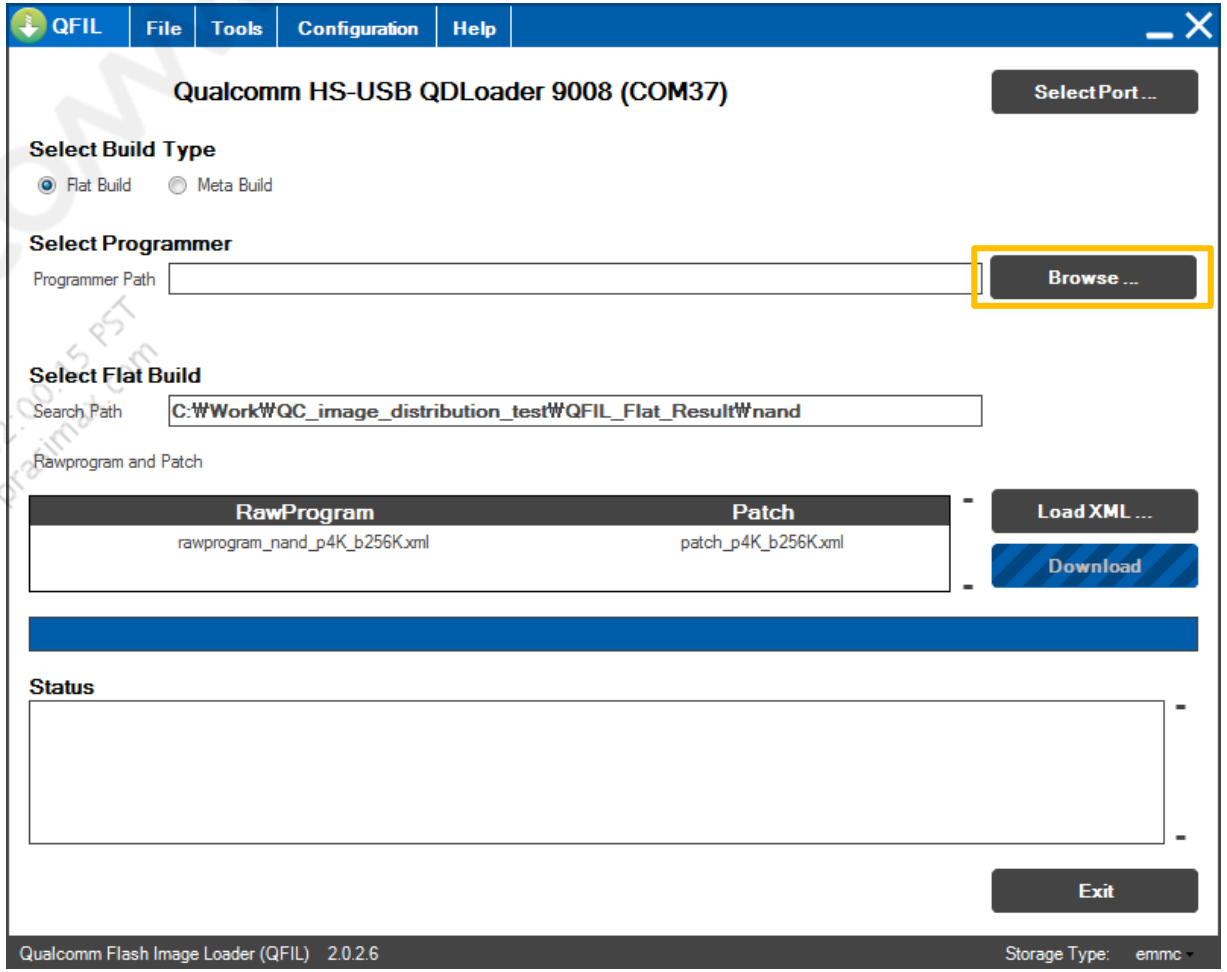
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load XML…’ button
- Open ‘rawprogram_nand’ xml file
- Open ‘patch’ xml file



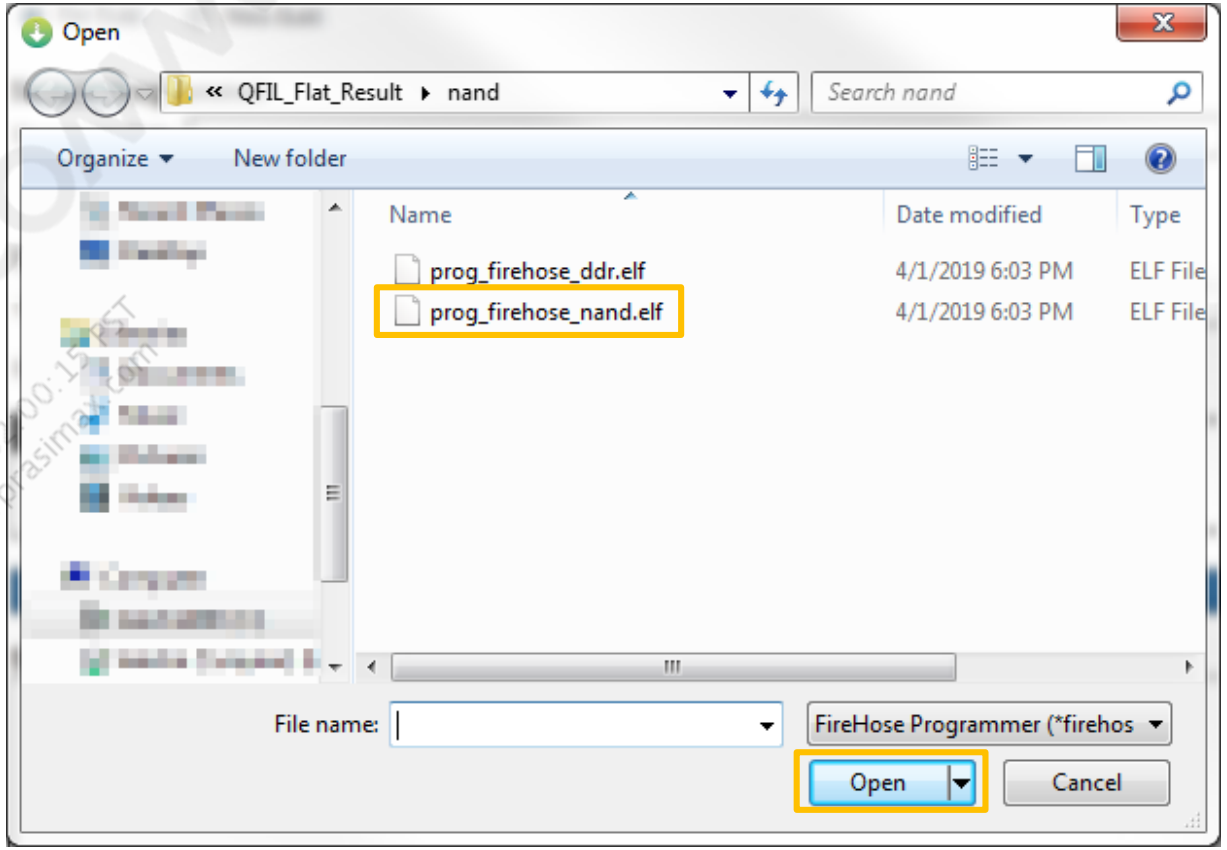
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load XML…’ button
- Open ‘rawprogram_nand’ xml file
- Open ‘patch’ xml file
- Click on ‘Programmer Path > Browse…’ button



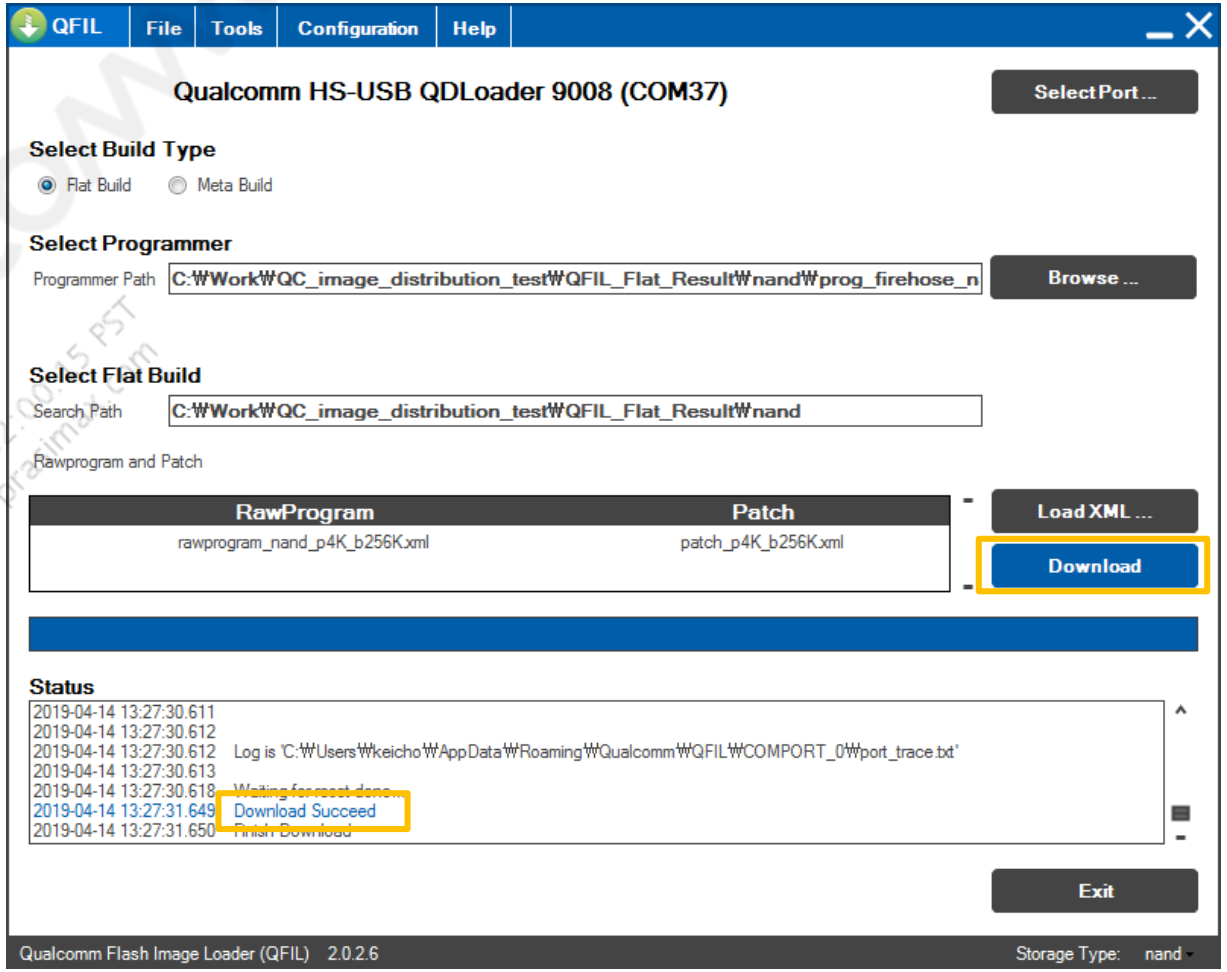
Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load XML…’ button
- Open ‘rawprogram_nand’ xml file
- Open ‘patch’ xml file
- Click on ‘Programmer Path > Browse…’ button
- Open ‘prog_firehose_nand.elf’ file



Writing Flat Build Images using QFIL

- Launch QFIL and Select Build Type to ‘Meta Build’
- Click on ‘Configuration > FireHose Configuration’ to check or modify settings for QCS403 (NAND)
- Set Download Configuration as below and then Click on ‘OK’
  - Device Type: nand
  - Reset After Download: Check ON
  - Erase All Before Download: Check ON
- Click on ‘Load XML…’ button
- Open ‘rawprogram_nand’ xml file
- Open ‘patch’ xml file
- Click on ‘Programmer Path > Browse…’ button
- Open ‘prog_firehose_nand.elf’ file
- Click on ‘Download’ button
- Make sure “Download Succeed” message
- DONE !

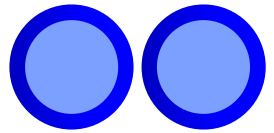


Section 8

Basic Operation Test

Basic Operation Test

- In this chapter, following process will be explained for successful AVS and Bluetooth testing setup
  - Bring up DUT
  - Writing Device Configuration
  - Setting up Haven License
  - Onboarding Wi-Fi
  - Onboarding AVS
  - Bluetooth Audio Streaming Test

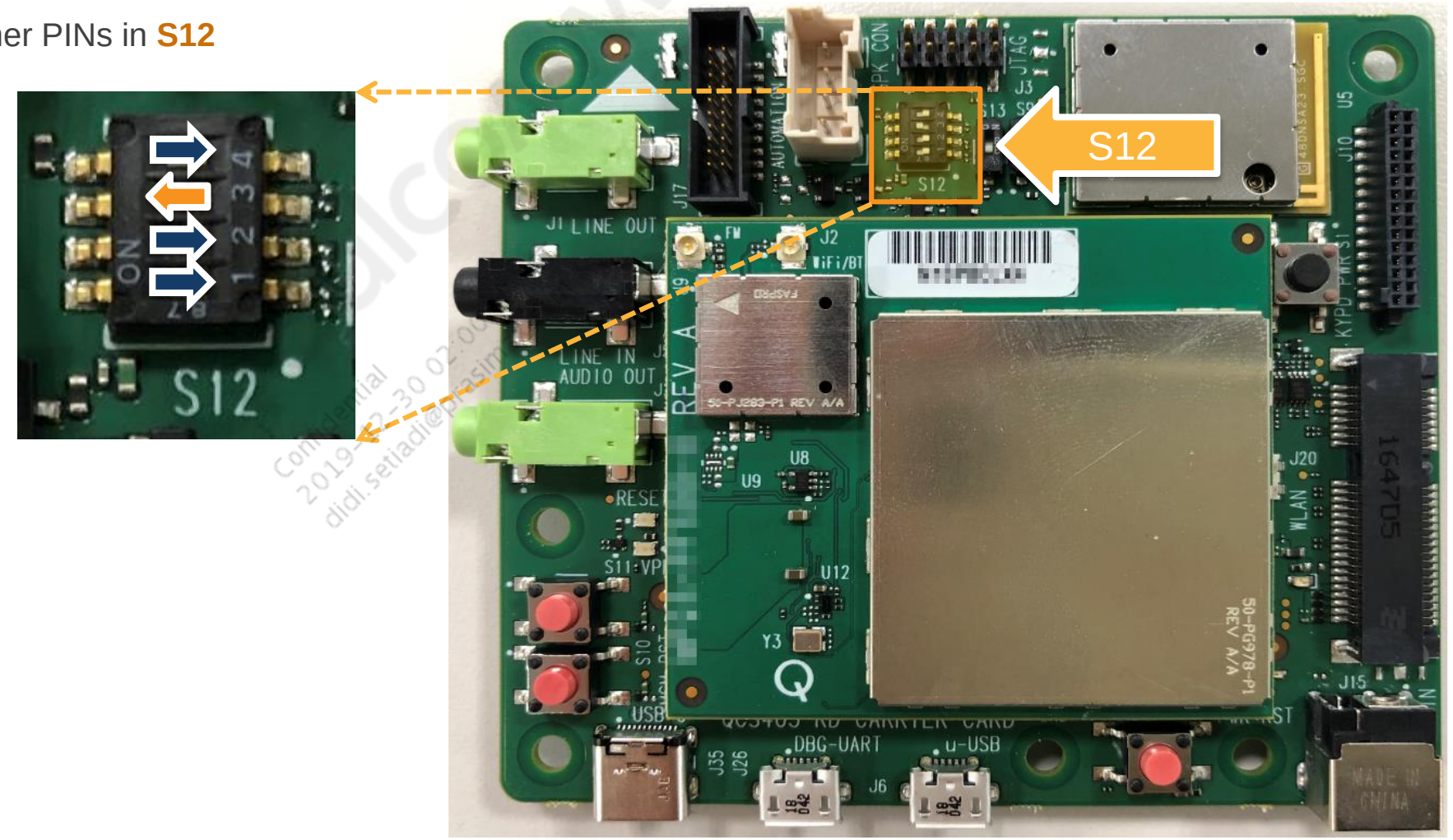


Section 8.1

Bring up DUT

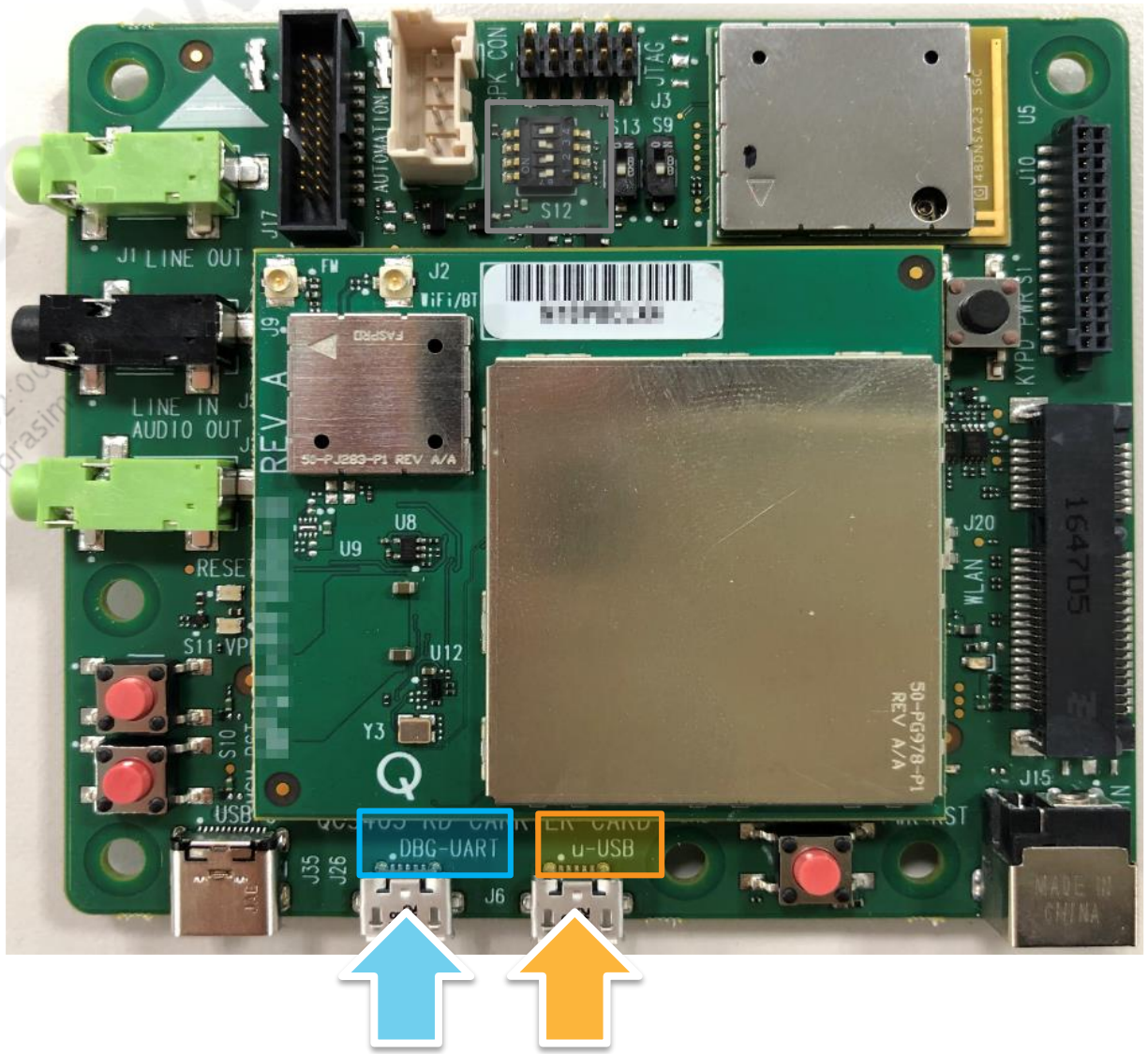
Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**



