QCA6595/QCA6696 Porting on Android Platform

User Guide

80-WL520-25 Rev. B

November 17, 2020

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

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

Revision history

Revision	Date	Description
A	June 2020	Initial release
B	November 2020	Add Chapter 11, Porting SDK.1.0 to Third Party Platform

Contents

Figures

Tables

1 Introduction

This document provides the basic information and steps for the customer who is going to support the QCA6595/QCA6696 on the 3rd party Android P platform.

2 AOSP HIDL Architecture for Bluetooth

2.1 Bluetooth Controller Initialization

The below MSC depicts the general bootstrap procedure of Qualcomm® QCA65x4/QCA6595/QCA6696 chip.

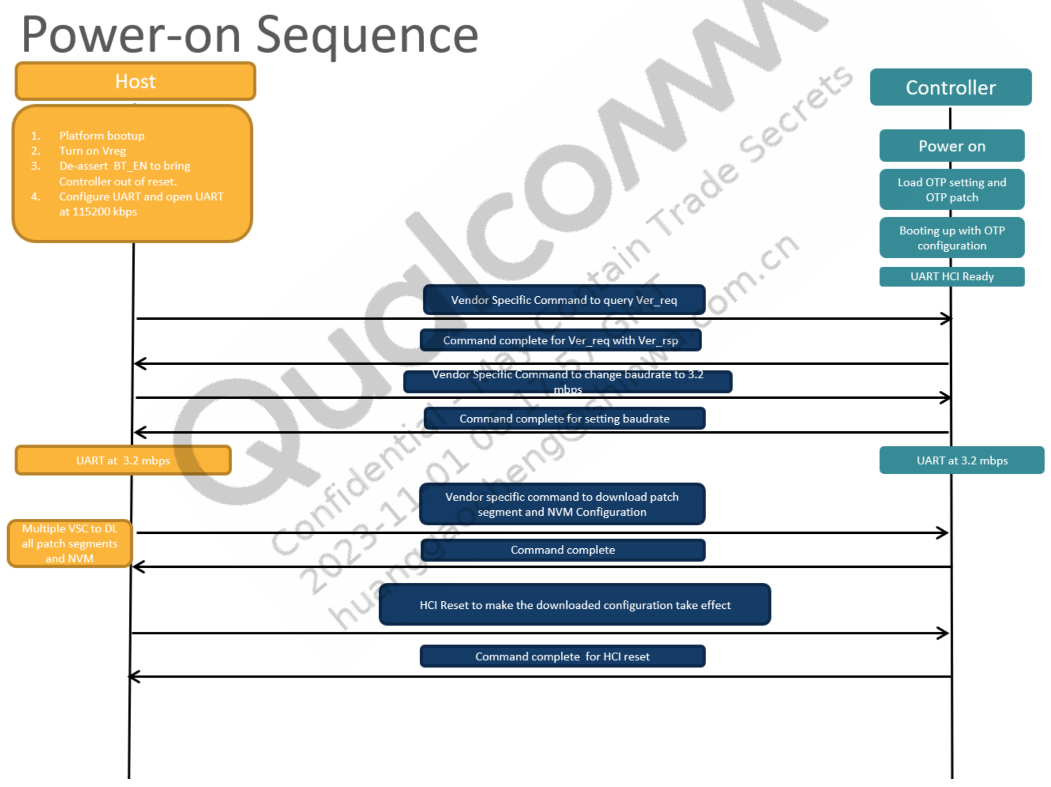

Power-on Sequence
Host
1. Platform bootup
2. Turn on Vreg
3. De-assert BT_EN to bring Controller out of reset.
4. Configure UART and open UART at 115200 kbps
Controller
Power on
Load OTP setting and OTP patch
Booting up with OTP configuration
UART HCI Ready
Vendor Specific Command to query Ver_req
Command complete for Ver_req with Ver_rsp
Vendor Specific Command to change baudrate to 3.2 mbps
Command complete for setting baudrate
UART at 3.2 mbps
UART at 3.2 mbps
Vendor specific command to download patch segment and NVM Configuration
Multiple VSC to DL all patch segments and NVM
Command complete
HCI Reset to make the downloaded configuration take effect
Command complete for HCI reset

Figure 2-1 the Bluetooth Bootstrap procedure

Refer to the following documents for more details.

- 80-CG635-1 BOOTING QCA BLUETOOTH CONTROLLER
- Section4 of 80-WL520-32 QCA65x4/QCA6595/QCA6696 Android P Bluetooth Software Training

2.2 AOSP HIDL Architecture

In Android system, the bootstrap of QCA Bluetooth chip is implemented in HIDL Daemon.

The below diagram depicts the AOSP HIDL architecture.

NOTE: Make sure the default AOSP HIDL is used in your software package, the QCA6595/QCA6696 BT bootstrap is based on the default AOSP HIDL.

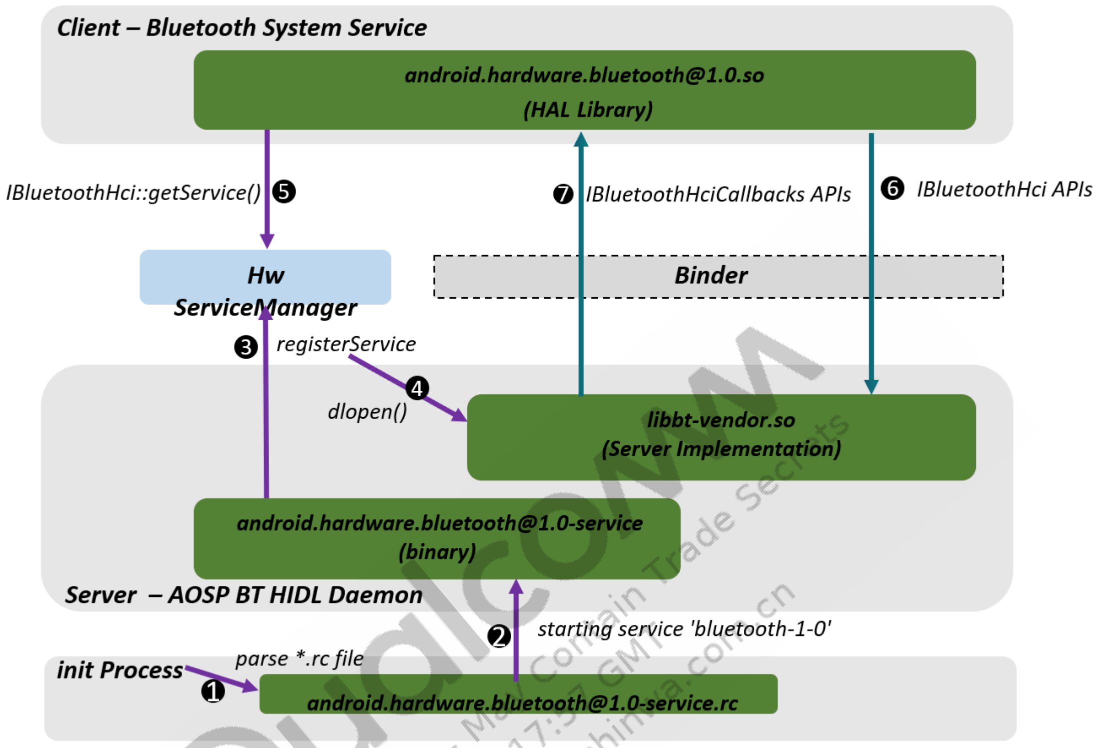

Client – Bluetooth System Service
android.hardware.bluetooth@1.0.so
(HAL Library)
IBluetoothHci::getService() ❺
❼ IBluetoothHciCallbacks APIs
❻ IBluetoothHci APIs
Hw ServiceManager
Binder
❸ registerService
❹
dlopen()
libbt-vendor.so
(Server Implementation)
android.hardware.bluetooth@1.0-service
(binary)
Server – AOSP BT HIDL Daemon
❷ starting service 'bluetooth-1-0'
init Process
parse *.rc file
❶
android.hardware.bluetooth@1.0-service.rc

Figure 2-2 The AOSP HIDL Architecture

3 Bluetooth libbt-vendor

BT Bootstrap is implemented in the libbt-vendor of AOSP HIDL, by default QCA65x4 is already supported. This chapter describes how to update the libbt-vendor to support QCA6595/QCA6696 BT bootstrap.

3.1 Fetch the AOSP libbt-vendor source code

1. repo init -u https://aosp.tuna.tsinghua.edu.cn/platform/manifest -b android-9.0.0_r9
2. modify the .repo/manifests/default.xml

```
--- a/default.xml
+++ b/default.xml
@@ -2,7 +2,7 @@
 <manifest>

   <remote  name="aosp"
-          fetch=".."
+          fetch="https://aosp.tuna.tsinghua.edu.cn"
           review="https://android-review.googlesource.com/" />
   <default revision="refs/tags/android-9.0.0_r9"
           remote="aosp"
```

3. repo sync platform/hardware/qcom/bt

NOTE: The domestic source aosp.tuna.tsinghua.edu.cn is utilized here, other source link also can be used if necessary.

3.2 The libbt-vendor source code directory

The libbt-vendor source code directory is as follows.

```
Qualcommm$:/local/mnt/workspace/AOSP_P$ tree -L 4
.
└── hardware
    └── qcom
        └── bt
            ├── Android.mk
            ├── libbt-vendor
            ├── msm8909
            ├── msm8909w_3100
            ├── msm8960
```

```
               ├── msm8992
               ├── msm8996
               └── msm8998

10 directories, 1 file
Qualcomm$:/local/mnt/workspace/AOSP_P$
```

NOTE: Use MSM8998 in the source code layout (hardware/qcom/MSM8998) as the work folder.

3.3 Support QCA6595/QCA6696 in libbt-vendor

To support the QCA6595/QCA6696 based on rome/QCA6x74 source code, the following modifications need to be applied.

Target source code path: `hardware/qcom/bt/MSM8998/libbt-vendor`

- Add new BT FW files definition and path for new BT chip.
- Handle the Vendor Specific Event properly during patch downloading.

3.3.1 BT FW files definition

In QCA 65x4/QCA6595/QCA6696 bootstrap, you can get Patch, Build, SOC version information by Vendor specific command EDL_PATCH_GETVER, refer to 80-WL520-14. Then select different BT FW files to download by these version information.

The code snap is as follows:

```
……
case EDL_APP_VER_RES_EVT:
                 if (!unified_hci) {
                   productid = (unsigned int)(rsp[PATCH_PROD_ID_OFFSET + 3] << 24 |
                                             rsp[PATCH_PROD_ID_OFFSET + 2] << 16 |
                                             rsp[PATCH_PROD_ID_OFFSET + 1] << 8 |
                                             rsp[PATCH_PROD_ID_OFFSET]  );
                   patchversion = (unsigned short)(rsp[PATCH_PATCH_VER_OFFSET + 1] << 8 |
                                                  rsp[PATCH_PATCH_VER_OFFSET] );
                   buildversion = (int)(rsp[PATCH_ROM_BUILD_VER_OFFSET + 1] << 8 |
                               rsp[PATCH_ROM_BUILD_VER_OFFSET] );

                   ALOGI("\t Current Product ID\t\t: 0x%08x", productid);
                   ALOGI("\t Current Patch Version\t\t: 0x%04x", patchversion);
                   ALOGI("\t Current ROM Build Version\t: 0x%04x", buildversion);
……
```

The example version are listed as follows:

Controller Version	Patch	ROM Build	SOC
QCA6574	0x0111	0x0302	0x00000044
QCA6595	0x0fb9	0x0200	0x400b0200

QCA6696	0x0d2b	0x0200	0x400a0200

After reading the version from the Controller, the libbt-vendor shall calculate the chipset_ver to select the proper tlv files. The code snap as below.

```
case EDL_APP_VER_RES_EVT:
                ……

                /* BT Chipset Version can be decided by Patch version and SOC version,
                Upper 2 bytes will be used for Patch version and Lower 2 bytes will be
                used for SOC as combination for BT host driver */
                chipset_ver = (buildversion << 16) |(soc_id & 0x0000ffff);
                ……
```

According to chipset_ver = (buildversion << 16) |(soc_id & 0x0000ffff); both QCA6695 and QCA6595 chipset_ver are same: 0x02000200, if your Software Product supports both QCA6696 and QCA6595 from software perspective, you need to distinguish them.

The following is an example definition for tlv files, so that the correct tlv files can be selected to be downloaded.

The following are the QCA6595/QCA6696 chip version related definitions.

Patch version

```
HASTINGS_PATCH_VER_0100 = 0x0100,

HASTINGS_PATCH_VER_0200 = 0x0200,

GENOA_PATCH_VER_0100 = 0xxxxxx,

GENOA_PATCH_VER_0200 = 0xxxxx
```

SOC_ID

```
HASTINGS_SOC_ID_0100 = 0x00000100,

HASTINGS_SOC_ID_0101 = 0x00000101,

HASTINGS_SOC_ID_0110 = 0x00000110,

HASTINGS_SOC_ID_0200 = 0x00000200,

GENOA_SOC_ID_0100 = 0xxxxxx,

GENOA_SOC_ID_0200 = 0xxxxxx
```

SOC Version

```
HASTINGS_VER_1_0 = ((HASTINGS_PATCH_VER_0100 << 16 ) |
HASTINGS_SOC_ID_0100),

HASTINGS_VER_1_0_1 = ((HASTINGS_PATCH_VER_0100 << 16 ) |
HASTINGS_SOC_ID_0101),

HASTINGS_VER_1_1 = ((HASTINGS_PATCH_VER_0100 << 16 ) |
HASTINGS_SOC_ID_0110),

HASTINGS_VER_2_0 = ((HASTINGS_PATCH_VER_0200 << 16 ) |
HASTINGS_SOC_ID_0200),
GENOA_VER_1_0 = ((GENOA_PATCH_VER_0100 << 16 ) | GENOA_SOC_ID_0100),

GENOA_VER_2_0 = ((GENOA_PATCH_VER_0200 << 16 ) | GENOA_SOC_ID_0200)
```

The following are the QCA6595/QCA6696 BT FW files related definitions.

Rampatch and NVM FW file path

```
#define HASTINGS_RAMPATCH_TLV_UART_1_0_PATH
"/lib/firmware/image/htbtfw10.tlv"

#define HASTINGS_NVM_TLV_UART_1_0_PATH
"/lib/firmware/image/htnv10.bin"

#define HASTINGS_RAMPATCH_TLV_UART_2_0_PATH
"/lib/firmware/image/htbtfw20.tlv"

#define HASTINGS_NVM_TLV_UART_2_0_PATH
"/lib/firmware/image/htnv20.bin"

#define GENOA_RAMPATCH_TLV_UART_1_0_PATH
"/lib/firmware/image/gnbtfw10.tlv"

#define GENOA_NVM_TLV_UART_1_0_PATH
"/lib/firmware/image/gnnv10.bin"

#define GENOA_RAMPATCH_TLV_UART_2_0_PATH
"/lib/firmware/image/gnbtfw20.tlv"

#define GENOA_NVM_TLV_UART_2_0_PATH
"/lib/firmware/image/gnnv20.bin"
```

3.3.2 Handle the Vendor Specific Event

QCA65x4

As mentioned, QCA65x4 BT bootstrap is already supported in the AOSP HIDL (libbt-vendor), the BT FW downloading procedure sequence is as following.

