Intel® Edison

Wi-Fi Guide

February 2015

Revision 001

Document Number: 331438-001

Contents

Figures

Tables

Revision History

Revision	Description	Date
001	Initial release.	February 4, 2015

§

1 Introduction

This document is for software developers providing Wi-Fi guidelines on the Intel® Edison development platform. The reader should have a basic knowledge of the Linux operating system and Wi-Fi connectivity.

1.1 References

Reference	Name	Number/location
331188	Intel® Edison Board Support Package User Guide	
331189	Intel® Edison Compute Module Hardware Guide	
331190	Intel® Edison Breakout Board Hardware Guide	
331191	Intel® Edison Kit for Arduino* Hardware Guide	
329686	Intel® Galileo and Intel® Edison Release Notes	
[GSG]	Intel® Edison Getting Started Guide	W: *http://www.intel.com/support/edison/sb/CS-035336.htm* M: *http://www.intel.com/support/edison/sb/CS-035344.htm* L: *http://www.intel.com/support/edison/sb/CS-035335.htm*
331438	Intel® Edison Wi-Fi Guide	(This document)
331704	Intel® Edison Bluetooth* Guide	
[YPQSG]	Yocto Project Quick Start Guide	*http://www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html*
[YDM]	Yocto Developer Manual	*http://www.yoctoproject.org/docs/current/dev-manual/dev-manual.html*
[YKDM]	Yocto Kernel Developer Manual	*http://www.yoctoproject.org/docs/latest/kernel-dev/kernel-dev.html*
	Yocto Project	*http://www.yoctoproject.org/docs/1.5.1/dev-manual/dev-manual.html*
	hostapd	*http://w1.fi/hostapd*
	Linux WPA/WPA2/IEEE 802.1X Supplicant	*http://w1.fi/wpa_supplicant*
	wpa_cli	*http://linux.die.net/man/8/wpa_cli*
	iwconfig	*http://www.linuxcommand.org/man_pages/iwconfig8.html*

1.2 Acronyms and abbreviations

Table 1. Terminology

Term	Definition
AES	Advanced Encryption Standard
AP	Access point
DHCP	Dynamic Host Configuration Protocol
DUT	Device under test
GO	Group owner
IBSS	Independent basic service set
MAC	Media Access Control
P2P	Peer to peer
PBC	Push button control
PIN	Personal identification number
PSK	Preshared key
SoftAP	Software-enabled access point
ssh	Secure shell
SSID	Service set identifier
STA	Station
UART	Universal asynchronous receiver/transmitter
WEP	Wired equivalent privacy
WPA	Wi-Fi protected access
WPA_CLI	WPA command line client
WPS	Wi-Fi protected setup

2 Software Architecture

The Intel ® Edison development board includes the BCM43340 —a combination Bluetooth/Wi-Fi chip. This connectivity module enables system-level connectivity features. This chapter includes an overview of the overall Wi-Fi stack and highlights the components that provide connectivity services to applications.

2.1 Features

Wi-Fi feature list

- 802.11 a/b/g/n support.
- WEP encryption.
- WPA encryption.
- WPA2 encryption.
- STA mode.
- AP mode.
- Wakeup on Wi-Fi.
- Multirole
- Wi-Fi Direct/P2P

Connection Manager

There are two user interfaces available to control network services:

- connman.
- wpa_cli.

Firmware and drivers

- Broadcom provides the open source Linux kernel drivers. (The firmware and drivers are included in our tar download.)

2.2 Services

Refer to Table 2 to determine whether *wpa_cli* or *connman* is the correct tool for your application. Both of these are capable of managing Wi-Fi connections, but each has advantages and disadvantages. Your choice may depend on your Wi-Fi and WLAN expertise and on how you intend to integrate WLAN into your middleware.

Table 2 Pros and cons of connman and wpa_cli

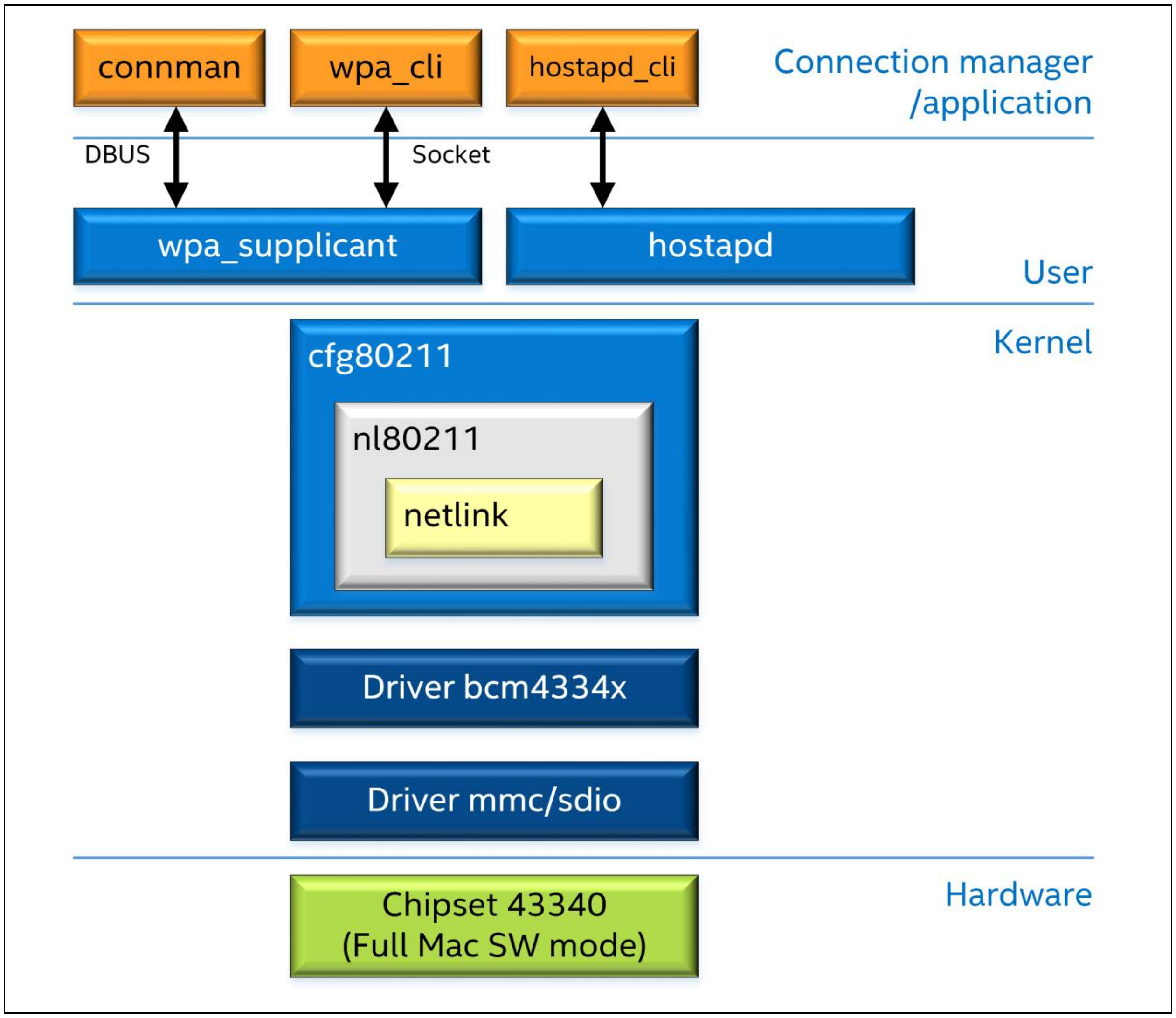
	connman	wpa_cli
Pros	• Supported on Intel® Galileo platform. • Easy to use. • Exports simple DBUS API for middleware. • Simple CLI commands. (You do not need to be a Wi-Fi expert.)	• Great support from the open source community. • Allows complex configurations. • Supports concurrent P2P/STA configurations.
Cons	• P2P connection not yet supported. • Not all EAP mode supported. • Concurrent P2P/STA configuration not supported.	• CLI commands are not as simple. • Integration within middleware is not as easy. • Requires DBUS API (a more complex variant of *wpa_supplicant*).

Once you have decided on the tool, download it:

- *wpa_cli* command line client: *http://linux.die.net/man/8/wpa_cli*
- *connman* command line: *https://wiki.archlinux.org/index.php/Connman*

Figure 1 shows the basic software architecture.

Figure 1 Software architecture



Note the following:

- *hostapd* is a user space daemon for access point and authentication server, which is used to create a virtual Wi-Fi access point.
- *wpa_supplicant* is a user space daemon that runs in the background and acts as the backend component controlling the wireless connection.
- Broadcom does not support STA/AP concurrent mode. So *hostapd* and *wpa_supplicant* cannot run simultaneously.
- In STA mode, you can use either *connman* or *wpa_cli* to manage the STA connection.
- In P2P mode, only *wpa_cli* is supported to manage the P2P connection.

§

3 First Steps

Before you attempt to connect your Intel® Edison device to a Wi-Fi network, make sure you complete these steps.

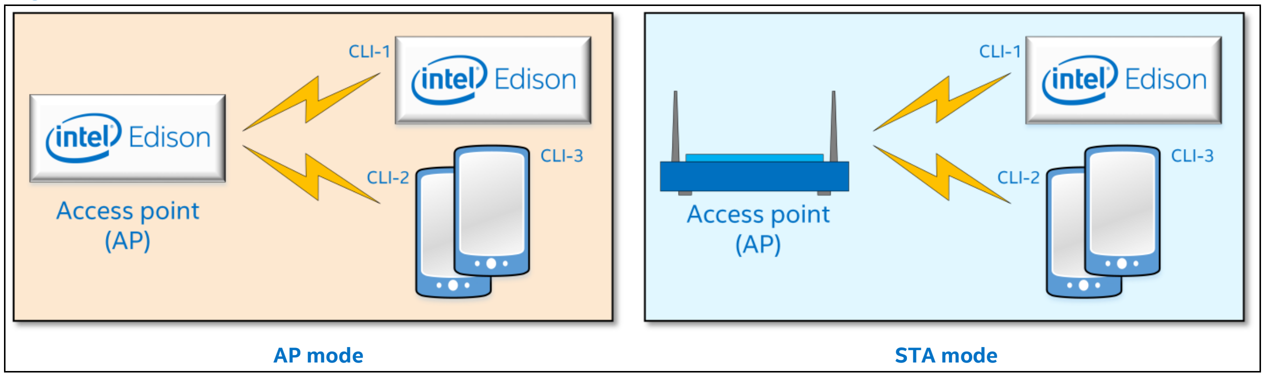
1. If you are using your Intel® Edison compute module for the first time, follow the OS-specific guidelines in the *How-tos* section of the Intel® Edison webpage: *http://www.intel.com/support/maker/edison.htm#how*.
2. Make sure that your device has the latest image. Visit the *Support and downloads* page to download the latest firmware: *http://www.intel.com/support/edison/sb/CS-035180.htm*.

3.1 Different ways to connect

After you complete the preparatory steps as described above, you can configure your Intel® Edison devices in either of the following operating modes:

- **Access point (AP) mode.** In this mode, an Intel® Edison device can act as a master to enable multiple Wi-Fi adapters (CLIs) to connect to it.
- **Station (STA) mode.** This is the default mode for any wireless driver. In STA mode, an Intel® Edison device can act as a client to connect with a Wi-Fi access point.

Figure 2 AP and STA modes



3.1.1 Serial connection

To connect an Intel® Edison device via serial console (UART), from a terminal enter the following:

```
$ sudo screen /dev/ttyUSB0 115200
```

You may need to enter your password. At the blank screen, press Enter twice, then log on as `root`. If you do not have "screen" installed, install it now by opening a new terminal window and entering `sudo apt-get install screen`.

3.1.2 SSH connection

To connect an Intel® Edison device via SSH (Ethernet over USB), from a terminal enter the following:

```
$ ssh root@192.168.2.15
```

Note: 192.168.2.15 is the Intel® Edison board's static IP address.

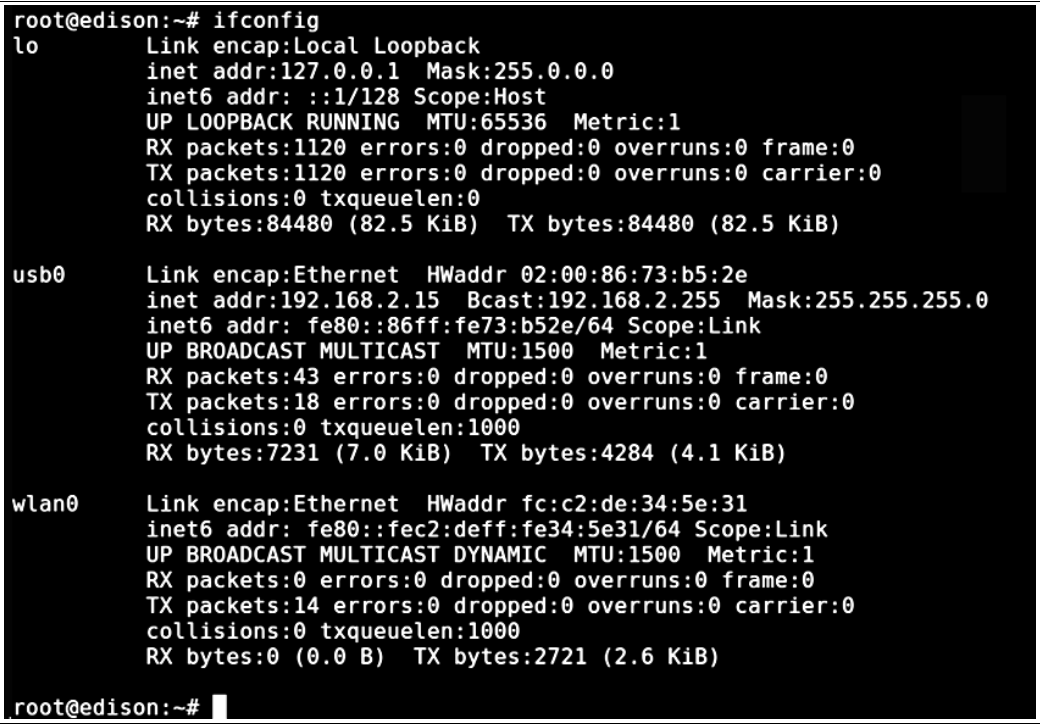
3.1.3 Manually powering Wi-Fi up or down

If the *ifconfig* command lists the *wlan0* interface in its results, Wi-Fi is powered up; if it does not list *wlan0,* Wi-Fi is powered down. You can power up or down the chipset with these commands:

```
$ ifconfig wlan0 up
$ ifconfig wlan0 down
```

From a terminal, use the *ifconfig* command to check if the network interface is configured (Figure 3).

Figure 3 ifconfig after Wi-Fi device has powered up



```
root@edison:~# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1120 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1120 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:84480 (82.5 KiB)  TX bytes:84480 (82.5 KiB)

usb0      Link encap:Ethernet  HWaddr 02:00:86:73:b5:2e
          inet addr:192.168.2.15  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::86ff:fe73:b52e/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:43 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7231 (7.0 KiB)  TX bytes:4284 (4.1 KiB)

wlan0     Link encap:Ethernet  HWaddr fc:c2:de:34:5e:31
          inet6 addr: fe80::fec2:deff:fe34:5e31/64 Scope:Link
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2721 (2.6 KiB)

root@edison:~# 
```

Note: The *rfkill unblock/block wlan* command is generally used to power up/down the WLAN.

§

4 Setting up Wi-Fi connections

This chapter shows how to set up Wi-Fi connections with standard command-line tools like *connman* and *wpa_cli*, which allow you to configure and connect to a Wi-Fi network.

By default, the Wi-Fi driver is loaded in the down state at boot time, so Wi-Fi is completely powered down. You will have to start *wpa_supplicant* or *hostapd* at boot time, depending on whichever one you decide to use. Intel provides the *configure_edison –wifi* script, which will start either one for you.