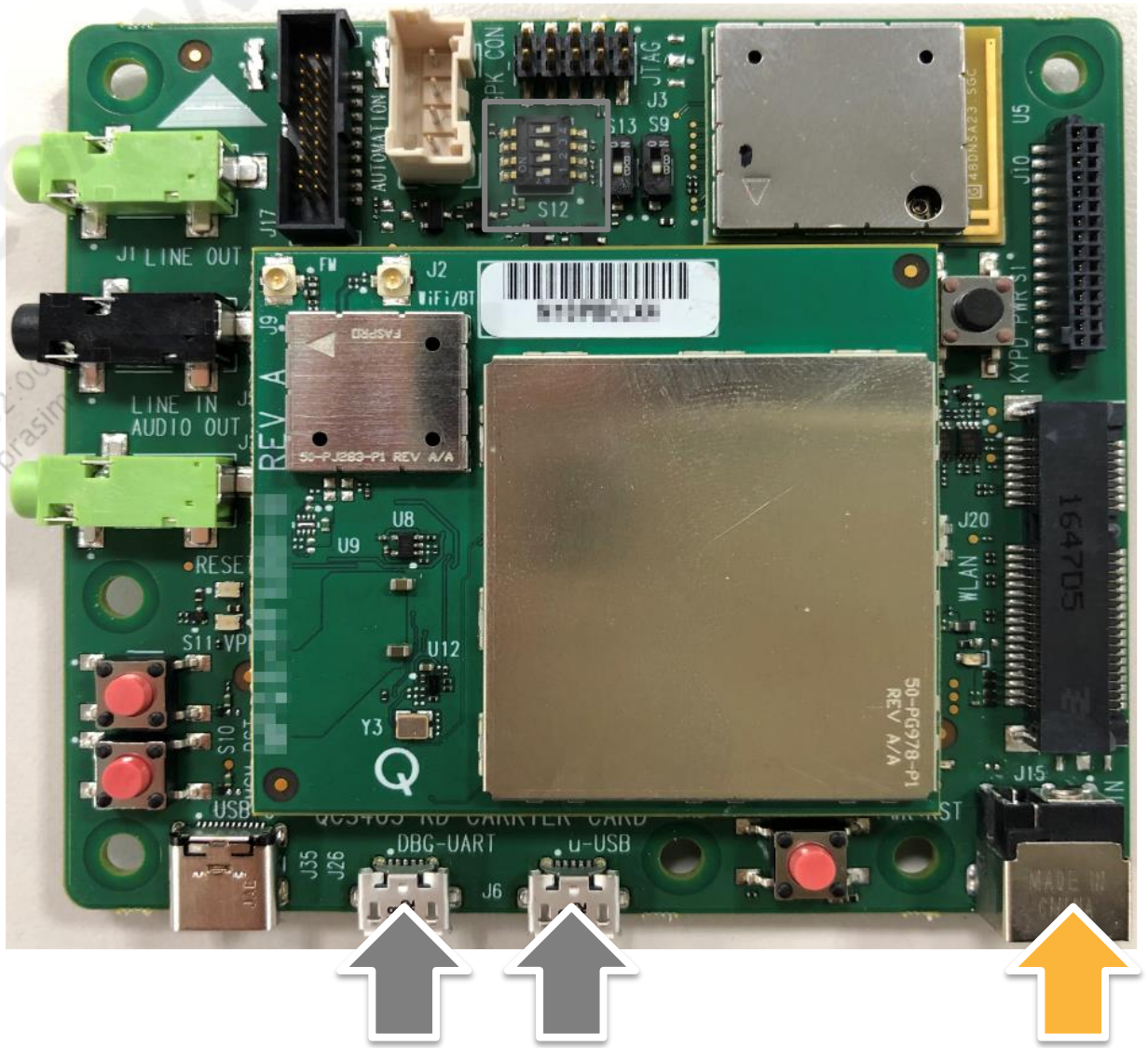
Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface



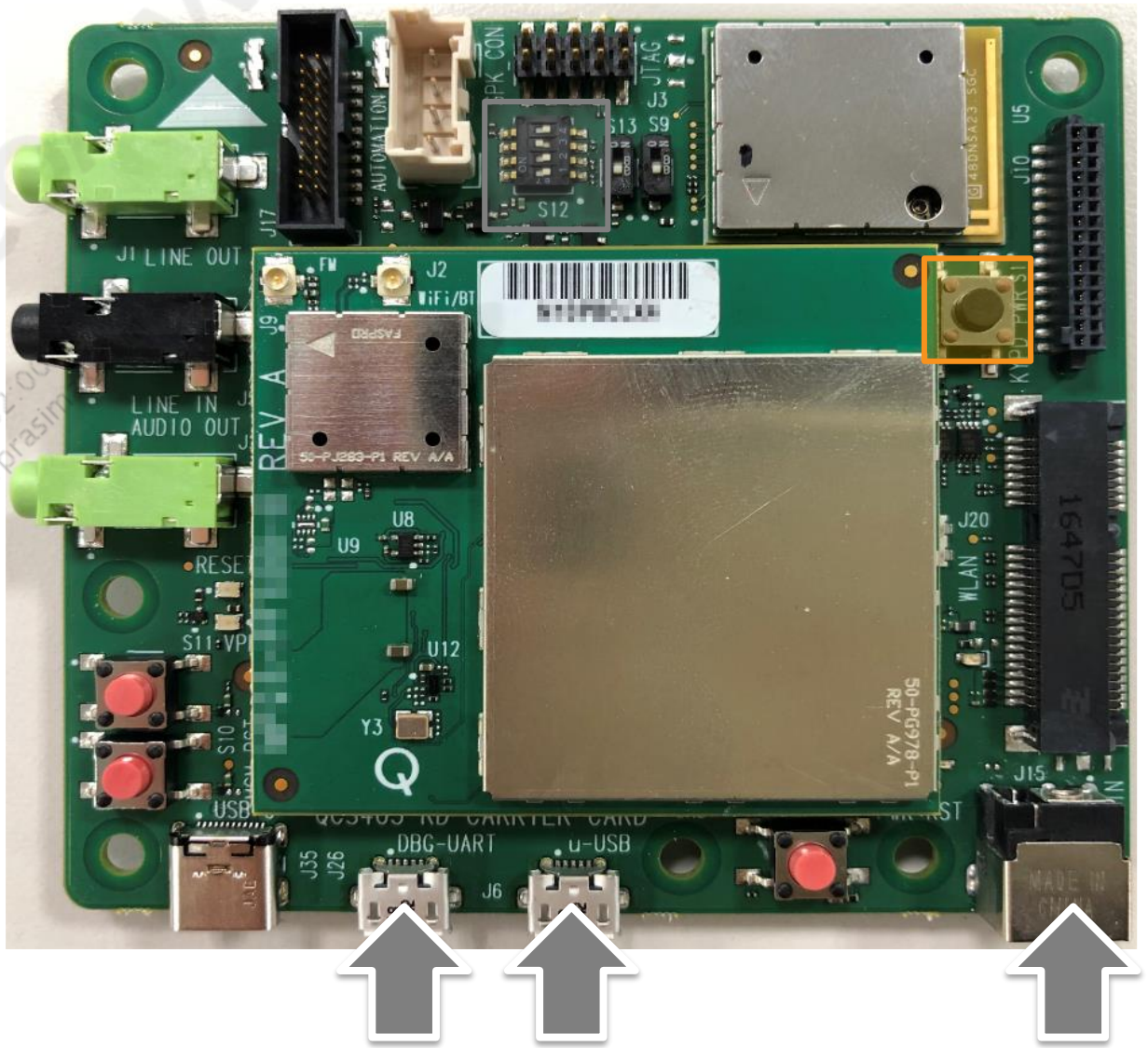
Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface
3. Connect the Main Power cable



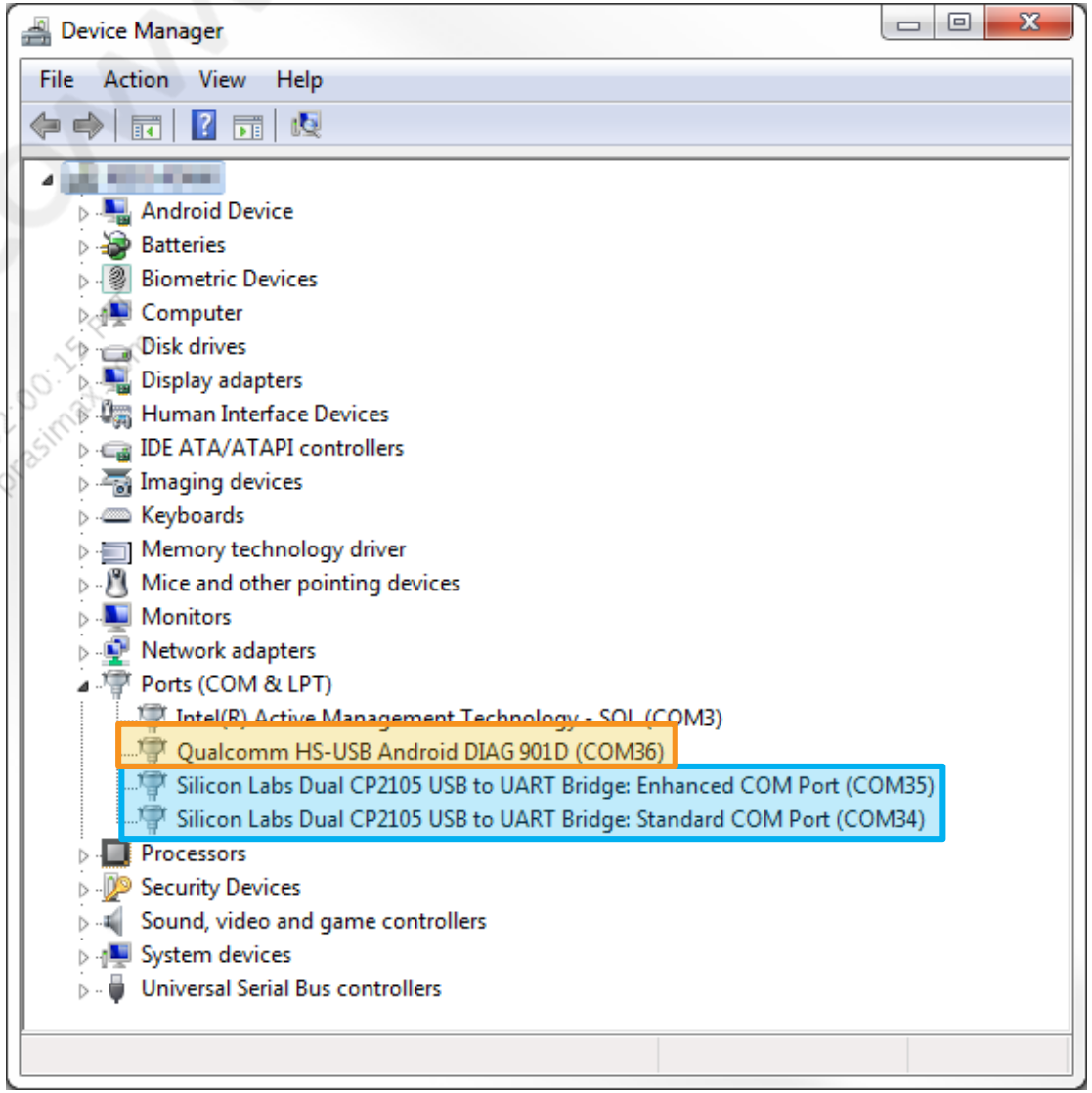
Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface
3. Connect the Main Power cable
4. Push the KYPD_PWR button



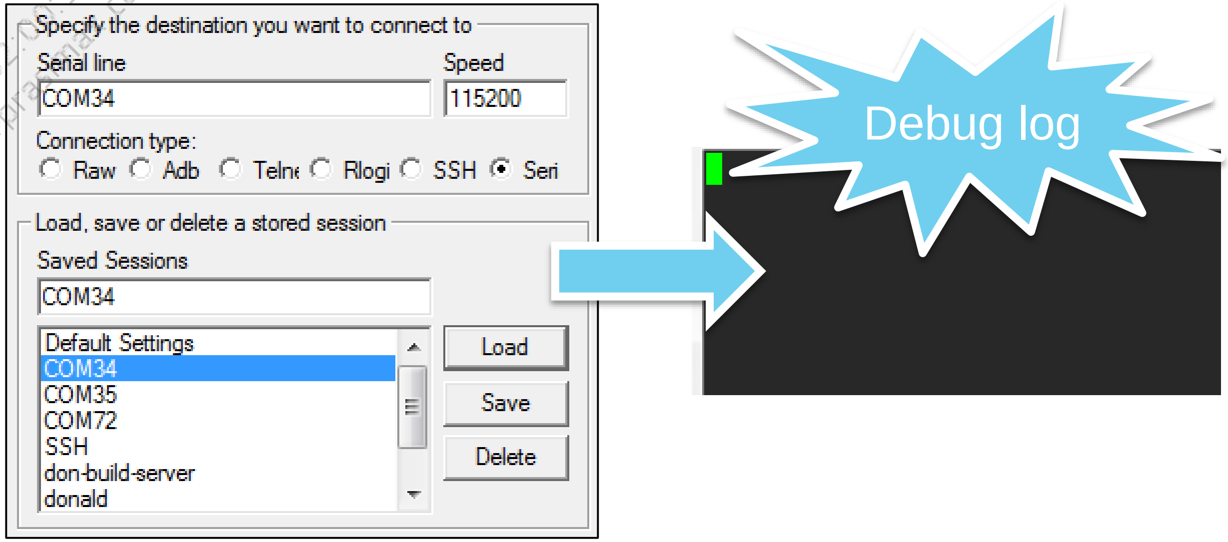
Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface
3. Connect the Main Power cable
4. Push the KYPD_PWR button
5. Check the USB connection on host PC > Device Manager
   - 'Qualcomm HS-USB Android DIAG…' for adb interface
   - 'Silicon Labs…' for debug log interface
     - Normally, among the two ports, the front number port is selected as debug port



Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface
3. Connect the Main Power cable
4. Push the KYPD_PWR button
5. Check the USB connection on host PC > Device Manager
   - 'Qualcomm HS-USB Android DIAG…' for adb interface
   - 'Silicon Labs…' for debug log interface
     - Normally, among the two ports, the front number port is selected as debug port
6. Open terminal windows for adb and debug log

