RS9113 n-Link® Software

Technical Reference Manual

Version 1.5.0

August 2016

Redpine Signals, Inc.
2107 N. First Street, #540
San Jose, CA 95131.
Tel: (408) 748-3385
Fax: (408) 705-2019
Email: info@redpinesignals.com
Website: www.redpinesignals.com

Disclaimer:

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only. Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

About this Document

This documents describes about the usage of the RS9113 n-Link® Driver for Wi-Fi, Bluetooth and ZigBee, including Driver Installation, Operation, Wi-Fi ioctl usage, Bluetooth hcitool usage and integration of the driver with specific processor platforms. The RS9113 n-Link® Software is named as OneBox-Mobile.

Table Of Contents

Table of Figures

Table of Tables

1 Introduction

The software provided for the RS9113 n-Link® modules is named as OneBox-Mobile. This software currently supports Wi-Fi (Access Point, Client, Wi-Fi-Direct (P2P), Sniffer and Monitor modes), Bluetooth Classic, Bluetooth Low Energy and ZigBee modes.

OneBox-Mobile Coexistence software supports the following combination of modes. They are as follows:

1. Wi-Fi only mode
2. Wi-Fi + Bluetooth Classic mode
3. Wi-Fi + Bluetooth Low Energy mode
4. Wi-Fi + ZigBee mode

Note:

The standard software package offers Coexistence modes such as modes 2, 3, and 4 as mentioned above, only when Wi-Fi is configured for Client mode operation. For other combinations, custom packages can be offered. For more details contact Redpine.

The subsequent sections explain the use of OneBox-Mobile software on an x86 platform. The Installation and operation of the driver on specific representative processor platforms have been explained in the Appendix sections.

2 Getting Started

This section lists the hardware and software requirements and the steps which is to be followed in order to get started with the OneBox-Mobile software.

2.1 Hardware Requirements

The Hardware requirements are as follows:

- RS9113 n-Link® Module
- Laptop/PC with SDIO or USB interface or any embedded platform with Linux Board support package.

Note:

If the Laptop/PC does not have an SDIO slot, a SDHC/SD/MMC to CardBus Adapter like the one available at http://www.hwtools.net/cardreader/SDCBA_C01.html can be used.

2.2 Software Requirements

The Software requirements are as follows:

- Linux with kernel version 2.6.35 and above – should enable the open source SDIO stack.
- DHCP Server (for Wi-Fi Access Point mode)
- Bluetooth Manager Application (for Bluetooth Classic and Low Energy modes)
- Compatible Bluetooth Host Stack, e.g., the Open Source BlueZ Stack v4.101
- ncurses and ncurses-devel libraries

Note:

- The OneBox-Mobile software has been tested up to kernel version 4.1.15
- For kernel versions 3.13 to 3.16, refer to the section on **18Appendix A: Configuration of Kernels 3.13 to 4.1.15** to ensure correct kernel configuration.
- User has to ensure the following flags are enabled in the Linux kernel configuration
  - CONFIG_WIRELESS=y
  - CONFIG_WIRELESS_EXT=y

2.3 Host Memory Requirements

Following are the memory requirements for the Host platform / Embedded board on which OneBox-Mobile software has to be run.

1) Ram size (Minimum 128MB)
2) CPU frequency (Minimum 400Mhz)

2.4 Software Package Contents

The OneBox-Mobile Software is delivered as a tarball named in the format RS9113.NXX.NL.GEN.LNX.x.y.z.tgz, where:

NXX – defines whether the package supports only Wi-Fi (N00) or Bluetooth Classic/Low Energy along with Wi-Fi (NB0) or ZigBee along with Wi-Fi (N0Z) or Bluetooth Classic/Low Energy and ZigBee along with Wi-Fi (NBZ).

x.y.z – identifies the software package.

The package contains the following files/folders:

- Readme.txt
- Releasenotes.txt
- Documents
- Binary_files (optional)
- source (optional)

Either of the Binary files or source folders might be empty depending on the request we have sent to Redpine and whether we have signed into a Software License Agreement for the source code.

If the source code has been provided, follow the instructions provided in the Compiling the Driver section.

3 Compiling the Driver

This section lists the steps to be followed to compile the OneBox-Mobile software for different platforms.

The steps are as follows:

1. Save the required configuration of Driver using the menuconfig utility.
   - Following are the options available in menuconfig:
     * Host Interface: SDIO or USB.
     * Operating system: Linux or Android
     * Nl80211 support
     * Hostapd Support
     * WIFI
     * BLUETOOTH
     * ZIGBEE

Below is the command to open menuconfig utility:

```
# make menuconfig
```

The given below images explains about the steps in using the menuconfig utility.