By default, the optimized rampatch file is used. There will be 1 Vendor Specific Event (VSE) reported for the last TLV_DOWNLOAD_REQ command.

If Non-optimized rampatch file is used, there will be 1 VSC and 1 Command Complete Event reported for each TLV_DOWNLOAD_REQ command.

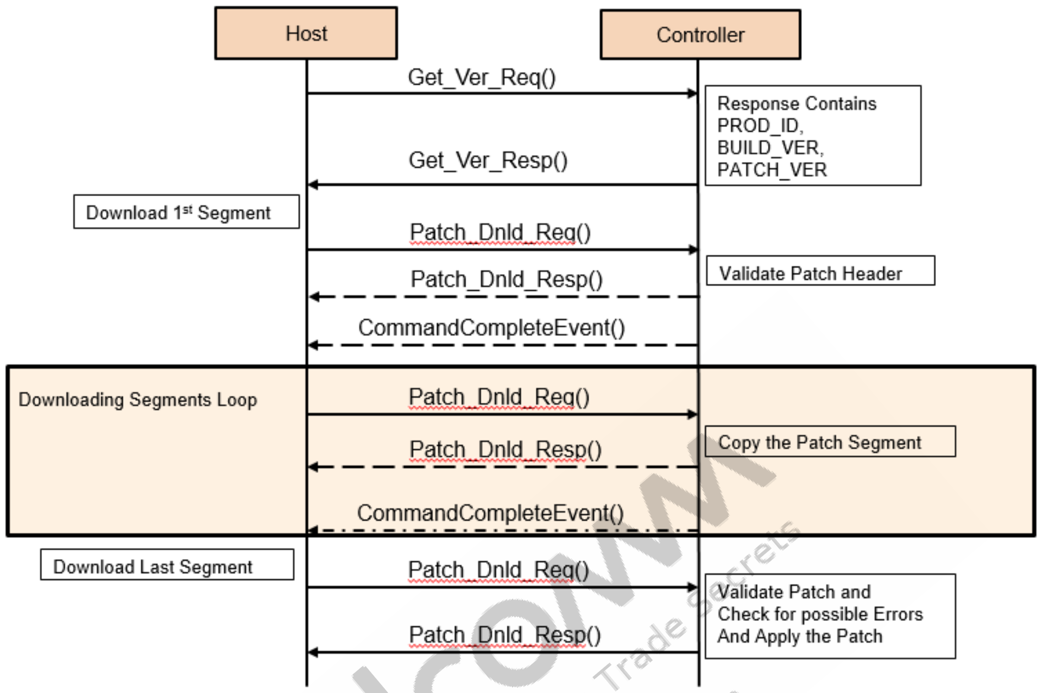


Figure 3-1 the QCA65x4 patch downloading procedure

For QCA65x4 NVM downloading, there will be 1 VSC and 1 Command Complete Event reported for each TLV_DOWNLOAD_REQ command.

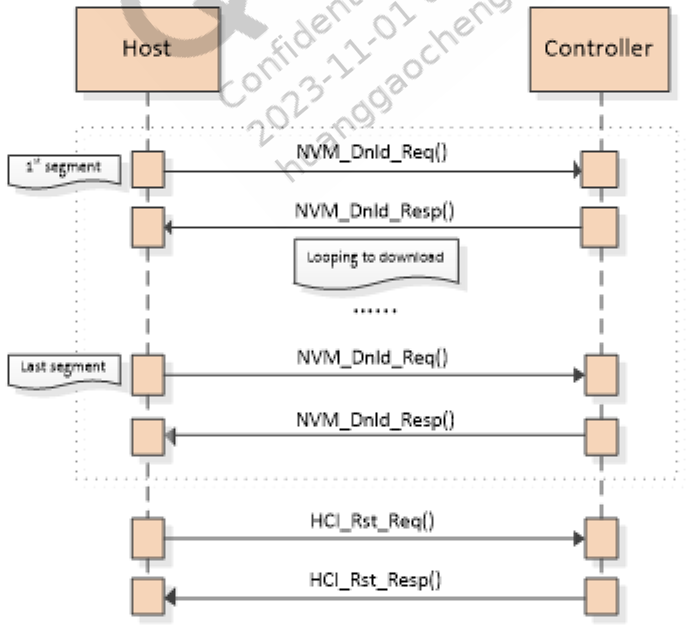


Figure 3-2 The QCA65x4 NVM downloading procedure

QCA6595/QCA6696

This section describes the QCA6595/QCA6696 rampatch and NVM downloading sequence.

By default, the optimized rampatch file is used. There will be 1 Complete Command Event with TLV_DOWNLOAD_REQ reported for the last TLV_DOWNLOAD_REQ command, please refer to 80-WL520-14.

If non-optimized rampatch file is used, there will be 1 Complete Command Event with TLV_DOWNLOAD_REQ reported for each TLV_DOWNLOAD_REQ command.

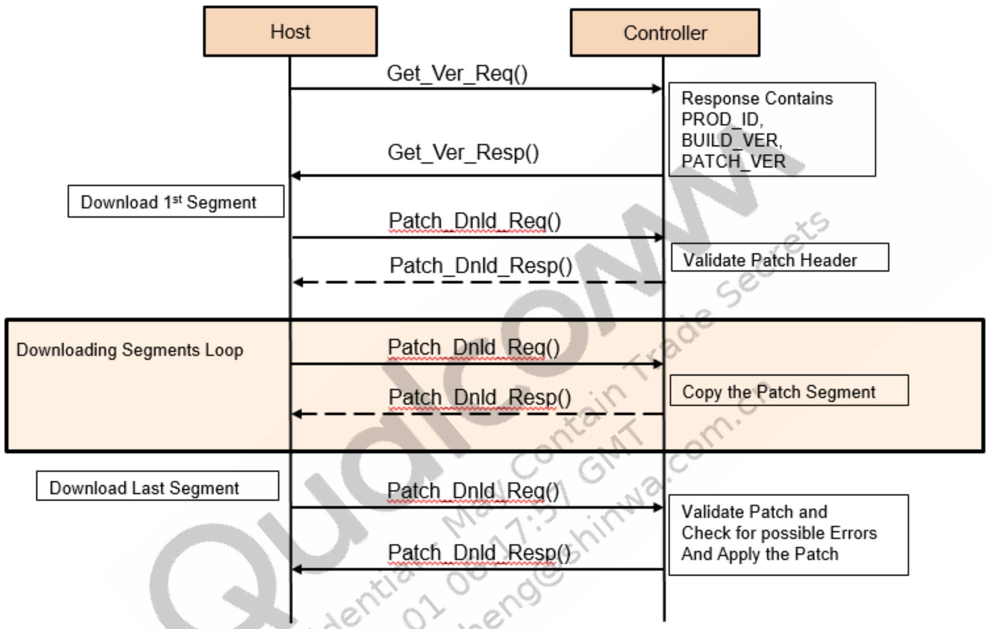


Figure 3-3 The QCA6595/QCA6696 Patch downloading procedure

For QCA6595/QCA6696 NVM downloading, there will be 1 VSC and 1 Command Complete Event reported for each TLV_DOWNLOAD_REQ command.

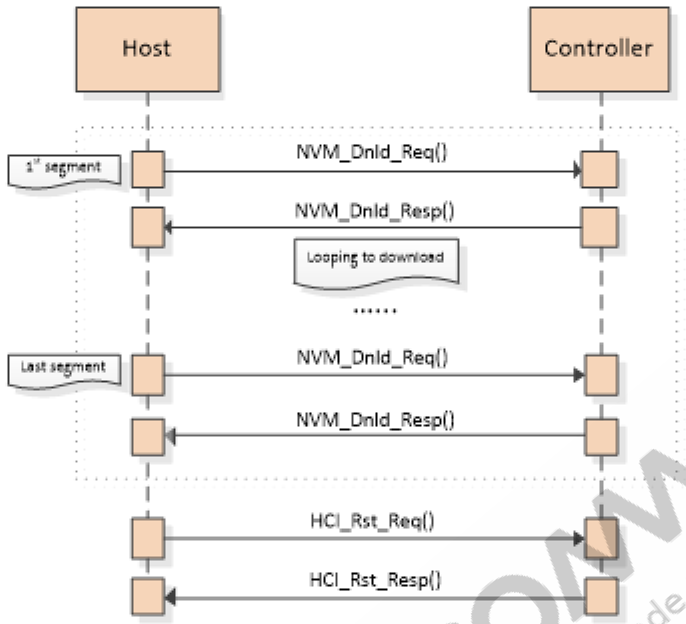


Figure 3-4 The QCA6595/QCA6696 NVM downloading procedure

3.3.3 The sequence of libbt-vendor handling

The Figure 4-5 describes the overall rampatch and NVM downloading sequence of libbt-vendor, to support QCA6595/QCA6696, the functions that will be added and modified in libbt-vendor are colored as green.

Related APIs of libbt-vendor

Rome_patch_ver_req()

get_vs_hci_event()

rome_tlv_dnld_req()

rome_tlv_dnld_segment()

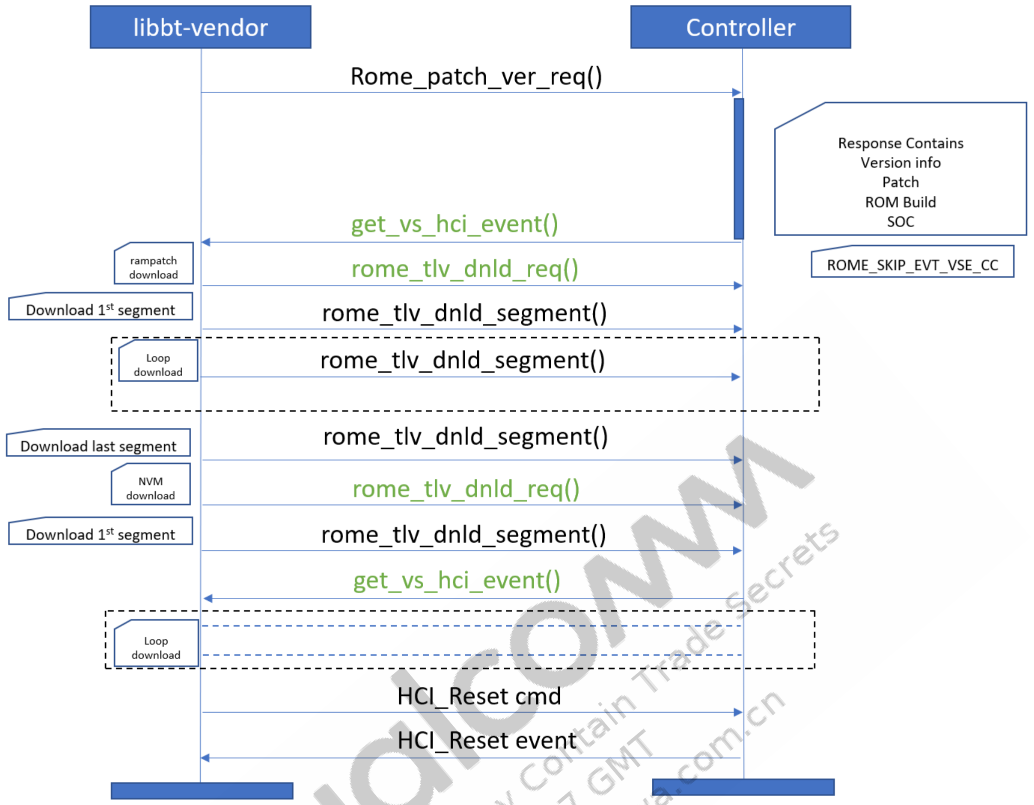


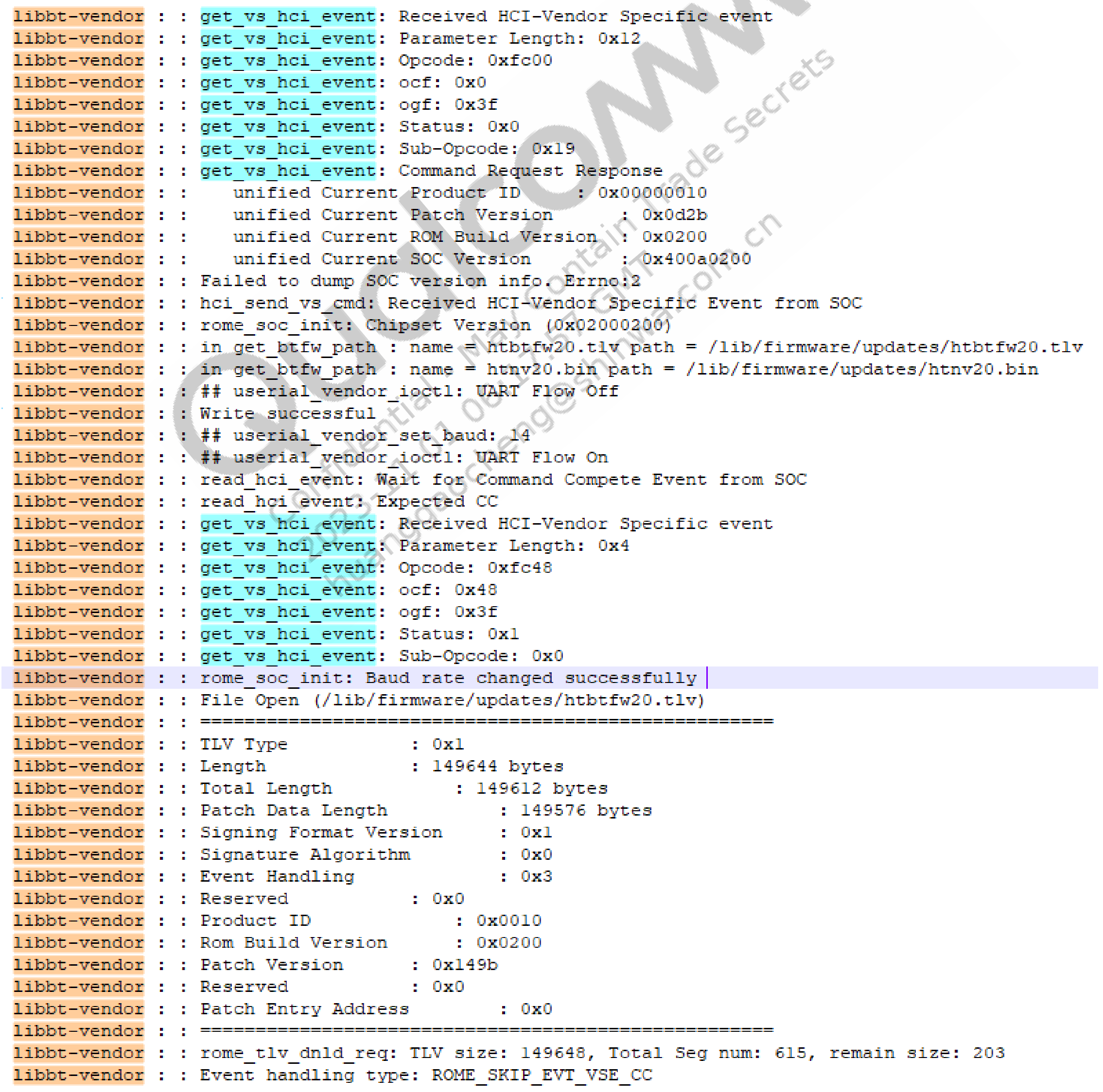
Figure 3-5 libbt-vendor patch & nvm downloading

4 Bluetooth Debug

4.1 libbt-vendor debug

The libbt-vendor debug info is saved to logcat log.