- *systemctl*: This is used to introspect and control the state of the *systemd* system and service manager. (Refer to the system man page for details.)
- *wpa_cli*: This text-based frontend program for interacting with *wpa_supplicant* is used to query current status, change configuration, trigger events, and request interactive user input. (Refer to the system man page for details.)

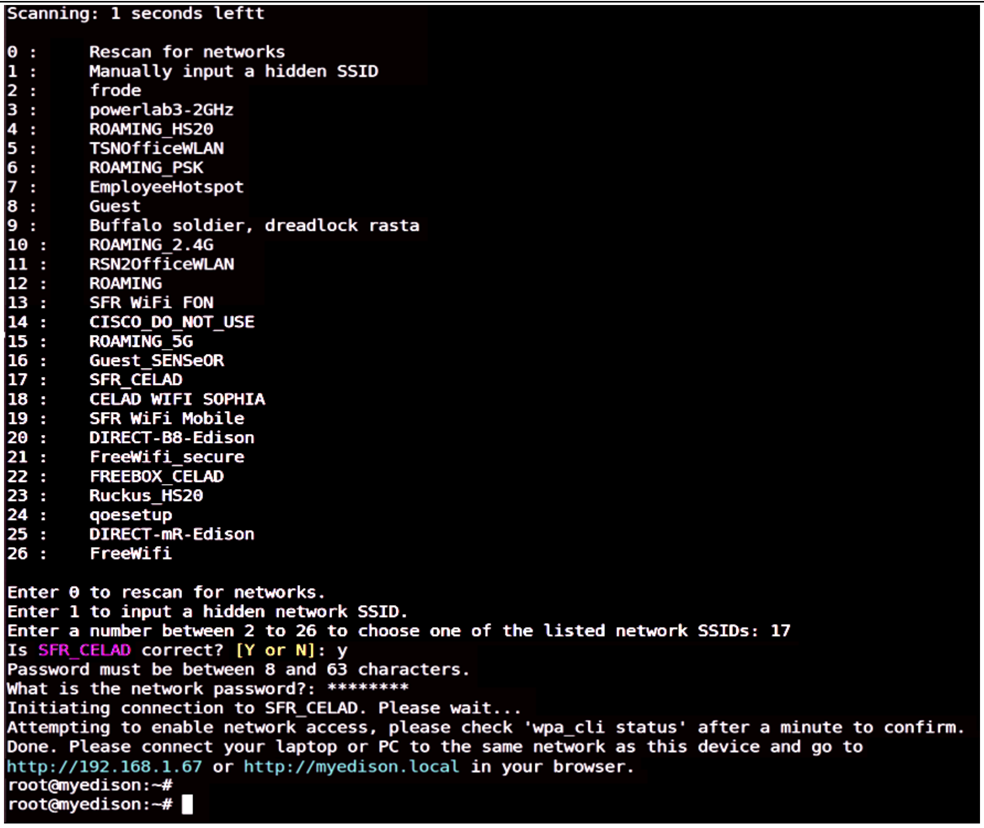
4.1 Configuring an Intel® Edison device as a Wi-Fi client

After successfully booting your Intel® Edison device, enter the following:

```
$ configure_edison --wifi
```

This command will scan for available networks and ask you which network you want to connect to. Enter the Wi-Fi AP number for the appropriate network (Figure 4). If your network does require a password or other information, enter the appropriate credentials. If the connection fails, you may have typed in your credentials incorrectly. Try again by typing `configure_edison --wifi` and repeat the procedure above.

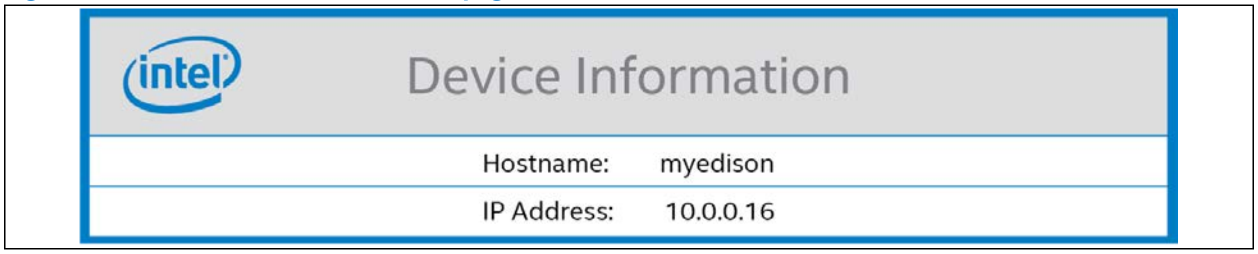
Figure 4 configure_edison --wifi



```
Scanning: 1 seconds leftt

0 :     Rescan for networks
1 :     Manually input a hidden SSID
2 :     frode
3 :     powerlab3-2GHz
4 :     ROAMING_HS20
5 :     TSNOfficeWLAN
6 :     ROAMING_PSK
7 :     EmployeeHotspot
8 :     Guest
9 :     Buffalo soldier, dreadlock rasta
10 :    ROAMING_2.4G
11 :    RSN2OfficeWLAN
12 :    ROAMING
13 :    SFR WiFi FON
14 :    CISCO_DO_NOT_USE
15 :    ROAMING_5G
16 :    Guest_SENSeOR
17 :    SFR_CELAD
18 :    CELAD WIFI SOPHIA
19 :    SFR WiFi Mobile
20 :    DIRECT-B8-Edison
21 :    FreeWifi_secure
22 :    FREEBOX_CELAD
23 :    Ruckus_HS20
24 :    qoesetup
25 :    DIRECT-mR-Edison
26 :    FreeWifi

Enter 0 to rescan for networks.
Enter 1 to input a hidden network SSID.
Enter a number between 2 to 26 to choose one of the listed network SSIDs: 17
Is SFR_CELAD correct? [Y or N]: y
Password must be between 8 and 63 characters.
What is the network password?: ********
Initiating connection to SFR_CELAD. Please wait...
Attempting to enable network access, please check 'wpa_cli status' after a minute to confirm.
Done. Please connect your laptop or PC to the same network as this device and go to
http://192.168.1.67 or http://myedison.local in your browser.
root@myedison:~#
root@myedison:~#
```

Once you connect to the Wi-Fi network, if your computer is connected to the same network, you can open up a browser and type in the IP address or name of your Intel® Edison device. For example, *http://192.168.1.67*. Visiting this address should open up a device information webpage (Figure 5).

Figure 5 Device information webpage



Device Information

Hostname: myedison

IP Address: 10.0.0.16

4.2 wpa_supplicant

The *wpa_supplicant* daemon is a minimal-code open-source implementation of IEEE 802.11i supplicant for Linux*. It runs in the background as the backend component controlling a wireless connection, implements key negotiation with a WPA authenticator, and controls roaming and IEEE 802.11 authentication/association of the WLAN driver.

By default, *wpa_supplicant* does not start at boot time. To have it start at boot time automatically, based on the last connection saved with *wpa_cli*, enter the following:

```
$ systemctl enable wpa_supplicant
$ systemctl disable wpa_supplicant (to remove the startup at boot time)
```

Use the following commands to start/stop *wpa_supplicant* manually:

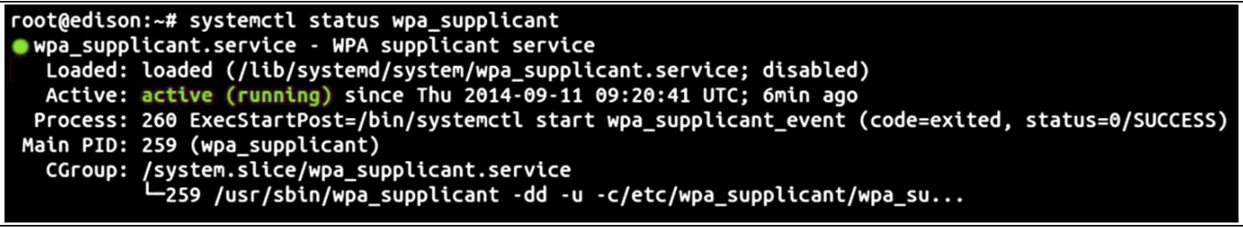
```
$ systemctl start wpa_supplicant
$ systemctl stop wpa_supplicant
```

Once *wpa_supplicant* has started, you can use *wpa_cli* to query current status, scan the network, select security mode, and connect to AP.

To verify that *wpa_supplicant* has started correctly, enter the following (see Figure 6):

```
$ systemctl status wpa_supplicant
```

Figure 6 wpa_supplicant status



```
root@edison:~# systemctl status wpa_supplicant
● wpa_supplicant.service - WPA supplicant service
   Loaded: loaded (/lib/systemd/system/wpa_supplicant.service; disabled)
   Active: active (running) since Thu 2014-09-11 09:20:41 UTC; 6min ago
  Process: 260 ExecStartPost=/bin/systemctl start wpa_supplicant_event (code=exited, status=0/SUCCESS)
 Main PID: 259 (wpa_supplicant)
   CGroup: /system.slice/wpa_supplicant.service
           └─259 /usr/sbin/wpa_supplicant -dd -u -c/etc/wpa_supplicant/wpa_su...
```

Because *connman* could automatically reconnect with saved connection parameters, when using *wpa_cli* to connect Wi-Fi, disable *connman* to avoid any connection conflicts. To stop *connman*, enter the following:

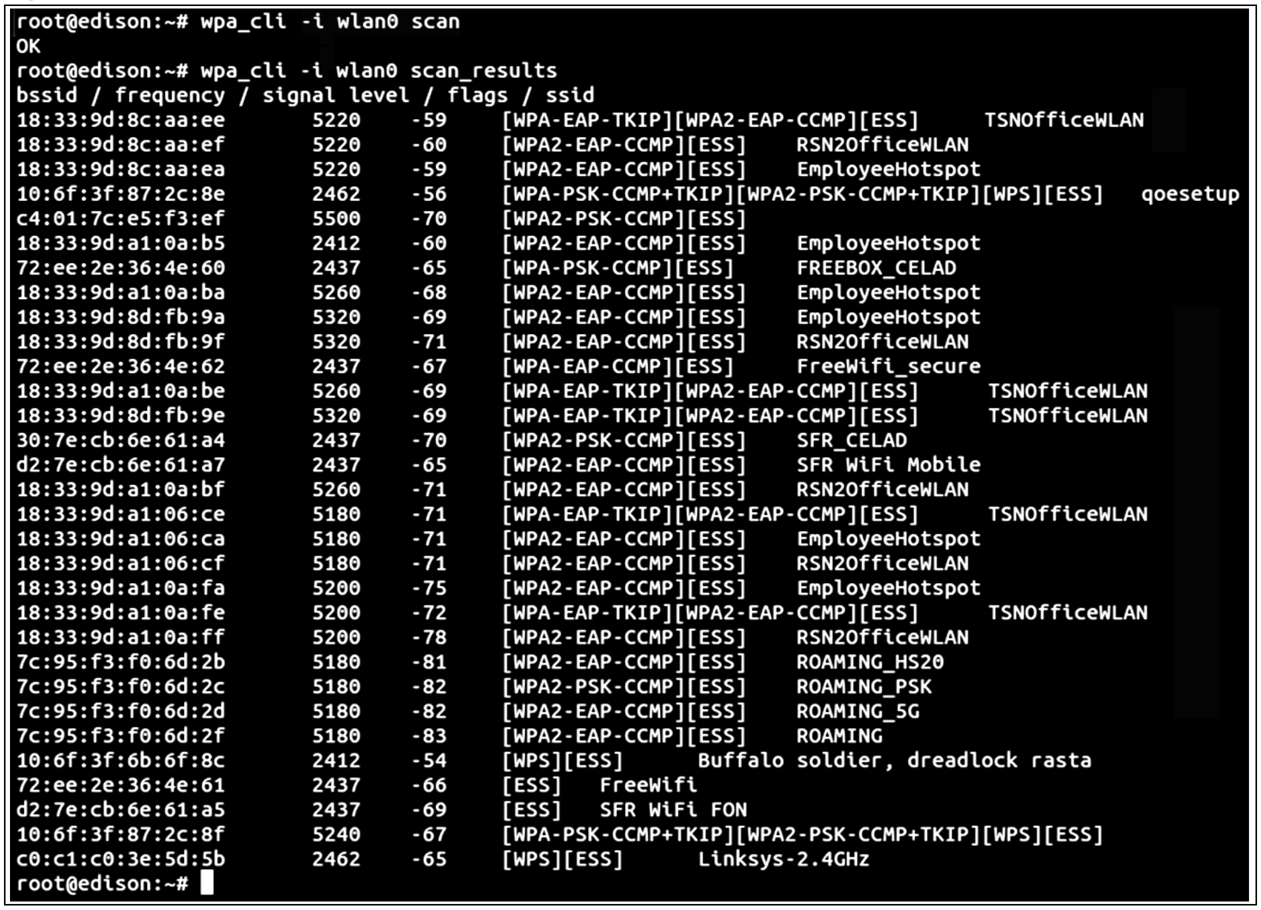
```
$ systemctl stop connman
```

4.2.1 Scan available networks

To display the list of configured networks, enter the following:

```
$ wpa_cli -i wlan0 scan
$ wpa_cli -i wlan0 scan_results
```

Figure 7 Scan result



```
root@edison:~# wpa_cli -i wlan0 scan
OK
root@edison:~# wpa_cli -i wlan0 scan_results
bssid / frequency / signal level / flags / ssid
18:33:9d:8c:aa:ee       5220    -59     [WPA-EAP-TKIP][WPA2-EAP-CCMP][ESS]      TSNOfficeWLAN
18:33:9d:8c:aa:ef       5220    -60     [WPA2-EAP-CCMP][ESS]    RSN2OfficeWLAN
18:33:9d:8c:aa:ea       5220    -59     [WPA2-EAP-CCMP][ESS]    EmployeeHotspot
10:6f:3f:87:2c:8e       2462    -56     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][WPS][ESS]   qoesetup
c4:01:7c:e5:f3:ef       5500    -70     [WPA2-PSK-CCMP][ESS]
18:33:9d:a1:0a:b5       2412    -60     [WPA2-EAP-CCMP][ESS]    EmployeeHotspot
72:ee:2e:36:4e:60       2437    -65     [WPA-PSK-CCMP][ESS]     FREEBOX_CELAD
18:33:9d:a1:0a:ba       5260    -68     [WPA2-EAP-CCMP][ESS]    EmployeeHotspot
18:33:9d:8d:fb:9a       5320    -69     [WPA2-EAP-CCMP][ESS]    EmployeeHotspot
18:33:9d:8d:fb:9f       5320    -71     [WPA2-EAP-CCMP][ESS]    RSN2OfficeWLAN
72:ee:2e:36:4e:62       2437    -67     [WPA-EAP-CCMP][ESS]     FreeWifi_secure
18:33:9d:a1:0a:be       5260    -69     [WPA-EAP-TKIP][WPA2-EAP-CCMP][ESS]      TSNOfficeWLAN
18:33:9d:8d:fb:9e       5320    -69     [WPA-EAP-TKIP][WPA2-EAP-CCMP][ESS]      TSNOfficeWLAN
30:7e:cb:6e:61:a4       2437    -70     [WPA2-PSK-CCMP][ESS]    SFR_CELAD
d2:7e:cb:6e:61:a7       2437    -65     [WPA2-EAP-CCMP][ESS]    SFR WiFi Mobile
18:33:9d:a1:0a:bf       5260    -71     [WPA2-EAP-CCMP][ESS]    RSN2OfficeWLAN
18:33:9d:a1:06:ce       5180    -71     [WPA-EAP-TKIP][WPA2-EAP-CCMP][ESS]      TSNOfficeWLAN
18:33:9d:a1:06:ca       5180    -71     [WPA2-EAP-CCMP][ESS]    EmployeeHotspot
18:33:9d:a1:06:cf       5180    -71     [WPA2-EAP-CCMP][ESS]    RSN2OfficeWLAN
18:33:9d:a1:0a:fa       5200    -75     [WPA2-EAP-CCMP][ESS]    EmployeeHotspot
18:33:9d:a1:0a:fe       5200    -72     [WPA-EAP-TKIP][WPA2-EAP-CCMP][ESS]      TSNOfficeWLAN
18:33:9d:a1:0a:ff       5200    -78     [WPA2-EAP-CCMP][ESS]    RSN2OfficeWLAN
7c:95:f3:f0:6d:2b       5180    -81     [WPA2-EAP-CCMP][ESS]    ROAMING_HS20
7c:95:f3:f0:6d:2c       5180    -82     [WPA2-PSK-CCMP][ESS]    ROAMING_PSK
7c:95:f3:f0:6d:2d       5180    -82     [WPA2-EAP-CCMP][ESS]    ROAMING_5G
7c:95:f3:f0:6d:2f       5180    -83     [WPA2-EAP-CCMP][ESS]    ROAMING
10:6f:3f:6b:6f:8c       2412    -54     [WPS][ESS]      Buffalo soldier, dreadlock rasta
72:ee:2e:36:4e:61       2437    -66     [ESS]   FreeWifi
d2:7e:cb:6e:61:a5       2437    -69     [ESS]   SFR WiFi FON
10:6f:3f:87:2c:8f       5240    -67     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][WPS][ESS]
c0:c1:c0:3e:5d:5b       2462    -65     [WPS][ESS]      Linksys-2.4GHz
root@edison:~# █
```

4.2.2 Connect to a protected AP

You will find a script connecting secure access point in section 9.2. You must run the script from the target (the Intel® Edison shell).