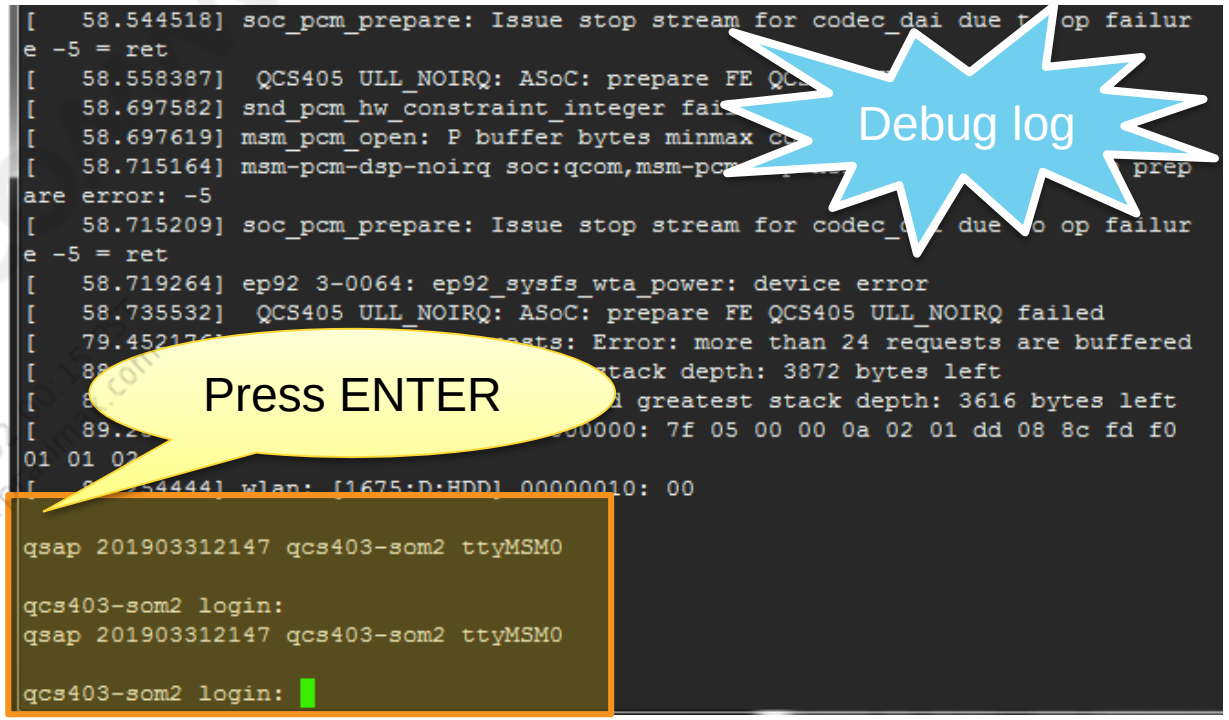


Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface
3. Connect the Main Power cable
4. Push the KYPD_PWR button
5. Check the USB connection on host PC > Device Manager
   - ‘Qualcomm HS-USB Android DIAG…’ for adb interface
   - ‘Silicon Labs…’ for debug log interface
     - Normally, among the two ports, the front number port is selected as debug port
6. Open terminal windows for adb and debug log
7. Reboot device and check the device booting log on debug log terminal

Bring up DUT

1. Turn ON PIN#3 and Turn OFF other PINs in **S12**
2. Connect device to host PC with USB cables
   - u-USB for adb interface
   - DBG-UART for debug log interface
3. Connect the Main Power cable
4. Push the KYPD_PWR button
5. Check the USB connection on host PC > Device Manager
   - 'Qualcomm HS-USB Android DIAG…' for adb interface
   - 'Silicon Labs…' for debug log interface
     - Normally, among the two ports, the front number port is decided as debug port
6. Open terminal windows for adb and debug log
7. Reboot device and check the device booting log on debug log terminal

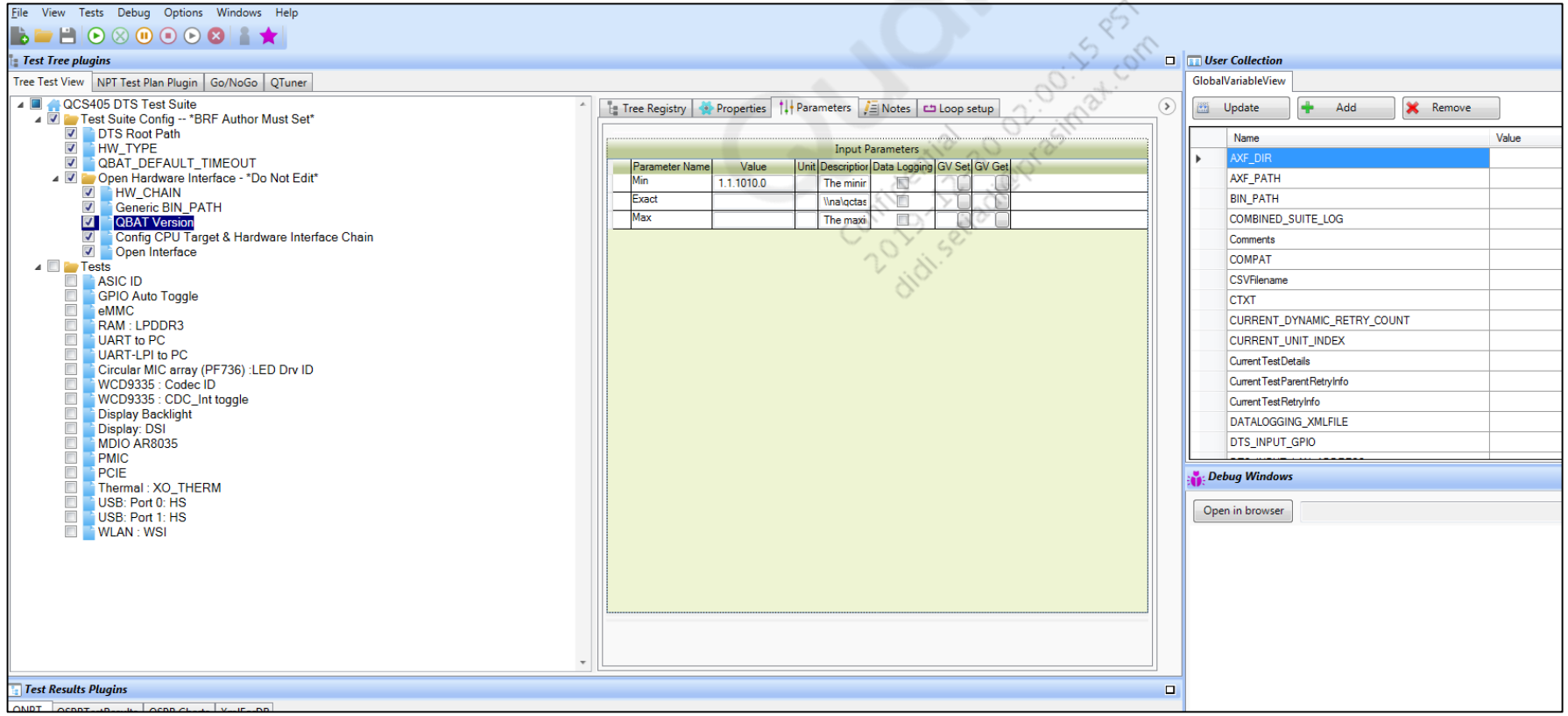


- In debug apps image, press ENTER key on the debug log window to login. Default password is 'oelinux123'
- In perf apps image, Login Function and apps debug log are not supported

Bring up DUT – Trouble shooting

- **DO NOT MIX USE** HLOS image with different version of non-HLOS image
- Otherwise, unexpected problems may occur even in the middle of operation

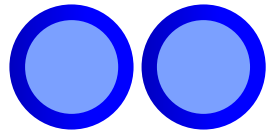
- If a problem occurs during the booting process, QBAT can be used to check the basic device status



Bring up DUT – Trouble shooting

- If you have any problem during operation, check the firmware version information from '/firmware/verinfo/Ver_info.txt' file
- And then please report the issue with the firmware version information to QCS Support Engineer

```
/ # cat /firmware/verinfo/Ver_Info.txt
{
    "Image_Build_IDs": {
        "adsp": "ADSP.IT.1.0-00370.3-QCS405-1",
        "apps": "LE.UM.2.4.1.r1-09304-qcs405.1-1",
        "boot": "BOOT.XF.0.1-00106-QCS405LAB-1",
        "btfm": "BTFM.CHE.2.1.4-00507-QCACHROM-1",
        "cdsp": "CDSP.IT.1.0-00249-QCS405-1",
        "common": "QCS405.LE.1.0.1-00293-INT.APC.NAND-1",
        "glue": "GLUE.QCS405_LE.1.0-00117-NOOP_TEST-1",
        "rpm": "RPM.BF.1.9-00030-QCS405AAAAANAAR-1",
        "tz": "TZ.XF.5.1-00200-Q405AAAAANAZT-1",
        "wcnss_qz": "CNSS_W.QZ.4.0-00023-QZHW4024-1",
        "wlan": "WLAN.HL.3.1-01018-QCAHLSWMTPL-1"
    },
    "Metabuild_Info": {
        "Meta_Build_ID": "QCS405.LE.1.0.1-00293-INT.APC.NAND-1",
        "Product_Flavor": "['asic']",
        "Time_Stamp": "2019-04-28 21:52:32"
    },
    "Version": "1.0"
}/ #
/ #
```



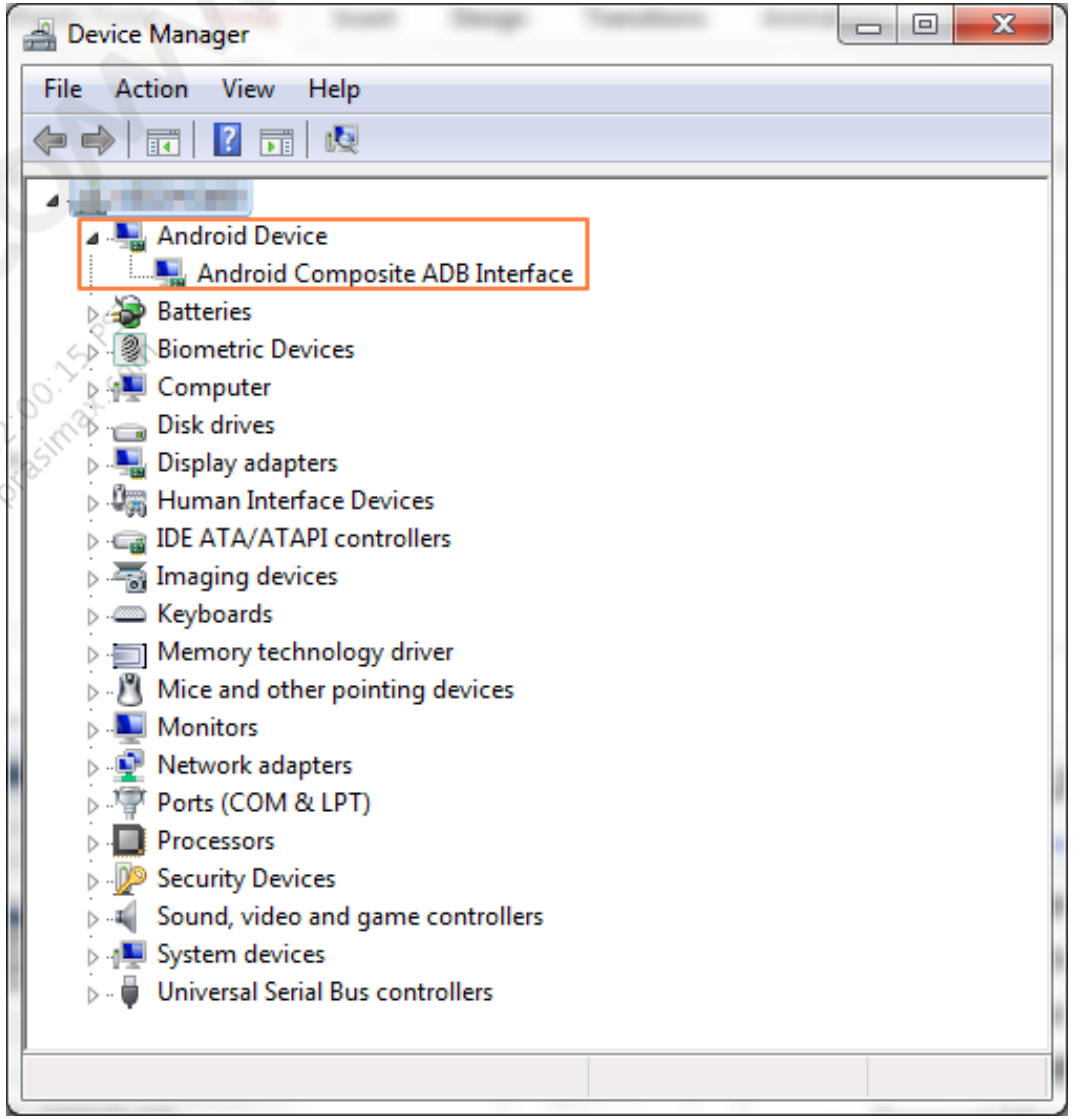
Section 8.2

Writing Device Configuration

Writing Device Configuration

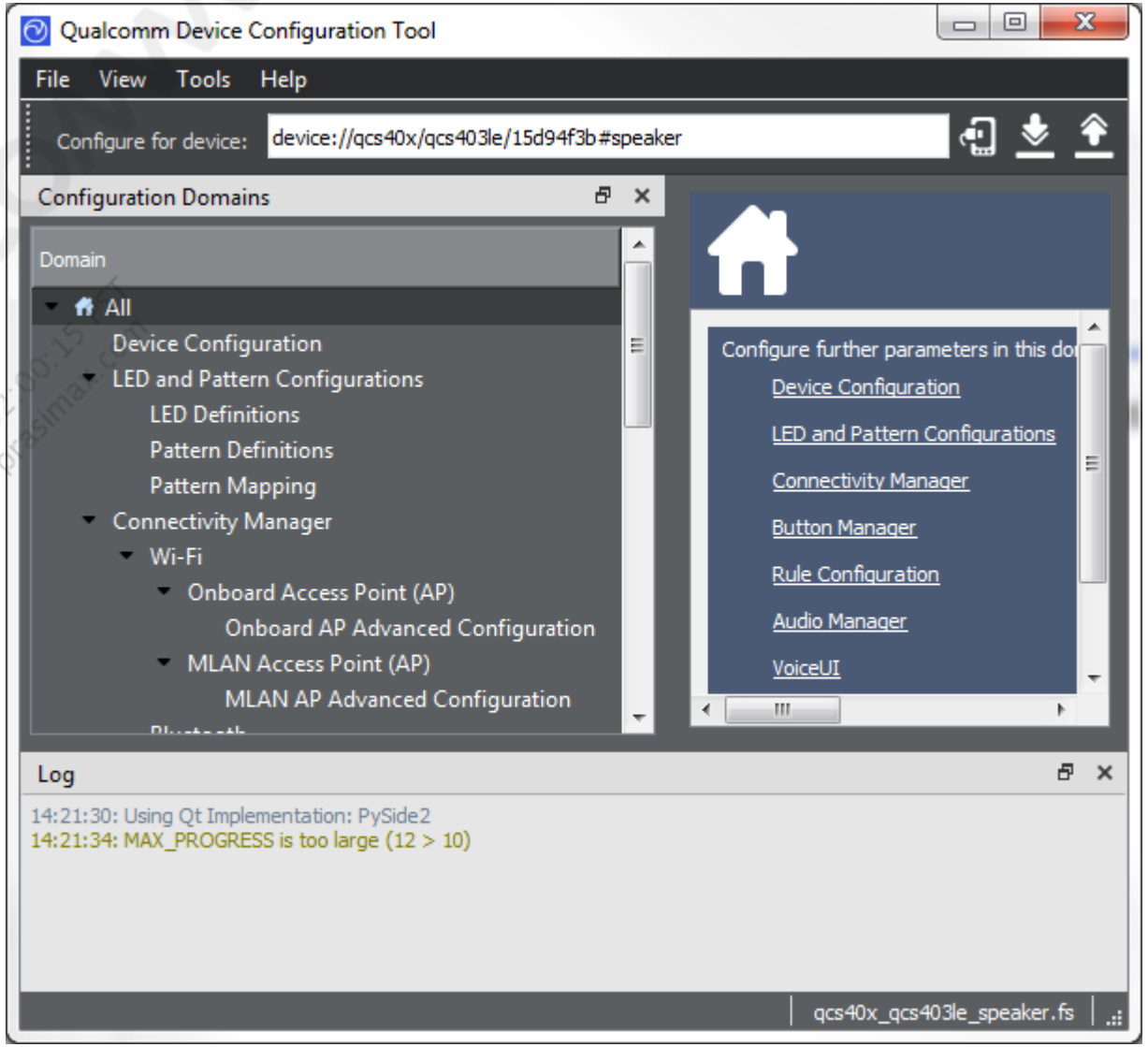
- To start many basic HLOS services and daemons, configuration data must be written to the device using Qualcomm Smart Audio Platform Configuration Tool

- To write configuration data, the device must be configured to normal boot mode
- Refer to:
  - Device Boot Configuration for SSRD – Normal boot mode