The following is the debug info snapshot.



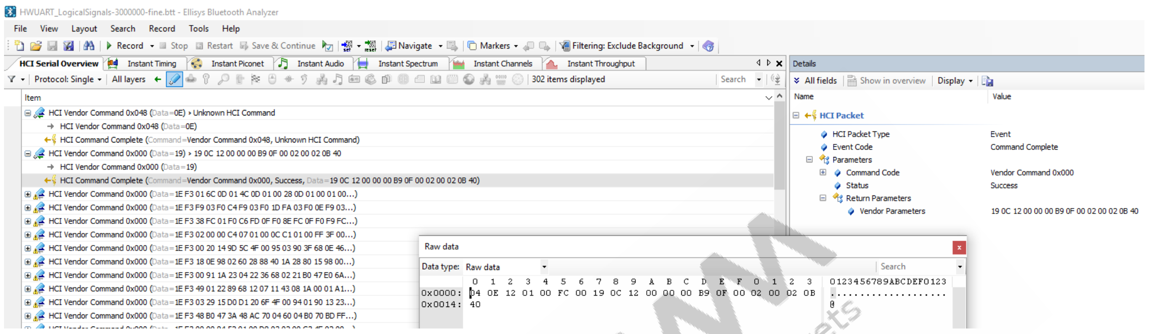
```
libbt-vendor : : get_vs_hci_event: Received HCI-Vendor Specific event
libbt-vendor : : get_vs_hci_event: Parameter Length: 0x12
libbt-vendor : : get_vs_hci_event: Opcode: 0xfc00
libbt-vendor : : get_vs_hci_event: ocf: 0x0
libbt-vendor : : get_vs_hci_event: ogf: 0x3f
libbt-vendor : : get_vs_hci_event: Status: 0x0
libbt-vendor : : get_vs_hci_event: Sub-Opcode: 0x19
libbt-vendor : : get_vs_hci_event: Command Request Response
libbt-vendor : :    unified Current Product ID      : 0x00000010
libbt-vendor : :    unified Current Patch Version       : 0x0d2b
libbt-vendor : :    unified Current ROM Build Version  : 0x0200
libbt-vendor : :    unified Current SOC Version         : 0x400a0200
libbt-vendor : : Failed to dump SOC version info. Errno:2
libbt-vendor : : hci_send_vs_cmd: Received HCI-Vendor Specific Event from SOC
libbt-vendor : : rome_soc_init: Chipset Version (0x02000200)
libbt-vendor : : in get_btfw_path : name = htbtfw20.tlv path = /lib/firmware/updates/htbtfw20.tlv
libbt-vendor : : in get_btfw_path : name = htnv20.bin path = /lib/firmware/updates/htnv20.bin
libbt-vendor : : ## userial_vendor_ioctl: UART Flow Off
libbt-vendor : : Write successful
libbt-vendor : : ## userial_vendor_set_baud: 14
libbt-vendor : : ## userial_vendor_ioctl: UART Flow On
libbt-vendor : : read_hci_event: Wait for Command Compete Event from SOC
libbt-vendor : : read_hci_event: Expected CC
libbt-vendor : : get_vs_hci_event: Received HCI-Vendor Specific event
libbt-vendor : : get_vs_hci_event: Parameter Length: 0x4
libbt-vendor : : get_vs_hci_event: Opcode: 0xfc48
libbt-vendor : : get_vs_hci_event: ocf: 0x48
libbt-vendor : : get_vs_hci_event: ogf: 0x3f
libbt-vendor : : get_vs_hci_event: Status: 0x1
libbt-vendor : : get_vs_hci_event: Sub-Opcode: 0x0
libbt-vendor : : rome_soc_init: Baud rate changed successfully
libbt-vendor : : File Open (/lib/firmware/updates/htbtfw20.tlv)
libbt-vendor : : ====================================================
libbt-vendor : : TLV Type           : 0x1
libbt-vendor : : Length             : 149644 bytes
libbt-vendor : : Total Length           : 149612 bytes
libbt-vendor : : Patch Data Length          : 149576 bytes
libbt-vendor : : Signing Format Version     : 0x1
libbt-vendor : : Signature Algorithm        : 0x0
libbt-vendor : : Event Handling             : 0x3
libbt-vendor : : Reserved           : 0x0
libbt-vendor : : Product ID             : 0x0010
libbt-vendor : : Rom Build Version      : 0x0200
libbt-vendor : : Patch Version      : 0x149b
libbt-vendor : : Reserved           : 0x0
libbt-vendor : : Patch Entry Address        : 0x0
libbt-vendor : : ====================================================
libbt-vendor : : rome_tlv_dnld_req: TLV size: 149648, Total Seg num: 615, remain size: 203
libbt-vendor : : Event handling type: ROME_SKIP_EVT_VSE_CC
```

Figure 4-1 the libbt-vendor debug info snapshot

4.2 tlv files downloading debug

The rampatch and NVM downloading message sequence can be captured by HSU or Ellisys equipment as Hardware UART log. The Hardware UART log also can help to debug the issues during BT chip bootstrap.

The following is the Hardware UART trace snapshot which is captured by Ellisys.


HWUART_LogicalSignals-3000000-fine.btt - Ellisys Bluetooth Analyzer
File View Layout Search Record Tools Help
Record Stop Restart Save & Continue Navigate Markers Filtering: Exclude Background
HCI Serial Overview Instant Timing Instant Piconet Instant Audio Instant Spectrum Instant Channels Instant Throughput
Protocol: Single All layers 302 items displayed Search
Details All fields Show in overview Display

```
Item
HCI Vendor Command 0x048 (Data=0E) › Unknown HCI Command
    HCI Vendor Command 0x048 (Data=0E)
    HCI Command Complete (Command=Vendor Command 0x048, Unknown HCI Command)
HCI Vendor Command 0x000 (Data=19) › 19 0C 12 00 00 00 B9 0F 00 02 00 02 0B 40
    HCI Vendor Command 0x000 (Data=19)
    HCI Command Complete (Command=Vendor Command 0x000, Success, Data=19 0C 12 00 00 00 B9 0F 00 02 00 02 0B 40)
HCI Vendor Command 0x000 (Data=1E F3 01 6C 0D 01 4C 0D 01 00 28 0D 01 00 01 00...)
HCI Vendor Command 0x000 (Data=1E F3 F9 03 F0 C4 F9 03 F0 1D FA 03 F0 0E F9 03...)
HCI Vendor Command 0x000 (Data=1E F3 38 FC 01 F0 C6 FD 0F F0 8E FC 0F F0 F9 FC...)
HCI Vendor Command 0x000 (Data=1E F3 02 00 00 C4 07 01 00 0C C1 01 00 FF 3F 00...)
HCI Vendor Command 0x000 (Data=1E F3 00 20 14 9D 5C 4F 00 95 03 90 3F 68 0E 46...)
HCI Vendor Command 0x000 (Data=1E F3 18 0E 98 02 60 28 88 40 1A 28 80 15 98 00...)
HCI Vendor Command 0x000 (Data=1E F3 00 91 1A 23 04 22 36 68 02 21 B0 47 E0 6A...)
HCI Vendor Command 0x000 (Data=1E F3 49 01 22 89 68 12 07 11 43 08 1A 00 01 A1...)
HCI Vendor Command 0x000 (Data=1E F3 03 29 15 D0 D1 20 6F 4F 00 94 01 90 13 23...)
HCI Vendor Command 0x000 (Data=1E F3 48 B0 47 3A 48 AC 70 04 60 04 B0 70 BD FF...)
```

Name	Value
HCI Packet	
HCI Packet Type	Event
Event Code	Command Complete
Parameters	
Command Code	Vendor Command 0x000
Status	Success
Return Parameters	
Vendor Parameters	19 0C 12 00 00 00 B9 0F 00 02 00 02 0B 40

```
Raw data
Data type: Raw data                                    Search
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0  1  2  3   0123456789ABCDEF0123
0x0000:  04 0E 12 01 00 FC 00 19 0C 12 00 00 00 B9 0F 00 02 00 02 0B   ....................
0x0014:  40                                                            @
```

Figure 4-2 the Hardware UART trace snapshot

4.2.1 How to get Hardware UART trace by HSU

Refer to section 'ComProbe High Speed Serial Sniffing Data Capture Method' of Frontline Help.

4.2.2 How to get Hardware UART trace by Ellisys

Refer to section 'To capture wired traffic' of Ellisys Bluetooth Analyzer User Manual.

5 Porting Bluetooth from SDK.1.0

Auto Connectivity Image Set SDK.1.0 provides the release for connectivity components. Each of these releases provide enhancements to WLAN and Bluetooth software functionality for various QTI connectivity chipsets. The software is organized by a set of images and software products. Images are the connectivity software bundles and software products are the testing context and delivery platform for the connectivity images.

5.1 Bluetooth Images in SDK.1.0

The 2020.R2 image set consists of the following Bluetooth images:

Image Name	Description
BTFM.GEN.2.0.0	Bluetooth firmware for QCA6595AU devices
BTFM.RM.2.4.1	Bluetooth firmware for QCA6574 and QCA6564 devices
BTFM.HST.2.0.0	Bluetooth firmware for QCA6696AU 2.0 devices
BTHost_AU.LA.3.1	QTI enhanced Android Q Bluetooth including BT HIDL

5.2 Booting QCA chips by SDK.1.0

5.2.1 Download the QCOM proprietary code

The code can be downloaded from https://chipcode.qti.qualcomm.com. Both firmware files and Bluetooth HIDL code are included in this package.

- Select Auto_Connectivity_Image_Set.SDK.1.0 from the products list.
- Click <customer name>/auto-connectivity-image-set-sdk-1-0_qca_oem to download the code.

5.2.2 Porting QCOM BT HIDL into Android Q

- By default, Android Q uses the Google BT HIDL service. Refer to `$android/hardware/interfaces/bluetooth/`
- To boot QCA BT chips, the QCOM BT HIDL service is required instead.
  a. Copy the QCOM BT HIDL service into Android Q. The code is from Qualcomm chipcode in HOST_LA, for example:

```
  # cp -rf apps_proc/vendor/qcom/proprietary/bluetooth
android/vendor/qcom/proprietary/
```

b. To install the QCOM HIDL related modules, you may need to modify .mk files in Android Q to ensure they will be built by adding the following lines.

```
BT += android.hardware.bluetooth@1.0-service-qti
BT += android.hardware.bluetooth@1.0-impl-qti
BT += android.hardware.bluetooth@1.0-service-qti.rc
PRODUCT_PACKAGES += $(BT)
```

c. Build your source code, the following files will be generated under $android/out

- android.hardware.bluetooth@1.0-impl-qti.so
- android.hardware.bluetooth@1.0-service-qti
- android.hardware.bluetooth@1.0-service-qti.rc

5.2.3 Push the Bluetooth firmware files

To boot the QCA chips, it still needs the firmware files on your board. The firmware files include the rampatch and NVM file. You can find them from the downloaded software from chipcode. In default, the bootstrap code in HIDL service is compatible with these chips and load them from the location below. Once loaded successfully, they will be downloaded into the chips.

Value	Description
/bt_firmware/image/btfw32.tlv	rampatch file for QCA65x4
/bt_firmware/image/btnv32.bin	BT NVM file for QCA65x4
/bt_firmware/image/htbtfw20.tlv	rampatch file for QCA6696
/bt_firmware/image/htnv20.bin	BT NVM file for QCA6696
/bt_firmware/image/gnbtfw20.tlv	rampatch file for QCA6595
/bt_firmware/image/gnnv20.bin	BT NVM file for QCA6595

5.2.4 Bluetooth Boot Sequence

The Boot sequence of QCA chips in QCOM HIDL service is shown as below. For more information on boot sequence, refer to 80-CG635-1.

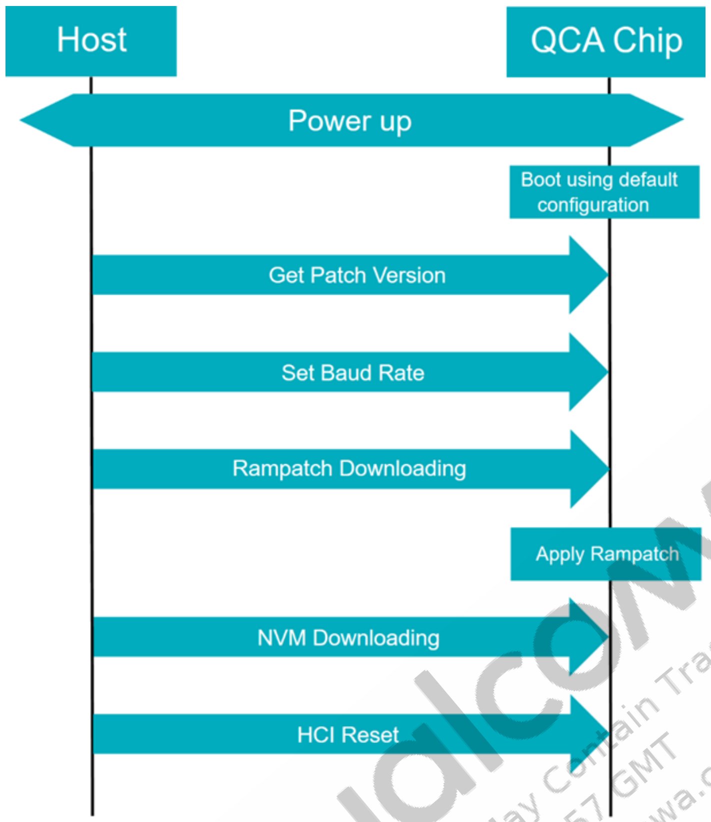
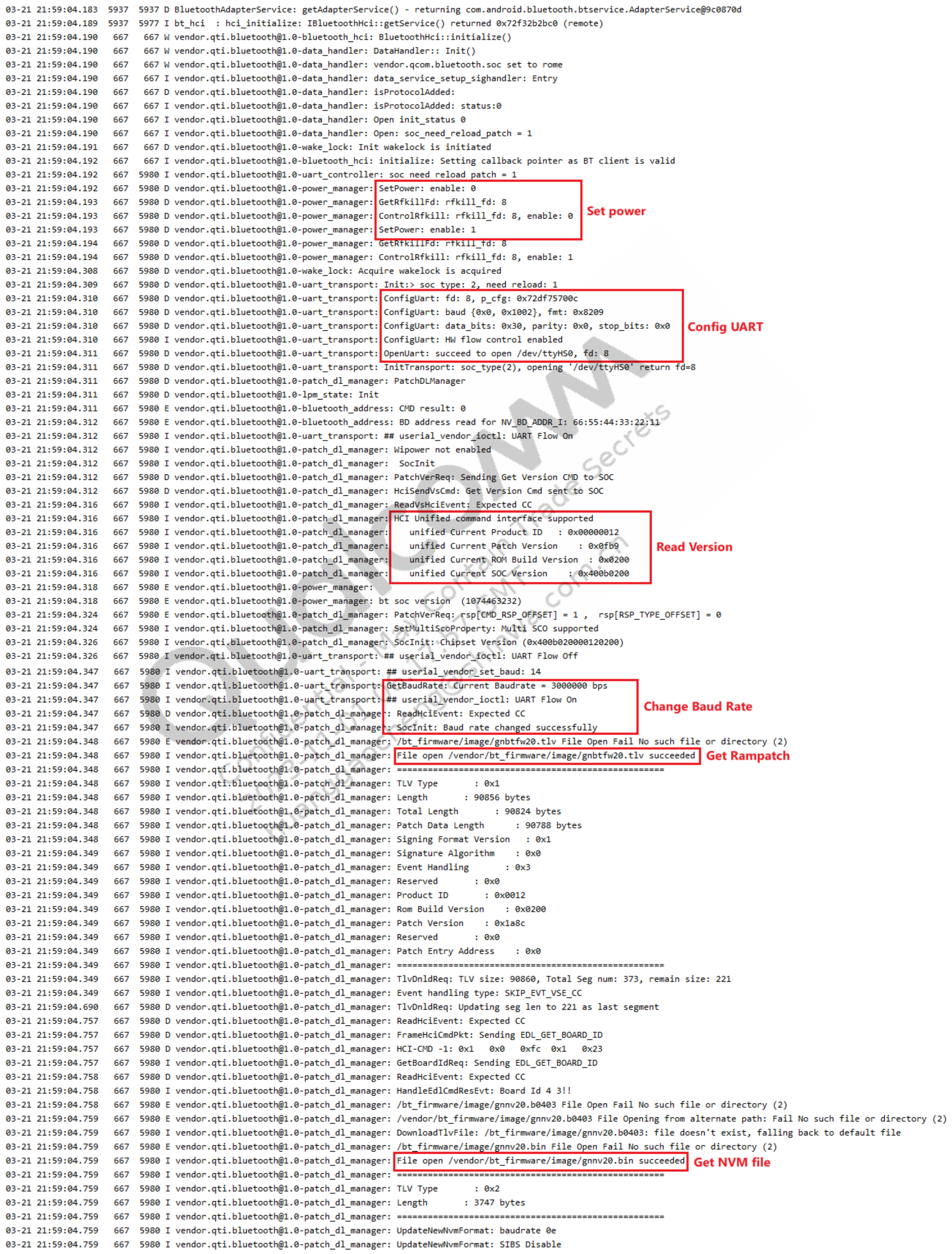