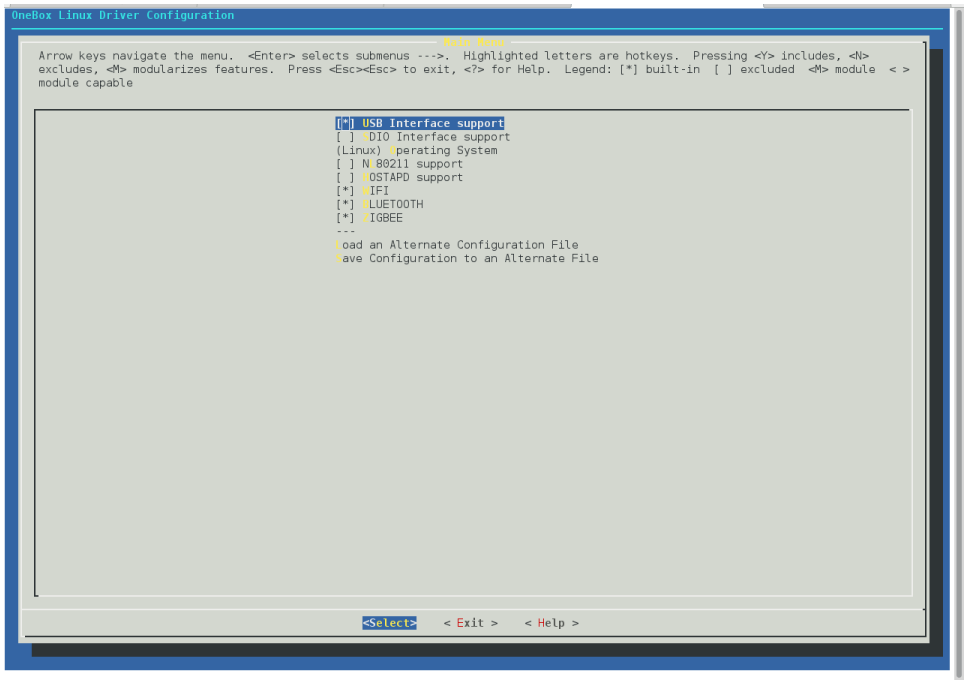

Figure 3-1: Main Page of menuconfig

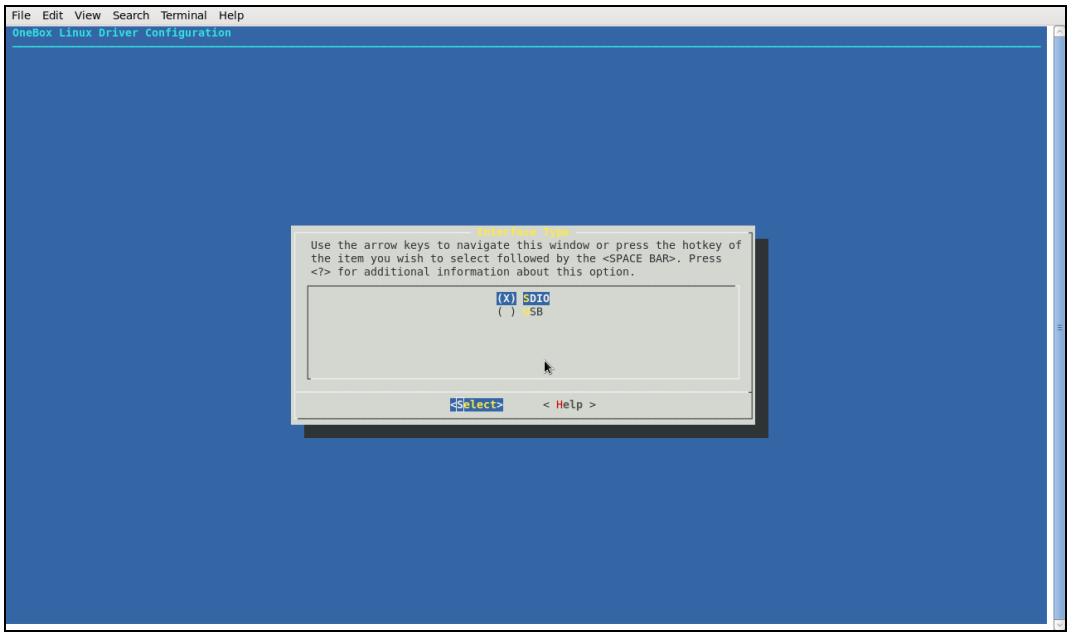

Figure 3-2: Selecting Host Interface

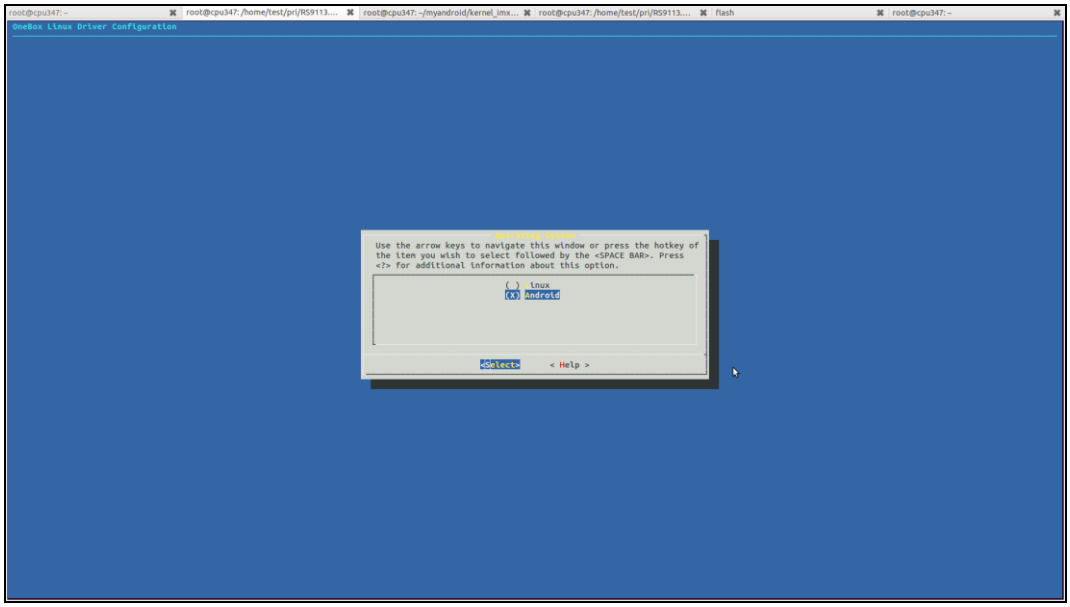

Figure 3-3: Selecting Operating System

By default, the driver package comes with “BSD” support. In case the user needs “Nl80211” support for Access point and Station modes, select the **menuconfig** accordingly.

“Hostapd” application is used as a configuration utility in case of AP mode with Nl80211 support.

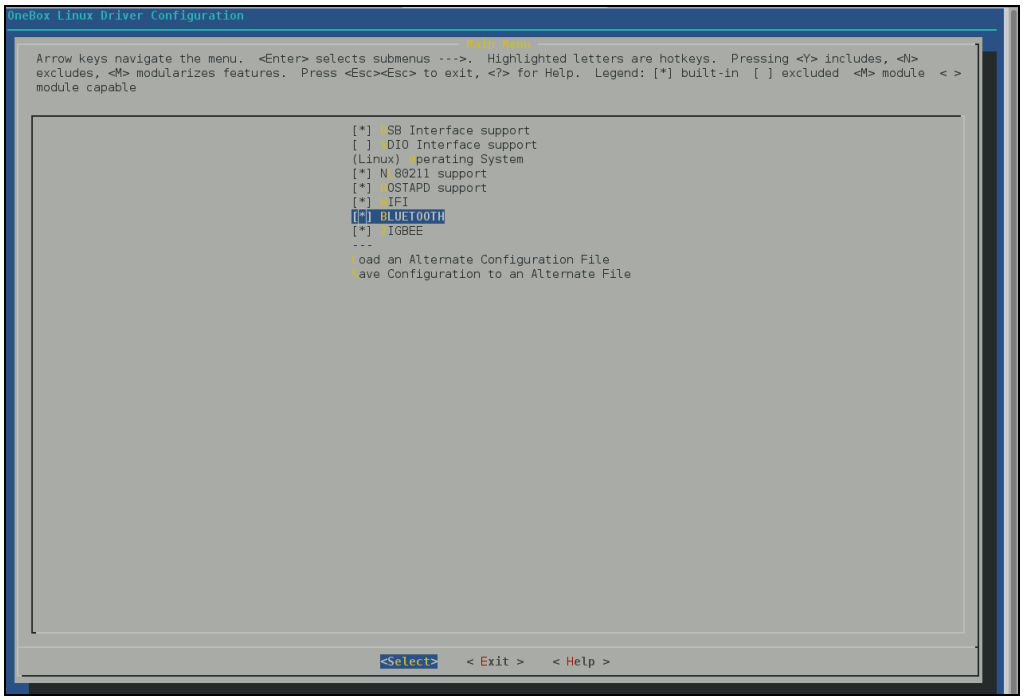

Figure 3-4: Selection of NL80211 and Hostapd Support

Note:

In case, if NL80211 support is enabled in the driver, make sure the that the following modules are loaded in the kernel prior running the driver in order to avoid module dependencies.

- modprobe cfg80211
- modprobe bluetooth

By default the configuration is enabled with WIFI, Bluetooth and ZigBee. If the User wants to compile driver for a particular Protocol, he can disable the unwanted protocols in **Menuconfig**.

In case of coex mode , The Wi-Fi must be enabled in conjunction with BT/ZigBee protocols.

For example, If the user wants to compile Driver only for Wi-Fi, then he can disable Bluetooth and ZigBee. Please refer the given below image of Menuconfig for the mentioned configuration:

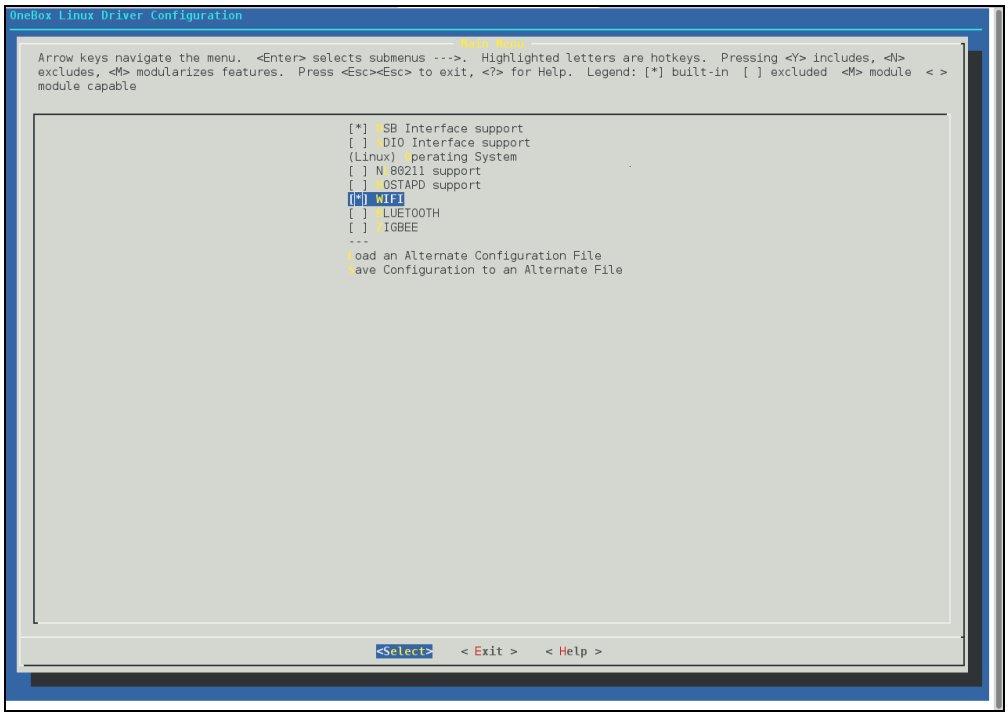

Figure 3-5: Selection of WIFI Only Mode

After Selecting the configuration, exit the menuconfig and save the configuration. Please refer the given below image of saving the configuration.