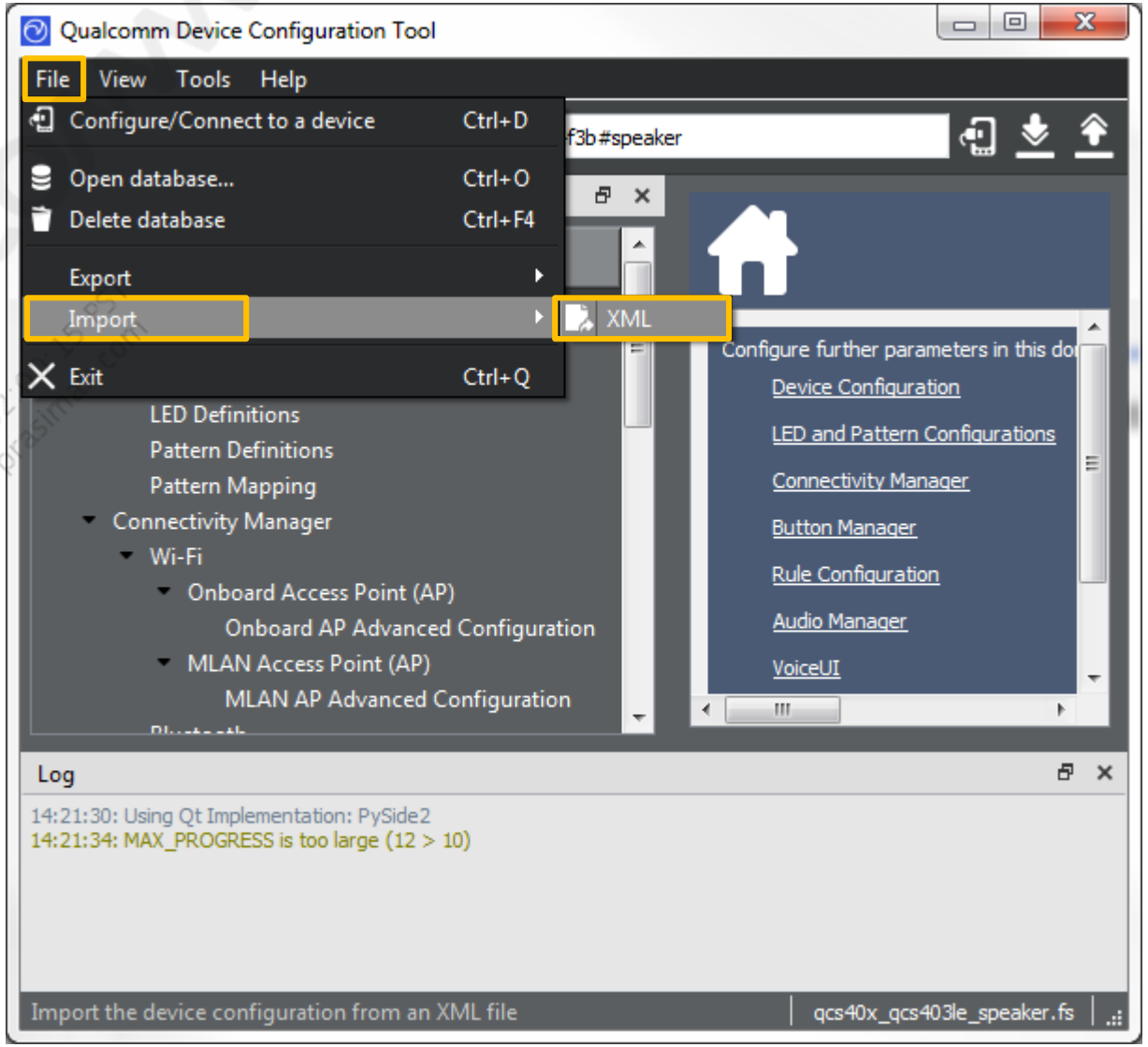
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool



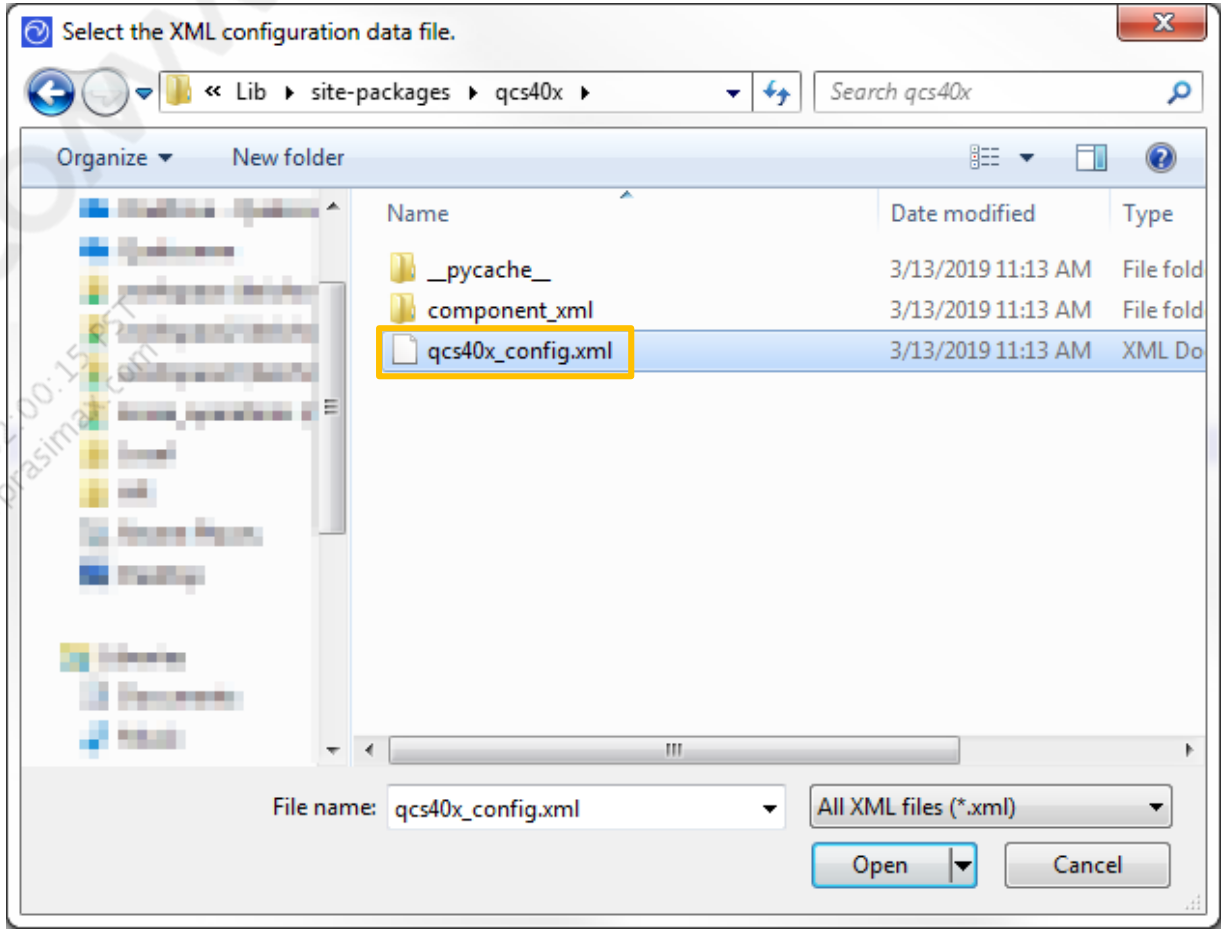
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu



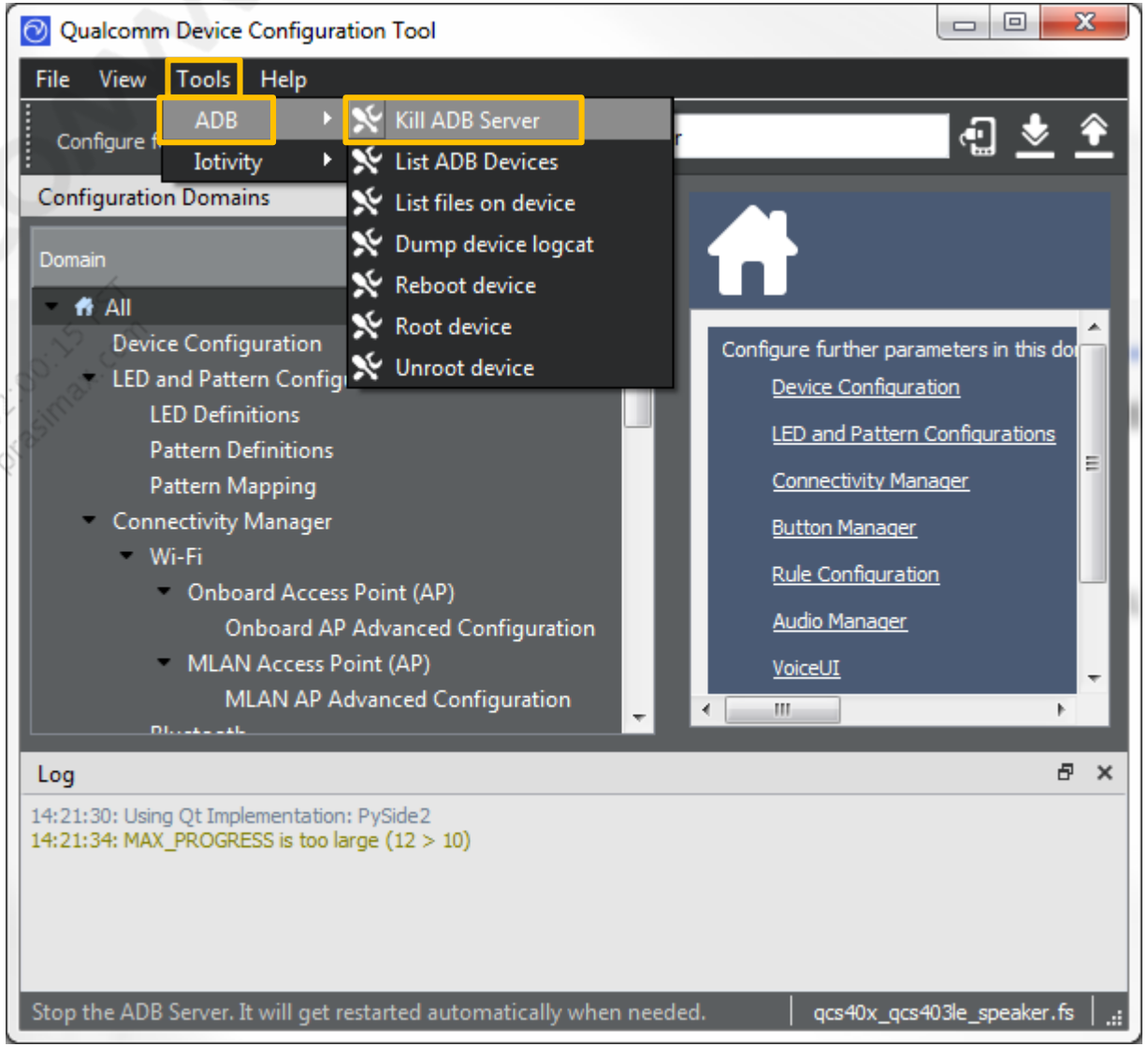
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x



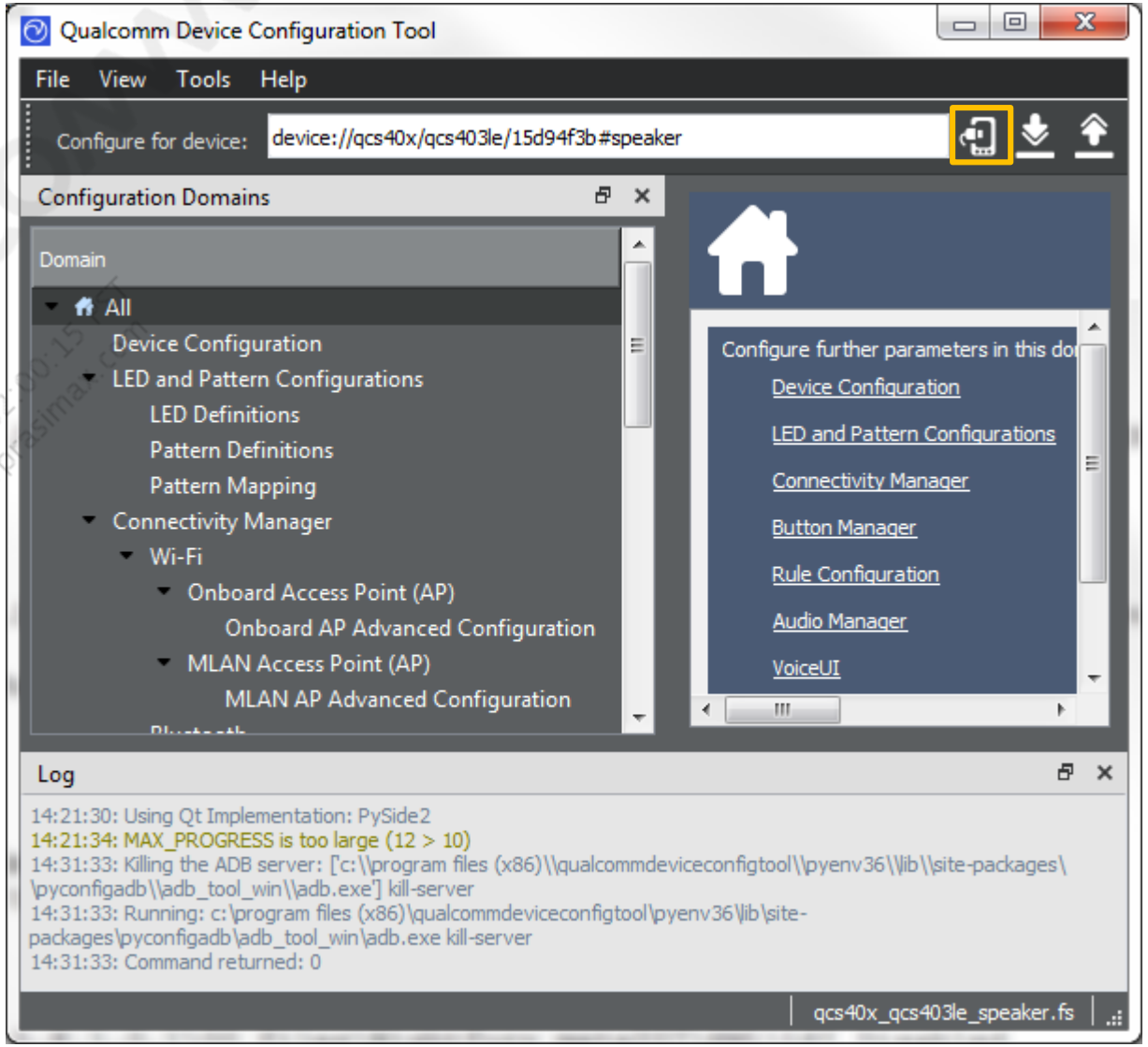
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu



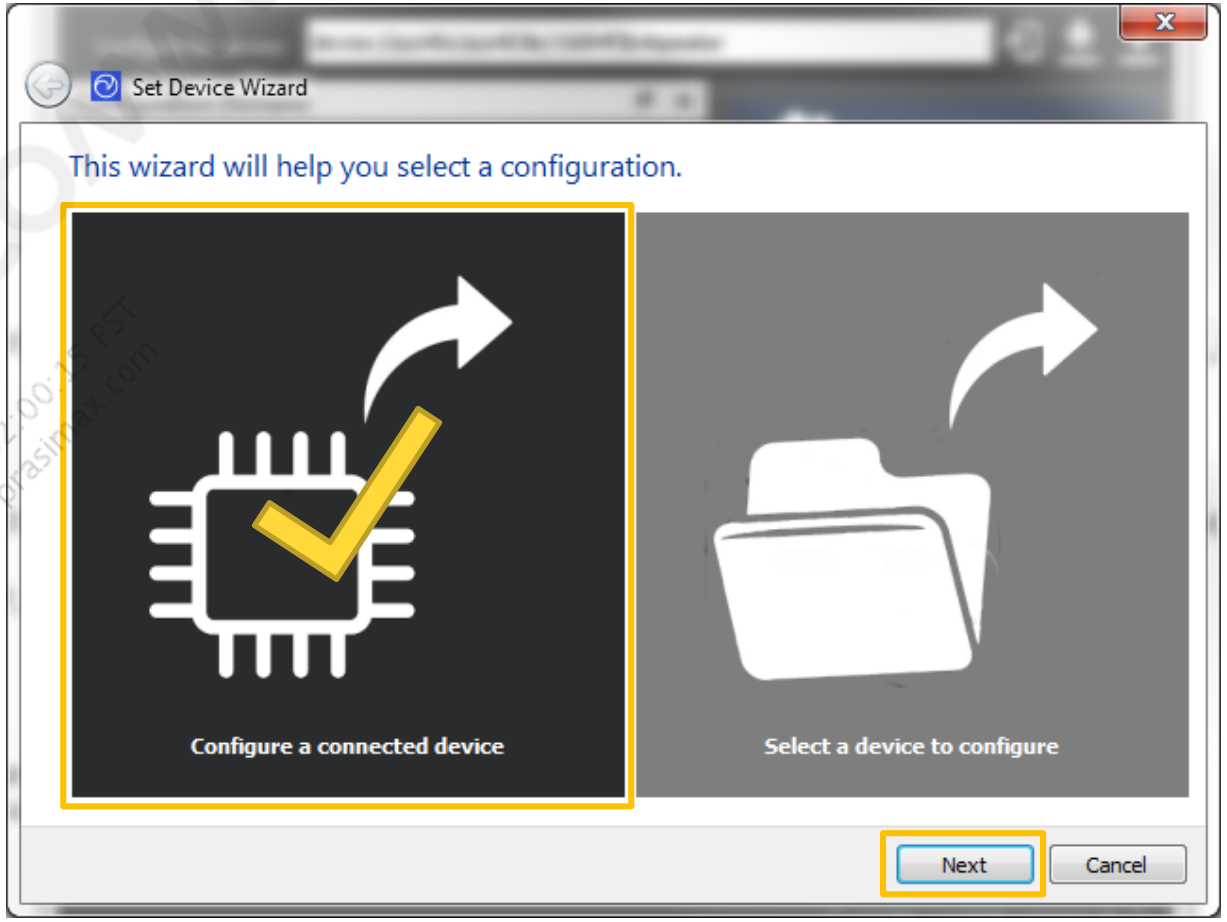
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu
- Click on ‘Choose the device to configure’ button



