Qualcomm Technologies, Inc.

QCA6174A/QCA9377-3 WLAN and Bluetooth on Linux x86

Porting Guide

80-YC636-1 Rev. C

September 4, 2018

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

Revision history

Revision	Date	Description
A	March 2018	Initial release
B	May 2018	Updated product number (non-technical change) Table 3-1 and Table 6-2, removed interface and Android references
C	September 2018	▪ Chapter 1 Introduction to software package: Added kernel 4.11 information, and adjusted the WLAN configuration file location ▪ Chapter 2 Set up Linux build environment with the kernel: Added kernel 4.11 case ▪ Chapter 3 Get host-based resources through AIO: Updated te-f20 to te-f30, and added kernel 4.11 ▪ Chapter 4 Build the Linux kernel 32 bit: Added kernel 4.11 ▪ Added Chapter 5 Build the Linux kernel 64 bit ▪ Chapter 6 Compile and build WLAN and Bluetooth drivers: Added kernel 4.11 ▪ Section 7.2 Resource location: Adjusted the location of the WLAN configuration files, and corrected typo ▪ Section 7.3 Configure the WLAN features: Added a reference ▪ Section 7.4 Install the WLAN host driver: Updated the installation procedure ▪ Section 7.5 Install the WLAN host driver: Added a reference and updated the FTM guidance ▪ Added Section 8.1.3 SAP-STA single-channel concurrency ▪ Added Section 8.1.4 mBSSID: STA-SAP-SAP SCC

Contents

Figures

Tables

1 Introduction to software package

This document focuses on the exclusive porting effort with All In One (AIO) for QCA6174A/QCA9377-3 chipsets on Linux x86 platforms. Basic knowledge of Linux is required to understand and perform the porting process.

The following hardware is required to complete the porting process:

- Linux x86 Ubuntu build platform
- Linux x86 target platform

The whole porting progress is divided into three steps.

1. Prepare the Linux build environment.
2. Perform the script-based porting procedure using AIO.
3. Bring up the WLAN or Bluetooth on target Linux x86 platforms.

The following table shows which components are needed for various functionalities

Table 1-1 WLAN and Bluetooth components

Component	Porting requirement	Path in AIO release
WLAN firmware	Copy to host file system	fixce/AIO/rootfs-<board_type>.build/lib/module
WLAN configuration files	Copy to host file system	INI_FILES/<interface>/qcom_cfg.ini
WLAN host driver	Patch kernel 4.4.15, 4.9.11, or 4.11 for target arch	fixce/AIO/rootfs-<board_type>.build/lib/module
wpa_supplicant	Must use version 2.7 or later Must compile with CONFIG_INTERWORKING=y CONFIG_WAPI=y CONFIG_FILS=y	fixce/AIO/rootfs-<board_type>.build/sbin/
hostapd	Must use version 2.7 or later Must compile with CONFIG_INTERWORKING=y	fixce/AIO/rootfs-<board_type>.build/sbin/

Download the package from the Qualcomm® Developer Network (QDN), https://developer.Qualcomm.com/. The package consists of the following components:

- Firmware binary code for WLAN and Bluetooth
- AIO, which is a group of scripts used to help build the proper host driver and set of application tools
- Porting guide for x86 Ubuntu platform (this document)

After all the necessary code or scripts and documents are downloaded, follow these steps to start the porting procedure:

1. Set up Linux build environment with kernel 4.4.15, 4.9.11, or 4.11.
2. Get the host-based resources using the AIO.
3. Build the Linux kernel.
4. Compile and build the WLAN and Bluetooth drivers.
5. Load the WLAN and Bluetooth drivers.
6. Enable the WLAN and Bluetooth features.

NOTE: WLAN and Bluetooth are separated from each other. Repeat Step 4 through Step 6 for WLAN and Bluetooth.

2 Set up Linux build environment with kernel 4.4.15, 4.9.11, or 4.11

Prerequisites:

```
$ sudo apt-get install ncurses-dev
$ sudo apt-get install libssl-dev
$ sudo apt-get install libnl-3-dev
$ sudo apt-get install libnl-genl-3-dev
$ sudo apt-get install bison
$ sudo apt-get install flex
$ sudo apt-get install automake libtool dpkg-dev libasound2-dev
```

Use AIO to enable patches to the Linux kernel. The patches enable certain WLAN functionality, such as DFS and FILS.