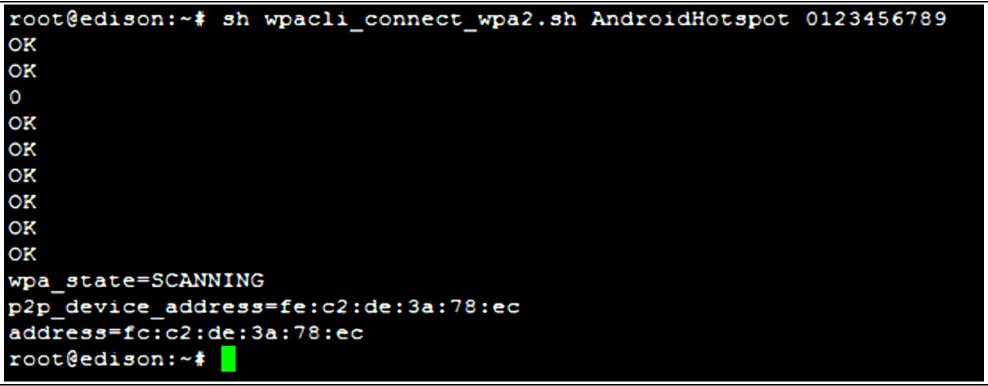
```
$ sh wpacli_connect_wpa2.sh <SSID> <Password>
```

To push files into the target, enter the following from your host terminal:

```
$ scp wpacli_connect_wpa2.sh root@192.168.2.15
```

In this example, the network SSID name is *AndroidHotspot* and the password for the network is *0123456789* (Figure 8).

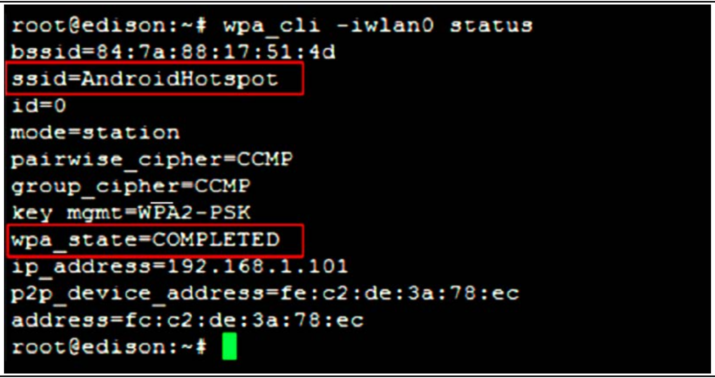
Figure 8 **Connecting to a protected AP**



```
root@edison:~# sh wpacli_connect_wpa2.sh AndroidHotspot 0123456789
OK
OK
0
OK
OK
OK
OK
OK
OK
wpa_state=SCANNING
p2p_device_address=fe:c2:de:3a:78:ec
address=fc:c2:de:3a:78:ec
root@edison:~#
```

Verify that you are correctly connected with the *status* command. The line `wpa_state = COMPLETED` in Figure 9 shows you are connected to the AP.

Figure 9 **Connecting to a protected AP**



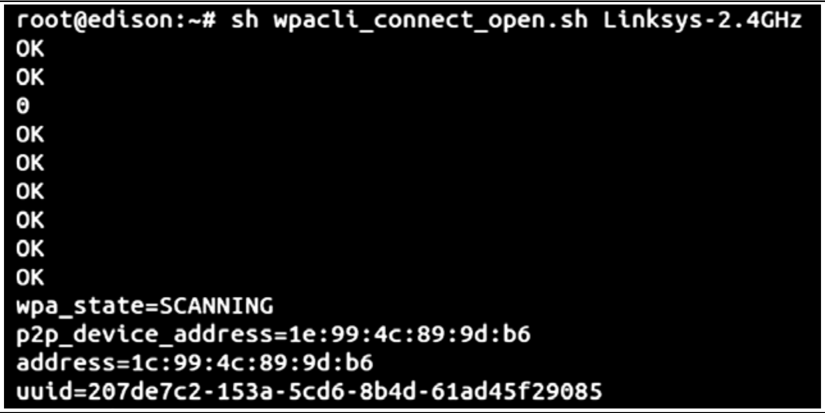
```
root@edison:~# wpa_cli -iwlan0 status
bssid=84:7a:88:17:51:4d
ssid=AndroidHotspot
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.1.101
p2p_device_address=fe:c2:de:3a:78:ec
address=fc:c2:de:3a:78:ec
root@edison:~#
```

4.2.3 Connect to an open AP

Section 9.1 details a WPA2-PSK script for connecting via secure access. You must run the script from the target.

```
$ sh wpacli_connect_open.sh <SSID>
```

Figure 10 **Connection to open AP with wpa_cli command**



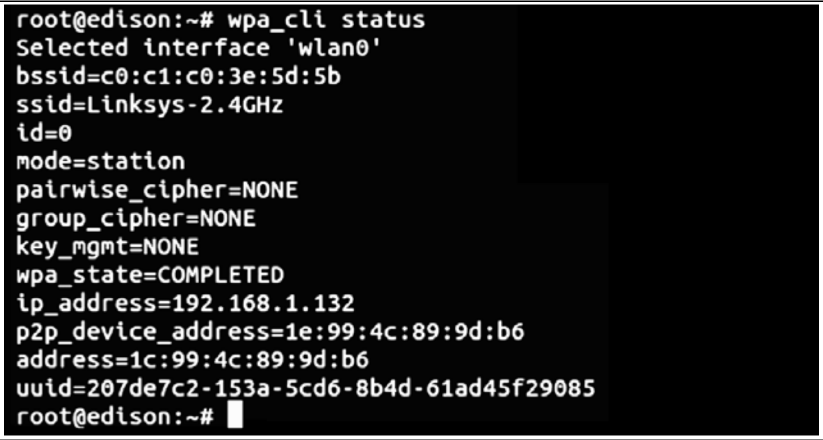
```
root@edison:~# sh wpacli_connect_open.sh Linksys-2.4GHz
OK
OK
0
OK
OK
OK
OK
OK
OK
wpa_state=SCANNING
p2p_device_address=1e:99:4c:89:9d:b6
address=1c:99:4c:89:9d:b6
uuid=207de7c2-153a-5cd6-8b4d-61ad45f29085
```

Verify that you are correctly connected with the following command:

```
$ wpa_cli status
```

The line `wpa_state = COMPLETED` in Figure 11 shows you are connected to the AP.

Figure 11 wpa_cli status



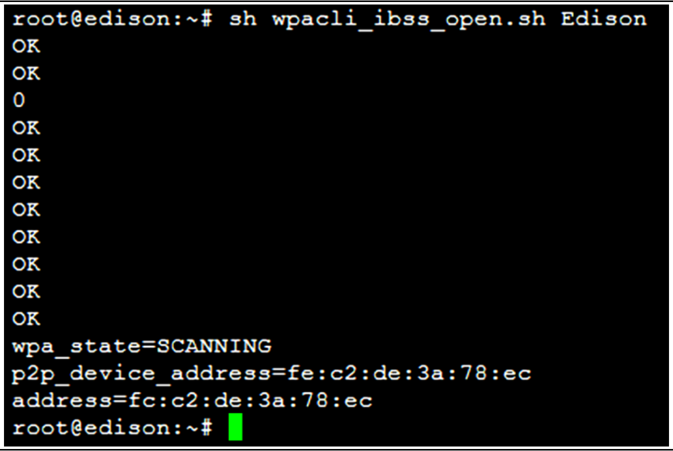
```
root@edison:~# wpa_cli status
Selected interface 'wlan0'
bssid=c0:c1:c0:3e:5d:5b
ssid=Linksys-2.4GHz
id=0
mode=station
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=192.168.1.132
p2p_device_address=1e:99:4c:89:9d:b6
address=1c:99:4c:89:9d:b6
uuid=207de7c2-153a-5cd6-8b4d-61ad45f29085
root@edison:~#
```

4.2.4 Create/connect to an ad-hoc network

You can use an ad-hoc network to create a wireless network or join an existing **independent basic service set** (IBSS) network without involving a master AP. To do so, do the following:

1. Run the *wpacli_ibss_open.sh* script listed in section 9.4 from the shell on the first Intel® Edison device to create an ad-hoc network with "*Edison*" as its SSID name (Figure 12).

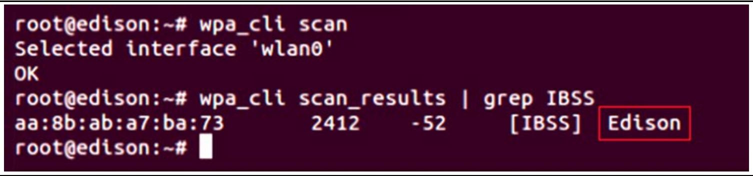
Figure 12 Creating an ad-hoc network



```
root@edison:~# sh wpacli_ibss_open.sh Edison
OK
OK
0
OK
OK
OK
OK
OK
OK
OK
OK
wpa_state=SCANNING
p2p_device_address=fe:c2:de:3a:78:ec
address=fc:c2:de:3a:78:ec
root@edison:~#
```

2. On the second Intel® Edison device, scan and list the ad-hoc interfaces (Figure 13).

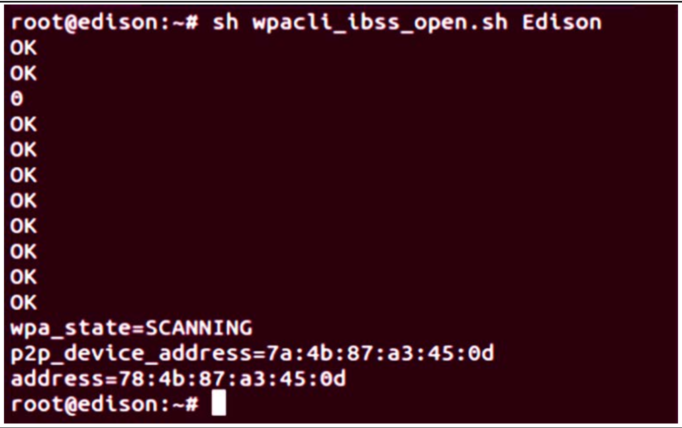
Figure 13 Scanning for an ad-hoc network



```
root@edison:~# wpa_cli scan
Selected interface 'wlan0'
OK
root@edison:~# wpa_cli scan_results | grep IBSS
aa:8b:ab:a7:ba:73       2412    -52     [IBSS]  Edison
root@edison:~#
```

3. From the second Intel® Edison device, connect to the newly created ad-hoc network by running the script *wpacli_ibss_open*.sh in section 9.4. (See Figure 14.)

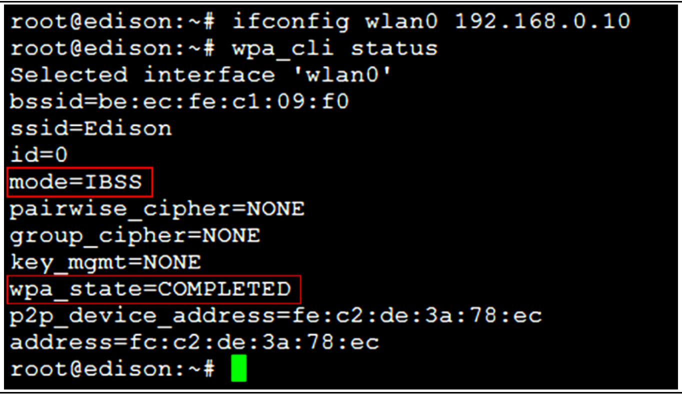
Figure 14 Connecting to an ad-hoc network



```
root@edison:~# sh wpacli_ibss_open.sh Edison
OK
OK
0
OK
OK
OK
OK
OK
OK
OK
OK
wpa_state=SCANNING
p2p_device_address=7a:4b:87:a3:45:0d
address=78:4b:87:a3:45:0d
root@edison:~#
```

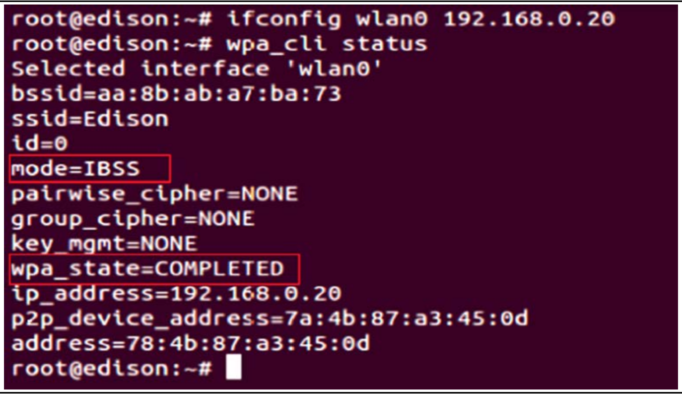
4. When you create an IBSS network or join an existing one, the DHCP client and DHCP server are not started, so you must manually assign the IP address of the wlan0 interface. You can statically configure the IP address on both of the Intel® Edison devices using the *ifconfig wlan0 <ip_address>* command (Figure 17).

Figure 15 Configuring a static IP address with ifconfig on the first device



```
root@edison:~# ifconfig wlan0 192.168.0.10
root@edison:~# wpa_cli status
Selected interface 'wlan0'
bssid=be:ec:fe:c1:09:f0
ssid=Edison
id=0
mode=IBSS
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
p2p_device_address=fe:c2:de:3a:78:ec
address=fc:c2:de:3a:78:ec
root@edison:~#
```

Figure 16 Configuring a static IP address with ifconfig on the second device



```
root@edison:~# ifconfig wlan0 192.168.0.20
root@edison:~# wpa_cli status
Selected interface 'wlan0'
bssid=aa:8b:ab:a7:ba:73
ssid=Edison
id=0
mode=IBSS
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=192.168.0.20
p2p_device_address=7a:4b:87:a3:45:0d
address=78:4b:87:a3:45:0d
root@edison:~#
```

5. Once the connection is established and an IP address is assigned to the peered devices, verify the connection using the *ping* command on both devices.

4.2.5 Display wpa supplicant traces

Because *wpa_supplicant* starts with a *systemd* service, *wpa_supplicant* traces display in the journal. You can retrieve these traces with the following command:

```
$ journalctl –f -u wpa_supplicant
```

By default, the level of *wpa_supplicant* of traces is set to INFO. You can check this with the following command:

```
$ wpa_cli log_level
```

You can increase the verbosity of *wpa_supplicant* with the following command:

```
$ wpa_cli log_level DEBUG
```

If you start *wpa_cli* without parameters, *wpa_cli* displays the events sent by *wpa_supplicant.* These events can help you troubleshoot problems.