Figure 3-6: Save the changes before exiting

Now to compile the driver, give the following command:

```
# make
```

The code is compiled and the binaries are generated in the source/host/release folder.

For embedded platforms, modify the path assigned to the **"DEF_KERNEL_DIR"** variable in the Makefile:

```
# cd RS9113.NXX.NL.GEN.LNX.x.y.z/source/host
# vim Makefile
```

The DEF_KERNEL_DIR variable has to be assigned with the path of the compiled kernel path. For an x86 based Linux platform, this path is usually ***"/lib/modules/<kernel_version>/build"*** and this is the path assigned in the **Makefile** provided in the package.

Example:

```
DEF_KERNEL_DIR:= /lib/modules/3.4.66/build
```

Next, use the "make" command to start compiling the driver. For embedded platforms, add the target platform and toolchain path as cross compile option to the "make" command.

For example, if the target platform is ARM and tool chain path is ***"/opt/freescale/usr/local/gcc-4.4.4-glibc-2.11.1-multilib-1.0/arm-fsl-linux-gnueabi/bin/arm-none-linux-gnueabi-"***, then the command is issued as :

```
# make ARCH=arm CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.4.4-glibc-2.11.1-multilib-1.0/arm-fsl-linux-gnueabi/bin/arm-none-linux-gnueabi-
```

4 Installing the Driver

4.1 Installation of Modules

After completion of compilation, Driver generates the following modules in release folder. They are as follows:

- onebox_common_gpl.ko
- onebox_gpl.ko
- onebox_nongpl.ko
- onebox_wlan_gpl.ko
- onebox_wlan_nongpl.ko
- onebox_bt_gpl.ko
- onebox_bt_nongpl.ko
- onebox_zb_gpl.ko
- onebox_zb_nongpl.ko
- wlan.ko
- wlan_wep.ko
- wlan_ccmp.ko
- wlan_tkip.ko
- wlan_acl.ko
- wlan_scan_sta.ko
- wlan_xauth.ko

Below is the explanation of the order of loading Modules:

1. Load onebox common gpl module
   - # insmod onebox_common_gpl.ko
2. Load protocol related Modules (Wi-Fi, BT, ZigBee)
3. load common hal Modules (onebox_nongpl.ko and onebox_gpl.ko).

   After loading the required modules follow the steps detailed in section 4.2 to enable required protocol(s).

4.2 Enabling a Protocol:

The given below command is used to enable required protocol(s):

./onebox_util rpine0 enable_protocol $protocol_value

Below are the given possible values of protocol. They are as follows:

- 1 – Enables Wi-Fi only
- 2 – Enables Bluetooth only
- 4 – Enables ZigBee only

- 3 - Enables both Wi-fi+Bluetooth
- 5 – Enables both Wi-fi+ZigBee

4.3 Disabling a Protocol:

The given below command is used to disable required protocol(s):

```
# ./onebox_util rpine0 disable_protocol $protocol_value
```

Below are the given possible values of protocol:

- 1 – Disables Wi-Fi only
- 2 – Disables Bluetooth only
- 4 – Disables ZigBee only
- 3 – Disables both Wi-fi+Bluetooth
- 5 – Disables both Wi-fi+ZigBee

Note:

- If the user selects only Wi-Fi in Menuconfig during the installation of the Driver, use the given below command:
  ```
  # sh wlan_enable.sh
  ```
- If the user selects only Bluetooth in Menuconfig during the installation of the Driver, use the given below Command:
  ```
  # sh bt_enable.sh
  ```
- If the user selects only ZigBee during the installation of the Driver, use the given below command:
  ```
  # sh zigb_enable.sh
  ```
- If Wi-Fi and Bluetooth are selected during the installation of the Driver, use the given below command:
  ```
  # sh wlan_bt_insert.sh
  ```
- If Wi-Fi and ZigBee are selected during the installation of the Driver, use the given below command:
  ```
  # sh wlan_zigb_enable.sh
  ```
- If all the protocols are selected during the installation of the Driver use the given below command:
  ```
  # sh onebox_insert.sh
  ```
- Similarly, for disabling the protocol(s) the following scripts are available:
- If the user wants to disable only WLAN, use the given below command:
  ```
  # sh wlan_disable.sh
  ```
- If the user wants to disable only Bluetooth, use the given below command:
  ```
  # sh bt_disable.sh
  ```
- If the user wants to disable only ZigBee, use the given below command:
  ```
  # sh zigb_disable.sh
  ```
- If the user wants to disable both WLAN and Bluetooth, use the given below command:
  ```
  # sh wlan_bt_disable.sh
  ```
- If the user wants to disable both WLAN and ZigBee, use the given below command:

```
# sh wlan_zigb_disable.sh
```

Note: Disabling of protocol is not recommended when WIFI is operating in AccessPoint mode.

The sections below list down the steps to be followed to install the driver for different modes of operation.

4.4 OneBox-Mobile in Wi-Fi Only Mode

The steps listed in this section start the Wi-Fi Only mode in Client, Access Point and Wi-Fi Direct modes.

1. Open the common_insert.sh file present in the “release” folder using an editor like vim.
2. Ensure that the DRIVER_MODE and COEX_MODE are set as below
   - DRIVER_MODE = 1
   - COEX_MODE = 1 (For Station Mode only/WIFI-Direct)
   - COEX_MODE = 2 (For ACESS POINT)’
   - COEX_MODE = 3 (For Both ACESS POINT and Station Mode)

Note:

For SDIO mode, ensure that the SDIO stack related modules are already inserted in the kernel.

The steps to be followed for this is as follows:

```
# cd release
# sh load_stack.sh
# lsmod
```

Verify that the output of the “lsmod” command lists the sdhci.ko, sdhci_pci.ko, mmc_block.ko and mmc_core.ko modules. This is a one-time process and need not be repeated unless the modules are explicitly removed by the user.

4.4.1 Installation in Wi-Fi Client Mode (with BSD interface support)

The steps to be followed for installation in Wi-Fi Client Mode are as follows:

1. Edit the “sta_settings.conf” file in the “release” folder and enter the parameters of the Wi-Fi network as below.
   - **For Open (non-Secure) mode**

     ```
     network={
     ssid=”<SSID of Access Point>”
     key_mgmt=NONE
     }
     ```

   - **For Open (non-Secure) mode connection to a Hidden SSID**

     ```
     network={
     ssid=”<SSID of Access Point>”
     scan_ssid=1
     ```

```
key_mgmt=NONE
}
```

- For WEP-64 mode

```
network={
ssid="<SSID of Access Point>"
key_mgmt=NONE
wep_key0=XXXXXXXXXX
wep_tx_keyidx=X
}
```

The key can be input either in ASCII or Hexadecimal formats:

ASCII Format: wep_key0="12345"

Hexadecimal Format: `wep_key0=1234567890`

The key index can vary between 0 and 3.

- **For WEP-128 mode**

```
network={
ssid="<SSID of Access Point>"
key_mgmt=NONE
wep_key0=XXXXXXXXXXXXXXXXXXXXXXXXXX
wep_tx_keyidx=X
}
```

The key can be input either in ASCII or Hexadecimal formats:

ASCII Format: `wep_key0="1234567890123"`

Hexadecimal Format: `wep_key0=12345678901234567890123456`

The key index can vary between 0 and 3.

- **For WEP-Shared (64-bit) mode**

```
network={
ssid="<SSID of Access Point>"
key_mgmt=NONE
wep_key0=XXXXXXXXXX
wep_tx_keyidx=X
auth_alg=SHARED
}
```

The key can be input either in ASCII or Hexadecimal formats:

ASCII Format: wep_key0="12345"

Hexadecimal Format: `wep_key0=1234567890`

The key index can vary between 0 and 3.