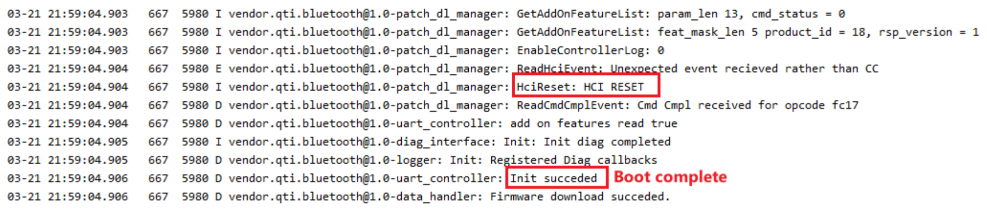
Figure 5-1 Bluetooth Boot Sequence

5.2.5 Boot logs and analysis

The below shows the logcat from QCOM HIDL service during the booting up on Android Q.



```
03-21 21:59:04.183  5937  5937 D BluetoothAdapterService: getAdapterService() - returning com.android.bluetooth.btservice.AdapterService@9c0870d
03-21 21:59:04.189  5937  5977 I bt_hci  : hci_initialize: IBluetoothHci::getService() returned 0x72f32b2bc0 (remote)
03-21 21:59:04.190   667   667 W vendor.qti.bluetooth@1.0-bluetooth_hci: BluetoothHci::initialize()
03-21 21:59:04.190   667   667 W vendor.qti.bluetooth@1.0-data_handler: DataHandler:: Init()
03-21 21:59:04.190   667   667 W vendor.qti.bluetooth@1.0-data_handler: vendor.qcom.bluetooth.soc set to rome
03-21 21:59:04.190   667   667 I vendor.qti.bluetooth@1.0-data_handler: data_service_setup_sighandler: Entry
03-21 21:59:04.190   667   667 D vendor.qti.bluetooth@1.0-data_handler: isProtocolAdded:
03-21 21:59:04.190   667   667 I vendor.qti.bluetooth@1.0-data_handler: isProtocolAdded: status:0
03-21 21:59:04.190   667   667 I vendor.qti.bluetooth@1.0-data_handler: Open init_status 0
03-21 21:59:04.190   667   667 I vendor.qti.bluetooth@1.0-data_handler: Open: soc_need_reload_patch = 1
03-21 21:59:04.191   667   667 D vendor.qti.bluetooth@1.0-wake_lock: Init wakelock is initiated
03-21 21:59:04.192   667   667 I vendor.qti.bluetooth@1.0-bluetooth_hci: initialize: Setting callback pointer as BT client is valid
03-21 21:59:04.192   667  5980 I vendor.qti.bluetooth@1.0-uart_controller: soc need reload patch = 1
03-21 21:59:04.192   667  5980 D vendor.qti.bluetooth@1.0-power_manager: SetPower: enable: 0
03-21 21:59:04.193   667  5980 D vendor.qti.bluetooth@1.0-power_manager: GetRfkillFd: rfkill_fd: 8
03-21 21:59:04.193   667  5980 D vendor.qti.bluetooth@1.0-power_manager: ControlRfkill: rfkill_fd: 8, enable: 0
03-21 21:59:04.193   667  5980 D vendor.qti.bluetooth@1.0-power_manager: SetPower: enable: 1
03-21 21:59:04.194   667  5980 D vendor.qti.bluetooth@1.0-power_manager: GetRfkillFd: rfkill_fd: 8
03-21 21:59:04.194   667  5980 D vendor.qti.bluetooth@1.0-power_manager: ControlRfkill: rfkill_fd: 8, enable: 1
03-21 21:59:04.308   667  5980 D vendor.qti.bluetooth@1.0-wake_lock: Acquire wakelock is acquired
03-21 21:59:04.309   667  5980 D vendor.qti.bluetooth@1.0-uart_transport: Init:> soc type: 2, need reload: 1
03-21 21:59:04.310   667  5980 D vendor.qti.bluetooth@1.0-uart_transport: ConfigUart: fd: 8, p_cfg: 0x72df75700c
03-21 21:59:04.310   667  5980 D vendor.qti.bluetooth@1.0-uart_transport: ConfigUart: baud {0x0, 0x1002}, fmt: 0x8209
03-21 21:59:04.310   667  5980 D vendor.qti.bluetooth@1.0-uart_transport: ConfigUart: data_bits: 0x30, parity: 0x0, stop_bits: 0x0
03-21 21:59:04.310   667  5980 I vendor.qti.bluetooth@1.0-uart_transport: ConfigUart: HW flow control enabled
03-21 21:59:04.311   667  5980 D vendor.qti.bluetooth@1.0-uart_transport: OpenUart: succeed to open /dev/ttyHS0, fd: 8
03-21 21:59:04.311   667  5980 D vendor.qti.bluetooth@1.0-uart_transport: InitTransport: soc_type(2), opening '/dev/ttyHS0' return fd=8
03-21 21:59:04.311   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: PatchDLManager
03-21 21:59:04.311   667  5980 D vendor.qti.bluetooth@1.0-lpm_state: Init
03-21 21:59:04.311   667  5980 E vendor.qti.bluetooth@1.0-bluetooth_address: CMD result: 0
03-21 21:59:04.312   667  5980 E vendor.qti.bluetooth@1.0-bluetooth_address: BD address read for NV_BD_ADDR_I: 66:55:44:33:22:11
03-21 21:59:04.312   667  5980 I vendor.qti.bluetooth@1.0-uart_transport: ## userial_vendor_ioctl: UART Flow On
03-21 21:59:04.312   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Wipower not enabled
03-21 21:59:04.312   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager:  SocInit
03-21 21:59:04.312   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: PatchVerReq: Sending Get Version CMD to SOC
03-21 21:59:04.312   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: HciSendVsCmd: Get Version Cmd sent to SOC
03-21 21:59:04.316   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: ReadVsHciEvent: Expected CC
03-21 21:59:04.316   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: HCI Unified command interface supported
03-21 21:59:04.316   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager:    unified Current Product ID   : 0x00000012
03-21 21:59:04.316   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager:    unified Current Patch Version    : 0x0fb9
03-21 21:59:04.316   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager:    unified Current ROM Build Version  : 0x0200
03-21 21:59:04.316   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager:    unified Current SOC Version    : 0x400b0200
03-21 21:59:04.318   667  5980 E vendor.qti.bluetooth@1.0-power_manager:
03-21 21:59:04.318   667  5980 E vendor.qti.bluetooth@1.0-power_manager: bt soc version  (1074463232)
03-21 21:59:04.324   667  5980 E vendor.qti.bluetooth@1.0-patch_dl_manager: PatchVerReq: rsp[CMD_RSP_OFFSET] = 1 ,  rsp[RSP_TYPE_OFFSET] = 0
03-21 21:59:04.324   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: SetMultiScoProperty: Multi SCO supported
03-21 21:59:04.326   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: SocInit: Chipset Version (0x400b020000120200)
03-21 21:59:04.326   667  5980 I vendor.qti.bluetooth@1.0-uart_transport: ## userial_vendor_ioctl: UART Flow Off
03-21 21:59:04.347   667  5980 I vendor.qti.bluetooth@1.0-uart_transport: ## userial_vendor_set_baud: 14
03-21 21:59:04.347   667  5980 I vendor.qti.bluetooth@1.0-uart_transport: GetBaudRate: Current Baudrate = 3000000 bps
03-21 21:59:04.347   667  5980 I vendor.qti.bluetooth@1.0-uart_transport: ## userial_vendor_ioctl: UART Flow On
03-21 21:59:04.347   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: ReadHciEvent: Expected CC
03-21 21:59:04.347   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: SocInit: Baud rate changed successfully
03-21 21:59:04.348   667  5980 E vendor.qti.bluetooth@1.0-patch_dl_manager: /bt_firmware/image/gnbtfw20.tlv File Open Fail No such file or directory (2)
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: File open /vendor/bt_firmware/image/gnbtfw20.tlv succeeded
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: ====================================================
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: TLV Type       : 0x1
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Length       : 90856 bytes
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Total Length       : 90824 bytes
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Patch Data Length      : 90788 bytes
03-21 21:59:04.348   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Signing Format Version   : 0x1
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Signature Algorithm    : 0x0
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Event Handling       : 0x3
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Reserved       : 0x0
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Product ID       : 0x0012
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Rom Build Version    : 0x0200
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Patch Version    : 0x1a8c
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Reserved       : 0x0
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Patch Entry Address    : 0x0
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: ====================================================
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: TlvDnldReq: TLV size: 90860, Total Seg num: 373, remain size: 221
03-21 21:59:04.349   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Event handling type: SKIP_EVT_VSE_CC
03-21 21:59:04.690   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: TlvDnldReq: Updating seg len to 221 as last segment
03-21 21:59:04.757   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: ReadHciEvent: Expected CC
03-21 21:59:04.757   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: FrameHciCmdPkt: Sending EDL_GET_BOARD_ID
03-21 21:59:04.757   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: HCI-CMD -1: 0x1   0x0   0xfc  0x1   0x23
03-21 21:59:04.757   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: GetBoardIdReq: Sending EDL_GET_BOARD_ID
03-21 21:59:04.758   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: ReadHciEvent: Expected CC
03-21 21:59:04.758   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: HandleEdlCmdResEvt: Board Id 4 3!!
03-21 21:59:04.758   667  5980 E vendor.qti.bluetooth@1.0-patch_dl_manager: /bt_firmware/image/gnnv20.b0403 File Open Fail No such file or directory (2)
03-21 21:59:04.759   667  5980 E vendor.qti.bluetooth@1.0-patch_dl_manager: /vendor/bt_firmware/image/gnnv20.b0403 File Opening from alternate path: Fail No such file or directory (2)
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: DownloadTlvFile: /bt_firmware/image/gnnv20.b0403: file doesn't exist, falling back to default file
03-21 21:59:04.759   667  5980 E vendor.qti.bluetooth@1.0-patch_dl_manager: /bt_firmware/image/gnnv20.bin File Open Fail No such file or directory (2)
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: File open /vendor/bt_firmware/image/gnnv20.bin succeeded
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: ====================================================
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: TLV Type       : 0x2
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: Length       : 3747 bytes
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: ====================================================
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: UpdateNewNvmFormat: baudrate 0e
03-21 21:59:04.759   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: UpdateNewNvmFormat: SIBS Disable
```



```
03-21 21:59:04.903   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: GetAddOnFeatureList: param_len 13, cmd_status = 0
03-21 21:59:04.903   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: GetAddOnFeatureList: feat_mask_len 5 product_id = 18, rsp_version = 1
03-21 21:59:04.903   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: EnableControllerLog: 0
03-21 21:59:04.904   667  5980 E vendor.qti.bluetooth@1.0-patch_dl_manager: ReadHciEvent: Unexpected event recieved rather than CC
03-21 21:59:04.904   667  5980 I vendor.qti.bluetooth@1.0-patch_dl_manager: HciReset: HCI RESET
03-21 21:59:04.904   667  5980 D vendor.qti.bluetooth@1.0-patch_dl_manager: ReadCmdCmplEvent: Cmd Cmpl received for opcode fc17
03-21 21:59:04.904   667  5980 D vendor.qti.bluetooth@1.0-uart_controller: add on features read true
03-21 21:59:04.905   667  5980 I vendor.qti.bluetooth@1.0-diag_interface: Init: Init diag completed
03-21 21:59:04.905   667  5980 D vendor.qti.bluetooth@1.0-logger: Init: Registered Diag callbacks
03-21 21:59:04.906   667  5980 D vendor.qti.bluetooth@1.0-uart_controller: Init succeded  Boot complete
03-21 21:59:04.906   667  5980 D vendor.qti.bluetooth@1.0-data_handler: Firmware download succeded.
```

6 AOSP Architecture for WLAN

Below diagram describe architecture for WLAN on android

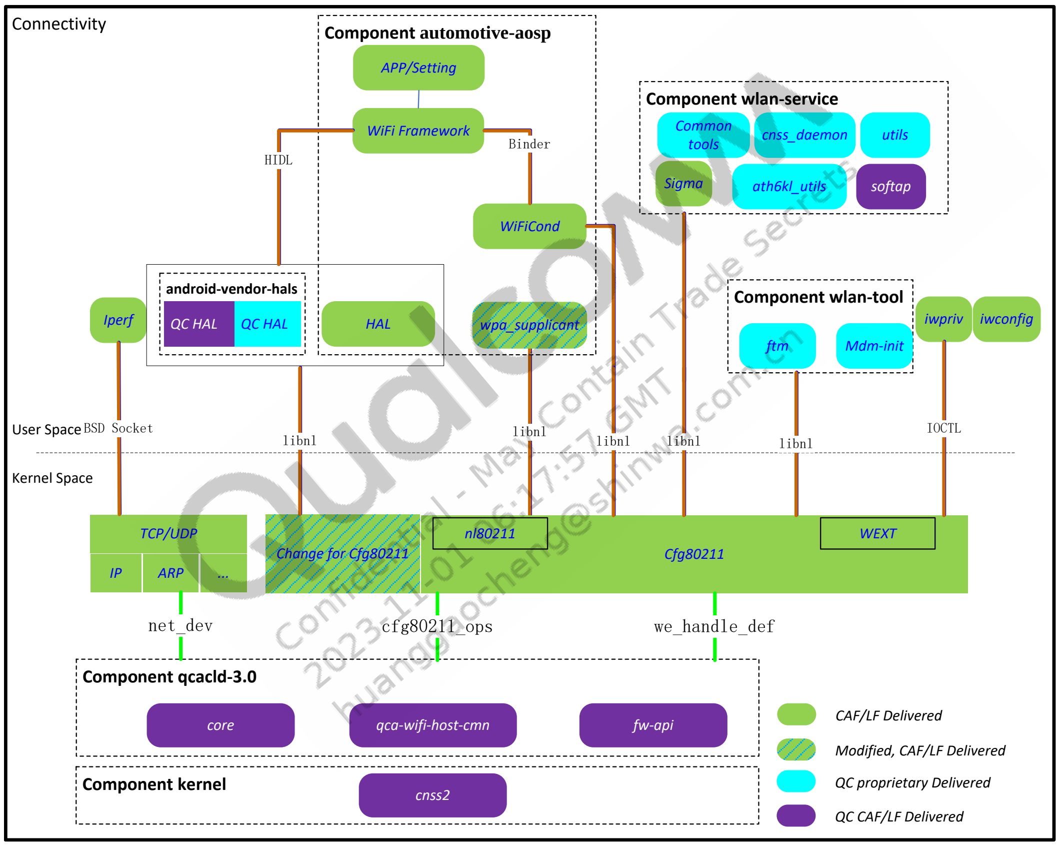


Figure 6-1 WLAN HIDL Architecture

7 AOSP Code Architecture for WLAN

Below table shows the code architecture at android system related to WLAN.