Note: Because *connman* uses *wpa_supplicant*, the information above also applies when using *connman*.

4.3 Connman

Connman is a connection manager daemon that lets you manage network connections over Wi-Fi and designed for use with embedded devices. We recommend using *connman* release 1.25 or above. To check the release version, enter the following:

```
$ connmand –v
```

4.3.1 Starting connman

Because *wpa_supplicant* could automatically reconnect with *wpa_cli*-saved connection parameters, when using *connman* to connect Wi-Fi, before starting *connman*, use the following commands to remove all networks that would have been configured or saved with *wpa_cli*:

```
$ wpa_cli -i wlan0 remove_network all
$ wpa_cli save
```

Use the following commands to start/stop *connman*:

```
$ systemctl start connman
$ systemctl stop connman
```

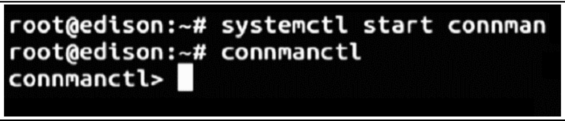
When the *connman* service starts, *wpa_supplicant* automatically starts. By default, *connman* does not start at boot time. To start it at boot time automatically, based on the last connection, enter the following:

```
$ systemctl enable connman
$ systemctl disable connman (to remove the startup at boot time)
```

Connman has a standard command line client *connmanctl*. It can run in two modes:

- In command mode, commands are entered as arguments to the *connmanctl* command, just like *systemctl*.
- Interactive mode is started by typing *connmanctl* without arguments. The prompt will change to `connmanctl>` to indicate it is waiting for user commands.

Figure 17 Starting connmanctl

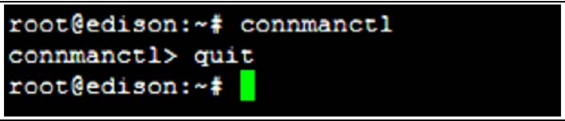


```
root@edison:~# systemctl start connman
root@edison:~# connmanctl
connmanctl> 
```

You can enter the *connman* command directly. Enter *help* to view all available commands.

To exit from the *connmanctl* interactive shell, enter *quit* or press Ctrl+D.

Figure 18 Quitting connmanctl



```
root@edison:~# connmanctl
connmanctl> quit
root@edison:~# 
```

4.3.2 Scan available networks

Enter following *connman* commands:

```
$ connmanctl enable wifi
$ connmanctl scan wifi
```

If you encounter an error when you run the *scan wifi* command, enter the following commands as a workaround, then later scan the Wi-Fi networks:

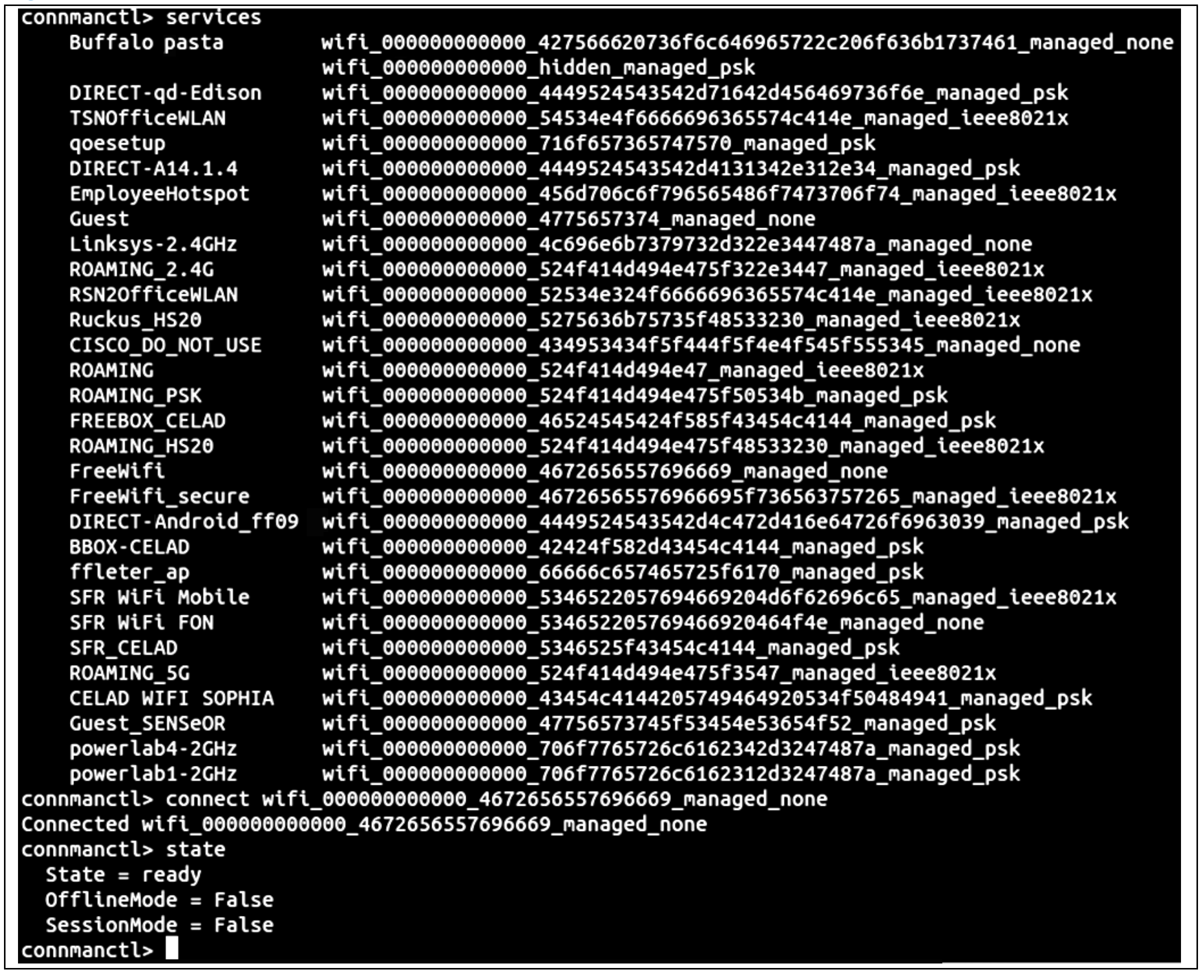
```
$ rfkill unblock wlan
$ systemctl start wpa_supplicant
$ systemctl start connman
```

To list the available networks found after a scan, enter the following:

```
$ connmanctl services
```

Figure 19 shows typical output from a network scan.

Figure 19 Network scan output



```
connmanctl> services
    Buffalo pasta        wifi_000000000000_427566620736f6c646965722c206f636b1737461_managed_none
                         wifi_000000000000_hidden_managed_psk
    DIRECT-qd-Edison     wifi_000000000000_4449524543542d71642d456469736f6e_managed_psk
    TSNOfficeWLAN        wifi_000000000000_54534e4f6666696365574c414e_managed_ieee8021x
    qoesetup             wifi_000000000000_716f657365747570_managed_psk
    DIRECT-A14.1.4       wifi_000000000000_4449524543542d4131342e312e34_managed_psk
    EmployeeHotspot      wifi_000000000000_456d706c6f796565486f7473706f74_managed_ieee8021x
    Guest                wifi_000000000000_4775657374_managed_none
    Linksys-2.4GHz       wifi_000000000000_4c696e6b7379732d322e3447487a_managed_none
    ROAMING_2.4G         wifi_000000000000_524f414d494e475f322e3447_managed_ieee8021x
    RSN2OfficeWLAN       wifi_000000000000_52534e324f6666696365574c414e_managed_ieee8021x
    Ruckus_HS20          wifi_000000000000_5275636b75735f48533230_managed_ieee8021x
    CISCO_DO_NOT_USE     wifi_000000000000_434953434f5f444f5f4e4f545f555345_managed_none
    ROAMING              wifi_000000000000_524f414d494e47_managed_ieee8021x
    ROAMING_PSK          wifi_000000000000_524f414d494e475f50534b_managed_psk
    FREEBOX_CELAD        wifi_000000000000_46524545424f585f43454c4144_managed_psk
    ROAMING_HS20         wifi_000000000000_524f414d494e475f48533230_managed_ieee8021x
    FreeWifi             wifi_000000000000_4672656557696669_managed_none
    FreeWifi_secure      wifi_000000000000_46726565576966695f736563757265_managed_ieee8021x
    DIRECT-Android_ff09  wifi_000000000000_4449524543542d4c472d416e64726f6963039_managed_psk
    BBOX-CELAD           wifi_000000000000_42424f582d43454c4144_managed_psk
    ffleter_ap           wifi_000000000000_66666c657465725f6170_managed_psk
    SFR WiFi Mobile      wifi_000000000000_5346522057694669204d6f62696c65_managed_ieee8021x
    SFR WiFi FON         wifi_000000000000_534652205769466920464f4e_managed_none
    SFR_CELAD            wifi_000000000000_5346525f43454c4144_managed_psk
    ROAMING_5G           wifi_000000000000_524f414d494e475f3547_managed_ieee8021x
    CELAD WIFI SOPHIA    wifi_000000000000_43454c4144205749464920534f50484941_managed_psk
    Guest_SENSeOR        wifi_000000000000_47756573745f53454e53654f52_managed_psk
    powerlab4-2GHz       wifi_000000000000_706f7765726c6162342d3247487a_managed_psk
    powerlab1-2GHz       wifi_000000000000_706f7765726c6162312d3247487a_managed_psk
connmanctl> connect wifi_000000000000_4672656557696669_managed_none
Connected wifi_000000000000_4672656557696669_managed_none
connmanctl> state
  State = ready
  OfflineMode = False
  SessionMode = False
connmanctl> █
```

4.3.3 Connect to an open AP

To connect to an open network, use the *connmanctl connect* command and enter the full “wifi_” name. To use an example from Figure 19:

```
$ connmanctl connect wifi_000000000000_716f657365747570_managed_psk
```

Verify that you are connected to the network by entering the following:

```
$ connmanctl state
```

4.3.4 Connect to a protected AP

For protected access points, you will need to provide some information to the *connman* daemon—at the very least a password or a passphrase. The commands in this section show how to run *connmanctl* in interactive mode, which is required for running the *agent* command. To start interactive mode, enter the following:

```
$ connmanctl
```

To scan for available Wi-Fi networks, enter the following:

```
connmanctl> scan wifi
```

To list services, enter the following:

```
connmanctl> services
```

To register the agent to handle user requests, enter the following:

```
connmanctl> agent on
```

To connect to one of the protected services, enter the following:

```
connmanctl> connect <service>
```

...where *<service>* is not the name of SSID but the full “wifi_” name, as shown in Figure 19. You might have to enter a passphrase if the selected Wi-Fi network requires authentication. For a WPS connection, enter an empty string for the passphrase so the system can then ask you to enter the WPS PIN or WPS PBC.

4.3.5 Display connman traces

Since *connman* starts with a *systemd* service, *connman* traces will display in the journal. You can retrieve these traces with the following command:

```
$ journalctl -f -u connman
```

§

5 Access Point Setup

5.1 Software-enable access point setup

Configure your device for Access Point mode by following the steps outlined here:

- *https://communities.intel.com/docs/DOC-23137*

SoftAP is supported through *hostapd* with the following configuration files for AP mode:

- */etc/hostapd/hostapd.conf*. This is the main configuration file, which you will need to edit to set up a SoftAP. The SSID is in the form *EDISON-XX-XX* (where the Xs are hexadecimal digits), and the passphrase is the serial number of the Intel® Edison compute module, both of which appear on the box label your Intel® Edison compute module shipped in. Verify the SSID and passphrase by comparing them to the values for *ssid* and *wpa_passphrase* in the *hostapd.conf* file.
- */etc/hostapd/udhcpd-for-hostapd.conf*. This is the configuration file for the DHCP server.

We recommend using the default *hostapd.conf* file, which comes with the Intel® Edison image package and is well documented with comments. You can edit and modify this file to suit your requirements. For example, instead of using the default *hostapd.conf*, you could use a minimal configuration setting with just the following:

```
interface=wlan0
driver=nl80211
ssid=test
channel=1
```

Start *hostapd*. Use the following to start/stop *hostapd*:

```
$ systemctl start hostapd
$ systemctl stop hostapd
```

To download the *hostapd* configuration file, visit this website: *http://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf*.

Note: Because the Broadcom module does not support STA/AP concurrently, *hostapd* and *wpa_supplicant* cannot run simultaneously.

Use the following commands to start/stop *wpa_supplicant* manually:

```
$ systemctl start wpa_supplicant
$ systemctl stop wpa_supplicant
```

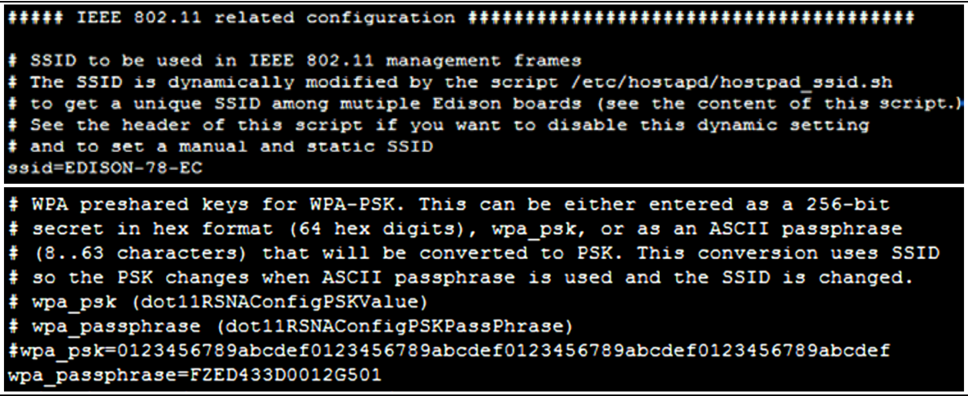
5.2 Software-enable access point setup using hidden SSID

We can configure an Intel® Edison device in AP mode with hidden SSID by setting the field *ignore_broadcast_ssid* to 1 (by default, the value is 0) in file */etc/hostapd/hostapd.conf*. This example uses two Intel® Edison devices: the first acting as an AP with hidden SSID, and the second acting as an STA.

On the first Intel® Edison device:

1. Obtain the first Intel® Edison compute module's SSID and passphrase. Open the */etc/hostapd/hostapd.conf* file and note the *ssid* and *wpa_passphrase* fields. In this example, the SSID is *EDISON-78-EC*, and the passphrase is *FZED433D0012G501* (Figure 20).

Figure 20 hostapd.conf file

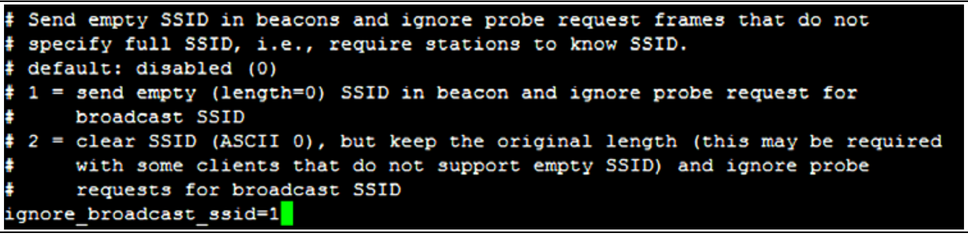


```
##### IEEE 802.11 related configuration #######################################

# SSID to be used in IEEE 802.11 management frames
# The SSID is dynamically modified by the script /etc/hostapd/hostpad_ssid.sh
# to get a unique SSID among mutiple Edison boards (see the content of this script.)
# See the header of this script if you want to disable this dynamic setting
# and to set a manual and static SSID
ssid=EDISON-78-EC
```

```
# WPA preshared keys for WPA-PSK. This can be either entered as a 256-bit
# secret in hex format (64 hex digits), wpa_psk, or as an ASCII passphrase
# (8..63 characters) that will be converted to PSK. This conversion uses SSID
# so the PSK changes when ASCII passphrase is used and the SSID is changed.
# wpa_psk (dot11RSNAConfigPSKValue)
# wpa_passphrase (dot11RSNAConfigPSKPassPhrase)
#wpa_psk=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
wpa_passphrase=FZED433D0012G501
```

2. Change the value of *ignore_broadcast_ssid* field to 1. Edit the */etc/hostapd/hostapd.conf* file, set the field to 1, and save the file.