NOTE: These patches are for use with kernel versions 4.4.15, 4.9.11, and 4.11. The patches might not work if applied to other kernel versions. WLAN has not been tested without using these patches. This document uses Linux kernel versions 4.4.15, 4.9.11, and 4.11as a reference. The AIO also facilitates downloading and patching of the Linux kernel. For more information, see Section 3. To download the kernel manually, instead, perform the following steps:

Download Linux kernel version 4.4.15, 4.9.11, or 4.11:

```
$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-
stable.git
$ cd linux-stable
$ git checkout v4.4.15      /* Or git checkout v4.9.11/v4.11 */
$ git log /* The First commit should be same as below*/
commit 35467dc7630af60abacc330f64029d081f160530
Author: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Date:   Mon Jul 11 09:31:24 2016 -0700

    Linux 4.4.15

Or /* In case of 4.9.11,git log should contain below line */

HEAD is now at eee1550... Linux 4.9.11

Or /* In case of 4.11,git log should contain below line */
HEAD is now at a351e9b... Linux 4.11
```

The AIO script applies the kernel patches.

3 Get host-based resources through AIO

3.1 Prepare the AIO

Before running the AIO, unzip the AIO package.

```
tar zxvf <your_qca_aio_package.tar.gz>
```

NOTE: Ensure the AIO version starts with *TBD*. Type **`./aio_gen_bit.sh -v`** under the folder `fixce/aio-gen` to get the AIO version.

In fixce/aio-gen/build/scripts/board-type/aio_gen.board-type:

- Set **CLONE_KERNEL** =**n**, if you do not want the fixce script to clone the kernel (ignore if the kernel is already cloned).

  Set **CLONE_KERNEL** =**y**, if you want the kernel to be cloned by fixce (ignore if the kernel is already cloned). After cloning, you might like to keep the value as **n** to avoid cloning again.

- Set **CLONE_KERNEL_PATH** to the directory that contains the Linux-stable directory.

  Example: If the Linux-stable is cloned to */home/user/linux-stable*, then

  in *fixce/aio-gen/build/scripts/board-type/aio_gen.board-type*,

  update **CLONE_KERNEL_PATH**= */home/user*

3.2 Run the AIO

Run the AIO script to get all host-side resources from the remote servers. The script gets the given version of the qcacld driver and the relevant patches from the remote open source server to local directories.

```
cd fixce/aio-gen
./aio_gen_bit.sh -r -t <board type> -k <kernel version>
//this takes default values from fixce/aio-gen/build/scripts/<board type>/release.<board type> file
```

OR

NOTE: The option [-r] should not be used in this command ,as [-r] and [-i/-w] are exclusive.

```
./aio_gen_bit.sh –t <board type> –w <wlan driver version> –b <Bluetooth-stack> -i <interface> –k <kernel version>
```

Where:

- -r: use release versions, as defined in the release configuration file (release. <board type>) under

  fixce\aio-gen\build\scripts\[board type], and make sure all options are correct. The [-r] option overrides the remaining options, besides kernel version and board type.

- -t: board type
- -k: kernel version
- -w: WLAN driver version
- -b: Bluetooth stack
- -i: interface

For more information on how to use aio_gen_bit.sh, type **`./aio_gen_bit.sh -h`**.

Table 3-1 AIO script parameters

Parameter	Description	Value
t	Board type option for the script	QCA6174A: re-f30 QCA9377-3: te-f30
k	Kernel version	v4.9.11, v4.4.15, or v4.11
i	Interface type	QCA6174A: SDIO/PCIE QCA9377-3: SDIO
w	WLAN driver version	Refer to fixce/aio-gen/build/scripts/<board - type>/release. <board - type>.
b	Bluetooth stack	FLUORIDE

NOTE:

- The kernel version should be specified to the aio_gen_bit.sh by specifying the **–k** option

  Example: **`-k v4.4.15/v4.9.11/v4.11`**; where *`v4.4.15/v4.9.11/v4.11`* is the git tag on the git.kernel.org Linux-stable git tree

- It is mandatory to provide the **–k** option even if **CLONE_KERNEL** is set to **n**. This is used to properly select the kernel patches directory from the AIO.
- *: Even customers specify the host driver version using the **-w** option, the value is covered by the value in fixce/aio-gen/build/scripts/<board - type>/release. <board - type>.
- If this script is not being run for the first time, check whether values are set as mentioned in following files. If not set, then set as follows:
  - In fixce/aio-gen/build/scripts/board-type/Makefile.board-type

    CLONED_KERNEL_PATH=kernel_dir_path
  - In fixce/aio-gen/build/scripts/board-type/bt.sh

    CLONED_KERNEL_PATH=kernel_dir_path

    IF_TYPE=interface_type

NOTE: QCA6174A/QCA9377-3 supports different WLAN and Bluetooth interfaces. For supported interfaces, refer to specific release notes.

NOTE: Make sure to configure the username and email address in the git configuration file. Otherwise, patching might fail.

NOTE: Ensure the right parameters are used for the specific chipset.