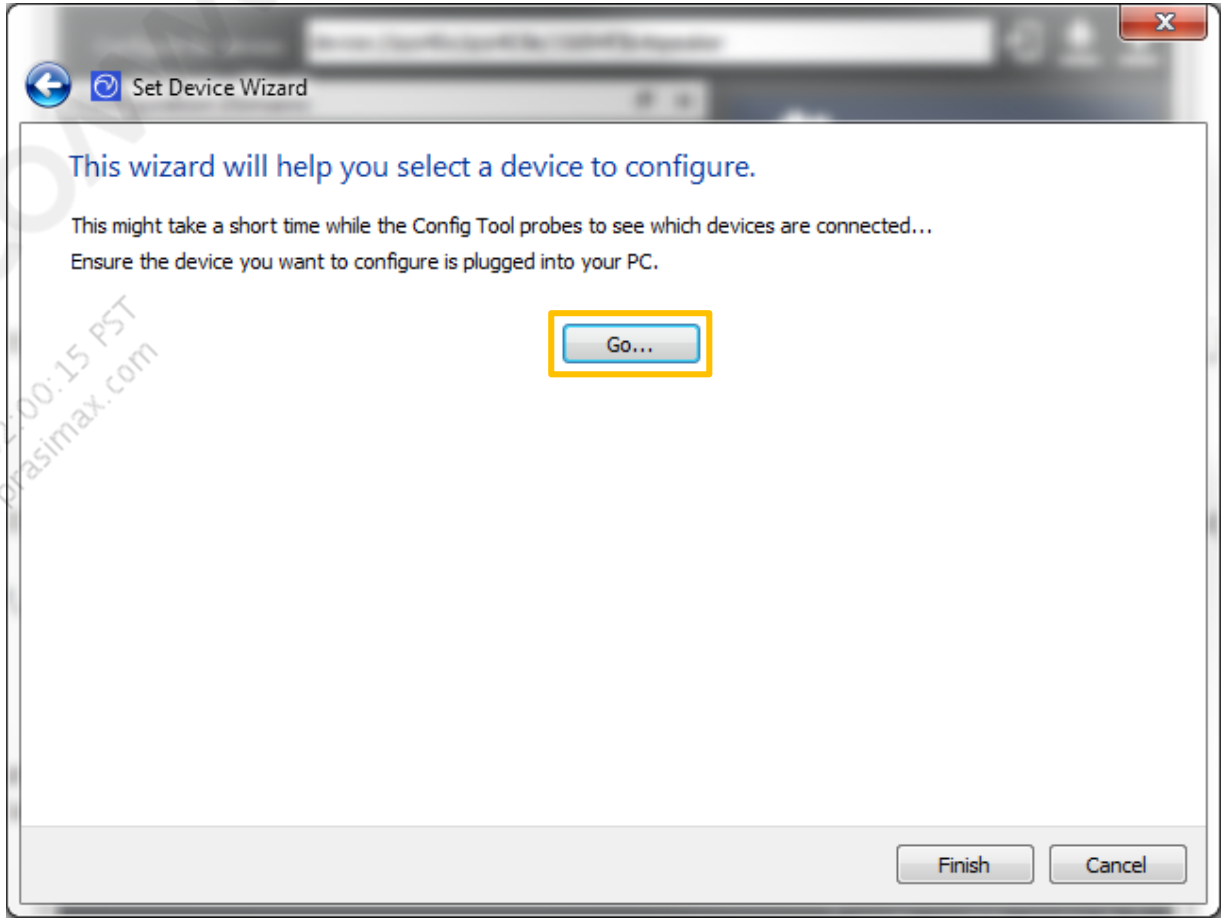
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu
- Click on ‘Choose the device to configure’ button
- Select ‘Configure a connected device’ and click on ‘Next’



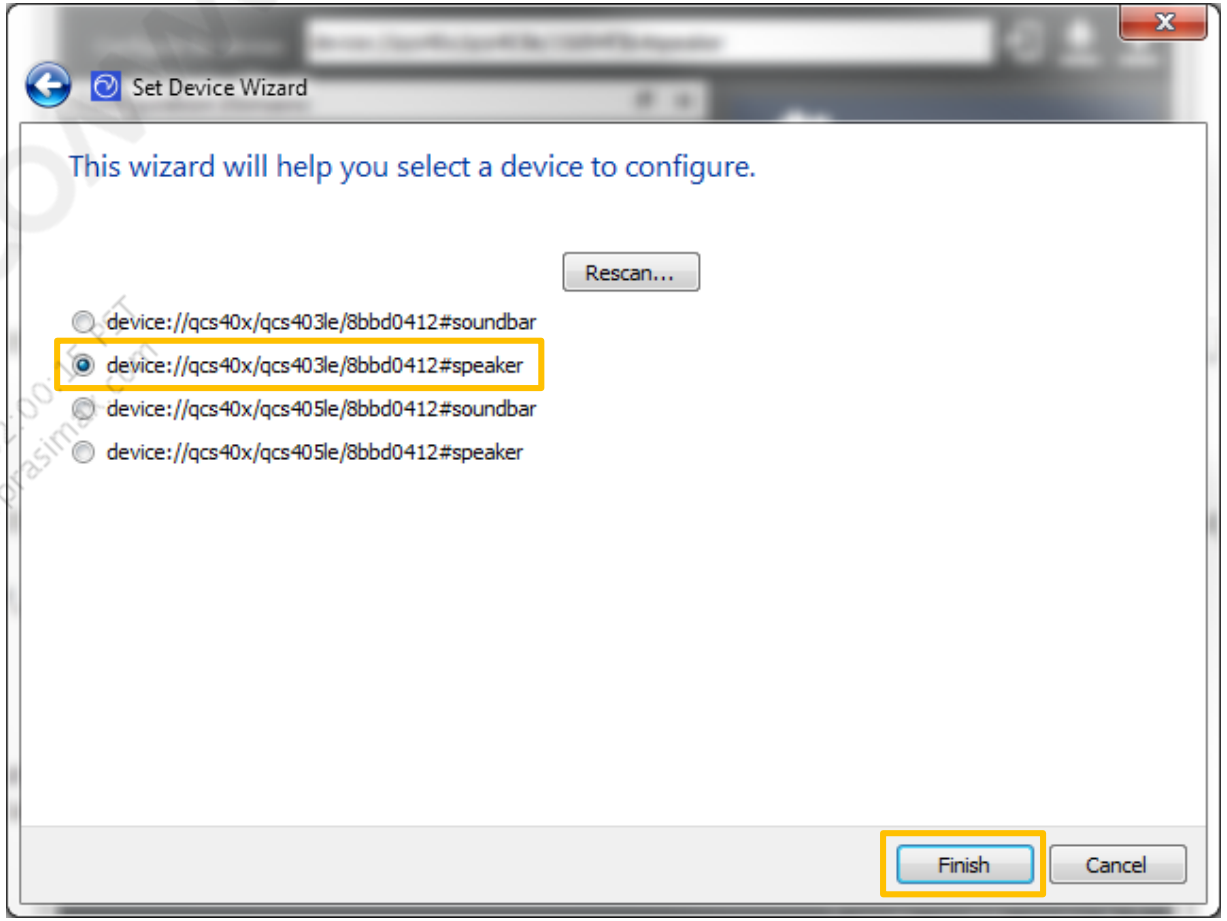
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu
- Click on ‘Choose the device to configure’ button
- Select ‘Configure a connected device’ and click on ‘Next’
- Click on ‘Go…’ button



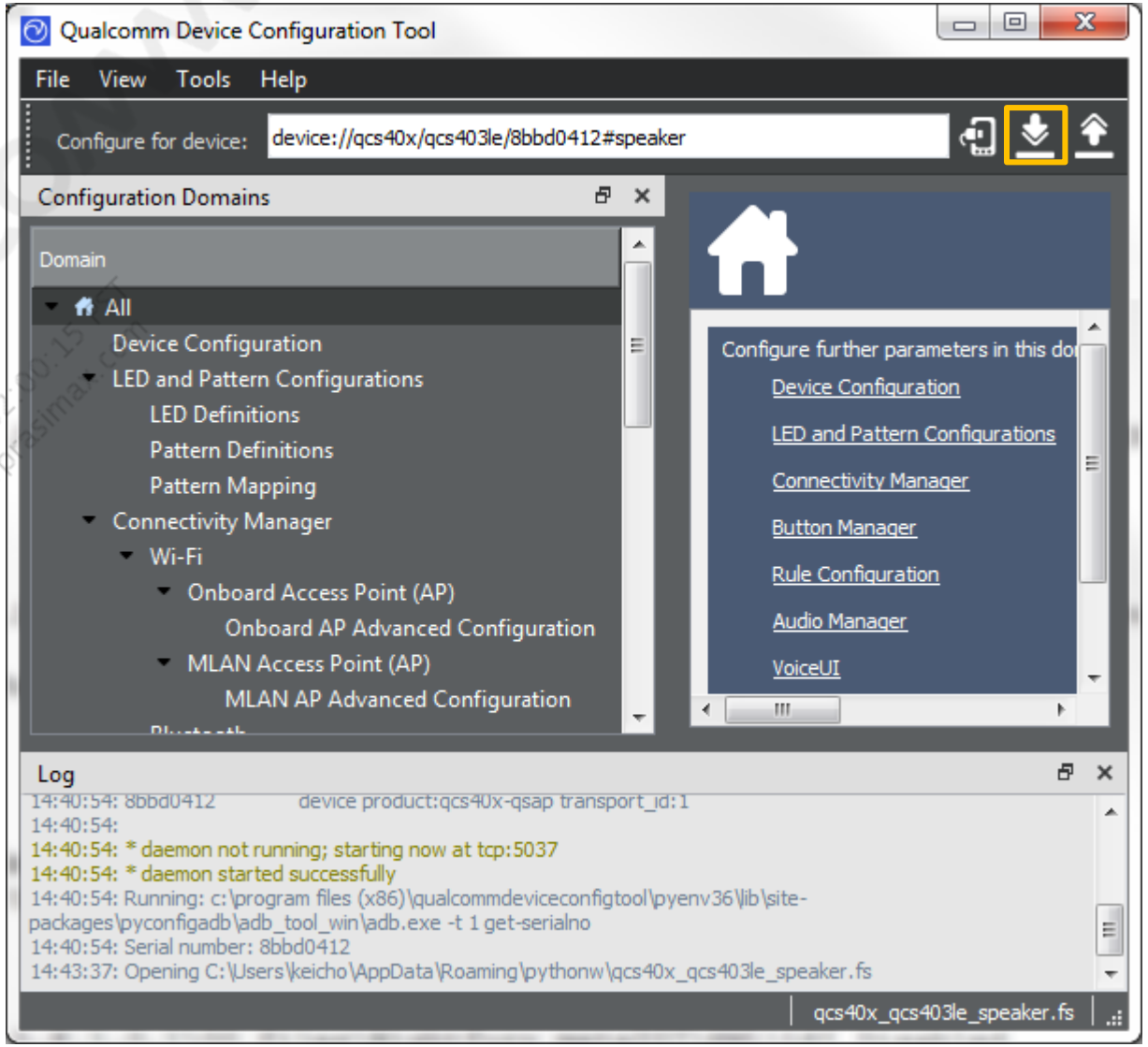
Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu
- Click on ‘Choose the device to configure’ button
- Select ‘Configure a connected device’ and click on ‘Next’
- Click on ‘Go…’ button
- Select ‘device://qcs40x/qcs403le/8bbd0412#speaker’ and click on ‘Finish’ button



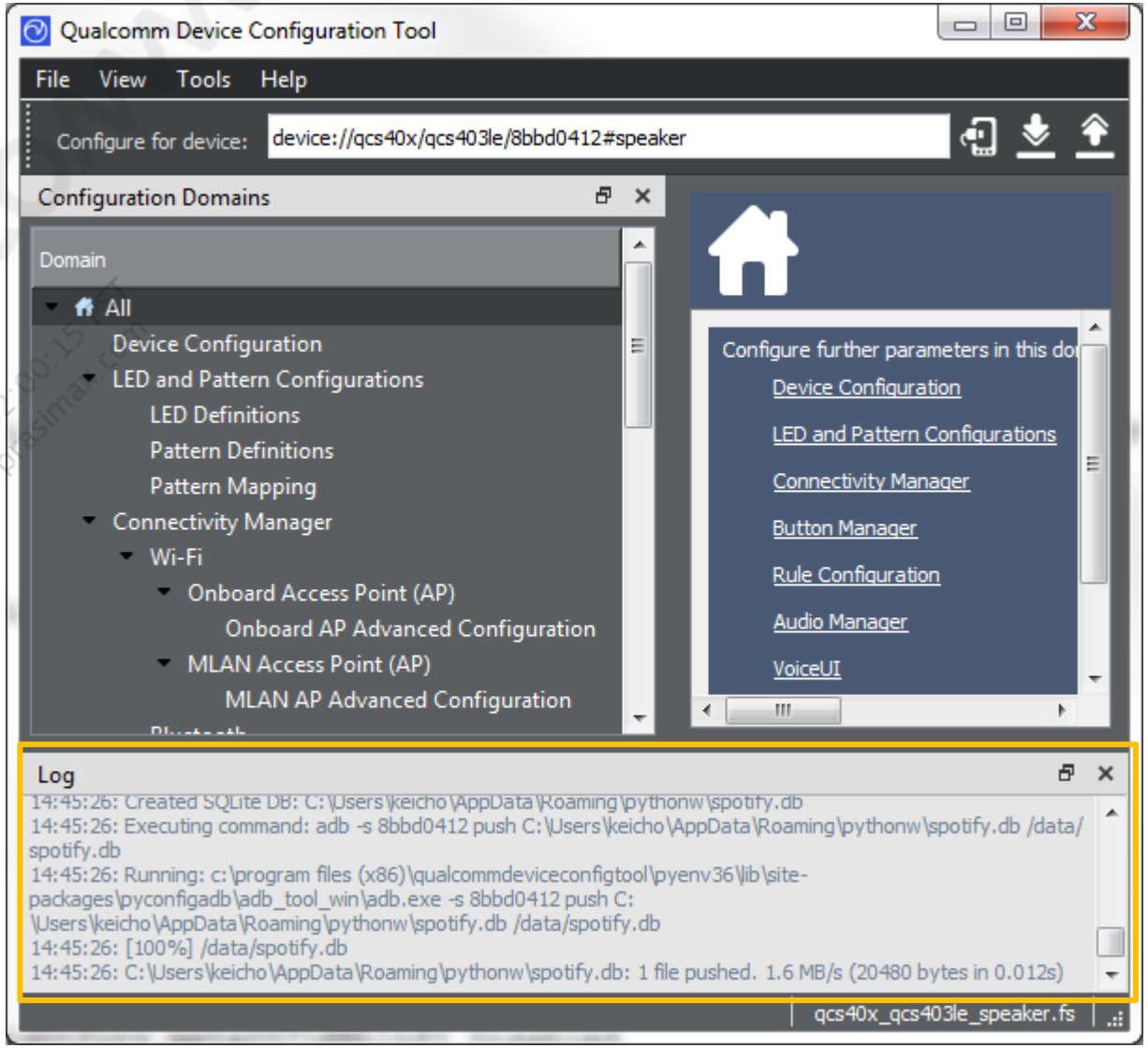
Writing Device Configuration

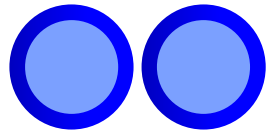
- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu
- Click on ‘Choose the device to configure’ button
- Select ‘Configure a connected device’ and click on ‘Next’
- Click on ‘Go…’ button
- Select ‘device://qcs40x/qcs403le/8bbd0412#speaker’ and click on ‘Finish’ button
- Click on ‘Push configuration to device’ button



Writing Device Configuration

- Launch Qualcomm Smart Audio Platform Configuration Tool
- Click on ‘File > Import > XML’ menu
- Open ‘qcs40x_config.xml’ file on the following folder:
  - C:\Program Files (x86)\QualcommDeviceConfigTool\pyenv36\Lib\site-packages\qcs40x
- Kill adb server with ‘Tools > ADB > Kill ADB Server’ menu
- Click on ‘Choose the device to configure’ button
- Select ‘Configure a connected device’ and click on ‘Next’
- Click on ‘Go…’ button
- Select ‘device://qcs40x/qcs403le/8bbd0412#speaker’ and click on ‘Finish’ button
- Click on ‘Push configuration to device’ button
- Make sure the result log and then reboot device
  - > adb reboot
- DONE !