Table 7-1 shows the component related to Wi-Fi at android P/Q.

Table 7-1 Code path in Android P/Q

Component	File path	Function
Setting	packages/apps/Settings/src/com/android/settings/wifi	Wi-Fi Application
Car UI Setting	packages/apps/Car/Settings/src/com/android/car/settings/wifi	Wi-Fi Application For Car UI
Framework	frameworks/opt/net/wifi/service frameworks/base/wifi frameworks/base/packages/SettingsLib/src/com/android/settingslib/wifi	Android core to schedule scans and select network.
wificond	system/connectivity/wificond frameworks/opt/net/wifi/libwifi_hal frameworks/opt/net/wifi/libwifi_system frameworks/opt/net/wifi/libwifi_system_iface	Starts/stops supplicant and hostapd and handles scan commands.
wpa_supplicant	external/wpa_supplicant_8	Wi-Fi middleware component, for example, RSN/WPS/EAP/P2P.
hostapd		
HAL	hardware/interfaces/wifi/	Load/unload driver, and QCA vendor commands

Table 2-2 shows the module info of Wi-Fi at android P/Q.

Table 7-2 Module info in Android P/Q

File path	Function
packages/apps/Settings/src/com/android/settings/wifi	setting.apk
packages/apps/Car/Settings/src/com/android/car/settings/wifi	CarSettings.apk
frameworks/opt/net/wifi/service frameworks/base/wifi frameworks/base/packages/SettingsLib/src/com/android/settingslib/wifi	framework.jar libwifi-service.so
system/connectivity/wificond	wificond
frameworks/opt/net/wifi/libwifi_hal	libwifi-hal.so
frameworks/opt/net/wifi/libwifi_system	libwifi-system.so

frameworks/opt/net/wifi/libwifi_system_iface	libwifi-system-iface.so
vendor/qcom/opensource/softap	libqsap_sdk.so (only for Android P)
external/wpa_supplicant_8	wpa_supplicant wpa_cli hostapd hostapd_cli
hardware/interfaces/wifi/	android.hardware.wifi@1.0.so android.hardware.wifi@1.1.so android.hardware.wifi@1.2.so android.hardware.wifi@1.3.so(Android Q) android.hardware.wifi.hostapd@1.0.so android.hardware.wifi.hostapd@1.1.so(Android Q) android.hardware.wifi.offload@1.0.so android.hardware.wifi.supplicant@1.0.so android.hardware.wifi.supplicant@1.1.so android.hardware.wifi.supplicant1.2.so (Android Q)

8 Sample Startup log of WLAN

8.1 WLAN boot-up sequence

Below picture shows the boot-up chants of WLAN.

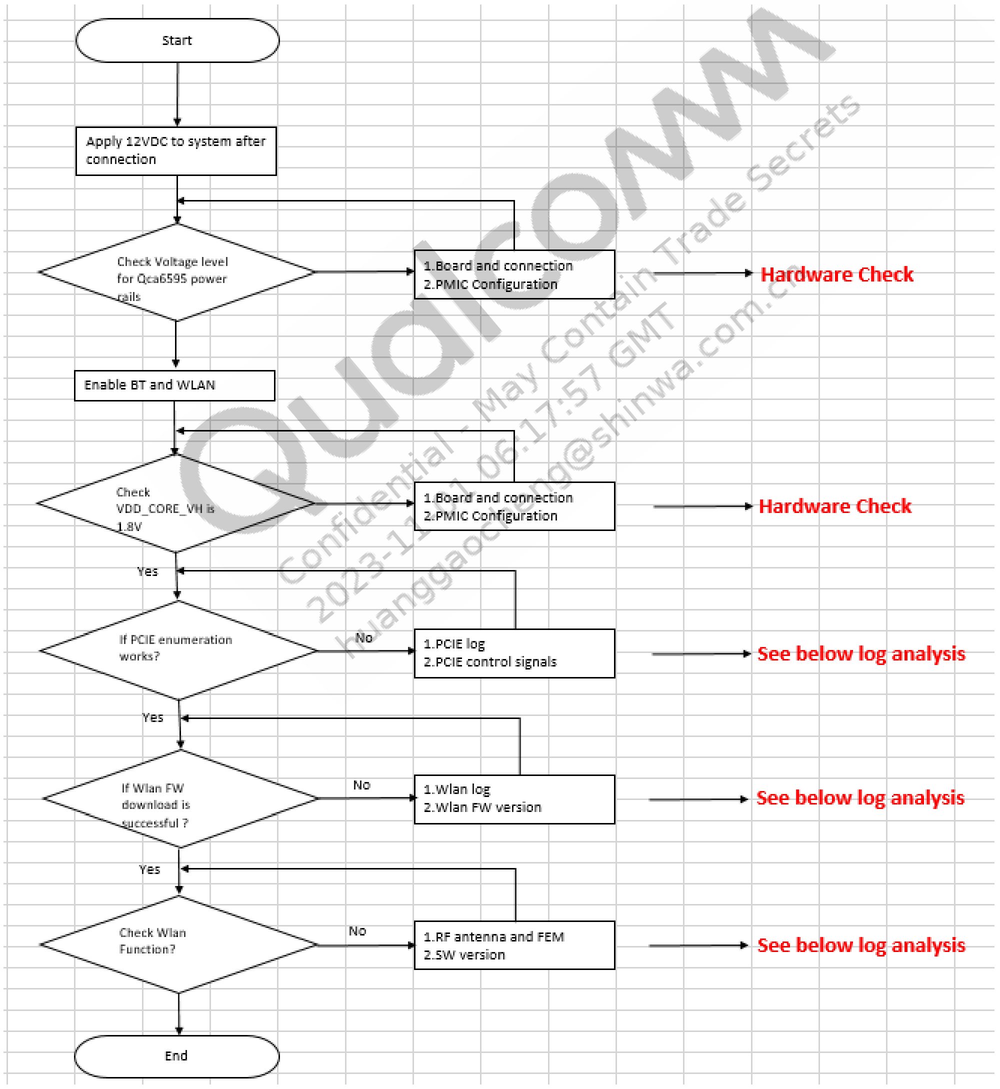
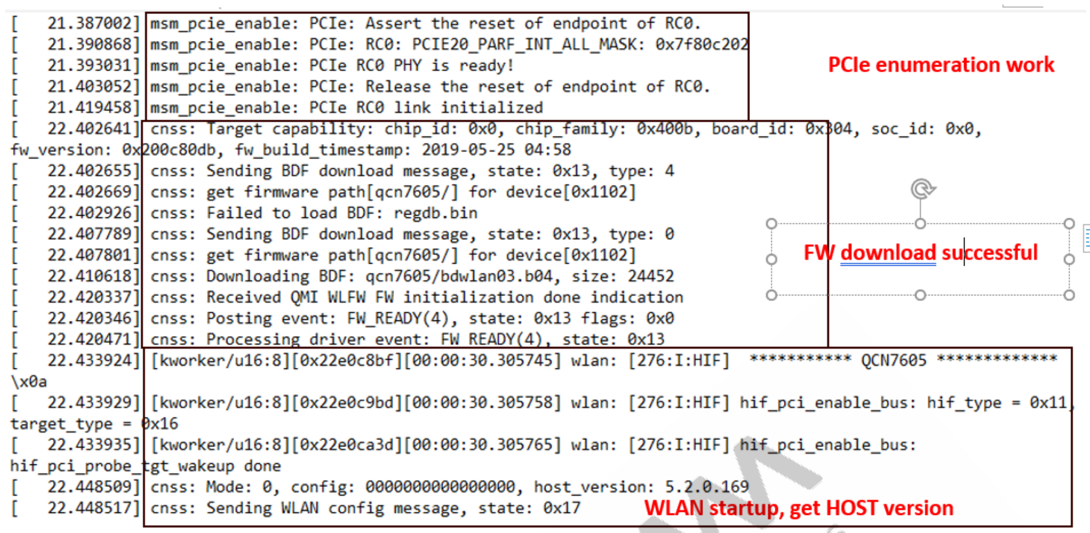


Figure 8-1 Boot-up chants of WLAN

Below log is based on Qualcomm SA8155 + QCA6595AU on android system. The logs are capture by `dmesg`, cut out the key message which shows different steps of WLAN boot-up.

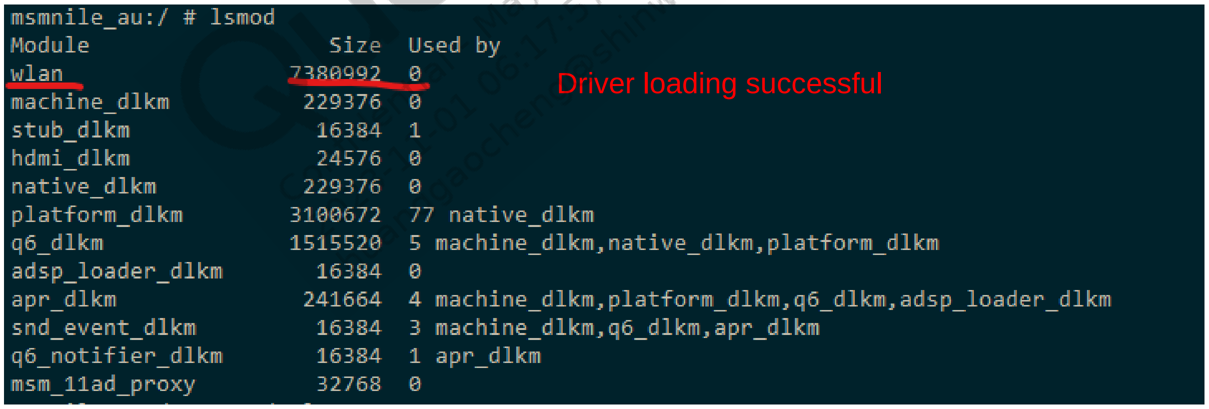


```
[   21.387002] msm_pcie_enable: PCIe: Assert the reset of endpoint of RC0.
[   21.390868] msm_pcie_enable: PCIe: RC0: PCIE20_PARF_INT_ALL_MASK: 0x7f80c202
[   21.393031] msm_pcie_enable: PCIe RC0 PHY is ready!
[   21.403052] msm_pcie_enable: PCIe: Release the reset of endpoint of RC0.
[   21.419458] msm_pcie_enable: PCIe RC0 link initialized
[   22.402641] cnss: Target capability: chip_id: 0x0, chip_family: 0x400b, board_id: 0x304, soc_id: 0x0,
fw_version: 0x200c80db, fw_build_timestamp: 2019-05-25 04:58
[   22.402655] cnss: Sending BDF download message, state: 0x13, type: 4
[   22.402669] cnss: get firmware path[qcn7605/] for device[0x1102]
[   22.402926] cnss: Failed to load BDF: regdb.bin
[   22.407789] cnss: Sending BDF download message, state: 0x13, type: 0
[   22.407801] cnss: get firmware path[qcn7605/] for device[0x1102]
[   22.410618] cnss: Downloading BDF: qcn7605/bdwlan03.b04, size: 24452
[   22.420337] cnss: Received QMI WLFW FW initialization done indication
[   22.420346] cnss: Posting event: FW_READY(4), state: 0x13 flags: 0x0
[   22.420471] cnss: Processing driver event: FW_READY(4), state: 0x13
[   22.433924] [kworker/u16:8][0x22e0c8bf][00:00:30.305745] wlan: [276:I:HIF]  *********** QCN7605 *************
\x0a
[   22.433929] [kworker/u16:8][0x22e0c9bd][00:00:30.305758] wlan: [276:I:HIF] hif_pci_enable_bus: hif_type = 0x11,
target_type = 0x16
[   22.433935] [kworker/u16:8][0x22e0ca3d][00:00:30.305765] wlan: [276:I:HIF] hif_pci_enable_bus:
hif_pci_probe_tgt_wakeup done
[   22.448509] cnss: Mode: 0, config: 0000000000000000, host_version: 5.2.0.169
[   22.448517] cnss: Sending WLAN config message, state: 0x17
```

8.2 Basic WLAN function verification

Use adb shell to check the WLAN driver module status. These parts are based

- `lsmod` (loaded modules)
- WLAN driver module be named wlan or qca7605

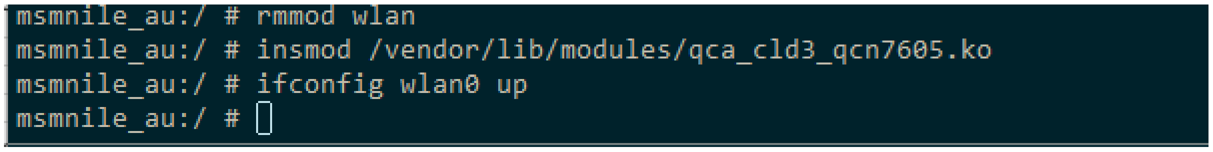


```
msmnile_au:/ # lsmod
Module                  Size  Used by
wlan                 7380992  0
machine_dlkm          229376  0
stub_dlkm              16384  1
hdmi_dlkm              24576  0
native_dlkm           229376  0
platform_dlkm        3100672  77 native_dlkm
q6_dlkm              1515520  5 machine_dlkm,native_dlkm,platform_dlkm
adsp_loader_dlkm       16384  0
apr_dlkm              241664  4 machine_dlkm,platform_dlkm,q6_dlkm,adsp_loader_dlkm
snd_event_dlkm         16384  3 machine_dlkm,q6_dlkm,apr_dlkm
q6_notifier_dlkm       16384  1 apr_dlkm
msm_11ad_proxy         32768  0
```