4 Build the Linux kernel 32 bit

Before building the rest of the components with the AIO, configure, build, and install the Linux kernel. Boot the target x86 system in Linux v4.4.15, v4.9.11, or v4.11 kernel mode.

```
$ cd <parent-dir>/linux-stable
$ make menuconfig
   load -> ok -> exit -> save
```

NOTE: Ensure **CONFIG_NL80211_TESTMODE** in the *.config* file is set as **CONFIG_NL80211_TESTMODE=y**. If any changes are made in the *.config* file, redo **`make menuconfig`**.

```
$ make
        Use “make –j8” to speed up this process
$ make modules
$ sudo make modules_install # requires super user priviliges
$ sudo make headers_install # requires super user priviliges
$ sudo make install # requires super user priviliges
$ sudo reboot

Select 4.4.15+/4.9.11+/4.11.0+ in Grub menu on startup.
```

NOTE: The **`menuconfig`** command helps to configure the Linux kernel that is built.

The command loads, by default, taking the existing configuration from the Linux system that is running. For example, if you are running an Ubuntu x86, Linux 32 bit system, the `menuconfig` command loads the configuration of this system as the default configuration and displays a text-based GUI with options to configure the kernel. Select each option by a `*` or an `M` or unselect by leaving the selection blank.

Suggested options are:

- MMC – MMC/SD/USB card support
- MMC_DEBUG – MMC debugging
- CONFIG_CFG80211_INTERNAL_REGDB=y – statically compiled regulatory rules data base
- CONFIG_CFG80211=m – wireless configuration APIs

Useful links on Linux kernel build:

https://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/scripts/README.Menuconfig

http://linuxwireless.org/en/developers/Regulatory/

5 Build the Linux kernel 64 bit

To create the Linux kernel 64-bit build, complete the build steps and kernel configuration described in Section 4, along with the configuration parameters in the kernel/arch/x86/configs/x86_64_defconfig file.

CONFIG_FRAME_WARN=2048

Perform this build on 64-bit Ubuntu.

6 Compile and build WLAN and Bluetooth drivers

After running the AIO script, all the required resources are located at the proper locations locally. You see the directory *AIO* aligned with *aio-gen*.

Continue only after compiling and loading the 4.4.15, 4.9.11, or 4.11 kernel.

6.1 Apply patches and build

The qcacld driver cannot be used directly because it is originally targeted onto a Linux kernel version 4.4.15, 4.9.11, or 4.11 x86 reference platform. Qualcomm Technologies, Inc., has supplied a series of patches for the x86 Ubuntu system.

```
cd AIO/build
make
```

NOTE: These steps build the WLAN host driver along with wpa_supplicant, hostapd, wpa_cli, and sigma_dut. Then Bluetooth-related patches are applied.

6.2 Install kernel headers

Execute the following commands for Fluoride stack compilation. This step must be executed after the system is rebooted with the new kernel (for example, 4.11, 4.9, or 4.4).

```
$ cd  <kernel_path>/linux-stable
$ sudo  make headers_install ARCH=i386 INSTALL_HDR_PATH=/usr
```

6.3 Build Fluoride and other components

The following commands must be executed to build Fluoride and other Bluetooth components.

```
cd AIO/build
chmod 777 ./scripts/board-type/bt.sh
sudo ./scripts/board-type/bt.sh
```

NOTE: Copy the nvm and rampatch files to */lib/firmware/ar3k* for firmware patch download. See the specific release notes to select the correct nvm and rampatch files.

7 Load WLAN and Bluetooth drivers

After performing the actions in Section 4 and Section 6, the WLAN resources, including host-driver binary files, are ready. You can find them in their respective directories. Later, they should be placed in the right location on the target x86 Ubuntu platform.

7.1 WLAN files description

WLAN-relevant files are composed of target firmware, the host driver, and configuration and applications or tools. Place the files in the proper folders on the target x86 platform.

NOTE: QCA6174A/QCA9377-3 supports different WLAN files. For details, see the specific release notes.

7.2 Resource location

Copy all the resources from the corresponding *source* directory to the proper *target* directories on the target x86 platform.

Table 7-1 WLAN source location and target position

Item	Type	Location	
1	Firmware	Source	firmware/wlan_firmware/<interface>/
		Target	/lib/firmware
2	Host driver	Source	fixce/AIO/rootfs-<board_type>.build/lib/module
		Target	/lib/firmware
3	Configuration files	Source	INI_FILES/<interface>/
		Target	/lib/firmware/wlan
4	Application: wpa_supplicant/hostapd/wpa_cli/hostapd_cli	SOURCE	fixce/AIO/rootfs-<board_type>.build/sbin
		TARGET	/usr/sbin

Specifically, when placing the configuration files on the target x86 platform, follow the naming rules in the following table.