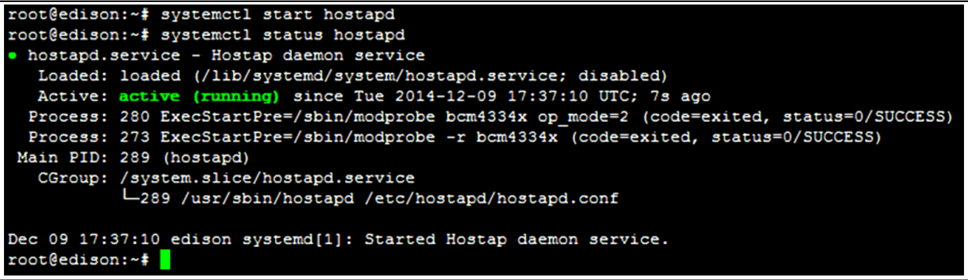
Figure 21 Ignore broadcast SSID



```
# Send empty SSID in beacons and ignore probe request frames that do not
# specify full SSID, i.e., require stations to know SSID.
# default: disabled (0)
# 1 = send empty (length=0) SSID in beacon and ignore probe request for
#     broadcast SSID
# 2 = clear SSID (ASCII 0), but keep the original length (this may be required
#     with some clients that do not support empty SSID) and ignore probe
#     requests for broadcast SSID
ignore_broadcast_ssid=1
```

3. To configure the Intel® Edison device as an access point, start *hostapd* (Figure 22).

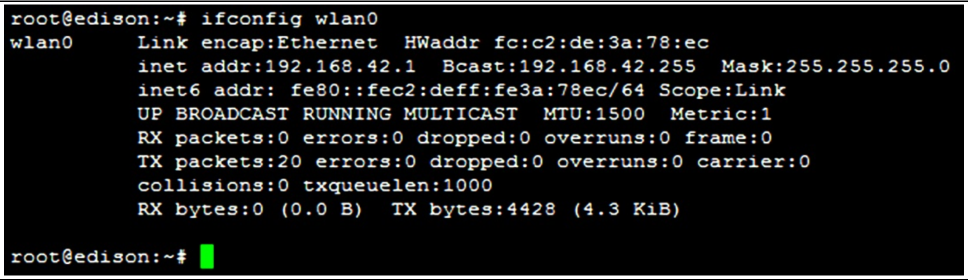
Figure 22 Configure as AP with hostapd



```
root@edison:~# systemctl start hostapd
root@edison:~# systemctl status hostapd
● hostapd.service - Hostap daemon service
   Loaded: loaded (/lib/systemd/system/hostapd.service; disabled)
   Active: active (running) since Tue 2014-12-09 17:37:10 UTC; 7s ago
  Process: 280 ExecStartPre=/sbin/modprobe bcm4334x op_mode=2 (code=exited, status=0/SUCCESS)
  Process: 273 ExecStartPre=/sbin/modprobe -r bcm4334x (code=exited, status=0/SUCCESS)
 Main PID: 289 (hostapd)
   CGroup: /system.slice/hostapd.service
           └─289 /usr/sbin/hostapd /etc/hostapd/hostapd.conf

Dec 09 17:37:10 edison systemd[1]: Started Hostap daemon service.
root@edison:~#
```

4. Once hostapd starts successfully, you can see the IP address assigned to the wlan0 interface.

Figure 23 IP address of the wlan0 interface



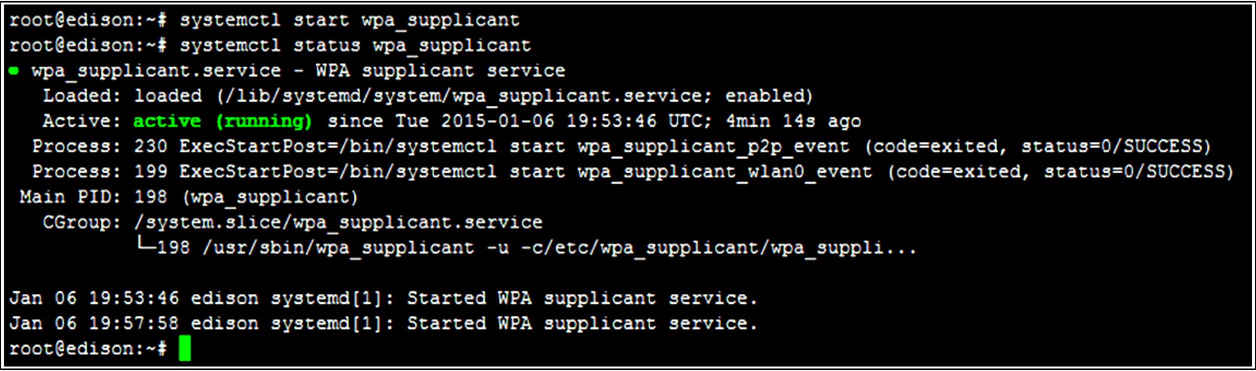
```
root@edison:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr fc:c2:de:3a:78:ec
          inet addr:192.168.42.1  Bcast:192.168.42.255  Mask:255.255.255.0
          inet6 addr: fe80::fec2:deff:fe3a:78ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:4428 (4.3 KiB)

root@edison:~#
```

At this point, the first Intel® Edison device is acting as an AP; however, its SSID is hidden.

On the second Intel® Edison device:

1. From the second Intel® Edison device, connect as an STA using *wpa_supplicant*. Check the status of *wpa_supplicant* and start it if is not started already (Figure 24).

Figure 24 wpa_supplicant status



```
root@edison:~# systemctl start wpa_supplicant
root@edison:~# systemctl status wpa_supplicant
● wpa_supplicant.service - WPA supplicant service
   Loaded: loaded (/lib/systemd/system/wpa_supplicant.service; enabled)
   Active: active (running) since Tue 2015-01-06 19:53:46 UTC; 4min 14s ago
  Process: 230 ExecStartPost=/bin/systemctl start wpa_supplicant_p2p_event (code=exited, status=0/SUCCESS)
  Process: 199 ExecStartPost=/bin/systemctl start wpa_supplicant_wlan0_event (code=exited, status=0/SUCCESS)
 Main PID: 198 (wpa_supplicant)
   CGroup: /system.slice/wpa_supplicant.service
           └─198 /usr/sbin/wpa_supplicant -u -c/etc/wpa_supplicant/wpa_suppli...

Jan 06 19:53:46 edison systemd[1]: Started WPA supplicant service.
Jan 06 19:57:58 edison systemd[1]: Started WPA supplicant service.
root@edison:~#
```

2. To display the list of configured networks, enter the following:

```
$ wpa_cli –i wlan0 scan
$ wpa_cli –i wlan0 scan_results
```

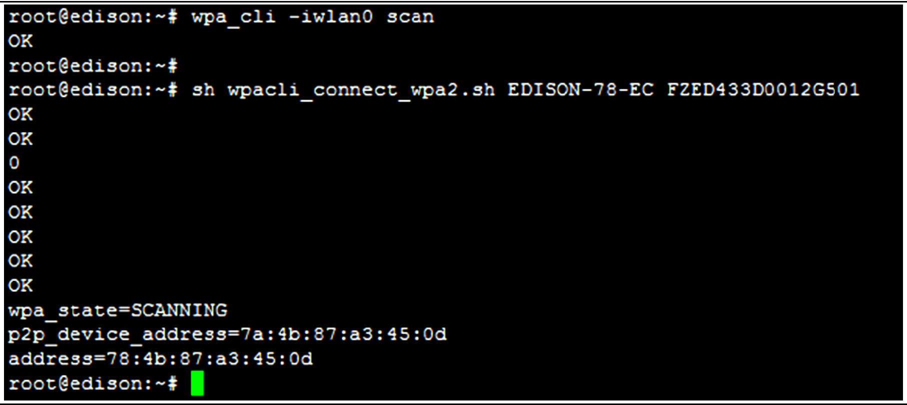
   Because the SSID of the first Intel® Edison device is hidden, it will not show in the discovered lists.

3. Section 9.2 features a script that lets you connect to a secure access point. Run the script from the target (the Intel® Edison shell).

```
$ wpacli_connect_wpa2.sh <ssid> <password>
```

   ...where *sid* and *passphrase* are the network ID and password of the first Intel® Edison device.

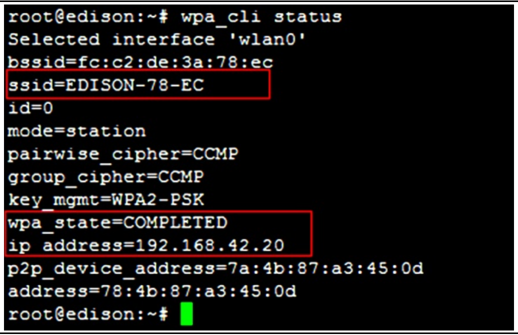
Figure 25 Running the secure AP script



```
root@edison:~# wpa_cli -iwlan0 scan
OK
root@edison:~#
root@edison:~# sh wpacli_connect_wpa2.sh EDISON-78-EC FZED433D0012G501
OK
OK
0
OK
OK
OK
OK
OK
wpa_state=SCANNING
p2p_device_address=7a:4b:87:a3:45:0d
address=78:4b:87:a3:45:0d
root@edison:~#
```

4. Check the status to see the connection was successful (Figure 26).

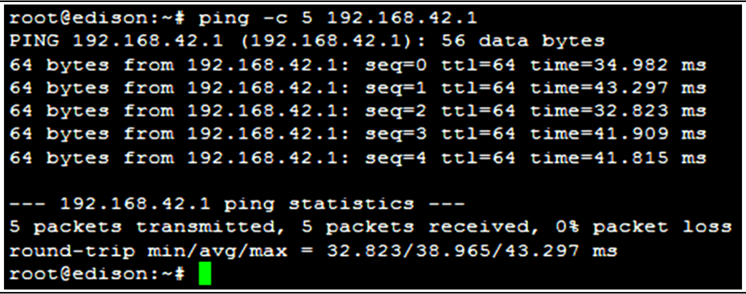
Figure 26 Connection status



```
root@edison:~# wpa_cli status
Selected interface 'wlan0'
bssid=fc:c2:de:3a:78:ec
ssid=EDISON-78-EC
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.42.20
p2p_device_address=7a:4b:87:a3:45:0d
address=78:4b:87:a3:45:0d
root@edison:~#
```

5. The line *wpa_state = COMPLETED* indicates that the connection is complete. Ping the first Intel® Edison device (in AP mode) to check the connection.

Figure 27 Ping



```
root@edison:~# ping -c 5 192.168.42.1
PING 192.168.42.1 (192.168.42.1): 56 data bytes
64 bytes from 192.168.42.1: seq=0 ttl=64 time=34.982 ms
64 bytes from 192.168.42.1: seq=1 ttl=64 time=43.297 ms
64 bytes from 192.168.42.1: seq=2 ttl=64 time=32.823 ms
64 bytes from 192.168.42.1: seq=3 ttl=64 time=41.909 ms
64 bytes from 192.168.42.1: seq=4 ttl=64 time=41.815 ms

--- 192.168.42.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 32.823/38.965/43.297 ms
root@edison:~#
```

§

6 Wi-Fi Direct

Wi-Fi Direct, also known as Wi-Fi Peer to Peer (P2P), allows Wi-Fi capable devices to connect directly to each other without a Wi-Fi hotspot. Only one of the participating devices must comply with Wi-Fi Direct to establish a peer-to-peer connection. In theory, devices having Wi-Fi Direct establish their own peer-to-peer networks with the WPA2-PSK security features. Wi-Fi Direct is much faster than Bluetooth* and allows devices to communicate or transfer data to each other across greater distances.

To set up Wi-Fi Direct on the Intel® Edison platform, we manage P2P connections with *wpa_supplicant*, which runs as a daemon and manages wireless communication.

Table 3 lists some basic P2P commands.

Table 3 Basic P2P commands

Command	Description
`p2p_find [timeout] [type=*]`	Find/search P2P devices for up to *[timeout]* seconds.
`p2p_stop_find`	Stop P2P devices search.
`p2p_connect <addr> <"pbc"|PIN> [ht40]`	Connect to a P2P device.
`p2p_listen [timeout]`	Listen for P2P devices for up to [timeout] seconds.
`p2p_group_remove <ifname>`	Remove P2P group interface. (Terminate group if GO.)
`p2p_group_add [ht40]`	Add a new P2P group. (Local end as GO.)
`p2p_prov_disc <addr> <method>`	Request provisioning discovery.
`p2p_invite <cmd> [peer=addr]`	Invite peer.
`p2p_peers [discovered]`	List known (optionally, only fully discovered) P2P peers.
`p2p_peer <address>`	Show information about known P2P peer.
`p2p_set <field> <value>`	Set a P2P parameter.
`p2p_flush`	Flush P2P state.
`p2p_cancel`	Cancel P2P group formation.
`p2p_remove_client <address|iface=address>`	Remove a peer from all groups.

6.1 Known limitations

Because of regulatory requirements, the Intel® Edison platform currently only supports P2P at 2.4 GHz.

6.2 Using Wi-Fi Direct

We can use Wi-Fi Direct to connect between two Intel® Edison devices (or between an Intel® Edison device and an Android* device) as peer devices. We can form P2P connections using these modes:

- PBC (push button control)
- PIN (pin mode)

6.3 Basic peer-to-peer setup

To set up a basic peer-to-peer connection, do the following:

1. Check the status of *wpa_supplicant* on both Intel® Edison devices. By default, *wpa_supplicant* does not start at boot time.

```
root@edison:~# systemctl status wpa_supplicant
● wpa_supplicant.service - WPA supplicant service
   Loaded: loaded (/lib/system/system/wpa_supplicant.service; disabled)
   Active: inactive (dead)
root@edison:~#
```

2. If the state is inactive, start *wpa_supplicant* on both of the Intel® Edison devices:

```
root@edison:~# systemctl start wpa_supplicant
root@edison:~# systemctl status wpa_supplicant
● wpa_supplicant.service - WPA supplicant service
   Loaded: loaded (/lib/system/system/wpa_supplicant.service; disabled)
   Active: active (running) since Fri 2015-01-02 19:08:27 UTC, 2s ago
  Process: 278 ExecStartPost=/bin/systemctl start wpa_supplicant_p2p_event
(code=exited, status=0/SUCCESS)
  Process: 275 ExecStartPost=/bin/systemctl start wpa_supplicant_wlan0_event
(code=exited, status=0/SUCCESS)
 MAIN PID: 274 (wpa_supplicant)
   CGroup: /system.slice/wpa_supplicant.service
           └ 274 /usr/sbin/wpa_supplicant –u –c/etc/wpa_supplicant/wpa...

Jan 02 19:08:27 edison systemd[1]: Started WPA supplicant service.
root@edison:~#
```

3. Check Wi-Fi Direct activation on both of the Intel® Edison devices: Activation of the P2P on the Intel® Edison device can be confirmed with the activation of the interface “p2p-dev-wlan0”.