To remove driver module

- `rmmod wlan` (remove module)

If module has a different name such as QCA7605 use `rmmod qca7605`



```
msmnile_au:/ # rmmod wlan
msmnile_au:/ # insmod /vendor/lib/modules/qca_cld3_qcn7605.ko
msmnile_au:/ # ifconfig wlan0 up
msmnile_au:/ #
```

Install driver module in mission mode

- insmod /vendor/lib/modules/qca_cld3_qcn7605.ko (install module)

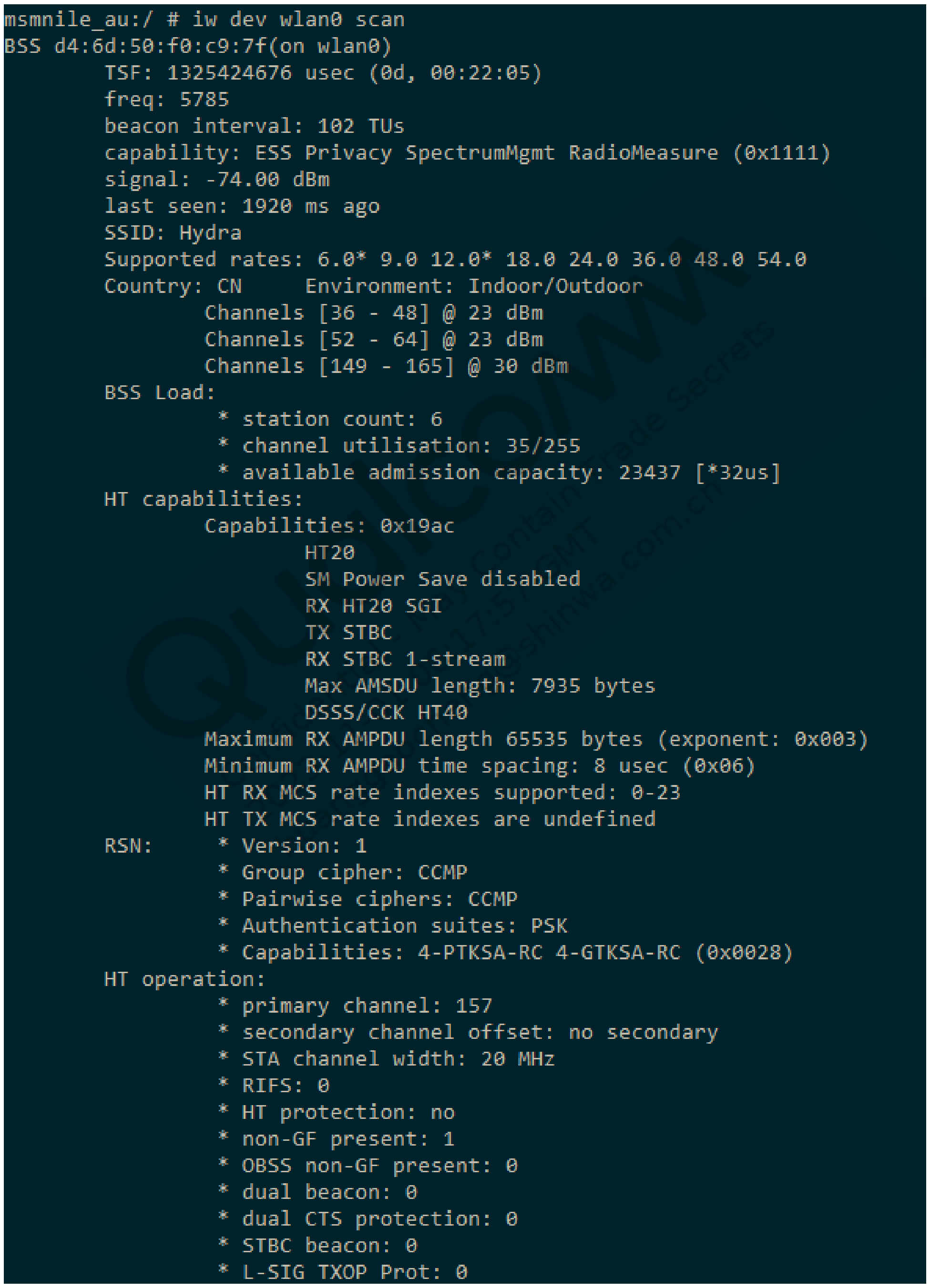
  The location is specific to different platform

- ifconfig wlan0 up

  bring wlan0 interface up

- iw dev wlan0 scan

  scan Wi-Fi device around and get the scan list



```
msmnile_au:/ # iw dev wlan0 scan
BSS d4:6d:50:f0:c9:7f(on wlan0)
        TSF: 1325424676 usec (0d, 00:22:05)
        freq: 5785
        beacon interval: 102 TUs
        capability: ESS Privacy SpectrumMgmt RadioMeasure (0x1111)
        signal: -74.00 dBm
        last seen: 1920 ms ago
        SSID: Hydra
        Supported rates: 6.0* 9.0 12.0* 18.0 24.0 36.0 48.0 54.0
        Country: CN     Environment: Indoor/Outdoor
                Channels [36 - 48] @ 23 dBm
                Channels [52 - 64] @ 23 dBm
                Channels [149 - 165] @ 30 dBm
        BSS Load:
                 * station count: 6
                 * channel utilisation: 35/255
                 * available admission capacity: 23437 [*32us]
        HT capabilities:
                Capabilities: 0x19ac
                        HT20
                        SM Power Save disabled
                        RX HT20 SGI
                        TX STBC
                        RX STBC 1-stream
                        Max AMSDU length: 7935 bytes
                        DSSS/CCK HT40
                Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
                Minimum RX AMPDU time spacing: 8 usec (0x06)
                HT RX MCS rate indexes supported: 0-23
                HT TX MCS rate indexes are undefined
        RSN:     * Version: 1
                 * Group cipher: CCMP
                 * Pairwise ciphers: CCMP
                 * Authentication suites: PSK
                 * Capabilities: 4-PTKSA-RC 4-GTKSA-RC (0x0028)
        HT operation:
                 * primary channel: 157
                 * secondary channel offset: no secondary
                 * STA channel width: 20 MHz
                 * RIFS: 0
                 * HT protection: no
                 * non-GF present: 1
                 * OBSS non-GF present: 0
                 * dual beacon: 0
                 * dual CTS protection: 0
                 * STBC beacon: 0
                 * L-SIG TXOP Prot: 0
```

9 Porting WLAN from QCA6595AU.LE.0.1

9.1 Download code

Get proprietary code from Qualcomm ChipCode™.

1. If you are new to the Qualcomm ChipCode, review the following link for up-to-date documentation and a set of tutorial videos:

   https://chipcode.qti.qualcomm.com/projects/help/wiki

2. Create another top-level directory on the build PC. Download the proprietary software from Qualcomm ChipCode.

   ChipCode link: https://chipcode.qti.qualcomm.com/qualcomm/qca6595au-le-0-1_qca_oem/tree/r00003.2

3. Unzip each of the subsystem images to generate the following directory structure. In this example, <qca6595_target_root> is the top-level directory.

```
<qca6595_target_root>
├── bt_firmware
│       └── btfm_proc
├── host
│       └── apps_proc
│             └── sources
│                   ├── meta-qti-connectivity-prop
│                   └── wlan-proprietary
├──meta_images
│       └── load_meta
│           ├── bt_firmware
│           ├── wlan_firmware
│           └── SD_Image
└── wlan_firmware
        └── wlan_proc
```

Get open source code from CAF.

1. Change working directory to <qca6595_target_root>/host/apps_proc

```
$ cd <qca6595_target_root>/host/apps_proc
```

2. Locate the <build_id> tag, which identifies the corresponding open source HLOS software build. Download from CAF.

```
$ repo init -u git://codeaurora.org/quic/le/le/manifest.git -b release -
m CHSS.LNX_FSLR.1.0.r1-00600-QCAAUTOHOSTHZ.xml --repo-
url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable
```

3. Sync the source codes.

```
$ repo sync
```

After that, <qca6595_target_root> folders show as follows.

```
<qca6595_target_root>
   ├── bt_firmware
   │     └── btfm_proc
   └── host
         └── apps_proc
               ├── .repo
               └── sources
                     ├── meta-fsl-bsp-release
                     ├── meta-openembedded
                     ├── meta-qt5
                     ├── meta-qti-connectivity
                     ├── meta-qti-connectivity-prop
                     ├── poky
                     ├── wlan-opensource
                     └── wlan-proprietary
   └── meta_images
         └── load_meta
               ├── bt_firmware
               ├── wlan_firmware
               └── SD_Image
   └── wlan_firmware
         └── wlan_proc
```

9.2 Prepare WLAN firmware and configuration files

Files used to bring up wlan driver

- **amss.bin**: WLAN target firmware
- **bdwlan.bin**: WLAN board data, customer can get specific file from module maker.
- **genoaftm.bin**: WLAN test mode firmware
- **qcom_cfg.ini:** driver configuration file.
- **wlan_mac.bin:** wlan mac address configuration file

get file "amss.bin", "bdwlan.bin", "genoaftm.bin" from folder

`<qca6595_target_root>\meta_image\load_meta\wlan_firmware\`

get file "QCA6595.LE.0.1_Genoa_PCIe_qcacld-3.0-iMX8.ini" from folder `<qca6595_target_root>\host\apps_proc\sources\wlan-opensource\mdm-init\wlan_standalone\`, and rename to "`qcom_cfg.ini`".

create file "wlan_mac.bin", the file format is as below

```
Intf0MacAddress=000AF58989FF
Intf1MacAddress=< MAC address for wlan1 interface >
END
```

9.3 Apply kernel Patch

Find the kernel patch at folder “<qca6595_target_root>\host\apps_proc\sources\meta-qti-connectivity\recipes-kernel\linux-kernel\files\lk-4.14”

Apply these kernel patch

```
"
0001-cfg80211-Updated-nl80211_commands-to-be-in-sync-with.patch
0002-cfg80211-nl80211-Optional-authentication-offload-to-.patch
0003-nl80211-Free-connkeys-on-external-authentication-fai.patch
0004-nl80211-Allow-SAE-Authentication-for-NL80211_CMD_CON.patch
0005-nl80211-Fix-external_auth-check-for-offloaded-authen.patch
0006-cfg80211-Authentication-offload-to-user-space-in-AP-.patch
0007-cfg80211-Sync-nl80211-commands-feature-with-upstream.patch
0008-nl80211-Allow-set-del-pmksa-operations-for-AP.patch
0009-cfg80211-nl80211-Offload-OWE-processing-to-user-spac.patch
0010-reg-qcom-call-regulatory-callback-for-self-managed-h.patch
0011-cfg80211-Add-flags-to-support-WPA3-STA-AP-function.patch
0012-imx-kernel.lnx.4.14.98-change-file-to-support-buildi.patch
"
```

9.4 Prepare WLAN driver and related modules

Files used to bring up wlan driver

- **wlan_cnss_core_pcie.ko:** wlan cnss core driver, include cnss2/mhi/diag/qrtr/qmi code
- **qrtr-ns:** qrtr-ns service, necessary unit for creating communication fort qrtr.
- **wlan-genoa-pcie.ko:** wlan driver for genoa chip

build wlan_cnss_core_pcie.ko:

```
   find code from "<qca6595_target_root>\host\apps_proc\sources\wlan-
opensource\wlan-cnss-core", use platform specific toolchain and kernel
header to build the code, below is one example from i.mx8 platform.
"
export TOOLCHAIN=/local/mnt/workspace/gongx/LE26/qca6574au-le-2-
6_qca_oem.git/host_4.14/apps_proc/build/tmp/work/aarch64-mx8m-poky-
linux/imx-m4-demos/2.3.0-r0/recipe-sysroot-native/usr/bin/aarch64-poky-
linux
export ARCH=arm64
export CROSS_COMPILE=${TOOLCHAIN}/aarch64-poky-linux-
make KERNEL_SRC=/local/mnt/workspace/gongx/LE26/qca6574au-le-2-
6_qca_oem.git/host_4.14/apps_proc/build/tmp/work-
shared/imx8mqevk/kernel-source
KERNEL_PATH=/local/mnt/workspace/gongx/LE26/qca6574au-le-2-
6_qca_oem.git/host_4.14/apps_proc/build/tmp/work-
shared/imx8mqevk/kernel-source
O=/local/mnt/workspace/gongx/LE26/qca6574au-le-2-
6_qca_oem.git/host_4.14/apps_proc/build/tmp/work-
shared/imx8mqevk/kernel-build-artifacts
```

"

build qrtr-ns:

find `code from "<qca6595_target_root>\host\apps_proc\sources\wlan-proprietary\qcmbr\qrtr-ns", refer the file "<qca6595_target_root>\host\apps_proc\sources\wlan-proprietary\qcmbr\qrtr-ns\Android.mk"` to build application

build wlan-genoa-pcie.ko:

find code from

```
"<qca6595_target_root>\host\apps_proc\sources\wlan-opensource\qca-wifi-host-cmn",
 "<qca6595_target_root>\host\apps_proc\sources\wlan-opensource\qcacld-3.0", "<qca6595_target_root>\host\apps_proc\sources\wlan-opensource\fw-api",
Then put them to one folder, refer the file
"<qca6595_target_root>\host\apps_proc\sources\ wlan-opensource\ qcacld-3.0\Andorid.mk" to build module
Add  "CONFIG_CNSS2=y" and "
WLAN_CFG_OVERRIDE="CONFIG_IPA_DISABLE_OVERRIDE=y""
To "KBUILD_OPTIONS" at "Android.mk".
```

9.5 Put the file to system

Here is an example of these binary at Yotco i.mx8 system, for android system, customer should put them to related folder.

```
/lib/firmware/qcn7605/amss.bin
/lib/firmware/qcn7605/bdwlan.bin
/lib/firmware/qcn7605/genoaftm.bin
/lib/firmware/qcn7605/wlan/qcom_cfg.ini
/lib/firmware/qcn7605/wlan/wlan_mac.bin
/lib/modules/wlan_cnss_core_pcie.ko
/lib/modules/wlan-genoa-pcie.ko
usr/bin/qrtr-ns
```

9.6 Start wlan driver

1. Check PCIe interfaces on target serial port command prompt, confirm the second PCIE device ID is 1102.

   ```
   # lspci
   0000:00:00.0 PCI bridge: Synopsys, Inc. Device abcd (rev 01)
   0000:01:00.0 Network controller: Qualcomm Atheros QCA6174 802.11ac
   Wireless Network Adapter (rev 32)
   0001:00:00.0 PCI bridge: Synopsys, Inc. Device abcd (rev 01)
   0001:01:00.0 Network controller: Qualcomm Device 1102 (rev 01)
   ```

2. Confirm wlan_cnss_core_pcie module is running

   ```
   # lsmod
   ```

```
Module                  Size  Used by
wlan_cnss_core_pcie   643072  3
if module wlan_cnss_core_pcie is not running, run command “insmod
wlan_cnss_core_pcie.ko” to insmod wlan_cnss_core_pcie module
# cd /lib/modules/${Kernel Version}/extra/
# insmod wlan_cnss_core_pcie.ko
```