Table 7-2 Configuration file path on the QDN

Item	Product	Path	Name
1	QCA9377-3	fixce/INI_FILES/SDIO	qcom_cfg.ini (for SDIO)
2.	QCA6174A	fixce/INI_FILES/SDIO fixce/INI_FILES/PCIE	qcom_cfg.ini (for SDIO/PCIE)

7.3 Configure the WLAN features

By default, some of the chipset features are not enabled. Modify **qcom_cfg.ini**, as required, to enable features before installing the WLAN host driver. For details, see *QCA6174A/QCA9377/QCA9379.LEA Configuration Parameters Specification* (80-YC321-11).

7.4 Install the WLAN host driver

Turn off the internal Wi-Fi from the BIOS. Run the following commands to the load host driver on the target x86 platform. While installing wlan.ko, the configuration files are parsed and firmware files are downloaded into the target chip with the proper configuration.

1. **rmmod cfg80211 # if already installed**
2. Remove dependent modules, if any.

   For example:
   mac80211 iwlwifi iwldvm ath10k_pci ath10k_core
3. Add the following to the `/etc/modprobe.d/blacklist.conf` file:
   ```
   blacklist ath10k_pci
   blacklist ath10k_core
   blacklist ath
   blacklist mac80211
   blacklist wl
   ```
4. **modprobe cfg80211**
5. **insmod fixce/AIO/rootfs-<board type>.build/lib/modules/wlan.ko**

After installing the WLAN host driver, you will see two WLAN interfaces installed correctly.

```
#iwconfig

Wlan0     Qcom:802.11n  ESSID:off/any  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power=0 dBm
          RTS thr=192000 B   Fragment thr=8000 B

p2p0      Qcom:802.11n  ESSID:off/any  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power=0 dBm
          RTS thr=192000 B   Fragment thr=8000 B
```

Figure 7-1 Iwconfig output after installing the WLAN driver

7.5 Install the WLAN host driver alternate modes

The WLAN host driver can be loaded into factory test mode (FTM), or monitor mode, depending on the con_mode argument given at the time of insmod. For details, see *QDART Connectivity for QCA61xx, QCA65xx, and QCA93xx User Guide* (80-WL400-32).

Load the WLAN host driver in FTM:

insmod fixce/AIO/rootfs-<board type>.build/lib/modules/wlan.ko con_mode=5

Load the WLAN host driver in monitor mode, and set the monitor parameters:

```
insmod fixce/AIO/rootfs-<board type>.build/lib/modules/wlan.ko con_mode=4
iwpriv wlan0 setMonChan <channel> <channel width>
# Valid channel width options: 0=20MHz, 1=40MHz, 2=80MHz
# Ex: iwpriv wlan0 setMonChan 36 2

iwpriv wlanx setMonFilter <filter option>
# Valid Frame filter options: 0=Management Frame, 1=Control Frame, 2=Data
Frame 3=All Frame type. Default value 3
```

7.6 Load Bluetooth

7.6.1 UART interface

No special driver is required for the UART interface. Uses a standard tty interface. If using a UART-to-USB converter, after connecting the cable, run **dmesg**, ensure that the device is detected, and ensure that a /dev/ttyUSBn device is available.

8 Enable WLAN and Bluetooth features

8.1 Enable the WLAN features

8.1.1 Static timing analysis mode

Connect to the open mode AP

Use iw, iwlist, ifconfig, and iwconfig commands to operate the WLAN.

1. **ifconfig wlan0 up**
2. **ifconfig wlan0 192.168.1.2**
3. **iwlist wlan0 scan |grep <ESSID>**
4. **iwconfig wlan0 essid <ESSID>**
5. **ping 192.168.1.1**

Connect to the WPA-PSK mode AP

1. Set up the AP in WPA-PSK mode with a *ref-AP* SSID and IP as 192.168.1.1 with DHCP server enable.
2. Edit the configuration file wpa-sta.conf content from the console.