Section 8.3

Setting up Haven License

Setting up Haven License

- In this chapter, it will be explained how to get Evaluation License
- Evaluation License allow to use the following audio features:
  - Far Field Voice (FFV)
  - Qualcomm® aptX™
  - Local ASR
  - Qualcomm® AllPlay™
- Below are the pre-defined Bluetooth Device Address for Evaluation License
  - 70:c9:4e:b7:f6:54
  - 70:c9:4e:7f:63:7a
  - 70:c9:4e:5b:b3:fe
  - 70:c9:4e:5b:b9:4e
  - 70:c9:4e:7f:6d:0e
- In this chapter, ‘70:c9:4e:b7:f6:54’ will be used as an example

- For more details, refer to *[80-YB405-129] QCS40x Software User Guide*

Setting up Haven License

1) Modify ‘bt_config.conf’ file

Modify ‘bt_config.conf’ file

```
// Run adb shell
> adb shell

// Enable Bluetooth
/ # btproperty &
/ # btapp
      gap_menu
      enable
      CTRL+C
      …
/ # killall btproperty
/ # exit

// Change Bluetooth address in bt_config.conf file
> adb pull /data/misc/bluetooth/bt_config.conf
==================================
…
[Adapter]
Address = 70:c9:4e:b7:f6:54
…
==================================
> adb push bt_config.conf /data/misc/bluetooth
> adb shell

// Change bt_config.conf access permission
/ # chmod 777 /data/misc/bluetooth/bt_config.conf
```

Setting up Haven License

1) Modify ‘bt_config.conf’ file
2) Change Bluetooth device address

Change device BT address

```
// Run adb shell
> adb shell

// Change Bluetooth device address
/ # setprop persist.vendor.service.bdroid.bdaddr 70:c9:4e:b7:f6:54

// Check Bluetooth device address is 70:c9:4e:b7:f6:54
/ # getprop persist.vendor.service.bdroid.bdaddr
70:c9:4e:b7:f6:54
```

Setting up Haven License

1) Modify ‘bt_config.conf’ file
2) Change Bluetooth device address
3) Check the license file is on the device, if not, push the license file to device and then reboot device
   - License file name: vipertooth_test_evaluation_license.pfm
   - File location: <QCS403_WS>/apps_proc/poky/meta-qti-audio-prop/recipes/alm/files/havenlicense/
   - License expiration: **June 30, 2019.**

Push license file to device

```
// Run adb shell
> adb shell

// Check the license file is in ‘/persist/data/pfm/licenses’
/ # ls /persist/data/pfm/licenses
900-930-jun-2019-770545770.pfm
// ****************************************************************
// If the license file does not exist, Run following steps
// ****************************************************************
// Create a folder where the license file is located to the device
/ # mkdir -p /persist/data/pfm/licenses
/ # exit

// Push license file
> adb push vipertooth_test_evaluation_license.pfm /persist/data/pfm/licenses
> adb shell

// Change access permission
/ # chmod -R 777 /persist/data
/ # exit

// Reboot device,
// NOTE: DO NOT USE ‘adb shell reboot’
> adb reboot
```

e.g. In LE.1.0 CS, the license file already exist in device

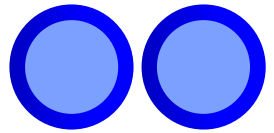
Setting up Haven License

1) Modify ‘bt_config.conf’ file
2) Change Bluetooth device address
3) Check the license file is on the device, if not, push the license file to device and then reboot device
   - License file name: vipertooth_test_evaluation_license.pfm
   - File location: <QCS403_WS>/apps_proc/poky/meta-qti-audio-prop/recipes/alm/files/havenlicense/
   - License expiration: **June 30, 2019.**
4) After reboot, check Bluetooth device address is 70:c9:4e:b7:f6:54

- DONE !

Check device BT address

```
// Check Bluetooth device address
> adb shell getprop persist.vendor.service.bdroid.bdaddr
70:c9:4e:b7:f6:54
```



Section 8.4

Onboarding Wi-Fi

Onboarding Wi-Fi

- There are several methods for Wi-Fi Onboarding:
  - Using QCMAP_CLI application (adb)
  - Using wpa_supplicant application (adb)
  - Using ADK message (adb)
  - Using Qualcomm Smart Audio App (Android, iOS)
  - Using Web browser (PC, SmartPhone, and so on…)

- In this chapter, ‘Using ADK message’ will be explained

Onboarding Wi-Fi

- If the default configuration was written successfully, the device's Wi-Fi is in 'STA + SoftAP' mode
  - / # ifconfig

```
/ # ifconfig
bridge0   Link encap:Ethernet  HWaddr 82:A4:A0:15:7A:AB
          inet addr:192.168.225.1  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::80a4:a0ff:fe12:77a8/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:3642 (3.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1960 (1.9 KiB)  TX bytes:1960 (1.9 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:0A:F5:BD:04:12
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan1     Link encap:Ethernet  HWaddr 02:0A:F5:42:04:12
          inet addr:169.254.1.1  Bcast:169.254.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a:f5ff:fe42:412/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:7 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:0 (0.0 B)  TX bytes:4853 (4.7 KiB)

/ #
```

Onboarding Wi-Fi

- If the default configuration was written successfully, the device's Wi-Fi is in 'STA + SoftAP' mode
  - / # ifconfig
- Start Wi-Fi Onboarding with following commands
  - / # adk-message-send 'connectivity_wifi_disable{}'
  - / # adk-message-send 'connectivity_wifi_enable{}'
  - / # adk-message-send 'connectivity_wifi_scan{}'
  - / # adk-message-send 'connectivity_wifi_connect {ssid:"MY_AP_SSID" password:"MY_AP_PASSWORD" homeap:true}'

```
/ # adk-message-send 'connectivity_wifi_disable{}'
Sending message:
connectivity_wifi_disable {
}
Attempted to unregister path (path[0] = com path[1] = qualcomm) which isn't registered
/ # adk-message-send 'connectivity_wifi_enable{}'
Sending message:
connectivity_wifi_enable {
}
Attempted to unregister path (path[0] = com path[1] = qualcomm) which isn't registered
/ # adk-message-send 'connectivity_wifi_scan{}'
Sending message:
connectivity_wifi_scan {
}
Attempted to unregister path (path[0] = com path[1] = qualcomm) which isn't registered
/ # adk-message-send 'connectivity_wifi_connect {ssid:"V30" password:"11112222"
homeap:true}'
Sending message:
connectivity_wifi_connect {
  ssid: "V30"
  password: "11112222"
  homeap: true
}
Attempted to unregister path (path[0] = com path[1] = qualcomm) which isn't registered
/ #
```

Onboarding Wi-Fi

- If the default configuration was written successfully, the device's Wi-Fi is in 'STA + SoftAP' mode
  - / # ifconfig
- Start Wi-Fi Onboarding with following commands
  - / # adk-message-send 'connectivity_wifi_disable{}'
  - / # adk-message-send 'connectivity_wifi_enable{}'
  - / # adk-message-send 'connectivity_wifi_scan{}'
  - / # adk-message-send 'connectivity_wifi_connect {ssid:"MY_AP_SSID" password:"MY_AP_PASSWORD" homeap:true}'
- Check 'wlan0' gets the IP address
  - / # ifconfig
  - / # ping 8.8.8.8

```
/ # ifconfig
bridge0   Link encap:Ethernet  HWaddr 82:A4:A0:15:7A:AB
          inet addr:192.168.225.1  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::80a4:a0ff:fe12:77a8/64 Scope:Link
          UP BROADCAST RUNNING PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:966 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:282 (282.0 B)  TX bytes:285517 (278.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1736 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1736 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:145888 (142.4 KiB)  TX bytes:145888 (142.4 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:0A:F5:BD:04:12
          inet addr:192.168.43.241  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:f5ff:febd:412/64 Scope:Link
          UP BROADCAST RUNNING PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:243 errors:0 dropped:0 overruns:0 frame:0
          TX packets:258 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:32429 (31.6 KiB)  TX bytes:32433 (31.6 KiB)

wlan1     Link encap:Ethernet  HWaddr 02:0A:F5:42:04:12
          inet addr:169.254.1.1  Bcast:169.254.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a:f5ff:fe42:412/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:98 errors:0 dropped:7 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:0 (0.0 B)  TX bytes:25350 (24.7 KiB)

/ # ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=99.1 ms
```

Onboarding Wi-Fi

- If the default configuration was written successfully, the device's Wi-Fi is in 'STA + SoftAP' mode
  - / # ifconfig
- Start Wi-Fi Onboarding with following commands
  - / # adk-message-send 'connectivity_wifi_disable{}'
  - / # adk-message-send 'connectivity_wifi_enable{}'
  - / # adk-message-send 'connectivity_wifi_scan{}'
  - / # adk-message-send 'connectivity_wifi_connect {ssid:"MY_AP_SSID" password:"MY_AP_PASSWORD" homeap:true}'
- Check 'wlan0' gets the IP address
  - / # ifconfig
  - / # ping 8.8.8.8
- Complete Wi-Fi Onboarding with following command
  - / # adk-message-send 'connectivity_wifi_completeonboarding{}'
- Wi-Fi will be reset automatically

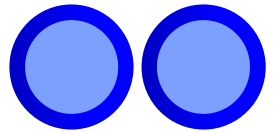
```
/ # adk-message-send 'connectivity_wifi_completeonboarding{}'
Sending message:
connectivity_wifi_completeonboarding {
}
Attempted to unregister path (path[0] = com path[1] = qualcomm) which isn't registered
/ # ifconfig
bridge0   Link encap:Ethernet  HWaddr 82:A4:A0:15:7A:AB
          inet addr:192.168.225.1  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::80a4:a0ff:fe12:77a8/64 Scope:Link
          UP BROADCAST PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1152 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:282 (282.0 B)  TX bytes:343880 (335.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1736 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1736 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:145888 (142.4 KiB)  TX bytes:145888 (142.4 KiB)
```

'wlan' disappears for a few seconds

Onboarding Wi-Fi

- If the default configuration was written successfully, the device's Wi-Fi is in 'STA + SoftAP' mode
  - / # ifconfig
- Start Wi-Fi Onboarding with following commands
  - / # adk-message-send 'connectivity_wifi_disable{}'
  - / # adk-message-send 'connectivity_wifi_enable{}'
  - / # adk-message-send 'connectivity_wifi_scan{}'
  - / # adk-message-send 'connectivity_wifi_connect {ssid:"MY_AP_SSID" password:"MY_AP_PASSWORD" homeap:true}'
- Check 'wlan0' gets the IP address
  - / # ifconfig
  - / # ping 8.8.8.8
- Complete Wi-Fi Onboarding with following command
  - / # adk-message-send 'connectivity_wifi_completeonboarding{}'
- Wi-Fi will be reset automatically
- Wait until Wi-Fi reconnects to MY_AP with 'STA only' mode
- DONE !

```
/ # ifconfig
bridge0   Link encap:Ethernet  HWaddr 82:A4:A0:15:7A:AB
          inet addr:192.168.225.1  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::80a4:a0ff:fe12:77a8/64 Scope:Link
          UP BROADCAST PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1152 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:282 (282.0 B)  TX bytes:343880 (335.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1736 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1736 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:145888 (142.4 KiB)  TX bytes:145888 (142.4 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:0A:F5:BD:04:12
          inet addr:192.168.43.241  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: 2001:2d8:e204:b77b:20a:f5ff:febd:412/64 Scope:Global
          inet6 addr: fe80::20a:f5ff:febd:412/64 Scope:Link
          UP BROADCAST RUNNING PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:466 errors:0 dropped:0 overruns:0 frame:0
          TX packets:488 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:69023 (67.4 KiB)  TX bytes:68527 (66.9 KiB)

/ #
```

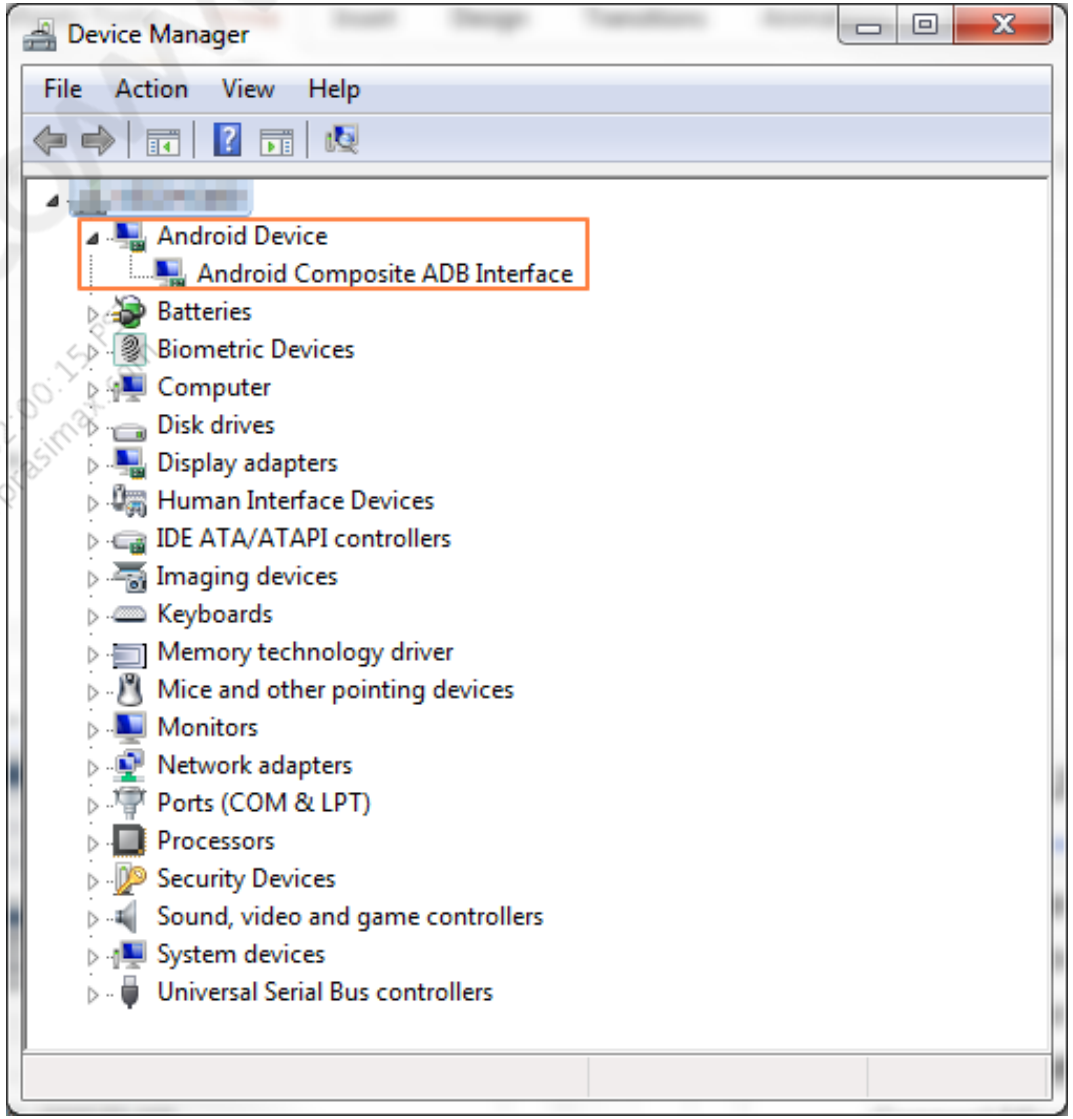


Section 8.5

Onboarding AVS

Onboarding AVS – Prerequisite

- Amazon User Account is required to onboard AVS
- The following below should have already been performed
  - Writing Device Configuration
  - Haven License
  - Onboarding Wi-Fi
- The device must be configured to normal boot mode
  - Refer to: Device Boot Configuration for SSRD – Normal boot mode

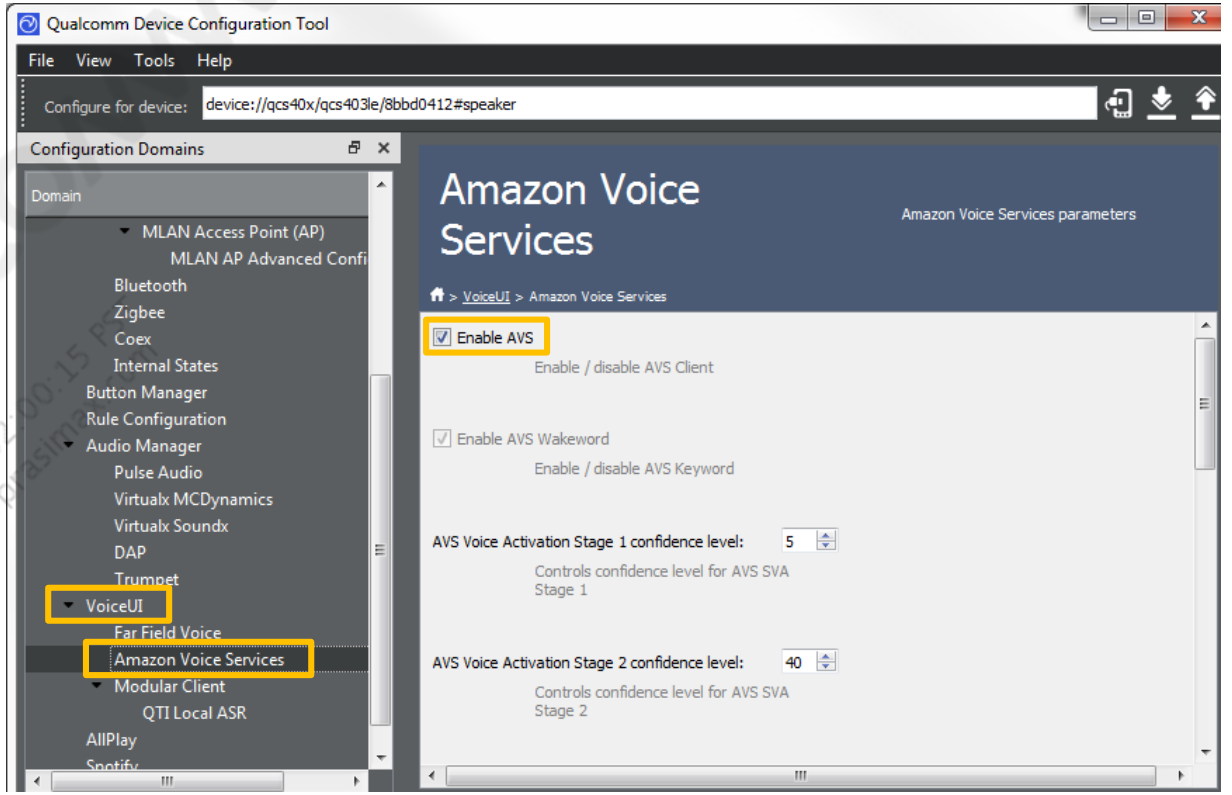


Onboarding AVS – Enable AVS Configuration

- AVS service is enabled in default configuration XML file, so AVS service would have already been enabled through the Writing Device Configuration step