   Use the following command to list all wireless interfaces:

```
root@edison:~# wpa_cli -iwlan0 interface
Available interfaces:
p2p-dev-wlan0
wlan0
root@edison:~#
```

4. Start discovery of the P2P device on both of the Intel® Edison devices:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_find
OK
root@edison:~#
```

5. Use the *p2p_peers* command to list and obtain the MAC address of the discovered peer devices on both of the Intel® Edison devices.

   First Intel® Edison device:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_peers
7a:4b:87:a3:45:0d
root@edison:~#
```

Second Intel® Edison device:

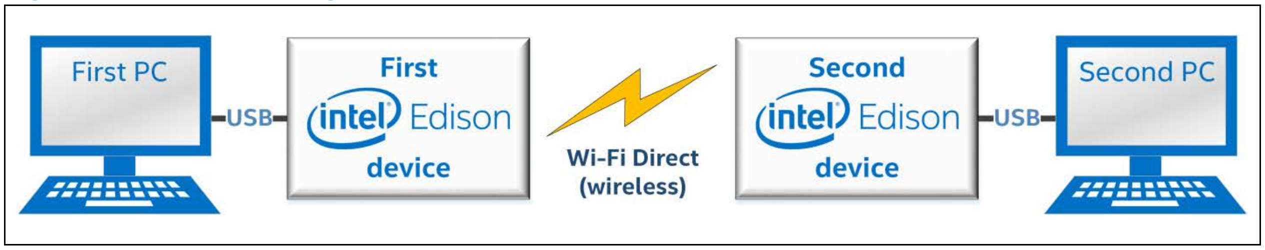
```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_peers
fe:c2:de:3a:78:ec
root@edison:~#
```

Note: Because the *p2p_find* command can take a few seconds before it finds a peer device, the *p2p_peers* command may return an empty value. If it does, wait a few seconds and enter the command again.

6. Optionally, you could list the detailed information of the peers, using their discovered MAC addresses:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_peer 7a:4b:87:a3:45:0d
  7a:4b:87:a3:45:0d
  pri_dev_type=0-00000000-0
  device_name=Edison
  manufacturer=Intel
  model_number=Edison
  serial_number=
  config_methods=0x188
  dev_capab=0x25
  group_capab=0x0
  level=0
  persistent=0
  age=336
  listen_freq=2437
  wps_method=not-ready
  interface_addr=7a:4b:87:a3:c5:0d
  member_in_go_dev=00:00:00:00:00:00
  member_in_go_iface =00:00:00:00:00:00
  go_neg_req_sent=0
  go_state=local
  dialog_token=1
  intended_addr=7a:4b:87:a3:c5:0d
  country=XX
  oper_freq=2412
  req_config_methods=0x0
  flags=[REPORTED][PREFER_PERSISTENT_GROUP]
  status=0
  wait_count=9
  invitation_reqs=0
  oper_ssid=DIRECT-3Y-Edison
root@edison:~#
```

Figure 28 Establishing a P2P connection between two Intel® Edison devices



6.4 P2P connection with PIN mode as autonomous GO

In this example, we will configure the first Intel® Edison device as a Group Owner and form a P2P group with the discovered peer using PIN mode. To do so, perform the procedure in section 6.3 to set up a basic peer-to-peer connection, then do the following:

1. Manually configure the first Intel® Edison device as a group owner (GO) with the following command. This command will create a new “autonomous GO” interface with the format *p2p-wlan0-x* (where *x* is an integer).

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_group_add
OK
```

2. You can list the new interface, along with existing interfaces, with the following command:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-0
p2p-dev-wlan0
wlan0
root@edison:~#
```

3. Enter the *wpa_cli -ip2p-wlan0-x wps_pin any* command, where *x* is an integer. This will return a PIN, an eight-digit number that you will later use to enter in the connection command for the second Intel® Edison device (the client). The *any* parameter allows any station to use the PIN. The PIN returned in the example below is *25889055*.

```
root@edison:~# wpa_cli -ip2p-wlan0-0 wps_pin any
25889055root@edison:~#
```

4. On the second Intel® Edison device, enter the following command to connect with the first Intel® Edison device with the PIN from above. The *join* parameter indicates that this is a command to join an existing group as a client.

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect fe:c2:de:3a:78:ec
25889055 join
OK
root@edison:~#
```

Note: When a P2P connection is established, the GO has to enable a DHCP server, in order to assign IP addresses to CLIs. It does so with a service named *wpa_cli_event*, which is called by *wpa_supplicant* when the wireless connection is activated.

5. Query the IP address of the *p2p-wlan0-x* interface with `ifconfig p2p-wlan0-x`:

   First Intel® Edison device:

```
root@edison:~# ifconfig p2p-wlan0-0
p2p-wlan0-1 Link encap:Ethernet HWaddr fe:c2:de:3a:f8:ec
    inet addr:192.168.42.1  Bcast:192.168.42.255  Mask:255.255.255.0
    inet6 addr: fe80::fcc2:deff:fe3a:f8ec/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
    RX packets:33 errors:0 dropped:0 overruns:0 frame:0
    TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:4376 (4.2 KiB)  TX bbytes:5061 (4.9 KiB)

root@edison:~#
```

   Second Intel® Edison device:

```
root@edison:~# ifconfig p2p-wlan0-0
p2p-wlan0-0 Link encap:Ethernet HWaddr 7a:4b:87:a3:c5:0d
    inet addr:192.168.42.20  Bcast:0.0.0.0  Mask:255.255.255.0
    UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
    RX packets:17 errors:0 dropped:0 overruns:0 frame:0
    TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:2549 (2.4 KiB)  TX bbytes:4896 (4.7 KiB)

root@edison:~#
```

6. Optionally, you can query the “status” as well.

   First Intel® Edison device:

```
root@edison:~# wpa_cli -ip2p-wlan0-0 status
  bssid=fe:c2:de:3a:f8:ec
  ssid=DIRECT-xf-Edison
  id=0
  mode=P2P GO
  pairwise_cipher=CCMP
  group_cipher=CCMP
  key_mgmt=WPA2-PSK
  wpa_state=COMPLETED
  ip_address=192.168.42.1
  p2p_device_address=fe:c2:de:3a:78:ec
  address=fe:c2:de:3a:f8:ec
root@edison:~#
```

Second Intel® Edison device:

```
root@edison:~# wpa_cli -ip2p-wlan0-0 status
  bssid=fe:c2:de:3a:f8:ec
  ssid=DIRECT-TS-Edison
  id=0
  mode=station
  pairwise_cipher=CCMP
  group_cipher=CCMP
  key_mgmt=WPA2-PSK
  wpa_state=COMPLETED
  ip_address=192.168.42.21
  p2p_device_address=7a:4b:87:a3:45:0d
  address=7a:4b:87:a3:45:0d
root@edison:~#
```

7. Verify the connection between the devices by cross-pinging them. From the command prompt of the first Intel® Edison device, ping the IP address of the second Intel® Edison device, and vice versa.

   First Intel® Edison device:

```
root@edison:~# ping -c 5 192.168.42.20
PING 192.168.42.20 (192.168.42.20): 56 data bytes
64 bytes from 192.168.42.20: seq=0 ttl=64 time=96.037 ms
64 bytes from 192.168.42.20: seq=1 ttl=64 time=116.956 ms
64 bytes from 192.168.42.20: seq=2 ttl=64 time=144.259 ms
64 bytes from 192.168.42.20: seq=3 ttl=64 time=163.489 ms
64 bytes from 192.168.42.20: seq=4 ttl=64 time=194.749 ms

--- 192.168.42.20 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 96.037/143.098/194.749 ms
root@edison:~#
```

   Second Intel® Edison device:

```
root@edison:~# ping -c 5 192.168.42.1
PING 192.168.42.1 (192.168.42.1): 56 data bytes
64 bytes from 192.168.42.1: seq=0 ttl=64 time=31.709 ms
64 bytes from 192.168.42.1: seq=1 ttl=64 time=30.777 ms
64 bytes from 192.168.42.1: seq=2 ttl=64 time=29.662 ms
64 bytes from 192.168.42.1: seq=3 ttl=64 time=39.124 ms
64 bytes from 192.168.42.1: seq=4 ttl=64 time=49.662 ms

--- 192.168.42.20 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 29.662/36.186/49.662 ms
root@edison:~#
```

6.5 P2P connection with PIN mode with group negotiation

Perform the procedure in section 6.3 to set up a basic peer-to-peer connection, then do the following:

1. Start the process of group negotiation without actually starting GO. The peer is expected to initiate a GO negotiation. On the first Intel® Edison device, enter the following *pin* command, which will return the PIN code that the peer device will use to connect. The PIN returned in the example below is *787062935*.

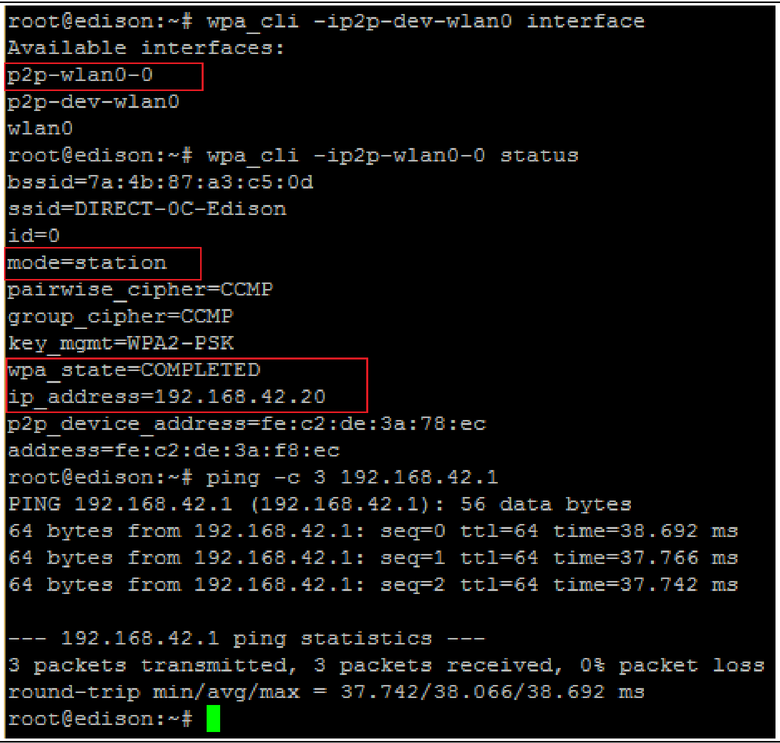
```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect 7a:4b:87:a3:45:0d
pin auth
78706293root@edison:~#
```

2. On the second Intel® Edison device, enter the following command with the returned PIN to connect with the peer device:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect fe:c2:de:3a:78:ec
78706293
OK
root@edison:~#
```

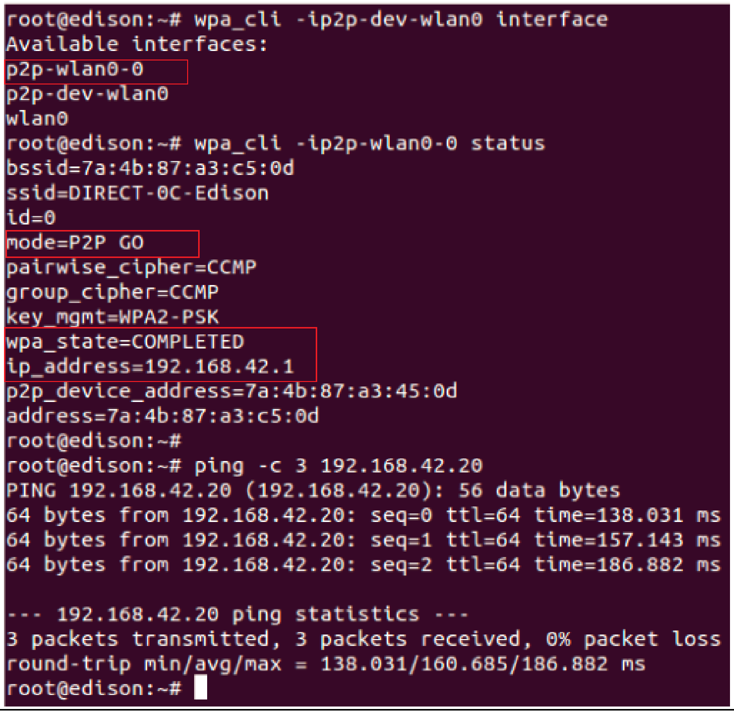
3. Once the connection is established, list the new interface, check the status of the interface, and cross-ping the devices (Figure 29 and Figure 30).

Figure 29 Verifying on the first Intel® Edison device



```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-0
p2p-dev-wlan0
wlan0
root@edison:~# wpa_cli -ip2p-wlan0-0 status
bssid=7a:4b:87:a3:c5:0d
ssid=DIRECT-0C-Edison
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.42.20
p2p_device_address=fe:c2:de:3a:78:ec
address=fe:c2:de:3a:f8:ec
root@edison:~# ping -c 3 192.168.42.1
PING 192.168.42.1 (192.168.42.1): 56 data bytes
64 bytes from 192.168.42.1: seq=0 ttl=64 time=38.692 ms
64 bytes from 192.168.42.1: seq=1 ttl=64 time=37.766 ms
64 bytes from 192.168.42.1: seq=2 ttl=64 time=37.742 ms

--- 192.168.42.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 37.742/38.066/38.692 ms
root@edison:~#
```

Figure 30 Verifying on the second Intel® Edison device



```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-0
p2p-dev-wlan0
wlan0
root@edison:~# wpa_cli -ip2p-wlan0-0 status
bssid=7a:4b:87:a3:c5:0d
ssid=DIRECT-0C-Edison
id=0
mode=P2P GO
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.42.1
p2p_device_address=7a:4b:87:a3:45:0d
address=7a:4b:87:a3:c5:0d
root@edison:~#
root@edison:~# ping -c 3 192.168.42.20
PING 192.168.42.20 (192.168.42.20): 56 data bytes
64 bytes from 192.168.42.20: seq=0 ttl=64 time=138.031 ms
64 bytes from 192.168.42.20: seq=1 ttl=64 time=157.143 ms
64 bytes from 192.168.42.20: seq=2 ttl=64 time=186.882 ms

--- 192.168.42.20 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 138.031/160.685/186.882 ms
root@edison:~#
```

6.6 P2P connection in PBC mode with first device as GO (with group negotiation)

Perform the procedure in section 6.3 to set up a basic peer-to-peer connection so that the peer devices discover each other, then do the following:

1. Start a P2P connection between the peer devices by passing the *pbc* parameter with the *p2p_connect* command. Use the optional *persistent* parameter if you want a persistent group to be formed.

   On the first Intel® Edison device, use a *go_intent* value of 15, which will make the DUT a GO:

   ```
   root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect 7a:4b:87:a3:45:0d
   pbc persistent go_intent=15
   OK
   root@edison:~#
   ```

   On the second Intel® Edison device, use a *go_intent* value of 0, which will make the DUT a client:

   ```
   root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect fe:c2:de:3a:78:ec
   pbc persistent go_intent=0
   OK
   root@edison:~#
   ```

2. Once the connection is established, list the new interface, check the status of the interface, and cross-ping the devices:

   ```
   wpa_cli -ip2p-dev-wlan0 interface
   wpa_cli -ip2p-wlan0-x status
   ping <p2p_ip_address>
   ```

6.7 Terminate and reinvoke a connection

You can terminate a P2P connection with the *p2p_group_remove* command, which removes the P2P group interface and disconnects the connection. (Use the network interface name of the group as an argument for this command.)

Note: As in earlier examples, the first Intel® Edison device is configured as GO (group owner), and the second Intel® Edison device is configured as CLI (client). You can reinvoke a previous connection, but only if you used the optional *persistent* connection parameter when you originally made the connection.

To terminate or reinvoke a connection, do the following:

1. On the first Intel® Edison device, enter the following command to break the connection and remove the group:

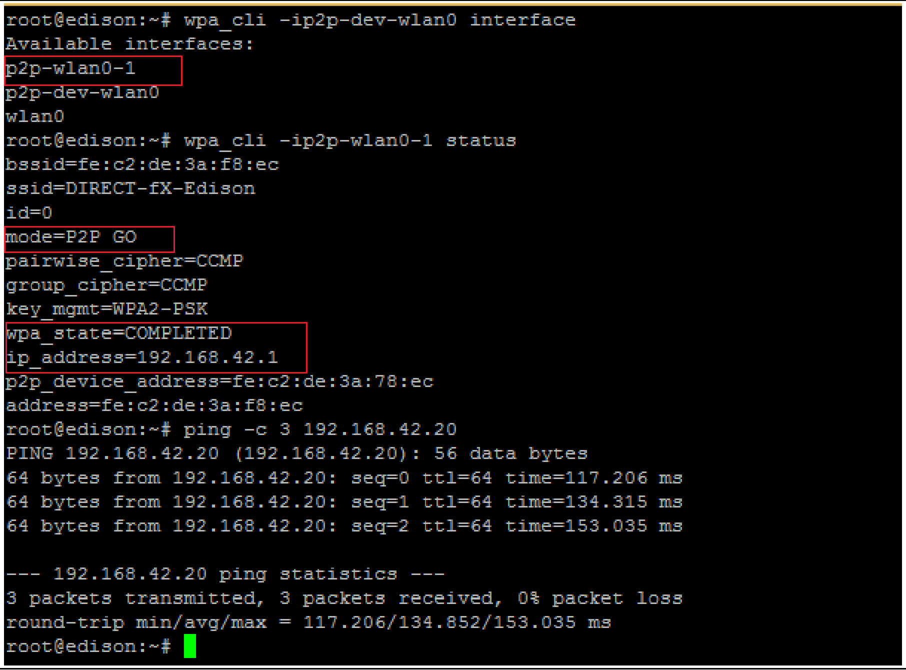
```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_group_remove p2p-wlan0-0
OK
root@edison:~#
```

2. To reconnect, use either the *p2p_find* command or the *p2p_listen* command on the client (the second Intel® Edison device in this case):
   - `wpa_cli -ip2p-dev-wlan0 p2p_find`
   - `wpa_cli -ip2p-dev-wlan0 p2p_listen`
3. On the first Intel® Edison device, reinvoke the connection using *p2p_invite* by passing the MAC address of the discovered peer device.

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_invite persistent=0
peer=7a:4b:87:a3:45:0d
OK
root@edison:~#
```