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network={
ssid="ref-AP"
proto=WPA
key_mgmt=WPA-PSK
auth_alg=OPEN
pairwise=CCMP
group=CCMP
psk="1234567890"
}
```

3. Enter the following command from the console to connect wlanX with the AP:
   wpa_supplicant –Dnl80211 -iwlanX -c wpa-sta.conf &

4. Enter the following command from the console, after wlan0 is connected with the AP, to set the IP address:
   ifconfig wlan0 192.168.1.2
5. Check the connectivity by pinging the AP IP address.
   ping 192.168.1.1

8.1.2 Soft AP mode

Set up the 11ng soft AP (SAP) with open mode

1. Edit the default configuration file `hostapd.conf` and save it on `/home/root/sbin` with hostapd together.
   interface=wlanX
   ssid=ref-AP
   hw_mode=g
   channel=1
   auth_algs=1
   ieee80211n=1
2. Install the udhcp server on the x86, and configure the DHCP server to assign a client IP address.
   a. **sudo apt-get install udhcpd**.
   b. Edit `/etc/udhcpd.conf`, set a range of IP addresses and interfaces, such as:
      Start 192.168.11.20
      End 192.168.11.254
      Interface wlanX
   c. Enable the DHCP server (see `/etc/default/udhcpd`).
   d. DHCPD_ENABLED="yes"
3. Stop the network service.
   service network-manager stop
4. Set up the SoftAP IP address.
   ifconfig wlanX 192.168.11.1 netmask 255.255.255.0
5. Run the DHCP server service.
   sercice udhcpd start
6. Run the hostapd with `hostapd.conf`.
   hostapd -dd hostapd.conf &
7. Set up the static timing analysis (STA) to connect to this SoftAP with open mode, and then use a ping to verify the connection.
   ping 192.168.11.1

Set up the 11ng SAP with WPA1/2 PSK mode

1. Edit the default configuration file `hostapd.conf`.

   i**nterface=wlanX**

   ssid=ref-AP

   hw_mode=g

   channel=1

   auth_algs=3

   ieee80211n=1

   wpa=3

   wpa_passphrase=12345678

   wpa_key_mgmt=WPA-PSK

   wpa_pairwise=CCMP

   rsn_pairwise=CCMP

2. Execute SAP open mode Step 2 through Step 4 in Section 7.1.2).

3. Start the hostapd with `hostapd.conf`.

   hostapd –dd hostapd.conf &

4. Set up the SAP IP address.

   Ifconfig wlanX 192.168.11.1

5. Set up the STA to connect to this SoftAP with WPA1/2 PSK mode, and then use a ping to verify the connection.

   ping 192.168.11.1

8.1.3 SAP-STA single-channel concurrency

This feature is used to force the SAP to follow the STA operating channel to get good performance (that is, the SAP and the STA run on the same channel). This functionality is called *single-channel concurrency* (SCC).

Enable SAP-STA concurrency by setting the ini parameter, gWlanMccToSccSwitchMode, to 2.

1. Load the wlan drivers.

   #insmod wlan.ko

2. Set up the SAP interface.

   #iw dev wlan0 interface add ap0 type __ap

   #ifconfig ap0 192.168.10.1 up

3. Configure IP address for STA wlan0.

   #ifconfig wlan0 192.168.2.102 netmask 255.255.255.0 up

4. Bring up the SAP-STA interface using hostapd and wpa_supplicant, respectively.

   There is no need to specify the channel in the sap.conf configuration file. The channel follows the STA channel.

   #hostapd sap.conf

```
nk@nk: hostapd:$sudo ./hostapd hostapd.conf
Configuration file: hostapd.conf
Using interface ap0 with hwaddr 02:03:7f:b8:12:11 and ssid "test"
ap0: interface state UNINITIALIZED->ENABLED
ap0: AP-ENABLED
ap0: IEEE 802.11 driver had channel switch: freq=5805, ht=0, offset=
0, width=0 (20 MHz (no HT)), cf1=5805, cf2=0
ap0: AP-CSA-FINISHED freq=5805 dfs=0
```

Figure 8-1 Start hostapd on channel 161

```
#wpa_supplicant –Dnl80211 –i wlan0 –c sta.conf
```

5. Connect the STA to a third-party device operating on channel 36.

   The SAP switches automatically to the STA channel.

   ```
   wpa_cli –i wlan0 enable_network 0
   ```

```
nk@nk: wpa_supplicant:$sudo ./wpa_supplicant -i wlan1 -c supp.conf
Successfully initialized wpa_supplicant
wlan1: Trying to associate with SSID '12345'
wlan1: Associated with 84:1b:5e:e9:41:e1
wlan1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan1: WPA: Key negotiation completed with 84:1b:5e:e9:41:e1 [PTK=CC
MP GTK=CCMP]
wlan1: CTRL-EVENT-CONNECTED - Connection to 84:1b:5e:e9:41:e1 comple
ted [id=0 id_str=]
```

Figure 8-2 Start wpa_supplicant

Connection on the STA interface triggers the channel change on the SAP interface.