- To check AVS service is enabled in default XML file:
  - Open Device Configuration Tool
  - Import default XML file
  - Click on 'VoiceUI > Amanzon Voice Services' item
  - Make sure the 'Enable AVS' is check ON

- If the 'Enable AVS' is checked OFF then check ON the 'Enable AVS' and write again the updated configuration



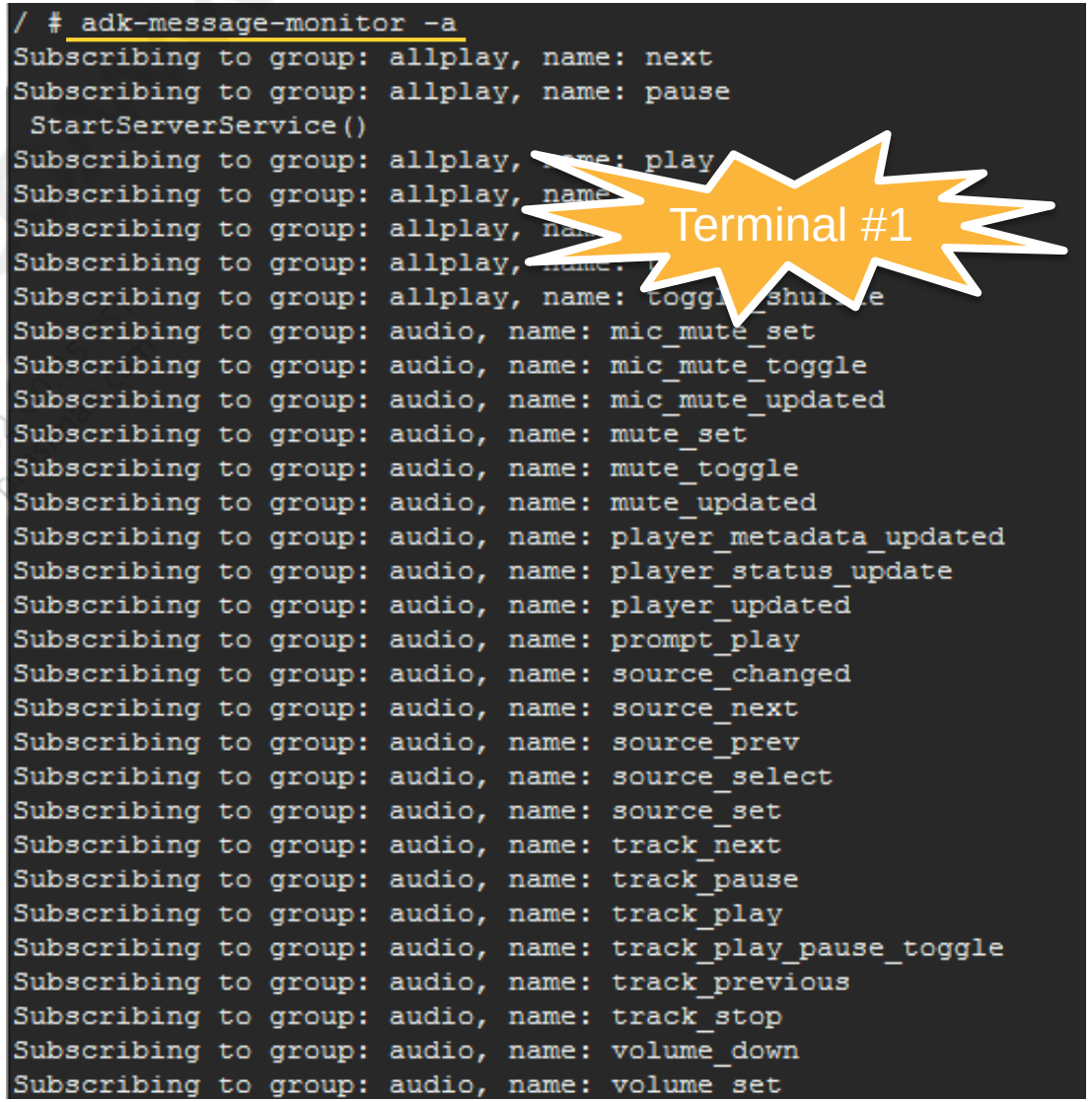
Onboarding AVS

1. Check Wi-Fi connection and VoiceUISampleApp process is running
   - / # ifconfig
   - / # ps | grep Voice

```
/ # ifconfig
bridge0   Link encap:Ethernet  HWaddr 0A:AC:04:5C:EC:0A
          inet addr:192.168.225.1  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::8ac:4ff:fe59:e907/64 Scope:Link
          UP BROADCAST PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:90 (90.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8368 (8.1 KiB)  TX bytes:8368 (8.1 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:0A:F5:BD:04:12
          inet addr:192.168.43.241  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: 2001:2d8:eb5c:ac2d:20a:f5ff:febd:412/64 Scope:Global
          inet6 addr: fe80::20a:f5ff:febd:412/64 Scope:Link
          UP BROADCAST RUNNING PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:40 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:5090 (4.9 KiB)  TX bytes:5373 (5.2 KiB)

/ # ps | grep Voice
 1656 root       0:00 /usr/bin/VoiceUISampleApp
 2011 root       0:00 {grep} /bin/busybox /bin/grep Voice
/ #
```

Onboarding AVS

1. Check Wi-Fi connection and VoiceUISampleApp process is running
   - / # ifconfig
   - / # ps | grep Voice
2. Terminal #1: Run ADK Message Monitor Application
   - / # adk-message-monitor -a



Onboarding AVS

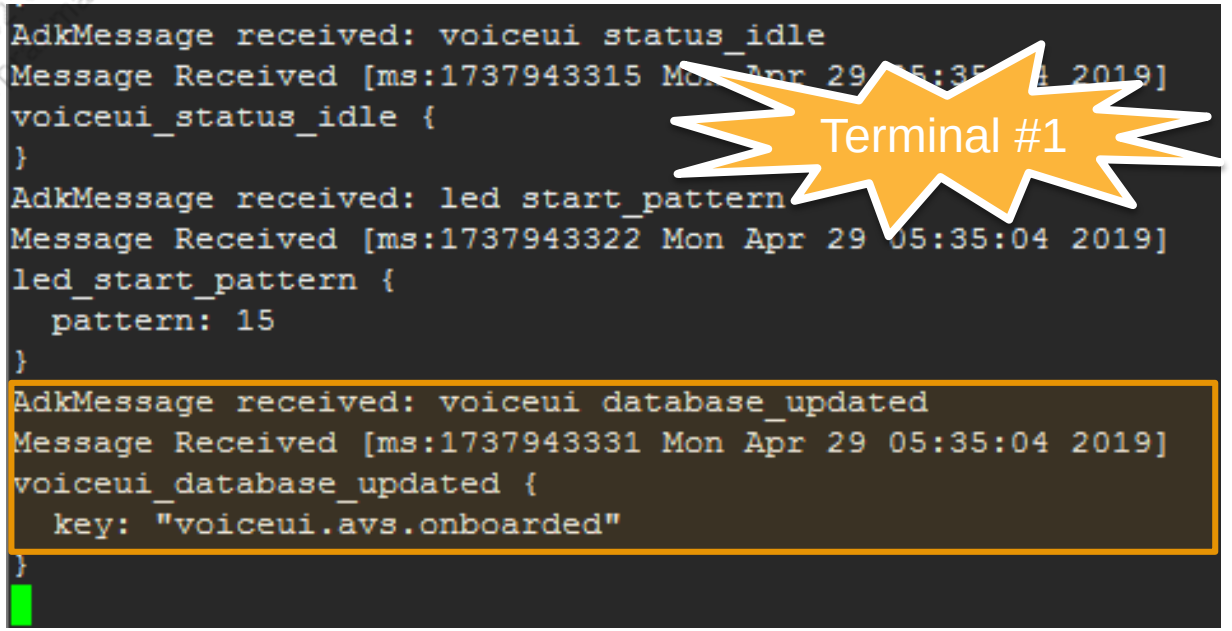
1. Check Wi-Fi connection and VoiceUISampleApp process is running
   - / # ifconfig
   - / # ps | grep Voice
2. Terminal #1: Run ADK Message Monitor Application
   - / # adk-message-monitor -a
3. Open another terminal window for adb shell - Terminal #2
4. Terminal #2: Run Onboarding AVS start command
   - / # adk-message-send 'voiceui_start_onboarding {client:"AVS"}'
5. Terminal #1: Check ‘code’ value

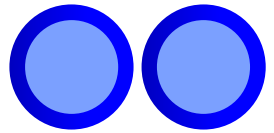
Onboarding AVS

1. Check Wi-Fi connection and VoiceUISampleApp process is running
   - / # ifconfig
   - / # ps | grep Voice
2. Terminal #1: Run ADK Message Monitor Application
   - / # adk-message-monitor -a
3. Open another terminal window for adb shell - Terminal #2
4. Terminal #2: Run Onboarding AVS start command
   - / # adk-message-send 'voiceui_start_onboarding {client:"AVS"}'
5. Terminal #1: Check 'code' value
6. Open Web Browser and Go: https://amazon.com/us/code
   - Login with User's Amazon Account
   - Input the 'code' value and click on 'Continue' button

Onboarding AVS

1. Check Wi-Fi connection and VoiceUISampleApp process is running
   - / # ifconfig
   - / # ps | grep Voice
2. Terminal #1: Run ADK Message Monitor Application
   - / # adk-message-monitor -a
3. Open another terminal window for adb shell - Terminal #2
4. Terminal #2: Run Onboarding AVS start command
   - / # adk-message-send 'voiceui_start_onboarding {client:"AVS"}'
5. Terminal #1: Check 'code' value
6. Open Web Browser and Go: https://amazon.com/us/code
   - Login with User's Amazon Account
   - Input the 'code' value and click on 'Continue' button
7. Make sure 'voiceui.avs.onboarded' message on Terminal #1
8. DONE ! Say "Alexa"





Section 8.6

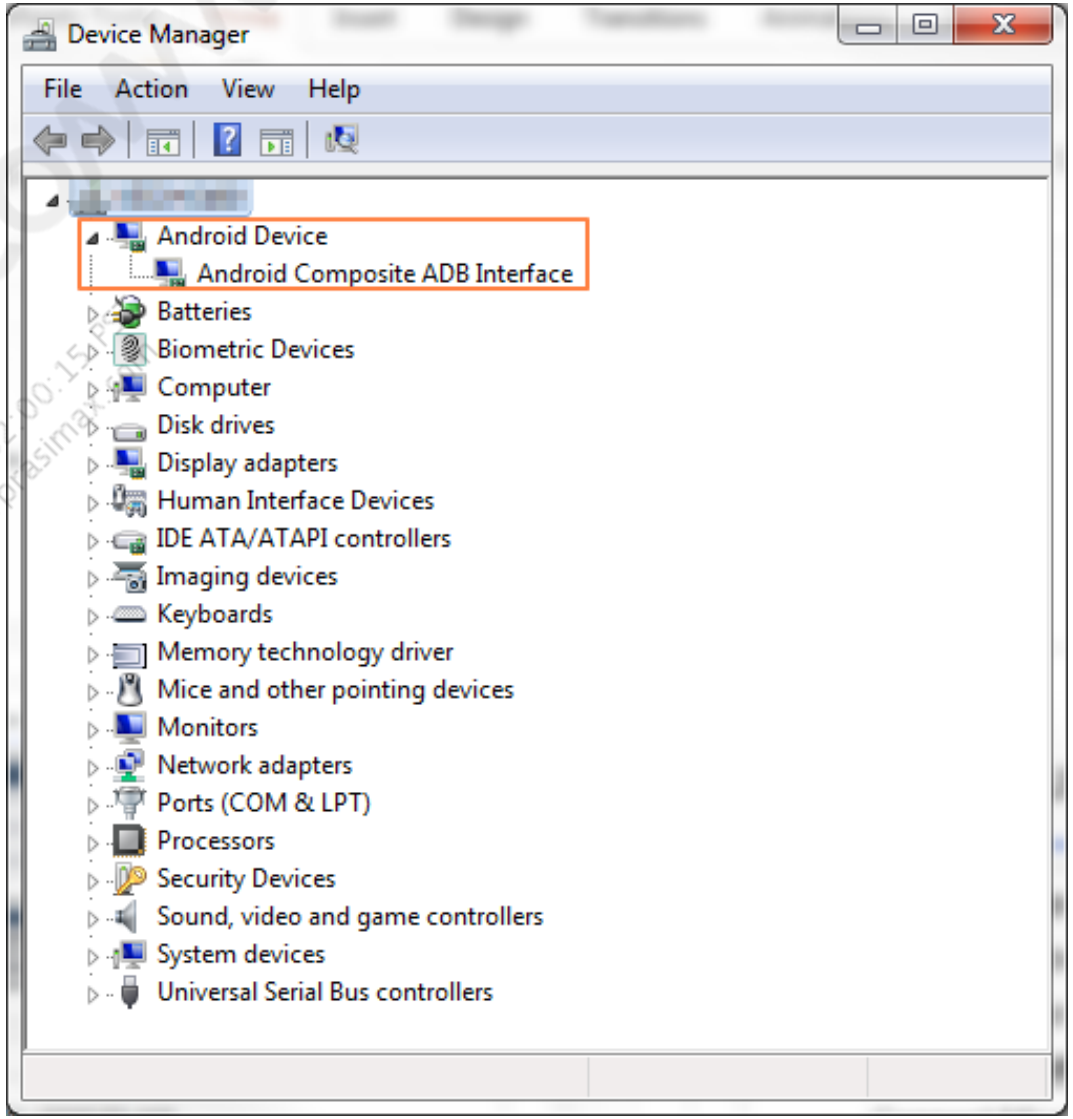
Local ASR + AVS Test

Local ASR + AVS Test – Prerequisite

- AVS is already enabled on Onboarding AVS chapter
- Local ASR is enabled by default function of VoiceUI

- The following below should have already been performed
  - Writing Device Configuration
  - Haven License

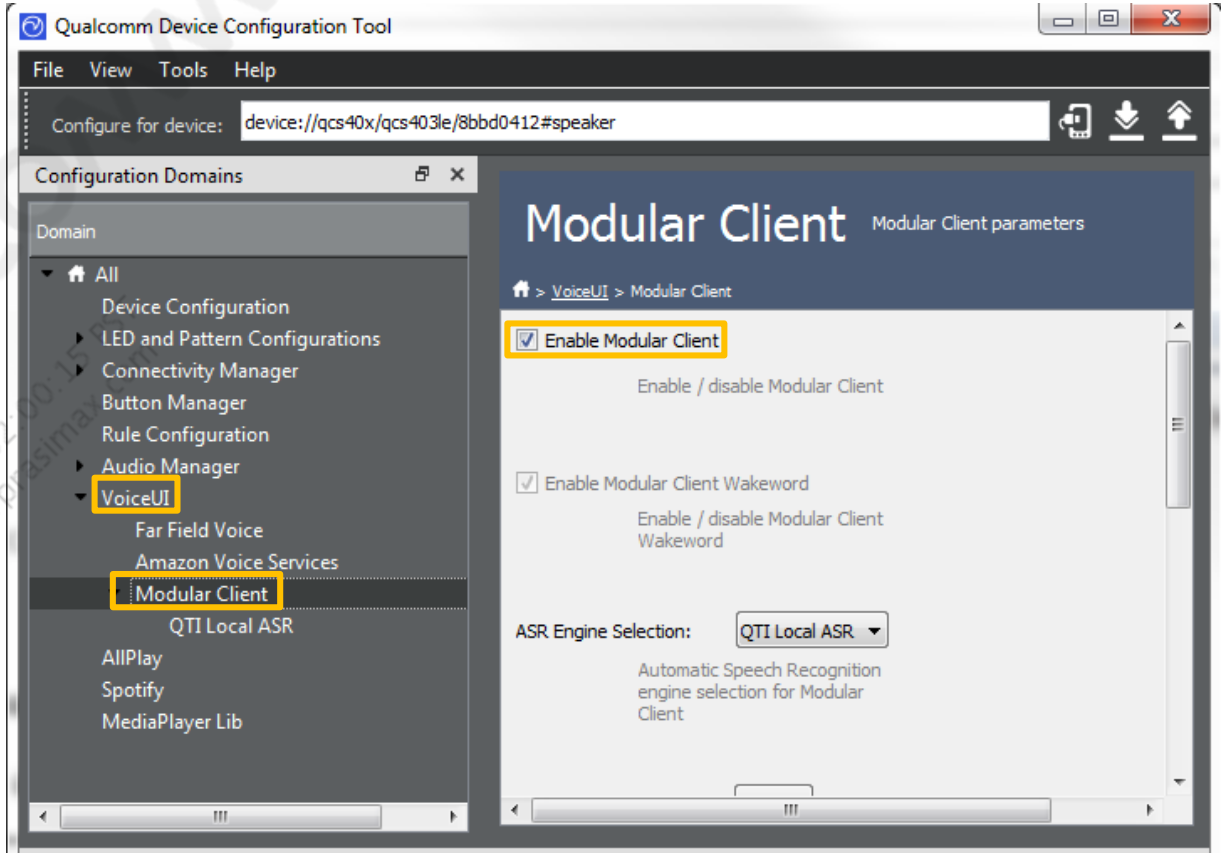
- The device must be configured to normal boot mode
  - Refer to: Device Boot Configuration for SSRD – Normal boot mode