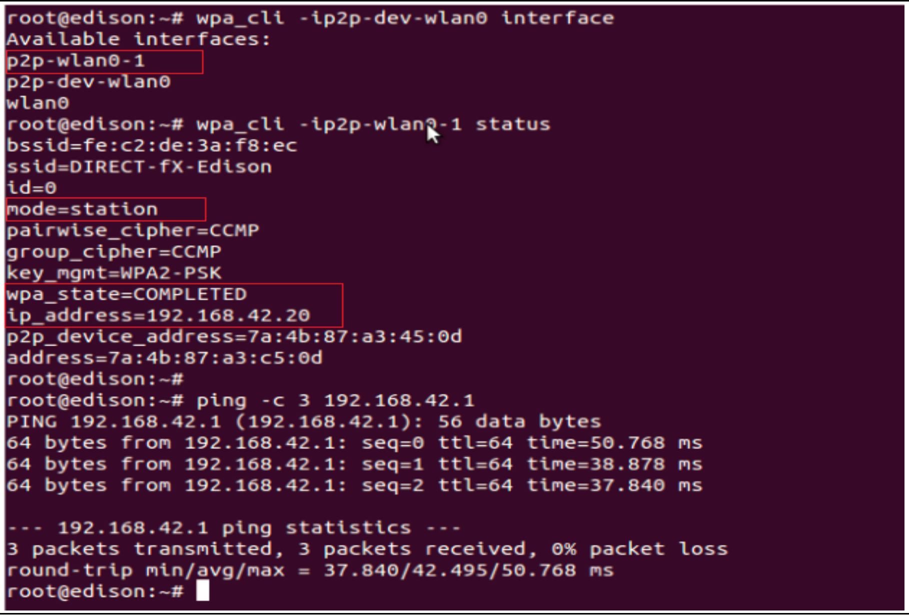
4. Once the connection is established, list the new interface, check the status of the interface, and cross-ping the devices:

Figure 31 Verifying on the first Intel® Edison device



```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-1
p2p-dev-wlan0
wlan0
root@edison:~# wpa_cli -ip2p-wlan0-1 status
bssid=fe:c2:de:3a:f8:ec
ssid=DIRECT-fX-Edison
id=0
mode=P2P GO
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.42.1
p2p_device_address=fe:c2:de:3a:78:ec
address=fe:c2:de:3a:f8:ec
root@edison:~# ping -c 3 192.168.42.20
PING 192.168.42.20 (192.168.42.20): 56 data bytes
64 bytes from 192.168.42.20: seq=0 ttl=64 time=117.206 ms
64 bytes from 192.168.42.20: seq=1 ttl=64 time=134.315 ms
64 bytes from 192.168.42.20: seq=2 ttl=64 time=153.035 ms

--- 192.168.42.20 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 117.206/134.852/153.035 ms
root@edison:~#
```

Figure 32 Verifying on the second Intel® Edison device



```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-1
p2p-dev-wlan0
wlan0
root@edison:~# wpa_cli -ip2p-wlan0-1 status
bssid=fe:c2:de:3a:f8:ec
ssid=DIRECT-fX-Edison
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.42.20
p2p_device_address=7a:4b:87:a3:45:0d
address=7a:4b:87:a3:c5:0d
root@edison:~#
root@edison:~# ping -c 3 192.168.42.1
PING 192.168.42.1 (192.168.42.1): 56 data bytes
64 bytes from 192.168.42.1: seq=0 ttl=64 time=50.768 ms
64 bytes from 192.168.42.1: seq=1 ttl=64 time=38.878 ms
64 bytes from 192.168.42.1: seq=2 ttl=64 time=37.840 ms

--- 192.168.42.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 37.840/42.495/50.768 ms
root@edison:~#
```

6.8 P2P connection with WPS-PBC mode as autonomous GO

In this example, we will configure the first Intel® Edison device as an autonomous group owner. To do so, perform the procedure in section 6.3 to set up a basic peer-to-peer connection so that the peer devices discover each other, then do the following:

1. Connect and form a P2P group with the discovered P2P peer using PBC mode.
2. Manually configure the first Intel ® Edison device as a group owner (GO) with the following command, which will create a new "autonomous GO" interface with the format *p2p-wlan0-x* (where *x* is an integer):

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_group_add
OK
```

3. You can display the newly created interface name with the *wpa_cli -ip2p-dev-wlan0 interface* command:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-0
p2p-dev-wlan0-0
wlan0
root@edison:~#
```

4. When the first Intel® Edison device has been configured as a GO, enter the following command, which starts the WPS PBC method and allows a single WPS enrollee to connect to the GO:

```
root@edison:~# wpa_cli -ip2p-wlan0-0 wps_pbc
OK
root@edison:~#
```

5. On the second Intel® Edison device, enter the following command to make a P2P connection. The *join* parameter indicates that this is a command to join an existing group as a client.

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect fe:c2:de:3a:78:ec
pbc join
OK
root@edison:~#
```

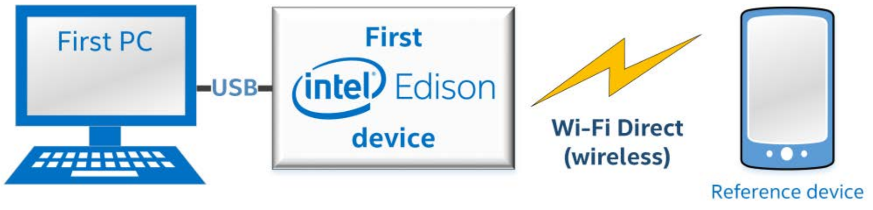
6. Once the connection is established, list the new interface, check the status of the interface, and cross-ping the devices:

```
wpa_cli -ip2p-dev-wlan0 interface
wpa_cli -ip2p-wlan0-x status
ping <p2p_ip_address>
```

6.9 P2P connection using PBC method

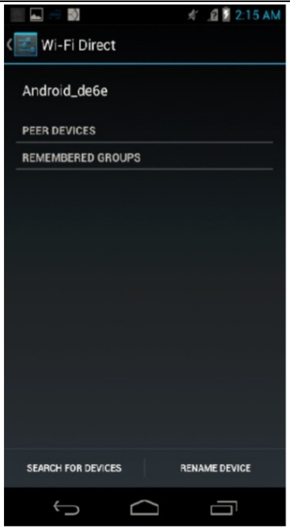
You can create a P2P connection using the PBC method if you want to connect several devices to the network and enable data encryption by pushing a button, either a physical button or a virtual one.

Figure 33 P2P connection between an Intel® Edison device and an Android* device using PBC method



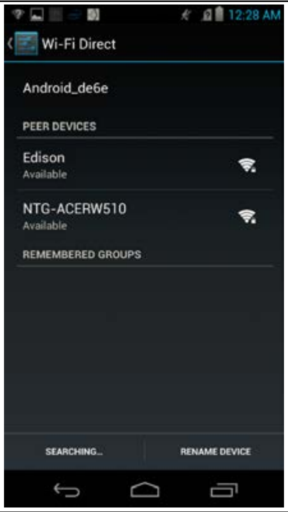
1. Enable Wi-Fi Direct on an Android* device that supports the Wi-Fi Direct feature (Figure 34).

Figure 34 Enabling Wi-Fi Direct on an Android* device



2. Perform the procedure in section 6.3 to set up a basic peer-to-peer connection. On the Android* device, you will see “Edison” in Peer Devices (Figure 35).

Figure 35 Peer devices on an Android*



3. Optionally, you can list the detailed information of the discovered peers with this command:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_peer 02:1e:67:80:a8:3b
  02:1e:67:80:a8:3b
  pri_dev_type=10-0050F204-5
  device_name=Android_de6e
  manufacturer=Intel
  model_name=merrifield
  model_number=R4
  serial_number=11111
  config_methods=0x188
  dev_capab=0x24
  group_capab=0x0
  level=-47
  persistent=8
  age=0
  listen_freq=2437
  wps_method=not-ready
  interface_addr=00:00:00:00:00:00
  member_in_go_dev=00:00:00:00:00:00
  member_in_go_iface =00:00:00:00:00:00
  go_neg_req_sent=no
  go_state=unknown
  dialog_token=0
  intended_addr=00:00:00:00:00:00
  country=00
  oper_freq=0
  req_config_methods=0x0
  flags=[REPORTED]
  status=0
  wait_count=0
  invitation_reqs=0
root@edison:~#
```

4. From the Intel® Edison device, connect with the peered Android* device:

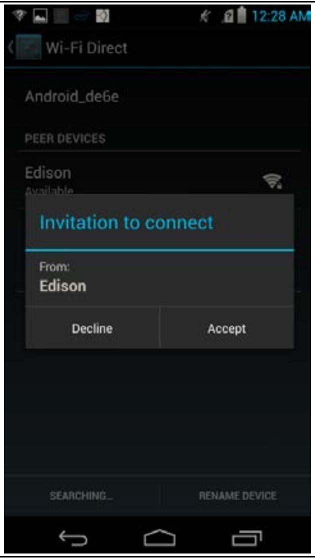
```
wpa_cli -ip2p-dev-wlan0 p2p_connect <discovered_mac_address> pbc
persistent go_intent=<0...15>
```

When using autonegotiation, an Intel® Edison device can be forced to GO for a P2P connection by assigning a value of 15 to the connect parameter *go_intent*. (For *go_intent*, a value of 15 means the device under test (DUT) will become GO; a value of 0 means the DUT will become client.)

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_connect 02:1e:67:80:a8:3b
pbc persistent go_intent=15
OK
root@edison:~#
```

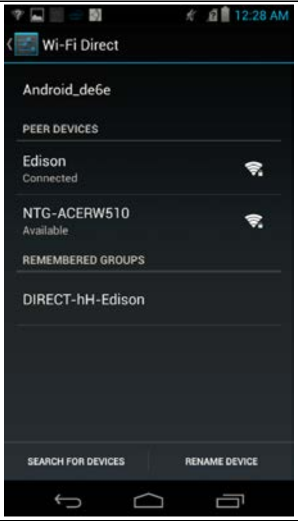
On the Android* device, you will get a notification to connect to the Intel® Edison device (Figure 36).

Figure 36 Android* invitation to connect



5. After you accept the connection on the Android* device, you should see the Intel® Edison device as *Connected* (Figure 37).

Figure 37 Intel® Edison connected

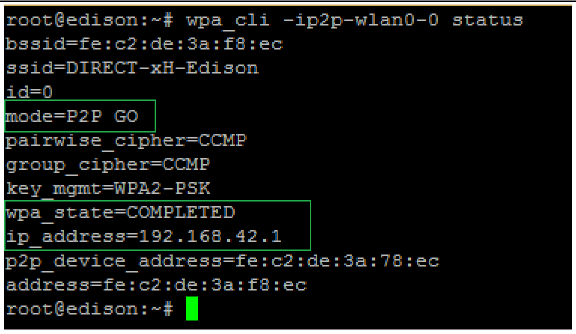


6. Once the connection is established, list the new interface, check the status of the interface, and cross-ping the devices:

```
wpa_cli -ip2p-dev-wlan0 interface
wpa_cli -ip2p-wlan0-x status
ping <p2p_ip_address>
```

7. Optionally, you can query the status with the *wpa_cli -ip2p-wlan0-x status* command (Figure 38).

Figure 38 Query status



```
root@edison:~# wpa_cli -ip2p-wlan0-0 status
bssid=fe:c2:de:3a:f8:ec
ssid=DIRECT-xH-Edison
id=0
mode=P2P GO
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.42.1
p2p_device_address=fe:c2:de:3a:78:ec
address=fe:c2:de:3a:f8:ec
root@edison:~#
```

6.9.1 Disconnecting and reconnecting

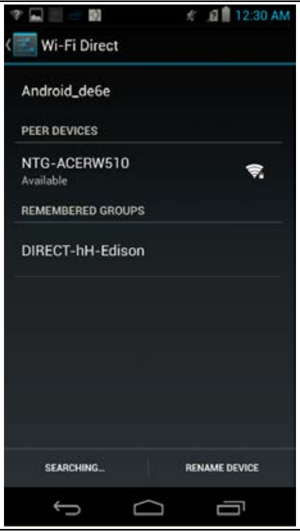
Disconnecting and reconnecting is quick and easy because once-connected Wi-Fi devices remember former P2P connections. To disconnect, do the following:

1. Use the *p2p_group_remove* command to terminate the connection between the Intel® Edison device and the Android* device.

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_group_remove p2p-wlan0-0
```

When you enter this command, the connection breaks and the Intel® Edison device will appear under *Remembered groups* in your Android* device (Figure 39).

Figure 39 Android* remembered groups



2. Reinvoke the connection by passing the peer device's MAC address with the *p2p_invite* command.

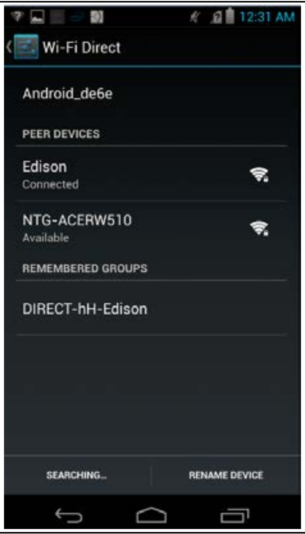
```
root@edison:~# wpa_cli -ip2p-dev-wlan0 p2p_invite persistent=0
peer=02:1e:67:80:a8:3b
OK
root@edison:~#
```

Once the connection is reestablished, you can see the new interface on the Intel® Edison device using:

```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
```

A “locked wireless” connection icon appears on the Android* device (Figure 40).

Figure 40 Android* reconnected


Wi-Fi Direct
Android_de6e
PEER DEVICES
Edison
Connected
NTG-ACERW510
Available
REMEMBERED GROUPS
DIRECT-hH-Edison

6.10 Wi-Fi Direct troubleshooting

When two P2P devices find each other, they alternately listen and scan on channels 1, 6, and 11. If you have trouble finding a peer device or you want to find it faster, you can tell each device independently to listen on a specific channel with the following command:

```
wpa_cli -ip2p-dev-wlan0 p2p_set listen_channel x
```

...where *x* is the channel (1, 6, or 11). We recommend using channel 1.