- **For WPA-PSK (TKIP) mode**

```
network={
ssid="<SSID of Access Point>"
key_mgmt=WPA-PSK
psk=<passphrase specified in the Access Point>
proto=WPA
pairwise=TKIP
group=TKIP
}
```

- For WPA2-PSK (CCMP) mode

```
network={
ssid="<SSID of Access Point>"
key_mgmt=WPA-PSK
psk=<passphrase specified in the Access Point>
proto=WPA2
pairwise=CCMP
group=CCMP
}
```

1. To connect to an Access Point whose SSID is not broadcast, add the following line in the above configurations.

   scan_ssid=1
2. Next, run the "start_sta.sh" script in the "release" folder to load the driver modules and the supplicant and connect to the Access Point specified in the "sta_settings.conf" file.

   # sh start_sta.sh
3. After issuing the above command, a virtual interface with the name "wifi0" will be created. You can view the list of interfaces using the following command:

   # ifconfig –a
4. You can check whether the connection to the Access Point is successful or not by running the following command:

   # iwconfig wifi0

   This command gives the status of the device. If the connection is successful, then the connected Access point SSID along with the MAC address is displayed. If it is not connected to an Access point a message "Not Associated" appears.
5. To view the list of Access Points scanned in each channel, you can run the following command in the "release" folder.

./wpa_cli –i wifi0 scan_results

6. To obtain an IP address using DHCP, start the DHCP client using the given command.

   # dhclient wifi0

4.4.2 Installation in Access Point Mode (with BSD interface support)

The “start_ap.sh” script present in the “release” folder needs to be run with the different configuration files provided in the same folder to install an Access Point in different security modes.

```
# sh start_ap.sh <conf_file>
```

The different configuration files (conf_file) provided are as follows:

- Access Point in Open Mode
  - Configuration File: wpa_supplicant_open.conf
  - This starts an Access Point with the following parameters:
    - SSID: REDPINE_AP
    - Channel 1 of 2.4GHz Band (2412 MHz)
    - Open (non-Secure) mode
- Access Point in WEP-64 Mode
  - Configuration File: wpa_supplicant_wep64.conf
  - This starts an Access Point with the following parameters:
    - SSID: onebox_wep
    - Channel 1 of 2.4GHz Band (2412 MHz)
    - Security Mode: WEP-64
    - WEP Key: 1234567890
    - Key Index: 0
- Access Point in WEP-128 Mode
  - Configuration File: wpa_supplicant_wep128.conf
  - This starts an Access Point with the following parameters:
    - SSID: onebox_wep
    - Channel 1 of 2.4GHz Band (2412 MHz)
    - Security Mode: WEP-128
    - WEP Key: 12345678901234567890123456
    - Key Index: 0
- Access Point in WPA-PSK (TKIP) Mode
  - Configuration File: wpa_supplicant_tkip.conf
  - This starts an Access Point with the following parameters:
    - SSID: onebox_tkip
    - Channel 1 of 2.4GHz Band (2412 MHz)
    - Security Mode: WPA-PSK (TKIP)
    - Passphrase: “12345678”
- Access Point in WPA2-PSK (CCMP) Mode
  - Configuration File: wpa_supplicant_ccmp.conf
- This starts an Access Point with the following parameters:

* SSID: onebox_ccmp
* Channel 1 of 2.4GHz Band (2412 MHz)
* Security Mode: WPA2-PSK (CCMP)
* Passphrase: “12345678”

Note:

All the above parameters can be modified in the respective configuration files by the user. The values provided are only for reference. The Access Point does not support WEP-Shared algorithm in the current release.

After running the **“start_ap.sh”** script a virtual interface with the name **“wifi1”** will be created. You can view the list of interfaces using the following command:

```
# ifconfig –a
```

You can check whether the Access Point has been started successfully by running the following command:

```
# iwconfig wifi1
```

This command gives the status of the device. It displays the Access Point’s SSID along with the MAC address and channel frequency. If the Access Point does not start, a message saying **“Exiting: Driver Initialization not completed even after waiting for xxms”** is displayed.

To start a DHCP server, use the commands below.

```
# sh dhcp_server.sh wifi1
```

4.4.3 Installation in Wi-Fi Direct Mode

The **“start_p2p.sh”** script present in the **“release”** folder needs to be run to start the supplicant and install the Wi-Fi Direct mode. The configurable parameters in the p2p.conf file are the listen channel, operating channel and GO Intent. After starting the supplicant the p2p_commands mentioned below should be executed.

- To find other P2P networks

```
# ./wpa_cli –i wifi0 p2p_find
```

- To find other P2P devices in range

```
# ./wpa_cli –i wifi0 p2p_peers
```

- To connect to a P2P network

```
# ./wpa_cli –i wifi0 p2p_connect <BSS ID> pbc go_intent=<intent value>
```

4.4.4 Autonomous GO Mode

The given below command is used to start the device in Autonomous GO mode:

```
# ./wpa_cli –i wifi0 p2p_group_add freq=<channel_freq>
```

The **"channel_freq"** input mentioned in the above command is the center frequency of the Wi-Fi channel in which the GO needs to start[1]. If this parameter is not provided, then the GO starts in the channel specified in the p2p.conf file.

Legacy Wi-Fi clients (non P2P clients) need a passphrase to connect to the P2P group. The command below generates the passphrase for legacy Wi-Fi clients.

```
# ./wpa_cli –i wifi0 p2p_get_passphrase
```

4.4.5 Installation in Wi-Fi Client Mode (with NL80211 support)

The steps for installing Wi-Fi Only mode in Client are as follows:

1. Open the common_insert.sh file present in the "release" folder using an editor like vim.
2. Ensure that the DRIVER_MODE and COEX_MODE are set as below
   - DRIVER_MODE = 1
   - COEX_MODE = 1 (For Station Mode only/WIFI-Direct)

     or

     COEX_MODE = 3 (For Both ACESS POINT and Station Mode)

Note:

For SDIO mode, ensure that the SDIO stack related modules are already inserted in the kernel. Follow the steps below for this:

cd release

sh load_stack.sh

lsmod

Verify that the output of the "lsmod" command lists the sdhci.ko, sdhci_pci.ko, mmc_block.ko and mmc_core.ko modules. This is a one-time process and need not be repeated unless the modules are explicitly removed by the user.

Ensure that in menuconfig, NL80211 support is enabled as mentioned in Figure-4.

- Compile the driver.
  - $ make
- Go to the release folder and start the device in station mode.
  - $ cd release
  - $ sh wlan_enable.sh or wlan_bt_enable.sh or wlan_zigb_enable.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1

- Issue the following command to get physical interfaces on which we can add wifi0 interface
  - $iw phy | grep phy

    output of the command will be phyX (X can be 1,2,3,… eg:phy1,phy2 etc)

[1] The OneBox-Mobile software supports DFS slave mode. However, DFS Channels need to be avoided till the module is certified for DFS.

Note:

In case of multiple phy's to identify the appropriate phy on which to run the command use the following command.

iw dev

The sample output of this command is

phy#3

Interface wlp0s26u1u2

ifindex 10

wdev 0x300000001

addr 00:23:a7:65:2a:ac

type managed

phy#0

Interface wlo1

ifindex 3

wdev 0x1

addr a4:17:31:a7:82:a3

type managed

In the above example "Phy3" will be picked as Redpine's Mac address starts with "00:23:a7:x:x:x"

Assuming the physical interface is detected as phy1, refer the below steps to create a virtual interface.

- Adds the wireless interface to the phy.
  - $service NetworkManager stop
  - $iw phy phy1 interface add wifi0 type managed

Instead of following the above 2 steps, we can directly create vap by using "Onebox_util" binary present in the release folder

cd release

./onebox_util rpine0 create_vap wifi0 sta sw_bmiss

Run the supplicant after configuring sta_settings.conf with required AP settings as mentioned in Installation in Wi-Fi Client Mode (with BSD interface support)

$ ./wpa_supplicant -i wifi0 -D nl802ll -c sta_settings.conf -ddddt > log &

4.4.6 Installation in Wi-Fi AP mode (with Nl80211 support)

The steps for installing Wi-Fi Only mode in AP are as follows:

1. Open the common_insert.sh file present in the "release" folder using an editor like vim.

2. Ensure that the DRIVER_MODE and COEX_MODE are set as below
   - DRIVER_MODE = 1
   - COEX_MODE = 2 (For ACESS POINT)'

     (Or)

     COEX_MODE = 3 (For Both ACESS POINT and Station Mode)

Ensure that in menuconfig, NL80211 support is enabled.