Local ASR + AVS Test – Enable Local ASR Configuration

- Local ASR service is enabled in default configuration XML file, so Local ASR service would have already been enabled through the Writing Device Configuration step

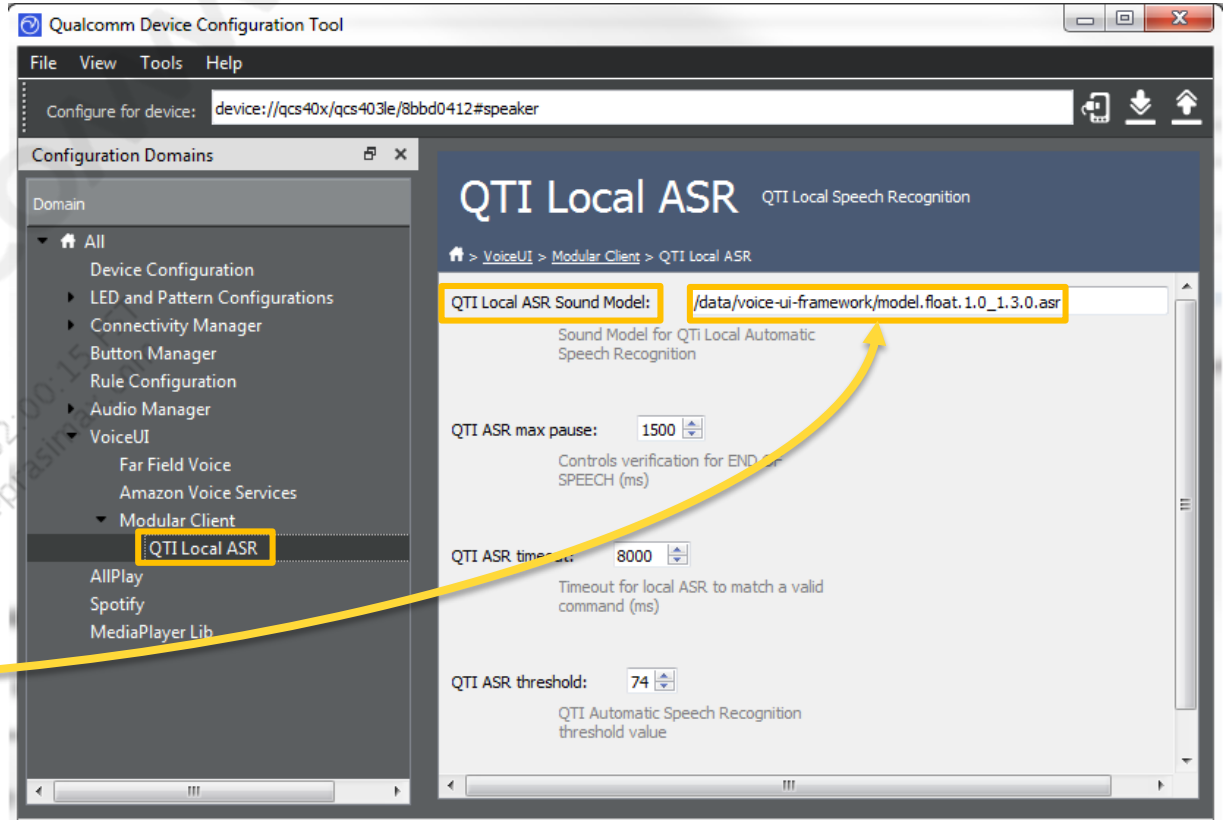
- To check Local ASR service is enabled in default XML file:
  - Open Device Configuration Tool
  - Import default XML file
  - Click on ‘VoiceUI > Modular Client’ item
  - Make sure the ‘Modular Client’ is check ON



Local ASR + AVS Test – Enable Local ASR Configuration

- In addition, make sure the QTI Local ASR Sound model file is in device
  - Click on ‘VoiceUI > Modular Client > QTI Local ASR’ item
  - Check ‘QTI Local ASR Sound Model’ file path
  - Check the file name in device

```
/data/voice-ui-framework # ls
Bedroom_Lights_OFF.mp3          Lights_OFF.mp3
Bedroom_Lights_ON.mp3           Lights_ON.mp3
Call_Ended.mp3                  Living_Room_Lights_OFF.mp3
Device_has_been_muted.mp3       Living_Room_Lights_ON.mp3
Device_has_been_unmuted.mp3     Thermostat_OFF.mp3
Front_Door_Closed.mp3           Thermostat_ON.mp3
Front_Door_Opened.mp3           Thermostat_temperature_has_changed.mp3
Garage_Door_Closed.mp3          Volume_Decreased.mp3
Garage_Door_Opened.mp3          Volume_Increased.mp3
HeySnapdragon.uim               Volume_changed.mp3
Incoming_call_Rejected.mp3      model.float.1.0_1.3.0.asr
Incoming_call_accepted.mp3      sm3_gmm_0703_cnn_jan08_Alexa.uim
Kitchen_Lights_ON.mp3           voiceDumpAVS
Kitchen_lights_OFF.mp3          voiceDumpLocalASR
/data/voice-ui-framework #
```



- If the file name on ConfigTool is not match with the file name in device, then modify the ConfigTool’s file name and write again the updated configuration

Local ASR + AVS Test

- Now AVS and Local ASR are available
- Reboot device and wait until Wi-Fi is connected

- Run ADK monitor application
  - adk-message-monitor -a
- Run Local ASR monitor
  - watch -n 2 "grep -i jston_str /var/log/messages | tail -1"

- Voice activation keywords are
  - For AVS: “Alexa”
  - For Local ASR: “Hey SnapDragon”

For example, Say
“Turn on the Kitchen Light”

```
AdkMessage received: connectivity zigbee_set_group_on_off_state
Message Received [ms:-2126565190 Sat May  4 05:09:23 2019]
connectivity_zigbee_set_group_on_off_state {
  group: "kitchen"
  state: true
}
AdkMessage received: led start_pattern
Message Received [ms:-2126565184 Sat May  4 05:09:23 2019]
led_start_pattern {
  pattern: 16
}
AdkMessage received: display notification
Message Received [ms:-2126565179 Sat May  4 05:09:23 2019]
display_notification {
  group: "voiceui"
  name: "LocalASR"
  message: "LightIntent"
  time_ms: "3000"
}
```

ADK Monitor

Local ASR Monitor

```
Every 2s: grep -i jston_str /var/log/messages | tail -1     2019-05-04 05:10:29

May  4 05:09:23 qcs403-som2 user.info VoiceUISampleApp: ModularClientManager::ProcessJSON -jston_str = {"slots":[{"
value":"on","name":"State"},{"name":"Room","value":"kitchen"}],"intent":"LightIntent"}
```

Local ASR + AVS Test

- To simply check the available Local ASR command list in current software, check the mp3 files in '/data/voice-ui-framework' folder

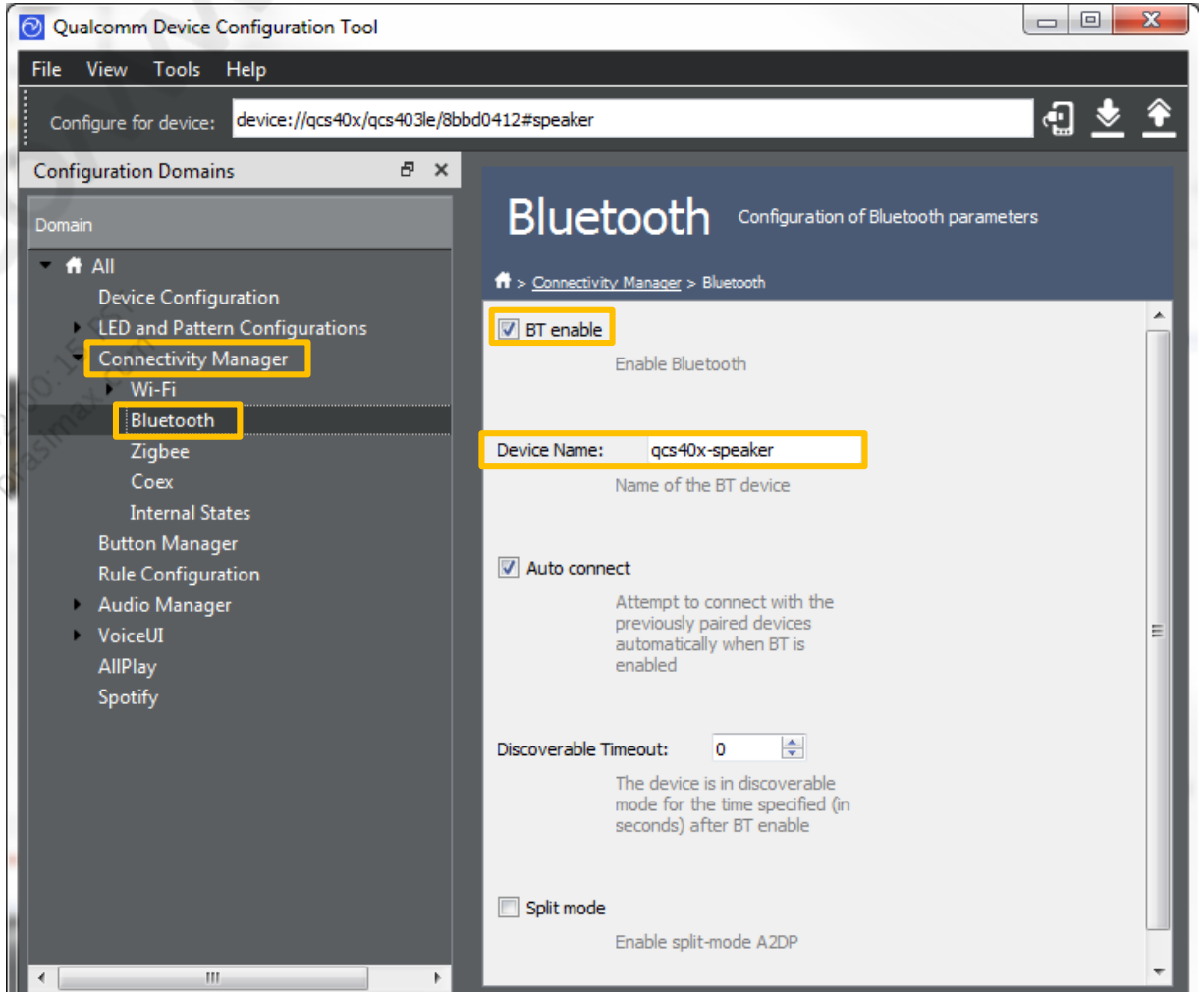
```
/data/voice-ui-framework # ls
Bedroom_Lights_OFF.mp3          Lights_OFF.mp3
Bedroom_Lights_ON.mp3           Lights_ON.mp3
Call_Ended.mp3                  Living_Room_Lights_OFF.mp3
Device_has_been_muted.mp3       Living_Room_Lights_ON.mp3
Device_has_been_unmuted.mp3     Thermostat_OFF.mp3
Front_Door_Closed.mp3           Thermostat_ON.mp3
Front_Door_Opened.mp3           Thermostat_temperature_has_changed.mp3
Garage_Door_Closed.mp3          Volume_Decreased.mp3
Garage_Door_Opened.mp3          Volume_Increased.mp3
HeySnapdragon.uim               Volume_changed.mp3
Incoming_call_Rejected.mp3      model.float.1.0_1.3.0.asr
Incoming_call_accepted.mp3      sm3_gmm_0703_cnn_jan08_Alexa.uim
Kitchen_Lights_ON.mp3           voiceDumpAVS
Kitchen_lights_OFF.mp3          voiceDumpLocalASR
/data/voice-ui-framework #
```

Section 8.7

Bluetooth Audio Streaming Test

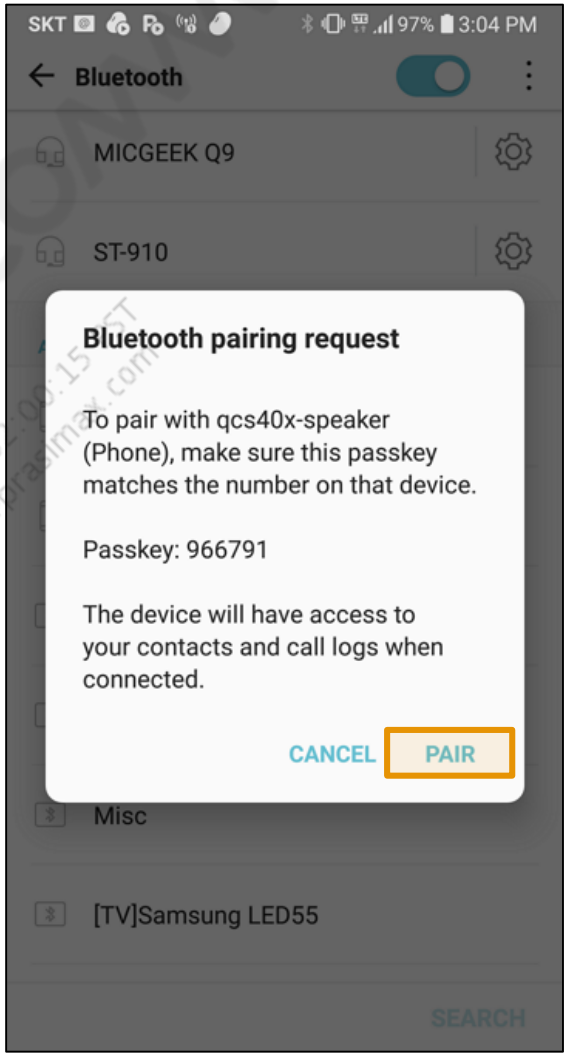
Bluetooth Audio Streaming Test

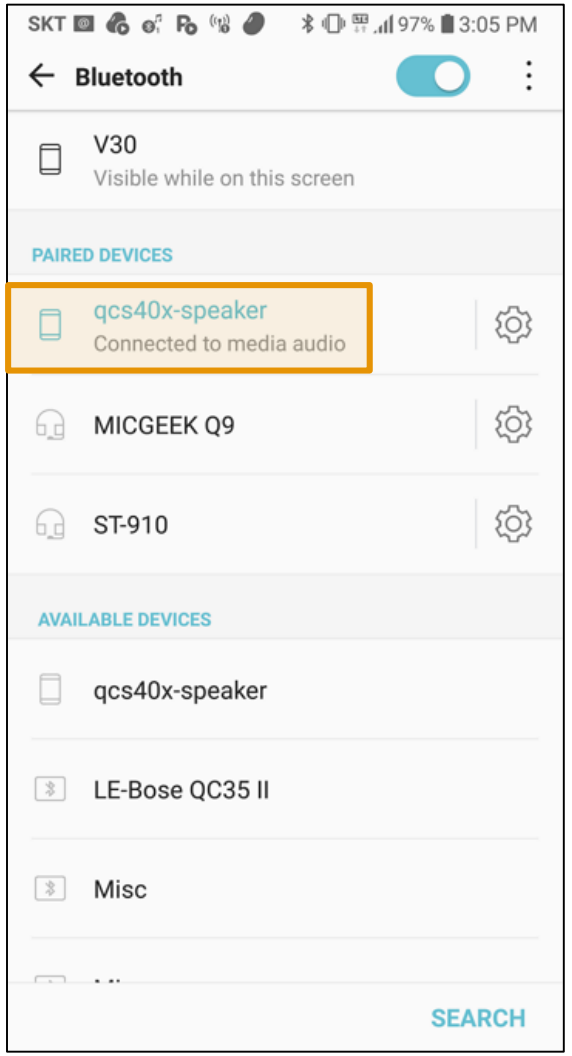
- Bluetooth feature is enabled in default configuration XML file
- To check Bluetooth feature is enabled in default XML file:
  - Open Device Configuration Tool
  - Import default XML file
  - Click on ‘Connectivity Manager > Bluetooth’ item
  - Make sure the ‘BT enable’ is check ON
- Make sure the ‘Device Name’ is set to something unique, default device name is ‘qcs40x-speaker’
- If the ‘BT enable’ is checked OFF then check ON the ‘BT enable’ and write again the updated configuration

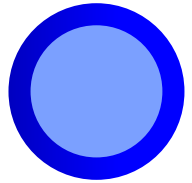


Bluetooth Audio Streaming Test

1. Open ‘Bluetooth’ setting on your Smart Phone
2. Search ‘qcs40x-speaker’ device
3. Pair and Bond with the device
   – When the ‘Bluetooth paring request’ window appears, just click ‘PAIR’ button, Passkey input is not required.
4. After connected to device, play some music with your Smart Phone
5. DONE !







Section 9

References

References

- [80-YB405-129] QCS40x Software User Guide
- [80-V1400-3] QUALCOMM PRODUCT SUPPORT TOOL (QPST) 2.7 USER GUIDE
- [80-NN120-1] QUALCOMM FLASH IMAGE LOADER (QFIL) USER GUIDE
- [80-YB420-4] BOOTLOADER

Questions?

For additional information or to submit technical questions, go to https://createpoint.qti.qualcomm.com

Thank You