```
nk@nk: hostapd:$sudo ./hostapd hostapd.conf
Configuration file: hostapd.conf
Using interface ap0 with hwaddr 02:03:7f:b8:12:11 and ssid "test"
ap0: interface state UNINITIALIZED->ENABLED
ap0: AP-ENABLED
ap0: IEEE 802.11 driver had channel switch: freq=5805, ht=0, offset=
0, width=0 (20 MHz (no HT)), cf1=5805, cf2=0
ap0: AP-CSA-FINISHED freq=5805 dfs=0
ap0: IEEE 802.11 driver had channel switch: freq=5180, ht=0, offset=
0, width=0 (20 MHz (no HT)), cf1=5180, cf2=0
ap0: AP-CSA-FINISHED freq=5180 dfs=0
```

Figure 8-3 Channel change in hostapd for enforcing SCC

8.1.4 mBSSID: STA-SAP-SAP SCC

Multiple base station subsystem identification (mBSSID) is a feature that enables the creation of multiple, virtual access points using a single, physical radio. The mBSSID allows configuration of multiple infrastructure networks. Each network has a different BSSID and different access characteristics, such as security mode and access control configurations. This design allows different security mechanisms for different clients on the same access point.

1. Add the following ini parameters to the qcom_cfg.ini file:

   gWlanApP2pGOConcurrencyEnable=1

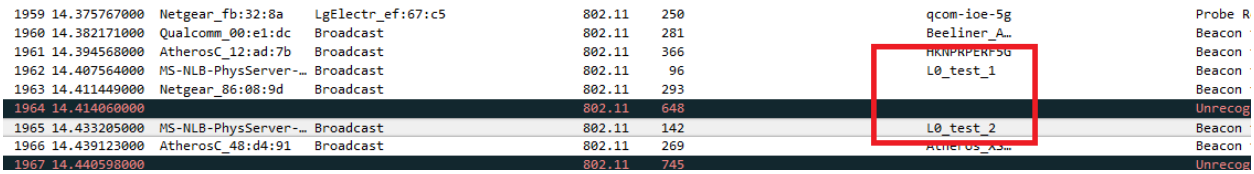
   gMaxConcurrentActiveSessions=3

2. Disable the network manager.
   #sudo service network-manager stop

3. Load the wlan drivers.
   #insmod wlan.ko

   #ifconfig p2p0 up

4. Run the hostapd using the following example softap.conf file and configuration:
   #hostapd -dd sap.conf
   <sap.conf>
   ctrl_interface=/var/run/hostapd
   use_driver_iface_addr=1
   interface=p2p0
   ssid=L0_test_1
   channel=36
   hw_mode=a
   bss=softap0
   ssid=L0_test_2
   wpa=3
   wpa_key_mgmt=WPA-PSK
   rsn_pairwise=CCMP
   wpa_pairwise=TKIP
   wpa_passphrase=password



NOTE: If hostapd fails on a laptop, try the following command to enable Wi-Fi and Bluetooth functionality on the laptop:

#sudo rfkill unblock all

8.2 Enable Bluetooth features

NOTE: The user cannot run multiple instances of the btapp. The user can run only one instance.

NOTE: The user cannot use the **kill -9 <btapp pid>.** Instead, the user can use **kill -2 <btapp pid>**, or CTRL+C.

8.2.1 Test Bluetooth

Open two terminals. Run following commands in one terminal:

```
$ cd <bt_workspace>/qcom-opensource/bt/property-ops
$ sudo ./btproperty
```

Run the following commands on the other terminal:

```
$ cd <bt_workspace>/qcom-opensource/bt/bt-app
$ sudo ./main/btapp
```

By this time, stack and other libraries are loaded.

Press **Enter** and go to gap_menu > enable. This enables the Bluetooth, and you see the following image.

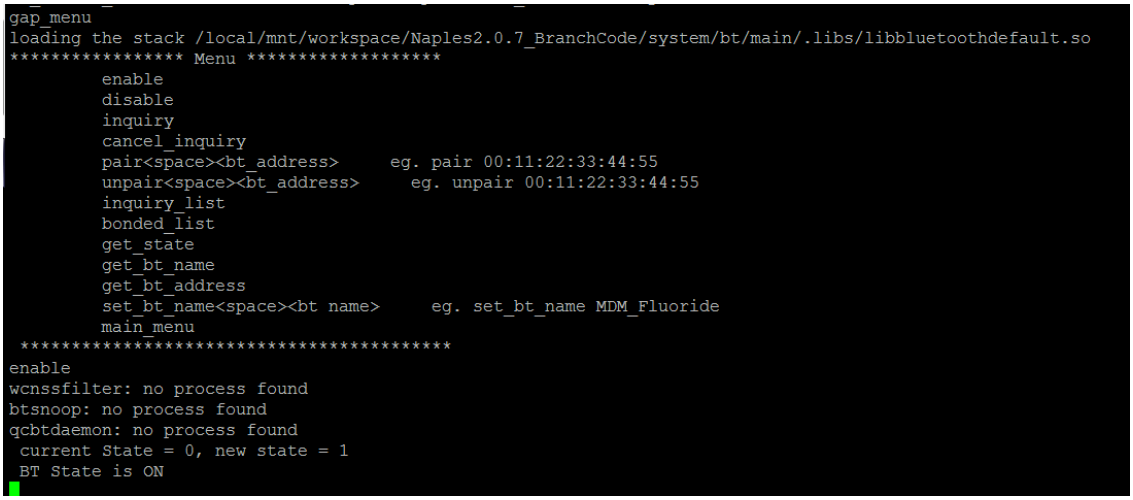


Figure 8-4 Sample output of the Bluetooth gap menu

8.2.2 Change the local Bluetooth device name