- Compile the driver.
  - $ make
- Go to the release folder and start the device in station mode.
  - $ cd release
  - $ sh wlan_enable.sh or wlan_bt_enable.sh or wlan_zigb_enable.sh or onebox_insert.sh script present in the "release" folder as per the instructions present in section Installation of Modules.

- Issue the following command to get physical interfaces on which we can add wifi0 interface
  - $iw phy | grep phy

output of the command will be phyX (X can be 1,2,3,… eg:phy1,phy2 etc)

- Now add wifi0 interface to phyX.
  - $service NetworkManager stop
  - $iw phy phy1 interface add wifi0 type __ap

Instead of following the above 2 steps, we can directly create vap by using "Onebox_util" binary present in the release folder

cd release

./onebox_util rpine0 create_vap wifi0 ap.

Configure the SSID Settings of the AP in the hostapd_open.conf file (say if you are starting AP in open mode).

Make sure in hostapd_open.conf file AP netdevice name is set to wifi0/wifi1 according to the interface obtained by following the above steps.

For eg:

- Interface = wifi0

Run hostapd with following command

- $ ./hostapd hostapd_open.conf –ddddt > log &

Note:

In the same way we can also configure required SSID and Passphrase and key management settings in hostapd_ccmp.conf,hostapd_wep.conf, hostapd_tkip.conf files accordingly.

4.5 OneBox-Mobile in Wi-Fi + Bluetooth Classic Coexistence Mode

This section explains about the installation of Wi-Fi and BT Classic modes. Note that to use the Coexistence mode, each protocol should be loaded individually one after the other.

- Open the common_insert.sh file present in the “release” folder.
- Ensure that the DRIVER_MODE and COEX_MODE as set as below
  - DRIVER_MODE = 1
  - COEX_MODE = 5(For WLAN Station and BT Classic Mode)
  - COEX_MODE = 6 (For WLAN Access Point and BT Classic Mode)

Note:

For SDIO mode, ensure that the SDIO stack related modules are already inserted in the kernel. Follow the steps as given below:

```
# cd release
# sh load_stack.sh
# lsmod
```

Verify that the output of the **“lsmod”** command lists the sdhci.ko, sdhci_pci.ko, mmc_block.ko and mmc_core.ko modules. This is a one-time process and need not be repeated unless the modules are explicitly removed by the user.

1. Follow the instructions in **4.4.1Installation in Wi-Fi Client Mode**, to install the Wi-Fi Client mode.
2. Run the **“bt_enable.sh”** or wlan_bt_insert.sh or onebox_insert.sh script present in the “release” folder as per the instructions in Section 4.1 Installing the Driver to start the Bluetooth Classic mode. This script inserts Bluetooth modules and common HAL modules if not already inserted.
3. You can check whether the BT Classic mode has been started successfully by running the following command:

   ```
   # hciconfig
   ```

   If the driver is loaded correctly, the above command displays a network adaptor named **“hciX”**. An example output if given below:

   ```
   hci0:    Type: BR/EDR  Bus: SDIO
   BD Address: 00:23:A7:00:05:68  ACL MTU: 1021:8  SCO MTU: 30:8
   UP RUNNING PSCAN
   RX bytes:478 acl:0 sco:0 events:20 errors:0
   TX bytes:331 acl:0 sco:0 commands:19 errors:0
   ```

4. After the device is up, we can pair it with the other devices or from other devices using the Bluetooth Manager application. Files can also be sent and received using Bluetooth

Manager. Instead of Bluetooth Manager, the device can be configured using **"hcitool"** or **"hciconfig"**. The procedure for using Bluetooth Manager is explained in the **20Appendix C: Using the Bluetooth Manager** section.

4.6 OneBox-Mobile in Wi-Fi + Bluetooth LE Coexistence Mode

This section describes the installation of Wi-Fi and Bluetooth LE (BLE) modes. Note that to use the Coexistence mode, each protocol should be loaded individually one after the other.

- Open the common_insert.sh file present in "release" folder.
- Ensure that the DRIVER_MODE and COEX_MODE as set as below
  - DRIVER_MODE = 1
  - COEX_MODE = 9(For WLAN Station and BT LE)

Note:

For SDIO mode, ensure that the SDIO stack related modules are already inserted in the kernel. Follow the steps as given below:

```
# cd release
# sh load_stack.sh
# lsmod
```

Verify that the output of the **"lsmod"** command lists the sdhci.ko, sdhci_pci.ko, mmc_block.ko and mmc_core.ko modules. This is a one-time process and need not be repeated unless the modules are explicitly removed by the user.

- Follow the instructions in section **4.4.1Installation in Wi-Fi Client Mode**, to install the Wi-Fi Client mode.
- Run the bt_enable.sh or wlan_bt_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1to start the Bluetooth LE mode. This script inserts Bluetooth modules and common HAL modules if not already inserted.
- You can check whether the BLE mode has been started successfully by running the following command:

  ```
  # hciconfig
  ```

  If the driver is loaded correctly, the above command displays a network adaptor named "hciX". An example output if given below:

  ```
  hci0:    Type: BR/EDR  Bus: SDIO
  BD Address: 00:23:A7:00:05:68  ACL MTU: 1021:8  SCO MTU: 30:8
  UP RUNNING PSCAN
  RX bytes:478 acl:0 sco:0 events:20 errors:0
  TX bytes:331 acl:0 sco:0 commands:19 errors:0
  ```

1) After the device is up, we can Advertise, Scan and Connect with other BLE devices. The device can be configured using hcitool or hciconfig.

4.6.1 Advertise, Scan, Connect Commands

- Enable Advertise
  - # hciconfig –a <hciX> leadv

- Disable Advertise
  - # hciconfig –a <hciX> noleadv
- Initiate Scan
  - # hcitool -i <hciX> lescan

The above command displays the scan responses and advertising information.

- Master Mode Connected State

Ensure that the remote device is in Advertise mode and then issue the command below.

  - # hcitool –i <hciX> lecc <remote_MAC_Addr>

The “remote_MAC_Addr” parameter above is the MAC address of the remote device, e.g., 00:23:AC:01:02:03.

- Slave Mode Connected State

Ensure that our device is in Advertise mode and then issue command below.

  - # hcitool –i <hciX> lecc <device_MAC_Addr>

The “device_MAC_Addr” parameter above is the MAC address of the Redpine module, e.g., 00:23:AC:01:02:03.

4.7 OneBox-Mobile in Wi-Fi + Bluetooth Classic + Bluetooth LE Coexistence Mode

This section describes the installation of Wi-Fi +Bluetooth Classic and Bluetooth LE modes.

Note that to use the Coexistence mode, each protocol should be loaded individually one after the other.

- Open the common_insert.sh file present in “release” folder.
- Ensure that the DRIVER_MODE and COEX_MODE as set as below
  - DRIVER_MODE = 1
  - COEX_MODE = 14(For WLAN Access Point ,BT Classic and BT LE)
  - COEX_MODE = 13(For WLAN Station ,BT Classic and BT LE)

Note:

For SDIO mode, ensure that the SDIO stack related modules are already inserted in the kernel. Follow the steps below for this:

```
# cd release
# sh load_stack.sh
# lsmod
```

Verify that the output of the “lsmod” command lists the sdhci.ko, sdhci_pci.ko, mmc_block.ko and mmc_core.ko modules. This is a one-time process and need not be repeated unless the modules are explicitly removed by the user.

1. Follow the instructions in **4.4.2Installation in Access Point Mode**, to install the Wi-Fi Access Point mode.
2. Run the bt_enable.sh or wlan_bt_insert.sh or onebox_insert.sh script present in the “release” folder as per the instructions in **Section 4.1** to start the Bluetooth LE mode. This

script inserts both WIFI, Bluetooth modules and common HAL modules if not already inserted.