3. Running qrtr-ns service

   Run qrtr-ns service by following command

```
# qrtr-ns
```

4. Insmod wlan host modules

```
# cd /lib/modules/${Kernel Version}/extra/
# insmod wlan-genoa-pcie.ko
# lsmod
Module                  Size  Used by
wlan_genoa_pcie      6746112  0
wlan_cnss_core_pcie   643072  6 wlan_genoa_pcie
```

5. Use *ifconfig* command to bring up WLAN interface on Linux

```
# ifconfig wlan0 up
# ifconfig
wlan0     Link encap:Ethernet  HWaddr 00:03:7F:61:10:31
          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)
```

9.7 Sample log at i.mx8 platform

Here is sample log at i.mx8 platform, customer can compare their log with it.

```
root@imx8mqevk:/lib/modules/4.14.98-2.3.1/extra# insmod
wlan_cnss_core_pcie.ko
[   63.729822] NET: Registered protocol family 42
[   63.735592] [__diag_rpmsg_init] DIAG_CTRL initialized fwd_ctxt:
0000000000000000 hdl: 0000000000000000
[   63.745002] [diagfwd_channel_open] p: 4 t: 1 considered opened
[   63.750890] [diag_state_open_rpmsg] DIAG_CTRL setting diag state to 1
[   63.751000] [__diag_rpmsg_init] DIAG_DATA initialized fwd_ctxt:
0000000000000000 hdl: 0000000000000000
[   63.766899] [__diag_rpmsg_init] DIAG_CMD initialized fwd_ctxt:
0000000000000000 hdl: 0000000000000000
[   63.776216] [__diag_rpmsg_init] DIAG_DCI_DATA initialized fwd_ctxt:
0000000000000000 hdl: 0000000000000000
[   63.785989] [__diag_rpmsg_init] DIAG_DCI_CMD initialized fwd_ctxt:
0000000000000000 hdl: 0000000000000000
[   63.795609] [diag_state_open_rpmsg] DIAG_CTRL setting diag state to 1
[   63.796412] [diag_mhi_init] mhi port 0 is initailzed
[   63.808073] [diag_mhi_init] mhi port 1 is initailzed
[   63.814117] unified_pdrv_init success
[   63.818566] subsys-restart: subsys_start(): [wlan]: before
wait_for_err_ready
[   63.825812] subsys-restart: subsys_start(): [wlan]: exit
[   63.831257] cnss2 qcom,cnss-qca-converged: for wlan segments only will
be dumped.
[   63.838992] cnss_pci 0001:01:00.0: BAR 0: assigned [mem 0x20100000-
0x201fffff 64bit]
[   63.846856] cnss_pci 0001:01:00.0: enabling device (0000 -> 0002)
[   63.873451] cnss: Platform driver probed successfully.
root@imx8mqevk:/lib/modules/4.14.98-2.3.1/extra# qrtr-ns
root@imx8mqevk:/lib/modules/4.14.98-2.3.1/extra# insmod wlan-genoa-pcie.ko
[   67.553107] wlan_genoa_pcie: Loading driver v5.2.0.185J; 2020-06-
04T02:31:43Z; cld:; cmn:;
[   67.562927] wlan_hdd_state wlan_genoa_pcie major(511) initialized
[   67.595254] [I][mhi_alloc_bhie_table] Allocating bytes:3145728
seg_size:524288 total_seg:7
[   67.604652] [I][mhi_alloc_bhie_table] Entry:0 Address:0x44300000
size:524288
[   67.612472] [I][mhi_alloc_bhie_table] Entry:1 Address:0x44380000
size:524288
[   67.620641] [I][mhi_alloc_bhie_table] Entry:2 Address:0x44400000
size:524288
[   67.628464] [I][mhi_alloc_bhie_table] Entry:3 Address:0x44480000
size:524288
[   67.636314] [I][mhi_alloc_bhie_table] Entry:4 Address:0x44500000
size:524288
[   67.644126] [I][mhi_alloc_bhie_table] Entry:5 Address:0x44580000
size:524288
[   67.651865] [I][mhi_alloc_bhie_table] Entry:6 Address:0x44134000 size:96
[   67.658590] [I][mhi_alloc_bhie_table] Successfully allocated bhi vec
table
[   67.665483] [E][mhi_prepare_for_power_up] Don't do memset_io for qcn7605
[   67.672201] [I][mhi_rddm_prepare] BHIe programming for RDDM
[   67.677794] [I][mhi_rddm_prepare] address:0x0000000044134000 len:0x60
sequence:557463079
[   67.685912] [I][mhi_async_power_up] Requested to power on
[   67.691478] [I][mhi_pm_st_worker] Transition to state:PBL
[   67.693096] [I][mhi_async_power_up] Power on setup success
[   67.696917] [I][mhi_fw_load_handler] Device current EE:PBL
[   67.835489] [I][mhi_fw_load_sbl] Starting BHI programming
[   67.840926] [I][mhi_fw_load_sbl] Waiting for image transfer completion
[   67.863658] [I][mhi_intvec_threaded_handlr] local ee:PBL device ee:PBL
dev_state:RESET
[   67.871608] [I][mhi_alloc_bhie_table] Allocating bytes:2740288
seg_size:524288 total_seg:7
```

```
[   67.880985] [I][mhi_alloc_bhie_table] Entry:0 Address:0x44600000
size:524288
[   67.888795] [I][mhi_alloc_bhie_table] Entry:1 Address:0x44680000
size:524288
[   67.896687] [I][mhi_alloc_bhie_table] Entry:2 Address:0x44700000
size:524288
[   67.904571] [I][mhi_alloc_bhie_table] Entry:3 Address:0x44780000
size:524288
[   67.913176] [I][mhi_alloc_bhie_table] Entry:4 Address:0x44800000
size:524288
[   67.920912] [I][mhi_alloc_bhie_table] Entry:5 Address:0x44880000
size:524288
[   67.928611] [I][mhi_alloc_bhie_table] Entry:6 Address:0x44135000 size:96
[   67.935336] [I][mhi_alloc_bhie_table] Successfully allocated bhi vec
table
[   67.942232] [I][mhi_fw_load_handler] Copying firmware image into vector
table
[   67.951143] [I][mhi_ready_state_transition] Waiting to enter READY state
[   67.990212] [I][mhi_ready_state_transition] Device in READY State
[   67.990224] [I][mhi_intvec_threaded_handlr] local ee:PBL device ee:SBL
dev_state:READY
[   68.004249] [I][mhi_init_mmio] Initializing MMIO
[   68.008877] [I][mhi_init_mmio] CHDBOFF:0x400
[   68.013168] [I][mhi_init_mmio] ERDBOFF:0xc00
[   68.017455] [I][mhi_init_mmio] Programming all MMIO values.
[   68.023065] [I][mhi_fw_load_handler] To Reset->Ready PM_STATE:POR
MHI_STATE:READY EE:PBL, ret:0
[   68.023416] [I][mhi_process_ctrl_ev_ring] MHI state change event to
state:M0
[   68.038830] [I][mhi_pm_m0_transition] Entered With State:READY
PM_STATE:POR
[   68.045832] [I][mhi_process_ctrl_ev_ring] MHI EE received event:SBL
[   68.052118] [I][mhi_fw_load_amss] Starting BHIe Programming
[   68.057716] [I][mhi_fw_load_amss] Upper:0x0 Lower:0x44135000 len:0x60
sequence:738279032
[   68.065827] [I][mhi_fw_load_amss] Waiting for image transfer completion
[   68.118691] [I][mhi_fw_load_handler] amss fw_load, ret:0
[   68.118698] [I][mhi_intvec_threaded_handlr] local ee:SBL device ee:SBL
dev_state:M0
[   68.124513] [I][mhi_pm_st_worker] Transition to state:SBL
[   68.125284] [I][mhi_process_ctrl_ev_ring] MHI EE received event:AMSS
[   68.143460] [I][mhi_pm_st_worker] Transition to state:MISSION MODE
[   68.149657] [I][mhi_pm_mission_mode_transition] Processing Mission Mode
Transition
[   68.157254] cnss: Unsupported MHI status cb reason: 6
[   68.162350] [I][mhi_pm_mission_mode_transition] Adding new devices
[   68.168753] [diag_mhi_probe] received probe for 0
[   68.173496] [diag_mhi_probe] diag: mhi device is ready to open
[   68.173501] [I][mhi_prepare_channel] Entered: preparing channel:4
[   68.187415] [I][mhi_prepare_channel] Chan:4 successfully moved to start
state
[   68.194578] [I][mhi_prepare_channel] Entered: preparing channel:5
root@imx8mqevk:/lib/modules/4.14.[   68.201802] [I][mhi_prepare_channel]
Chan:5 successfully moved to start state
98-2.3.1/extra# [   68.211167] [__mhi_open] opened mhi read/write channel,
port: 0
[   68.218530] [mhi_read_work_fn] queueing a read buf ffff8000b9075000, ch:
MDM
[   68.218703] [I][mhi_prepare_channel] Entered: preparing channel:16
[   68.225645] [mhi_read_work_fn] queueing a read buf ffff8000b9072000, ch:
MDM
[   68.238907] [mhi_read_work_fn] queueing a read buf ffff8000b9076000, ch:
MDM
[   68.239910] [I][mhi_prepare_channel] Chan:16 successfully moved to start
state
[   68.245984] [mhi_read_work_fn] queueing a read buf ffff8000b9071000, ch:
MDM
[   68.253306] [I][mhi_prepare_channel] Entered: preparing channel:17
[   68.253310] TX CTRL: cmd:0x2 node[0x1]
[   68.260272] [mhi_read_work_fn] queueing a read buf ffff8000b9073000, ch:
MDM
[   68.267625] [I][mhi_prepare_channel] Chan:17 successfully moved to start
state
[   68.270225] [mhi_read_work_fn] queueing a read buf ffff8000b9077000, ch:
MDM
[   68.274901] RX CTRL: cmd:0x2 node[0x7]
[   68.274923] RX CTRL: cmd:0x4 SVC[0x45:0x101] addr[0x7:0x1]
[   68.275148] cnss: QMI WLFW service connected, state: 0x11
[   68.275170] TX DATA: Len:0x3a CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[20000000 10003300] [kworker/u8:6]
[   68.277342] [I][mhi_pm_mission_mode_transition] Exit with ret:0
[   68.284531] [mhi_read_work_fn] queueing a read buf ffff8000b9070000, ch:
MDM
[   68.293685] RX DATA: Len:0x19 CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[20000002 02001200]
[   68.295345] [mhi_read_work_fn] queueing a read buf ffff8000b4098000, ch:
MDM
[   68.306224] [mhi_read_work_fn] queueing a read buf ffff8000b409f000, ch:
MDM
[   68.321291] [mhi_read_work_fn] queueing a read buf ffff8000b409c000, ch:
MDM
[   68.321298] [mhi_read_work_fn] queueing a read buf ffff8000b409a000, ch:
MDM
[   68.336191] [mhi_read_work_fn] queueing a read buf ffff8000b409b000, ch:
MDM
[   68.350303] [mhi_read_work_fn] queueing a read buf ffff8000b4099000, ch:
MDM
```

```
[   68.350307] [mhi_read_work_fn] queueing a read buf ffff8000b409e000, ch:
MDM
[   68.350311] [mhi_read_work_fn] queueing a read buf ffff8000b409d000, ch:
MDM
[   68.364427] [mhi_read_work_fn] queueing a read buf ffff8000b94de000, ch:
MDM
[   68.371531] TX DATA: Len:0x25 CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[34000100 10001e00] [kworker/u8:6]
[   68.378525] [mhi_read_work_fn] queueing a read buf ffff8000b94df000, ch:
MDM
[   68.387403] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[34000102 02000700]
[   68.392642] [mhi_read_work_fn] queueing a read buf ffff8000b94dd000, ch:
MDM
[   68.399702] RX DATA: Len:0x1d CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[35000104 01001600]
[   68.440729] TX DATA: Len:0x2d CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[36000200 01002600] [kworker/u8:6]
[   68.462425] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[36000202 02000700]
[   68.470209] RX DATA: Len:0x7 CF:0x1 src[0x7:0x1] dst[0x1:0x4001]
[00000000 00000000]
[   68.478010] TX CTRL: cmd:0x7 addr[0x1:0x4001]
[   68.499964] TX DATA: Len:0x7 CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[00000000 00000000] [kworker/u8:0]
[   68.518658] RX DATA: Len:0x38 CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[24000302 02003100]
[   68.526681] cnss: Failed to load BDF: regdb.bin
[   68.542753] TX DATA: Len:0x182d CF:0x1 src[0x1:0x4001] dst[0x7:0x1]
[25000400 01182600] [kworker/u8:0]
[   68.556163] RX CTRL: cmd:0x7 addr[0x7:0x1]
[   68.560283] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[25000402 02000700]
[   68.568230] TX DATA: Len:0x182d CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[25000500 01182600] [kworker/u8:0]
[   68.593664] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[25000502 02000700]
[   68.601678] TX DATA: Len:0x182d CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[25000600 01182600] [kworker/u8:0]
[   68.631150] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[25000602 02000700]
[   68.639239] TX DATA: Len:0x17b1 CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[25000700 0117aa00] [kworker/u8:0]
[   68.668651] RX DATA: Len:0x7 CF:0x1 src[0x7:0x1] dst[0x1:0x4001]
[00000000 00000000]
[   68.676435] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[25000702 02000700]
[   68.676444] TX CTRL: cmd:0x7 addr[0x1:0x4001]
[   68.720449] [kworke][0x4172bc2][13:07:10.707938] wlan: [5:E:HDD]
hdd_dp_trace_init: 13103: busBandwidthComputeInterval is 0, using defaults
[   68.733097] TX DATA: Len:0xb CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[2f000800 10000400] [kworker/u8:0]
[   68.743650] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[2f000802 02000700]
[   68.751454] [kworke][0x417a4e3][13:07:10.738948] wlan: [5:I:HIF]
hif_pci_enable_bus: con_mode = 0x0, device_id = 0x1102
[   68.762272] [kworke][0x417cf24][13:07:10.749765] wlan: [5:I:HIF]
hif_pci_enable_bus: hif_enable_pci done
[   68.771793] [kworke][0x417f453][13:07:10.759283] wlan: [5:I:HIF]
*********** QCN7605 *************
[   68.771793]
[   68.782338] [kworke][0x4181d87][13:07:10.769832] wlan: [5:I:HIF]
hif_pci_enable_bus: hif_type = 0x11, target_type = 0x16
[   68.793232] [kworke][0x4184815][13:07:10.780726] wlan: [5:I:HIF]
hif_pci_enable_bus: hif_pci_probe_tgt_wakeup done
[   68.803608] [kworke][0x418709b][13:07:10.791100] wlan: [5:E:TXRX]
hif_print_hal_shadow_register_cfg: CONFIG_SHADOW_V2 not defined
[   68.816200] TX DATA: Len:0x227 CF:0x1 src[0x1:0x4001] dst[0x7:0x1]
[23000900 10022000] [kworker/u8:0]
[   68.837409] RX CTRL: cmd:0x7 addr[0x7:0x1]
[   68.841550] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[23000902 02000700]
[   68.849393] TX DATA: Len:0x12 CF:0x0 src[0x1:0x4001] dst[0x7:0x1]
[22000a00 01000b00] [kworker/u8:0]
[   68.874904] RX DATA: Len:0xe CF:0x0 src[0x7:0x1] dst[0x1:0x4001]
[22000a02 02000700]
[   68.888197] [kworke][0x419bb05][13:07:10.875686] wlan: [5:I:HIF]
hif_configure_irq: E
[   68.896258] [kworke][0x419da87][13:07:10.883752] wlan: [5:W:HIF]
hif_napi_create: bad IRQ value for CE 1: -22
[   68.906215] [kworke][0x41a016c][13:07:10.893709] wlan: [5:W:HIF]
hif_napi_create: bad IRQ value for CE 3: -22
[   68.916156] [kworke][0x41a2842][13:07:10.903651] wlan: [5:W:HIF]
hif_napi_create: bad IRQ value for CE 8: -22
[   68.926216] [kworke][0x41a4f8d][13:07:10.913710] wlan: [5:E:mlme]
mlme_init_reg_cfg: 2350: null pdev
[   68.935413] [kworke][0x41a737a][13:07:10.922907] wlan: [5:E:QDF]
cds_get_context: 1463: Module ID 66 context is Null
[   68.945982] [kworke][0x41a9cc4][13:07:10.933477] wlan: [5:E:HDD]
hdd_wlan_start_modules: 3718: WBUFF init unsuccessful; status: 11
[   68.959255] [kworke][0x41ad09a][13:07:10.946747] wlan: [5:E:QDF]
cds_get_context: 1463: Module ID 66 context is Null
[   68.969922] [kworke][0x41afa47][13:07:10.957416] wlan: [5:F:WMA] WMA -->
wmi_unified_attach - success
[   68.983790] [kworke][0x41b3072][13:07:10.971283] wlan: [5:E:QDF]
htc_wait_target: 657: Target Ready! TX resource : 2 size:1748,
MaxMsgsPerHTCBundle = 1
```

```
[   68.997372] [kworke][0x41b6581][13:07:10.984866] wlan: [5:E:QDF]
htc_setup_target_buffer_assignments: 549: SVS Index : 1 TX : 0x100 :
alloc:2
[   69.012250] [kworke][0x41b9f9f][13:07:10.999744] wlan: [3080:I:WMA]
wma_rx_service_ready_event: Firmware build version : 220b8066
[   69.023935] [kworke][0x41bcd44][13:07:11.011429] wlan: [3080:I:WMA]
wma_rx_service_ready_event: Board id: 0, Board version: 0 0 0 0 0
[   69.035963] [kworke][0x41bfc40][13:07:11.023457] wlan: [3080:I:TXRX]
ol_tx_set_desc_global_pool_size: 131: Global pool size: 2000
 [   69.050238] [kworke][0x41c3403][13:07:11.037732] wlan: [3080:E:WMA]
wma_update_supported_bands: wrong supported band
[   69.060968] [kworke][0x41c5deb][13:07:11.048460] wlan: [3080:E:QDF]
copy_fw_abi_version_tlv: 7060: copy_fw_abi_version_tlv: INIT_CMD version:
1, 0, 0x5f414351, 0x4c4d, 0x0, 0x0
[   69.103560] [kworke][0x41d044a][13:07:11.091051] wlan: [3080:E:QDF]
ready_extract_init_status_tlv: 9252: ready_extract_init_status_tlv:0
[   69.116035] [kworke][0x41d3508][13:07:11.103529] wlan: [3080:I:dfs]
WLAN_DEBUG_DFS_ALWAYS : wlan_dfs_pdev_obj_create_notification: 431:
dfs_offload 0
[   69.129749] [kworke][0x41d6a9a][13:07:11.117243] wlan: [3080:E:dfs]
WLAN_DEBUG_DFS_ALWAYS : dfs_init_radar_filters: 252: Unknown dfs domain 0
[   69.142597] [kworke][0x41d9ccb][13:07:11.130092] wlan: [3080:I:HDD]
hdd_update_tgt_cfg: 2306: hw_mac is zero
[   69.152783] [kworke][0x41dc493][13:07:11.140277] wlan: [5:I:WMA]
wma_wait_for_ready_event: 6968: FW ready event received
[   69.196750] [kworke][0x41e704d][13:07:11.184238] wlan: [5:I:HTT] max
pool size 64 pool filled 64
[   69.211966] [kworke][0x41eabc0][13:07:11.199457] wlan: [5:I:SYS]
Creating periodic timer for OLBC UPDATE CACHE TIMEOUT
 [   69.224657] [schedu][0x41edd57][13:07:11.212151] wlan: [3371:E:PE]
pe_register_callbacks_with_wma: 1384: Registering roaming callbacks with
WMA failed
[   69.224737] [kworke][0x41edda6][13:07:11.212231] wlan: [5:E:REGULATORY]
reg_freq_width_to_chan_op_class: 591: invalid frequency 5660
[   69.260775] [kworke][0x41f6a69][13:07:11.248266] wlan: [5:I:HDD]
hdd_initialize_mac_address: 11807: using MAC address from wlan_mac.bin
[   69.276043] [kworke][0x41fa60d][13:07:11.263533] wlan: [5:I:HDD]
hdd_psoc_idle_timer_start: 10543: psoc idle timer is disabled
```