```
$sudo gedit /etc/bluetooth/bt_app.conf
```

From the file, change *`BtLocalDeviceName=WhatEverULike`*.

8.2.3 A2DP sink

1. Go to gap_menu > enable.
2. Connect to the device under test DUT using a remote device (for example, from a phone).
3. Do one of the following:
   - If the passphrase that appears on the screen matches the passphrase on the phone, type **`yes`** and press **Enter**.

     The on-screen message `A2DP Sink Connected to <remote_bd_address>` indicates a successful connection.
   - If the passphrase that appears on the screen does not match the passphrase on the phone, type **no** and press **Enter**, return to Step 2.

Use the start, stop, forward, backward, next, previous, volume change, and disconnect functions from the remote device only.

NOTE:

- Either A2DP sink or A2DP source is supported at any time.
- Only one device is supported at any time (for example, max a2dp connection is 1.)
- Currently, only the SBC codec is supported.
- Ensure the following settings in */etc/bluetooth/bt_app.conf*

  BtA2dpSourceEnable=false

  BtA2dpSinkEnable=true

  BtAvrcpEnable=true

8.2.4 A2DP source

1. Go to gap_menu > enable.
2. Connect to the DUT using a remote device (for example, from a phone).
3. Do one of the following:
   - If the passphrase that appears on the screen matches the passphrase on the phone, type **`yes`** and press **Enter**.

     The on-screen message `A2DP Source Connected to <remote_bd_address>` indicates a successful connection.
   - If the passphrase that appears on the screen does not match the passphrase on the phone, type **`no`** and press **Enter**, return to Step 2.

OR

1. Go to gap_menu > enable.
2. pair<space><bt_address of the sink device> (for example: `pair 00:11:22:33:44:55`).
3. Do one of the following:
   - If the passphrase that appears on the screen matches the passphrase on the sink device, type **`yes`** and press **Enter**.

     The on-screen message `Pairing state for <Device name> is BONDED` indicates a successful connection.
   - If the passphrase that appears on the screen does not match the passphrase on the sink device, type **`no`**, press **Enter**, and return to Step 2.
4. Go to main_menu > a2dp_source_menu.
5. Perform a profile-level connection by using command **`connect<space><bt_address>`** (for example: **`connect 00:11:22:33:44:55`**).

   The on-screen message `A2DP Source Connected to <remote_bd_address>` indicates a successful connection.

NOTE:

- Currently, the player is not integrated into the A2DP source. Only one file can be played. Before starting the source, ensure that the file named pcm.wav is copied to the */usr/local* folder. This file should be 16 bit PCM data and 44.1 kHz. You can generate this using tools like audacity.
- Either A2DP sink or A2DP source is supported at any time.
- Only one device is supported at any time (for example, max a2dp connection is 1).
- Currently, only the SBC codec is supported.
- a2dp_source_menu currently supports connect, disconnect, start, stop, suspend, volume up, and volume down.
- Ensure the following settings in */etc/bluetooth/bt_app.conf*:

  BtA2dpSourceEnable=true

  BtA2dpSinkEnable=false

  BtAvrcpEnable=false

8.2.5 HID

1. Go to gap_menu > enable.
2. Start inquiry, cancel_inquiry, list through inquiry_list.
3. Pair <bd_add> from gap_menu.
4. The passkey shown on the screen must be entered on the keyboard (not a mouse).
5. View this device in bonded_list, at this point.
6. Go to main_menu > hid_menu.
7. Connect <bd_addr> for profile-level HID connection.

OR

1. Go to gap_menu > enable.
2. Start inquiry, cancel_inquiry, list through inquiry_list.
3. Go to main_menu > hid_menu.
4. Connect <bd_addr> for profile-level HID connection.
5. The passkey shown on the screen must be entered on the keyboard (not a mouse).

   After connecting, this can be seen in hid_list from gap_menu.

8.2.6 HOGP

1. Go to gap_menu > enable.
2. Start inquiry, cancel_inquiry, list through inquiry_list.
3. Go to main_menu > hid_menu.
4. Use connect <bd_addr> from hid_menu.

5. The passkey shown on the screen must be entered on the keyboard (not a mouse).
6. After connecting, this can be seen in hid_list from gap_menu.

NOTE: It cannot be paired from the pair of gap_menu because LE is over GATT.

8.2.7 SPP

Enable SPP (client and server) by using the following settings in */etc/bluetooth/bt_app.conf*:

BtSppServerEnable=true

BtSppClientEnable=true

8.2.7.1 SPP server

Steps to receive a file

1. Go to gap_menu > enable.
2. Go to main_menu > spp_server_menu.
3. Start (this starts the SPP server, and SDP registration happens only after this).