3. To check whether the Bluetooth Classic and Bluetooth LE mode has been started successfully, run the following command:

   ```
   # hciconfig
   ```

   If the driver has been installed successfully, the above command displays a network adapter named “hciX”. An example output is given below:

   ```
   hci0:    Type: BR/EDR  Bus: SDIO
   BD Address: 00:23:A7:xx:xx:xx  ACL MTU: 1021:8  SCO MTU: 30:8
   UP RUNNING PSCAN
   RX bytes:478 acl:0 sco:0 events:20 errors:0
   TX bytes:331 acl:0 sco:0 commands:19 errors:0
   ```

4. After the device is up, we can Advertise, Inquiry, Scan and Connect with other BT Classic and BLE devices. The device can be configured using hcitool or hciconfig applications.
5. After the device is up, we can pair it with the other devices or from other devices using the Bluetooth Manager application. Files can also be sent and received using Bluetooth Manager. Instead of Bluetooth Manager, the device can be configured using **“hcitool”** or “hciconfig”. The procedure for using Bluetooth Manager is explained in the **20Appendix C: Using the Bluetooth Manager** section.

4.8 OneBox-Mobile in Wi-Fi + ZigBee Coexistence Mode

This section explains about the installation of Wi-Fi and ZigBee (ZB) modes. Note that to use the Coexistence mode, each protocol should be loaded individually one after the other.

1. Open the common_insert.sh file present in “release” using the gvim editor.
2. Ensure that the DRIVER_MODE and COEX_MODE are set as given below

- DRIVER_MODE = 1
- COEX_MODE = 17 (For Wlan Station and ZigBee)

Note:

For SDIO mode, ensure that the SDIO stack related modules are already inserted in the kernel. Follow the given below steps:

```
# cd release
# sh load_stack.sh
# lsmod
```

Verify that the output of the **“lsmod”** command lists the sdhci.ko, sdhci_pci.ko, mmc_block.ko and mmc_core.ko modules. This is a one-time process and need not be repeated unless the modules are explicitly removed by the user.

3. Follow the instructions given in the section **4.4.1Installation in Wi-Fi Client Mode**, to install the Wi-Fi Client mode.
4. Run the **“zigb_insert.sh”** script present in the **“release”** folder to start the ZigBee mode. This script inserts ZigBee modules and common HAL modules if not already inserted.

5. You can check whether the ZigBee mode has been started successfully by running the following command:

   ```
   # ifconfig –a
   ```

   If the driver is loaded correctly, the above command displays a network adapter named **"zigb0"**.

4.8.1 Building and Running the Sample Home Automation Switch Application

To help evaluate the ZigBee mode, a sample Home Automation switch application is made available with the release. You will need a 3rd party ZigBee Coordinator and ZigBee-enabled Light bulb which support the Home Automation Profile. Ensure that the Coordinator and Light bulb are switched on and connected before proceeding further.

4.8.1.1 About the Sample Application

This is the ZigBee Home Automation-defined switch application using Host APIs. This application connects to the light parent and tries to match the simple descriptors by using Match Descriptor command.

After exchanging the simple descriptors, it will send the toggle command to the light continuously.

4.8.1.2 Host API Folder Structure

The folder structure for host API along with sample applications has been given below. This folder structure is available in the **"ZigBee/utils"** folder.

The folders in the ZigBee/utils folder are as follows:

- apis – contains the core APIs and sample application
  - core – contains the host mode API implementation.
  - ref_apps – contains the reference HA switch application.
  - build - contains Makefile to compile core and ref_apps irrespective of reference project.
- reference_projects – contains code related to netlink sockets which is used to communicate with driver.

4.8.1.3 Building and Running the Home Automation Sample Application

The steps for building and running the home automation sample application is as follows:

1. Go to the folder **"ZigBee/utils/reference_projects/src"**
2. Clean the existing builds by giving the following command
   - `# make clean`
3. Build the Home Automation Switch application using the following command
   - `# make switch`
4. run the switch app by giving the following command
   - `# ./rsi_wsc_zigb_app`

4.9 Driver Uninstallation Procedure

The driver can be uninstalled along with the different modules using the scripts provided in the "release" folder.

1. remove_all.sh: Uninstall the complete driver and all modules, including the common HAL modules .

- `# sh remove_all.sh`

4.10 Driver Information

4.10.1 Driver Statistics

Use the command below to view Wi-Fi driver statistics:

```
# cat /proc/rpine<$id>/stats
```

`<$id> Indicates Id of Wi-Fi device. For example if rpine0 is created for module then to view Wi-Fi related statistics related to module then Use the below command:`

```
# cat /proc/rpine0/stats
```

When 2nd usb device is connected to same host then rpine1 will get created, In order to see the Wi-Fi related statistics related to 2nd usb module use the below command:

```
# cat /proc/rpine1/stats
```

This command prints statistics related to the total management packets, total data packets with respect to a given access category sent to/from the driver, buffer full status as well as semi buffer full status, FSM states etc.

4.10.2 Disabling Driver Debug Prints

You may opt to disable the debug prints of the driver appearing on the console using the command below. Ensure that the driver is installed correctly before using this command for SDIO interface.

```
# echo 0x0 > /proc/onebox-hal/debug_zone
```

`For USB interface, the proc name is onebox-mobile$devnum$busnum.`

```
# echo 0x0 > /proc/onebox-hal<$devnum$busnum>/debug_zone
```

5 Wi-Fi ioctl Usage Guide

This section explains about the usage of various ioctl commands in the OneBox-Mobile driver. The user has control over multiple settings such as device settings, radio, aggregation, fragmentation thresholds, power save configurations etc.

5.1 Configuring using Wireless Extensions

"iwconfig" is a generic Linux based wireless tool used to set parameters of a wireless network interface. It may be used in lieu of the Wi-Fi supplicant provided as part of the OneBox-Mobile software. However, care has to be taken to follow the correct sequence of commands while using **"iwconfig"**. Redpine Signals recommends usage of the supplicant provided in the software package.

This section describes the use of **"iwconfig"** in conjunction with the Onebox-Mobile driver. For a detailed description of the tool, refer to the relevant mainn pages in Linux.

"iwconfig" only works when the driver is operating in the 'BSD' mode.

The details of the Access Point to which the n-Link® is connected in Client mode can be viewed using the command below:

```
# iwconfig <vap_name>
```

The table below describes the usage of the command in more detail.

Set ESSID (only in Access Point mode)	
Description	This command is used to set the ESSID or Network Name of the n-Link® Module.
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) SSID Name (string of maximum 32 characters)
Output Parameter	None
Reset Required	Yes. In order to set the ESSID information the virtual interface has to be reset.
Usage	`# iwconfig <vap_name> essid <network_name>`
Example	The commands below reset the VAP to set the ESSID to "Redpine_AP": Issue the below delete and create commands if wifi1 interface is already created and started beaconing else go to step 3.. 1) `# ./onebox_util rpine0 delete_vap wifi1` 2) `# ./onebox_util rpine0 create_vap wifi1 ap` 3) `# iwconfig wifi1 essid Redpine_AP`

	4) # ifconfig wifi1 up (Issue this command to Run the interface after configuring required settings)
Set Channel/Frequency (only in Access Point mode)	
Description	This command is used to set the Channel for the n-Link® module.[2]
Default value	1
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Channel number[3]
Output Parameter	None
Reset required	Yes. In order to set the ESSID information the virtual interface has to be reset.
Usage	# iwconfig <vap_name> freq <channel_no> (OR) # iwconfig <vap_name> channel <channel_no>
Example	Issue the given below delete and create commands only if wifi0 interface is already created and started beaconing or else jump to step 3 . 1) # ./onebox_util rpine0 delete_vap wifi0 2) # ./onebox_util rpine0 create_vap wifi0 ap 3) # iwconfig wifi0 freq 36 4) # ifconfig wifi1 up (Issue this command to Run the interface after configuring required settings)
Set Data Transmit Rate	
Description	This command is used to set the data rate for transmission.[4]

[2]

[3] The OneBox-Mobile software supports DFS slave mode. However, DFS Channels need to be avoided till the module is certified for DFS.

Default value	0 (Auto Rate)
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value as per the mapping below: Auto Rate – 0 1 Mbps – 2 2 Mbps – 4 5.5 Mbps – 11 11 Mbps – 22 6 Mbps – 12 12 Mbps – 24 18 Mbps – 36 24 Mbps – 48 36 Mbps – 72 48 Mbps – 96 54 Mbps – 108 MCS0 – 13 MCS1 – 26 MCS2 – 39 MCS3 – 52 MCS4 – 78 MCS5 – 104 MCS6 – 117 MCS7 – 130
Output Parameter	None
Reset required	No
Usage	`# iwconfig <vap_name> rate <rate_val>`
Set RTS/CTS Threshold (only in Access Point mode)	
Description	This command is used to set the RTS/CTS threshold of the n-Link® Module.