10 Porting WLAN from SDK.1.0

Auto Connectivity Image Set SDK.1.0 provides the release for connectivity components. Each of these releases provide enhancements to WLAN and Bluetooth software functionality for various QTI connectivity chipsets. The software is organized by a set of images and software products. Images are the connectivity software bundles and software products are the testing context and delivery platform for the connectivity images.

10.1 WLAN Software Images in SDK.1.0

The 2020.R2 image set consists of the following WLAN software images:

Image Name	Description
WLAN.GNO.2.2	WLAN firmware for QCA6595AU 2.0 devices
WLAN.RM.4.5.1	WLAN firmware for QCA6574AU devices
WLAN.RM.4.5.3	WLAN firmware for QCA6574AU devices,
WLAN.HST.1.0.2	WLAN firmware for QCA6696AU 2.0 devices
WLANHost_AU.LE.2.1	WLAN Host driver, wpa_supplicant, tools for LE project
WLANHost_AU.LA.3.1	WLAN Host driver, wpa_supplicant, Framework, tools for LA project

10.2 Porting SDK.1.0 WLAN To Third Party Android

Download the QCOM SDK.1.0 Package

The package can be downloaded from https://chipcode.qti.qualcomm.com.

- Select Auto_Connectivity_Image_Set.SDK.1.0 from the products list.
- Click <customer name>/auto-connectivity-image-set-sdk-1-0_qca_oem to download the code.

10.2.1 Bring up Corresponding LE project

SDK.1.0 do not include CNSS driver for QCA6574/QCA6595/QCA6696, and MHI driver for QCA6595/QCA6696. You need to bring up standard LE project to make CNSS/MHI driver work before Android platform porting.

The standard LE project can be download from https://chipcode.qti.qualcomm.com. They are QCA6574AU.LE.2.2.1, QCA6595.LE.0.1, QCA6696.LE.0.1.

Based on the LE project, bring up the target platform Linux kernel and WLAN host driver.

10.2.2 Porting SDK.1.0 into Android Q

- Download Android Q for the target platform.
- Sync the code in HOST_LA, replace these components into target platform Android Q.

  ```
  cd auto-connectivity-image-set-sdk-1-0_qca_oem/HOST_LA/apps_proc
  ```

  - wlan host driver

    ```
    git clone https://source.codeaurora.org/quic/la/platform/vendor/qcom-
    opensource/wlan/fw-api/ -b wlan-api.lnx.1.0.c20.2
    "vendor/qcom/opensource/wlan/fw-api"
    ```

    ```
    git clone https://source.codeaurora.org/quic/la/platform/vendor/qcom-
    opensource/wlan/qcacld-3.0/ -b wlan-cld3.driver.lnx.2.0.c20.2
    "vendor/qcom/opensource/wlan/qcacld-3.0"
    ```

    ```
    git clone https://source.codeaurora.org/quic/la/platform/vendor/qcom-
    opensource/wlan/qca-wifi-host-cmn/ -b wlan-cmn.driver.lnx.2.0.c20.2
    "vendor/qcom/opensource/wlan/qca-wifi-host-cmn"
    ```

  - wpa_supplicant

    ```
    git clone
    https://source.codeaurora.org/quic/la/platform/external/wpa_supplican
    t_8/ -b q-aosp-automotive.c20.2 "external/wpa_supplicant_8"
    ```

  - Android Q framework

    ```
    git clone
    https://source.codeaurora.org/quic/la/platform/frameworks/opt/net/wif
    i/ -b  q-aosp-automotive.c20.2 "frameworks/opt/net/wifi"
    ```

  - Android hardware

    ```
    git clone
    https://source.codeaurora.org/quic/la/platform/hardware/qcom/wlan -b
    wlan-aosp.lnx.5.0.c20.2 "hardware/qcom/wlan"
    ```

  - wlan configuration file

    ```
    git clone
    https://source.codeaurora.org/quic/la/platform/vendor/qcom/wlan -b
    wlan-service.lnx.5.0.c20.2 "device/qcom/wlan"
    ```

    ```
    git clone
    https://source.codeaurora.org/quic/la/platform/vendor/qcom/MSMnileau
    -b qcom-devices-auto.lnx.2.1.r1-rel "device/qcom/MSMnile_au"
    ```

  - sigma-dut tool

    ```
    git clone https://source.codeaurora.org/quic/la/platform/vendor/qcom-
    opensource/wlan/utils/sigma-dut/ -b wlan-service.lnx.5.0.c20.2
    "vendor/qcom/opensource/wlan/utils/sigma-dut"
    ```

- Check the WLAN configuration file

  ```
  WCNSS_qcom_cfg_qca6174.ini at /device/qcom/wlan/msmnile_au
  WCNSS_qcom_cfg_qca6390.ini at /device/qcom/wlan/msmnile_au
  WCNSS_qcom_cfg_qca7605.ini at /device/qcom/wlan/msmnile_au
  ```

- Add Qualcomm proprietary code to target Android Q

  ```
  The proprietary code path:
  Linux SP:
  [SP NAME]/host_proc/vendor/qcom/proprietary/
  Android SP:
  [SP NAME]/ LINUX/android/vendor/qcom/proprietary
  /host_proc/vendor/qcom/proprietary/wlan/oem/oem-ss
  /host_proc/vendor/qcom/proprietary/wlan/utils
  /host_proc/vendor/qcom/proprietary/wlan/ath6kl-utils
  /host_proc/vendor/qcom/proprietary/wlan/cnss-daemon
  /host_proc/vendor/qcom/proprietary/wlan/common-tools
  /host_proc/vendor/qcom/proprietary/ftm
  ```

- Integrate WLAN into Android Q and install WLAN
  - Modify android.mk file to add following parameters.

    ```
    WPA := wpa_supplicant.conf
    WPA += wpa_supplicant
    WPA += hostapd.conf
    WPA += hostapd
    WPA += hostapd.deny
    WPA += hostapd.accept
    WPA += hs20-osu-client
    WLAN := qca_cld3_qca6174.ko
    WLAN += qca_cld3_qcn7605.ko
    WLAN += qca_cld3_qca6390.ko
    PRODUCT_PACKAGES += $(WPA)
    PRODUCT_PACKAGES += $(WLAN)
    ```

  - Modify the init.target.rc in Android to add following parameters.

    ```
    insmod /vendor/lib/modules/qca_cld3/qca_cld3_qca6390.ko setprop
    wlan.driver.status "ok"
    ```

  - Modify defconfig on the target platform to add following parameters.

    ```
    CONFIG_CFG80211=y
    CONFIG_CFG80211_INTERNAL_REGDB=y
    CONFIG_IPC_ROUTER=y
    CONFIG_IPC_ROUTER_SECURITY=y
    CONFIG_WCNSS_MEM_PRE_ALLOC=y
    CONFIG_CNSS_CRYPTO=y
    CONFIG_ATH_CARDS=y
    CONFIG_CLD_LL_CORE=y
    CONFIG_CNSS2=y
    ```

```
CONFIG_CNSS2_DEBUG=y
```

- □ Install WLAN firmware in Android.

```
QCA6574:
INI file path: /device/qcom/wlan/MSMnile_au
Copy WCNSS_qcom_cfg_qca6174.ini to /lib/firmware/image/wlan/
Rename /firmware/image/wlan/WCNSS_qcom_cfg_qca6174.ini to
qcom_cfg.ini.
Firmware path:
/wlan_rome/cnss_proc/wlan/fw/target/.out/AR6320/hw.3/bin/
Copy qwlan30.bin to the target platform /lib/firmware/qca6174/.
Copy utf30.bin to the target platform /lib/firmware/qca6174/.
Copy bdwlan30.bin to the target platform /lib/firmware/qca6174/.
Copy otp30.bin to the target platform /lib/firmware/qca6174/.

QCA6595:
INI file path: /device/qcom/wlan/MSMnile_au
Copy WCNSS_qcom_cfg_qcn7605.ini to /firmware/image/wlan/
Rename /firmware/image/wlan/WCNSS_qcom_cfg_qcn7605.ini to
qcom_cfg.ini.
Firmware path:
/wlan_gen/wlan_proc/build/ms/bin/7605.wlanfw.eval_v2_TO_ll/
Copy amss.bin to the target platform /lib/firmware/qcn7605/.
FTM firmware path:
/wlan_gen/wlan_proc/build/ms/bin/7605.wlanfw.eval_v2_TO_ll_ftm/
Copy genoaftm.bin to the target platform /lib/firmware/qcn7605/.
Copy bdwlan.bin to the target platform /lib/firmware/qcn7605/.

QCA6696:
INI file path: /device/qcom/wlan/MSMnile_au
Copy WCNSS_qcom_cfg_qca6390.ini to /firmware/image/wlan/
Rename /firmware/image/wlan/WCNSS_qcom_cfg_qca6390.ini to
qcom_cfg.ini.
Firmware path:
/wlan_hst/wlan_proc/config/bsp/cnss_ram_v2_TO_link_patched/build/6390
.wlan.eval_v2_TO/
Copy amss20.bin to the target platform /lib/firmware/qca6390/.
Fireball firmware path:
/wlan_hst/wlan_proc/wlan/subsys/phyucode_binary/image_hastings/
Copy m3.bin to the target platform /lib/firmware/qca6390/.
Copy bdwlan02.e* to the target platform /lib/firmware/qca6390/.
```

```
*note: All BDF files should be contact Qualcomm HW support to confirm
the version should be used.
```

- Rebuild Android image with WLAN component.

A Terms and Acronyms

Term	Definition
BT	Bluetooth
AOSP	Android Open-Source Project
HIDL	Hardware Interface Definition Language
HCI	Host controller interface
LE	Low Energy
UART	Universal asynchronous receiver/transmitter
VSC	Vendor-specific Command

B Reference

Document	Reference
QCA6595/QCN7605/QCN7606 Bluetooth NVM Configuration Parameters Application Notes	80-WL621-26
QCA639x/QCA6595/QCA6696/QCN760x HCI Vendor Specific Commands	80-WL520-14
BOOTING QCA BLUETOOTH CONTROLLER	80-CG635-1
QCA65x4/QCA6595/QCA6696 Android P Bluetooth Software Training	80-WL520-32