4. Use the Bluetooth SPP Pro application (the SPP client application from the phone) to discover the DUT.
5. Initiate pairing from the Bluetooth SPP Pro application.
6. After pairing is complete, initiate a connection from the Bluetooth SPP Pro application.
7. After the connection is established, you see `Select communication mode` in the Bluetooth SPP Pro application.
8. On the DUT side, go to main_menu > spp_server_menu.
9. Use option *receive <filename>* to receive the file.

   Example: Receive /home/administrator/test_sppserver_recv.txt
10. From the Bluetooth SPP Pro application, select byte stream mode.
11. Send data from the Bluetooth SPP Pro application (use the > button).
12. Exit the Bluetooth SPP Pro application (disconnects the connection)

    You can also disconnect from the DUT side using the disconnect option.
13. Go back to the main_menu.

NOTE: If the remote device (phone) is already paired, initiate a scan from the Bluetooth SPP Pro application, and select the DUT from the list of discovered devices. Use the connect option to connect to the DUT, then follow the procedure beginning with Step 7.

Steps to send a file

1. Go to gap_menu > enable.
2. Go to main_menu > spp_server_menu.

3. Start (this starts the SPP server, and SDP registration happens only after this).
4. Use the Bluetooth SPP Pro application (the SPP client application from the phone) to discover the DUT.
5. Initiate pairing from the remote device using the Bluetooth SPP Pro application.
6. After pairing is complete, initiate a connection from the Bluetooth SPP Pro application.
7. After the connection is established, you see `Select communication mode` in the Bluetooth SPP Pro application.
8. From the Bluetooth SPP Pro application, select byte stream mode.
9. On the DUT side, go to main_menu > spp_server_menu.
10. Use option *send <filename>* to send the file.

    Example: Send /home/administrator/test_spps_send.txt
11. Verify that the data is received in the Bluetooth SPP Pro application.
12. Exit the Bluetooth SPP Pro application (disconnects the connection).

    You can also disconnect from the DUT side using the disconnect option.
13. Go back to the main_menu.

NOTE: If the remote device (phone) is already paired, initiate a scan from the Bluetooth SPP Pro application, and select the DUT from the list of discovered devices. Use the connect option to connect to the DUT, then follow the procedure beginning with Step 7.

8.2.7.2 SPP client

Steps to receive a file

1. Go to gap_menu > enable.
2. Switch on Bluetooth on the phone side and ensure it is discoverable.
3. Start inquiry from the DUT.
4. Discover and pair the remote phone from the DUT.
5. Start the BlueSPP application (the SPP server application on the phone).
6. On the DUT, go to main_menu > spp_client_menu.
7. From the DUT, initiate a connection with *`connect <BD Addr>`*.

    Example: `connect bc:f5:ac:9c:0d:e6`

    NOTE: If you have previously tested using the BlueSPP application, it loads the history (send and receive data) unless you disable the default option.
8. From the DUT, use the command **`receive <file>`**.

    Example: **`receive /home/administrator/test_sppclient_recv.txt`**
9. From the phone, use the BlueSPP application to send data to the DUT.

10. Exit the BlueSPP application (disconnects the connection).

    You can also disconnect from the DUT side using the disconnect option.

11. Go back to the main_menu.

NOTE: If the remote device (phone) is already paired, begin this procedure from Step 8.

Steps to send a file

1. Go to gap_menu > enable.
2. Switch on Bluetooth on the phone side, and ensure it is discoverable.
3. Start inquiry from the DUT.
4. Discover and pair the remote phone from the DUT.
5. Start the BlueSPP application (the SPP server application in the phone).
6. On the DUT, go to main_menu > spp_client_menu.
7. From the DUT, initiate a connection with *`connect <BD Addr>`*.

   Example: `connect bc:f5:ac:9c:0d:e6`

   NOTE: If you have previously tested using the BlueSPP application, it loads the history (send and receive data) unless you disable the default option.

8. From the DUT, use the command **`send <file>`**.

   Example: **`send /home/administrator/test_sppclient_send.txt`**

9. Verify the data in the BlueSPP application on the phone.
10. Exit the BlueSPP application (disconnect the connection).

    You can also disconnect from the DUT side using the disconnect option.

11. Go back to the main_menu.

NOTE: If the remote device (phone) is already paired, begin this procedure from Step 8.