[4] For Access Point mode, this command has to be issued after the Set Mode command if the VAP is started using **"iwconfig"** commands and not using the supplicant provided by Redpine Signals. For Client mode, the Set Mode command is not mandatory.

Default Value	2346
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer between 256 and 2346
Output Parameter	None
Reset Required	No.
Usage	`# iwconfig <vap_name> rts <payload_size>`
Example	The command below sets the RTS/CTS threshold to 1008 bytes: `# iwconfig wifi0 rts 1008`
Set Transmit Power[5]	
Description	This command is used to set the transmit power of the n-Link® Module
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value in dBm
Output Parameter	None
Reset Required	No.
Usage	`# iwconfig <vap_name> txpower <val_in_dBm>`
Example	`# iwconfig wifi0 txpower 10` **Note:** Txpower setting is the minimum value picked from the max regulatory power settings and any user defined value and the maximum the radio can support. So it is not guaranteed that the user defined value gets effected when this settings is done.

Table 1: iwconfig Usage

5.2 Private (Driver-Specific) Commands for Access Point and Client Modes

The **"iwpriv"** command is used to set parameters specific to the OneBox-Mobile software. The table below lists the usage of the **"iwpriv"** command for setting and getting parameters common for the Access Point and Client modes.

[5] If the value of transmit power set in the above command exceeds the maximum allowable power supported by the channel specified by the regulatory domain, then the minimum of the two values shall be used.

Set Short GI	
Description	This command is used to set the Short GI mode of the n-Link® Module.[6]
Default Value	1 (Short GI enabled for 20 MHz Bandwidth)
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer mapped as below: 0 –Disable Short GI 1 –Enable Short GI for 20MHz Bandwidth 2 –Enable Short GI for 40MHz Bandwidth 3 –Enable Short GI for 20MHz and 40MHz Bandwidths
Output Parameter	None
Reset Required	Yes. Refer to the example for the reset process for Client and Access Point modes.
Usage	`# iwpriv <vap_name> short_gi <value>`

Example	The commands given below set the Short GI for 20MHz bandwidth in Client mode and then reset the Client for the command to take effect:
	`# iwpriv wifi0 short_gi 1.`
	`# ./onebox_util rpine0 reset_adapter`
	The commands given below tells about the delete and create the VAP to set the Short GI for 20MHz bandwidth in Access Point mode:
	`#./onbox_util rpine0 delete_vap wifi0`
	`# ./onebox_util rpine0 create_vap wifi0 ap`
	`# iwpriv wifi0 short_gi 1.`
	`# ./wpa_supplicant –i wifi0 –Dbsd –c wpa.conf &`
	Note:
	Issue this ioctl before starting the supplicant.

Get Short GI	

[6] Issue this command before starting the supplicant in Access Point mode.

Description	This command is used to get the value programmed for Short GI mode of the n-Link® Module
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.)
Output Parameter	Integer mapped as below: 0 – Disable Short GI 32 – Enable Short GI for 20MHz Bandwidth 64 – Enable Short GI for 40MHz Bandwidth 96 – Enable Short GI for 20MHz and 40MHz Bandwidths
Reset Required	No.
Usage	`# iwpriv <vap_name> get_short_gi`
Example	The command below gets the Short GI programmed in the module: `# iwpriv wifi0 get_short_gi`
Get Privacy	
Description	This command is used to get the Privacy bit of the n-Link® Module
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.)
Output Parameter	Integer value mapped as below: 0 – Privacy is disabled 1 – Privacy is enabled
Reset Required	No.
Usage	`# iwpriv <vap_name> get_privacy`
Example	The command given below tells about like how to get the Privacy information in the module: `# iwpriv wifi0 get_privacy`
Get Mode	
Description	This command is used to get the Wi-Fi mode

	of the n-Link® Module
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.)
Output Parameter	String mapped as below: '6' – All data rates of 802.11 a/b/g/n are supported. '11AN' – All data rates of 802.11 a/g/n are supported. 802.11b rates are not supported.
Reset Required	No.
Usage	`# iwpriv <vap_name> get_mode`
Example	The command given below tells about like how to get Wi-Fi mode of operation: `# iwpriv wifi0 get_mode`
Set WMM (only in Access Point mode)	
Description	This command is used to enable the WMM (QoS) feature of the n-Link® Module[7]
Default Value	1 (Enabled)
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value mapped as below: 0 – Disable 1 – Enable
Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> wmm <value>`
Example	The command below sets the WMM mode for the module: `# iwpriv wifi0 wmm 1`
Set AMPDU	
Description	This command is used to enable AMPDU Aggregation in the n-Link® Module

[7] Issue this command before starting the supplicant in Access Point Mode.

Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value mapped as below: 0 – Disable AMPDU Aggregation 1 – Enable AMPDU Aggregation for Transmit, disable for Receive 2 – Enable AMPDU Aggregation for Receive, disable for Transmit
Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> ampdu_set <value>`

Example	The command below disables A-MPDU aggregation:
	`# iwpriv wifi0 ampdu_set 0`
	The command below enables A-MPDU aggregation for Transmit:
	`# iwpriv wifi0 ampdu_set 1`

Set Bandwidth	
Description	This command is used to enable or disable 20/40 MHz Bandwidths in the n-Link® Module.[8]
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value mapped as below: 1- Enable only 20MHz 2- Enable only 40MHz 3 – Enable both 20 and 40MHz
Output Parameter	None
Reset Required	Yes. Refer to the example for the reset process for Client and Access Point modes
Usage	`# iwpriv <vap_name> set_htconf <value>`

[8] Issue this command before starting the supplicant in Access Point Mode.

Example	The commands given below is used to delete and create the VAP to set the bandwidth in Access Point mode:
	`# ./onebox_util rpine0 delete_vap wifi0`
	`# ./onebox_util rpine0 create_vap wifi0 ap`
	`# iwpriv wifi0 set_htconf $value`
	`# ./wpa_supplicant –i wifi0 wpa_supplicant_open.conf &`
	Note:
	Issue this ioctl before starting the supplicant.
	The commands given below is used to set the 20MHz bandwidth in Client mode and reset the Client for the command to take effect:
	`# iwpriv wifi0 set_htconf 1`
	`# ./onebox_util rpine0 reset_adapter.`

Set Debug Zone	
Description	This command is used to select the debug zone for Wifi.
Default Value	0x4000
Input Parameters	Zone value. Integer value mapped as follows: 0 – Disable zone. 0x4000 – Error Zone.
Output Parameter	None
Reset Required	No
Usage	# iwpriv <vap name> set_dbg_zone <zone_value>
Example	The following command disables debug zone level. # iwpriv wifi0 set_dbg_zone 0
Set Mgmt Rate	
Description	This command is used to set the mgmt rate

Default Value	0
Input Parameters	value*2
Output Parameter	None
Reset Required	No
Usage	# iwpriv <vap name> mgmt_rate <value*2>
Example	The Following command sets the mgmt rate to 5.5Mbps #iwpriv wifi0 mgmt_rate 11 To disable the mgmt_rate use the below command: #iwpriv wifi0 mgmt_rate 0

Table 2: iwpriv Usage for Access Point and Client Modes

5.3 Private (Driver- Specific) Commands for Access Point Mode

The table below lists the usage of the "iwpriv" command for setting and getting parameters common for the Access Point Mode.

Set DTIM Period	
Description	This command is used to set the DTIM period in the n-Link® Module.[9]
Default Value	1
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value between 1 and 15
Output Parameter	None
Reset Required	Yes. In order to set the DTIM period, the virtual interface has to be reset.
Usage	`# iwpriv <vap_name> dtim_period <value>`

Example	The commands given below is used to reset the VAP and set the DTIM period:
	`# sh remove_all.sh`
	`# sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or`

[9] Issue this command before starting the supplicant.

	onebox_insert.sh script present in the "release" folder as per the instructions in **Section 4.1**
	`# ./onebox_util rpine0 create_vap wifi1 ap`
	`# iwpriv wifi1 dtim_period $value`
	`# ./wpa_supplicant –I wifi1 –Dbsd –c wpa_supplicant_open.conf –ddddt > log &`
	Note:
	Issue this ioctl before starting the supplicant in Access Point.