Once the devices find each other and the connection is triggered, the peer devices negotiate a channel on which the P2P group starts. You can force the P2P group channel by adding *freq=<frequency>* to the *p2p_connect* command, where *<frequency>* is an integer in megahertz. For example:

```
wpa_cli -ip2p-dev-wlan0 p2p_connect 11:22:33:44:55:66 pbc persistent
go_intent=0 freq=2412
```

6.10.1 Cycling power

Because Wi-Fi-Direct is a complex technology, we recommend cycling power on all devices and retrying the Wi-Fi-Direct connection whenever you experience interoperability problems. You can power Intel® Edison devices off/on with these commands:

```
systemctl stop wpa_supplicant
modprobe –r bcm4334x
modprobe bcm4334x
systemctl start wpa_supplicant
```

You can enter all of these commands in interactive mode, by starting `wpa_cli`:

```
wpa_cli –i p2p-dev-wlan0
```

6.10.2 Wi-Fi multirole

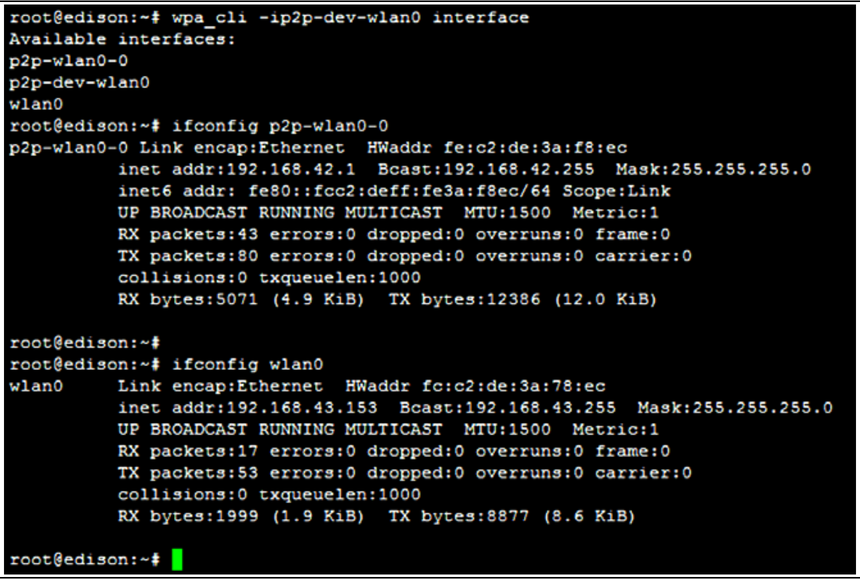
The multirole feature is the capability to have simultaneously a station (STA) connection (to an AP) and a P2P connection (GO role). The BCM-43340 Wi-Fi driver supports this multirole feature.

To enable multirole from a test setup perspective, do the following:

1. Establish a P2P connection by following any of the methods described in this chapter.
2. Make an STA connection by following any of the methods described in this chapter.

   Note: Due to current software limitations, if you first make an STA connection and then try to establish a P2P connection, the P2P connection may fail.
3. When you have a Wi-Fi Direct connection and STA connection to an access point at the same time, you can list all wireless interfaces using the *wpa_cli -ip2p-dev-wlan0 interface* command (Figure 41).

Figure 41 Available wireless interfaces



```
root@edison:~# wpa_cli -ip2p-dev-wlan0 interface
Available interfaces:
p2p-wlan0-0
p2p-dev-wlan0
wlan0
root@edison:~# ifconfig p2p-wlan0-0
p2p-wlan0-0 Link encap:Ethernet  HWaddr fe:c2:de:3a:f8:ec
          inet addr:192.168.42.1  Bcast:192.168.42.255  Mask:255.255.255.0
          inet6 addr: fe80::fcc2:deff:fe3a:f8ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:43 errors:0 dropped:0 overruns:0 frame:0
          TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5071 (4.9 KiB)  TX bytes:12386 (12.0 KiB)

root@edison:~#
root@edison:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr fc:c2:de:3a:78:ec
          inet addr:192.168.43.153  Bcast:192.168.43.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:53 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1999 (1.9 KiB)  TX bytes:8877 (8.6 KiB)

root@edison:~#
```

4. Once the connection is established, list the new interface, check the status of the interface, and cross-ping the devices:

```
wpa_cli -ip2p-dev-wlan0 interface
wpa_cli -ip2p-wlan0-x status
ping <p2p_ip_address>
```

§

7 Additional Configurations

This chapter explains a few miscellaneous configurations.

7.1 Disable power management

Disabling power management might be useful for some debug/performance reasons. You can turn power management off with the following command:

```
$ iwconfig wlan0 power off
```

To disable power management at boot time so that it's always disabled, create a file in the */lib/systemd/system* folder named *wifi-power-management-off.service* with the content as shown in section 9.3.

To disable power management, use the following command:

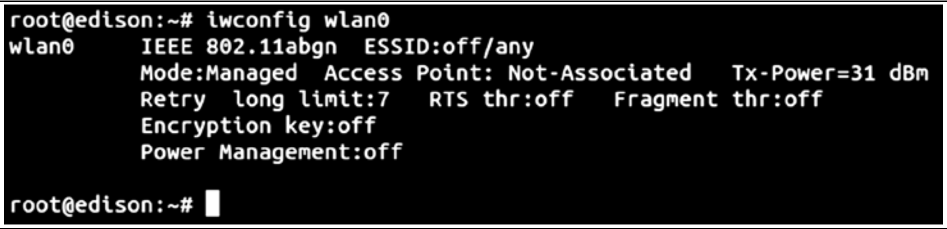
```
$ systemctl start wifi-power-management-off
```

To verify that power management is disabled, use the following command:

```
$ iwconfig wlan0
```

You should see *Power Management:off*.

Figure 42 Power Management status



```
root@edison:~# iwconfig wlan0
wlan0     IEEE 802.11abgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=31 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

root@edison:~# 
```

Note: You can only disable power management after the Intel® Edison device connects to AP; disabling power management *before* the Intel® Edison device connects to AP will have no effect.

7.2 DHCP client vs. static IP address

By default, the DHCP client dynamically assigns an IP address to the *wlan0* interface as soon as the Wi-Fi connection is made:

- When *connman* makes the Wi-Fi connection, it uses the built-in DHCP client.
- When *wpa_cli* makes the Wi-Fi connection, the */etc/wpa_supplicant/wpa_cli-actions.sh* script calls and uses the DHCP client *udhcp*.

When assigning a static IP address, disable the DHCP client to avoid conflicts:

- When *connman* makes the Wi-Fi connection, execute the following command before making the connection:

```
connmanctl>config <service> --ipv4 off
```

- When *wpa_cli* makes the Wi-Fi connection, comment out the following line in the */etc/wpa_supplicant/wpa_cli-actions.sh* script:

```
udhcpc –I $IFNAME  -p /var/run/udhcpc-$IFNAME.pid -S
```

When the DHCP client is disabled and the Wi-Fi connection is complete, you can assign the IP address with the following command:

```
ifconfig wlan0 <ip_address>
```

If you want the static address assigned after an autoconnection following a reboot or Wi-Fi reconnection, do the following:

- When *connman* makes the Wi-Fi connection, execute the following command before making the connection:

  ```
  connmanctl>config <service> --ipv4 manual <ip_address> <netmask> <gateway>
  ```

- When *wpa_cli* makes the Wi-Fi connection, add the following line in the */etc/wpa_supplicant/wpa_cli-actions.sh* script:

  ```
  ifconfig wlan0 <ip_address> (where the call to udhcpc has been made)
  ```

7.3 Autoreconnect after a reboot

With *connman*, once the connection is made, the connection parameters are automatically saved in the specific *connman* folder. In this case, nothing is saved in */etc/wpa_supplicant/wpa_supplicant.conf*.

With *wpa_cli,* by default, the connection parameters are not saved. To save them, you must enter the command:

```
$ wpa_cli –i wlan0 save
```

This adds a new network, along with its connection parameters, in */etc/wpa_supplicant/wpa_supplicant.conf*. At the next reboot, *wpa_supplicant* reads the conf file and tries to connect to the “network” configured.

7.4 Ping latency

When pinging, mainly from AP device to STA device, you can observe a latency. This is related to the overall power management scheme to reduce the power consumption. Hence, when there is minimal data traffic, after some inactivity timeouts, the following events occur:

- The WLAN chipset automatically switches from active mode to passive mode. (This is related to the Wi-Fi power management mentioned above.)
- The SDIO host controller automatically powers down.
- The Intel® Edison CPU is automatically clock-gated.

So when a packet is received, there are some delays to restore the components in an active state for the reception of the packet. This delay is then a relevant part of the ping delay.

Note: Because the interval between two packets during normal data traffic is shorter than the smaller inactivity timeout, the hardware above the components remains in an “active” state, and there is no significant delay per packet.

§

8 Wi-Fi Software Components

8.1 Source folders

These are the Wi-Fi source folders:

• broadcom_cws/wlan/driver_bcm43x:	Source files for the Broadcom Linux Driver.
• wpa_supplicant / wpa_cli:	Source files for wpa_supplicant / wpa_cli 2.1
• linux-kernel/net/wireless:	Source files for the full Mac model defined in mac80211

8.2 Binary files

These are the Wi-Fi configuration/binary files:

• broadcom_cws/wlan/firmware:	Firmware and NVRAM files of Broadcom chipset 43340.
• device-software/meta-edison-distro/recipes-connectivity/ wpa_supplicant/wpa-supplicant/wpa_supplicant.conf-sane:	Configuration file of wpa_supplicant.

8.3 Recipe folders

These are the Wi-Fi recipe folders:

• device-software/meta-edison-distro/recipes-connectivity/ wpa_supplicant/wpa-supplicant:	Recipe folder of wpa_supplicant.
• device-software/meta-edison/recipes-kernel/bcm43340:	Recipe folder for the Broadcom driver and FW/NVRAM.

8.4 Build commands

The *bitbake edison-image* command builds the overall image; you can also rebuild solitary recipes with the following:

```
bitbake <recipe> -c clean -f -v
bitbake <recipe> -c compile -f -v
```

Recipes include:

• bcm43340-mod:	Broadcom driver module bcm4334x.ko
• wpa-supplicant:	wpa_supplicant / wpa_cli

8.5 Files copied onto root of the Intel® Edison file system

The following binary/configuration files are copied onto root of the Intel® Edison file system:

• /etc/firmware/fw_bcmdhd.bin:	Chipset firmware.
• /etc/firmware/bcmdhd.cal:	Chipset NVRAM file for the Murata 43340 module on the Intel® Edison board.
• /etc/modprobe.d/bcm4334x.conf:	Kernel module configuration file used when inserting the kernel driver with the command modprobe.
• /lib/modules/3.10.17-poky-edison\+/extra/bcm4334x.ko:	Driver module.
• /etc/wpa_supplicant/wpa_supplicant.conf:	Configuration file for wpa_supplicant.

8.6 Driver modules

To list driver modules, enter following:

```
$ lsmod
```

To load the Broadcom driver module, enter the following:

```
$ modprobe bcm4334x
```

To remove the Broadcom driver module, enter the following:

```
$ modprobe –r bcm4334x
```

This powers down the chipset, if it is not already powered down.

§

9 Sample Scripts

The scripts in this chapter work on the Intel® Edison board from the */tty/USBx* console or from an SSH connection. To execute these generic scripts through SSH, you must store them in the same folder on your computer that has the *edison_env* file.

9.1 Script to connect to an open AP

The *wpacli_connect_open.sh* script initially configures the wlan0 interface into a disconnected state and removes all networks from the list of configured networks. It also adds a network with the wlan0 interface and configures and sets network variables such SSID (as provided from the command line input) and some other attributes, such as authentication algorithm, key management protocol, and security protocol. It then selects and enables the network with Index 0.

The *wpacli_connect_open.sh* script prompts for one input: the **SSID** of the target Wi-Fi network.

Run the script from the shell command prompt on the Intel® Edison device:.

```
if [ $# != 1 ] ; then
        echo "$0 <SSID>"
        exit
fi

wpa_cli -iwlan0 disconnect
wpa_cli -iwlan0 remove_network all
wpa_cli -iwlan0 add_network
wpa_cli -iwlan0 set_network 0 mode 0
wpa_cli -iwlan0 set_network 0 ssid \"$1\"
wpa_cli -iwlan0 set_network 0 auth_alg OPEN
wpa_cli -iwlan0 set_network 0 key_mgmt NONE
wpa_cli -iwlan0 set_network 0 scan_ssid 1
wpa_cli -iwlan0 select_network 0
wpa_cli -iwlan0 enable_network 0
wpa_cli -iwlan0 reassociate
wpa_cli -iwlan0 status
```

9.2 Script to connect to a secure AP (with WPA2-PSK)

The *wpacli_connect_wpa2.sh* script initially configures the wlan0 interface into a disconnected state and removes all networks from the list of configured networks. It also adds a network with the wlan0 interface, and configures and sets the network variables such SSID, WPA passphrase (as provided from the command line), and some other attributes, such as authentication algorithm, key management protocol, and security protocol. It then selects and enables the network with Index 0.

The *wpacli_connect_wpa2.sh* script prompts for two inputs: the **SSID** and **password** of the target Wi-Fi network.

Run the script from the shell command prompt on the Intel® Edison device:

```
if [ $# != 2 ] ; then
        echo "$0 <SSID> <passphrase>"
        exit
fi

wpa_cli -iwlan0 disconnect
wpa_cli -iwlan0 remove_network all
wpa_cli -iwlan0 add_network
wpa_cli -iwlan0 set_network 0 mode 0
wpa_cli -iwlan0 set_network 0 ssid \"$1\"
wpa_cli -iwlan0 set_network 0 auth_alg OPEN
wpa_cli -iwlan0 set_network 0 key_mgmt WPA-PSK
wpa_cli -iwlan0 set_network 0 proto RSN
wpa_cli -iwlan0 set_network 0 psk \"$2\"
wpa_cli -iwlan0 set_network 0 scan_ssid 1
wpa_cli -iwlan0 select_network 0
wpa_cli -iwlan0 enable_network 0
wpa_cli -iwlan0 reassociate
wpa_cli -iwlan0 status
```

9.3 Script to disable power management

To disable power management at boot time so that it's always disabled, you need to create a file with the name *wifi-power-management-off.service* and save it in the */lib/systemd/system* folder.

The *wifi-power-management-off.service* file should look like this:

```
[Unit]
Description=Disable power management for wlan0
Requires=sys-subsystem-net-devices-wlan0.device
After=sys-subsystem-net-devices-wlan0.device

[Service]
Type=oneshot
ExecStart=/sbin/iwconfig wlan0 power off
```

The *systemd* service manager daemon will invoke this file on bootup, and the service will disable power management for the wlan0 interface.

9.4 Script to create/connect to an ad-hoc network

The *wpacli_ibss_open.sh* script lets you join an IBSS network or create one on channel 1, without security:

```
if [ $# != 1 ] ; then
        echo "$0 <SSID>"
        exit
fi

$rmt wpa_cli -iwlan0 disconnect
$rmt wpa_cli -iwlan0 remove_network all
$rmt wpa_cli -iwlan0 add_network
$rmt wpa_cli -iwlan0 set_network 0 frequency 2412
$rmt wpa_cli -iwlan0 set_network 0 mode 1
if [  ! -n "$rmt" ] ; then
$rmt wpa_cli -iwlan0 set_network 0 ssid \"$1\"
else
$rmt wpa_cli -iwlan0 set_network 0 ssid '\"'$1'\"'
fi
$rmt wpa_cli -iwlan0 set_network 0 auth_alg OPEN
$rmt wpa_cli -iwlan0 set_network 0 key_mgmt NONE
$rmt wpa_cli -iwlan0 set_network 0 scan_ssid 1
$rmt wpa_cli -iwlan0 select_network 0
$rmt wpa_cli -iwlan0 enable_network 0
$rmt wpa_cli -iwlan0 status
```

§