Get DTIM Period	
Description	This command is used to get the DTIM period in the n-Link® Module.
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.)
Output Parameter	Integer value between 1 and 15
Reset Required	No.
Usage	`# iwpriv <vap_name> get_dtim_period`
Example	The command given below is used to get the DTIM period programmed in the module: `#iwpriv wifi0 get_dtim_period`
Get Beacon Interval	
Description	This command is used to get the Beacon Interval programmed in the n-Link® Module
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.)
Output Parameter	Integer value
Reset Required	No.
Usage	`# iwpriv <vap_name>`

	`get_bintval`
Example	The command given below is used to get the Beacon interval programmed in the module: `#iwpriv wifi0 get_bintval`
MAC Command	
Description	This command is used to set the Access Policy based on MAC address. The Access Policy can be disabled or used to Allow or Deny traffic from the MAC address.[10] Note: All the acl policy commands need to be issued before starting the wpa_supplicant.
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value mapped as follows: 0 – Disable Access Policy 1 – Enable Access Policy and Allow traffic 2 – Enable Access Policy and Deny traffic
Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> maccmd <value>`

Example	The command below enables the ACL Policy and allows traffic:
	`# iwpriv wifi0 maccmd 1`
	The command below enables the ACL Policy and denies traffic:
	`# iwpriv wifi0 maccmd 2`

Add MAC Address for Access Policy	
Description	This command is used to add a MAC address for the Access Policy in the n-Link® Module.[11]

[10] Issue this command before starting the supplicant.

[11] Issue this command before a Station connects to the module.

Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) 48-bit MAC Address in hexadecimal format with colon separation. e.g., 00:23:A7:01:02:03
Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> addmac <mac_addr>`
Example	The command below adds a MAC Address (10:10:A9:12:13:14) to the ACL Policy: `# iwpriv wifi0 addmac 10:10:a9:12:13:14`
Delete MAC Address from Access Policy list	
Description	This command is used to delete a MAC address from the Access Policy list in the n-Link® Module
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) 48-bit MAC Address in hexadecimal format with colon separation. e.g., 00:23:A7:01:02:03
Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> delmac <mac_addr>`
Example	The command given below is used to delete a MAC Address (10:10:A9:12:13:14) from the ACL Policy: `# iwpriv wifi0 delmac 10:10:a9:12:13:14`
Set Hidden SSID	
Description	This command is used to stop broadcasting of the SSID of the Access Point in the n-Link® Module's beacons and probe responses.[12]

[12] This command has to be issued before starting the supplicant in Access Point mode.

Default Value	0 (Hidden SSID Disabled)
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value mapped as follows: 0 – Disable Hidden SSID (SSID is broadcast) 1 – Enable Hidden SSID (SSID is not broadcast)
Output Parameter	None
Reset Required	Yes. In order to move from/to Hidden SSID mode, the virtual interface has to be reset.
Usage	`# iwpriv <vap_name> hide_ssid <value>`

Example	The command given below is used to start the Access Point in hidden mode:
	`# ./onebox_util rpine0 create_vap wifi0 ap`
	`# iwpriv wifi0 hide_ssid 1`
	`# ./wpa_supplicant –i wifi0 –D bsd –c wpa.conf &`
	Note:
	Issue this ioctl before starting the supplicant.

Set DFS channel to switch to	
Description	This command is used to select a channel to switch to in case of Radar Detection in Access Point mode. This is used only when the bsd driver is used.
Default Value	Disabled (A channel gets picked at random)
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Frequency of the channel to switch to in case of radar detection.
Output Parameter	None
Reset Required	No
Usage	`# iwpriv <vap name> dfs_chan_switch <frequency>`

Example	The following command sets the channel 36 as the channel to switch to. `# iwpriv wifi0 dfs_chan_switch 5180`

Table 3: iwpriv Usage for Access Point Mode

5.4 Private (Driver- Specific) Commands for Client Mode

The table below lists the usage of the **"iwpriv"** command for setting and getting parameters common for the Client Mode.

Deauthenticate while Roaming	
Description	This command is used to de authenticate the n-Link® Module from "old" Access Point while roaming.
Default Value	NULL Data
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value - 0 or 1 0 – Inform Access Point that the module is going to be in power save mode. 1 – De authenticate from the previous Access Point during Roaming.
Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> setparam 12`[13] `<value>`
Example	The command below sends de authentication during roaming. `# iwpriv wifi0 setparam 12 1`
Set Keep Alive Period	
Description	This command is used to set the Keep Alive period in the n-Link® Module.
Default Value	90 seconds
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) Integer value between 1 and 300 (seconds)

[13] The value 12 is used for setting Roaming related parameters for the setparam command.

Output Parameter	None
Reset Required	No.
Usage	`# iwpriv <vap_name> keep_alive <value>`
Example	The command given below sets the Keep Alive period to 100 seconds: `# iwpriv wifi0 keep_alive 100`

Table 4: iwpriv Usage for Access Point Mode

5.5 Configuring Using onebox_util

The "onebox_util" program is provided to configure the n-Link® module for parameters which are not specific to a virtual interface (VAP). The table below lists the usage of the "onebox_util" command for setting and getting the parameters.

Create a VAP[14]	
Description	This command is used to create a virtual interface (VAP) in the operating mode specified.
Default Value	-
Input Parameters	Base Interface (string like rpine0) VAP Name (string like wifi0, wifi1, etc.) Operating Mode (string): ap – Access Point Mode sta – Station/Client Mode p2p – P2P Mode mon – Monitor Mode[15] Beacon Filtering after connecting to an Access Point (only for Client mode). Valid inputs are: sw_bmiss – Beacon filtering disabled. All beacons of connected Access Point provided to Host driver. hw_bmiss – Beacon filtering enabled. Beacon is provided to Host driver when there is a change in the Beacon from the connected Access Point. This feature also programs the device to indicate to the Host driver when 20 consecutive beacons are not received by the device.

[14] The OneBox-Mobile software allows creation of 4 VAPs
[15] Refer to the section **11Monitor Mode** for more details.

Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> create_vap <vap_name> <op_mode>`
Example	The command given below creates a virtual interface named wifi0 in the Client mode with Beacon filtering disabled. `# ./onebox_util rpine0 create_vap wifi0 sta sw_bmiss`
Delete a VAP	
Description	This command is used to delete an existing virtual interface (VAP).
Default Value	-
Input Parameters	Base Interface (string like rpine0) VAP Name (string like wifi0, wifi1, etc.)
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> delete_vap <vap_name>`
Example	The command given below deletes a virtual interface named wifi0. `# ./onebox_util rpine0 delete_vap wifi0`
Print VAP Statistics	
Description	This command is used to print the statistics of the transmitted and received packets of an existing virtual interface (VAP).
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) [-v] – Get description of the fields in the statistics Filename (string) to which the statistics will be written
Output Parameter	Statistics like: 1) Number of Beacons transmitted (for Access Point/P2P GO modes) 2) Number of Beacons received (for Client/P2P

	Client modes) 3) Number of Management packets received 4) Number of packets received from a different BSS etc.
Reset Required	No.
Usage	`# ./onebox_util <vap_name> print_vap_stats [-v] [-f filename]`
Example	The command given below prints the statistics of the transmitted and received packets of the interface wifi0 into the file "stats" `# ./onebox_util wifi0 print_vap_stats -v stats`
Print Station Statistics (only in Access Point mode)	
Description	This command is used to print the statistics of the packets exchanged between the Access Point and a Station.
Default Value	-
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) 48-bit MAC Address in hexadecimal format with colon separation. e.g., 00:23:A7:01:02:03 [-v] – Get description of the fields in the statistics Filename (string) to which the statistics will be written
Output Parameter	Statistics like: 1) Number of Beacons received 2) Number of Management packets transmitted/received 3) Number of Unicast/Multicast packets transmitted/received 4) Number of data packets transmitted/received 5) Number of Probe Request/Response packets transmitted/received etc.
Reset Required	No.
Usage	`# ./onebox_util <vap_name> print_station_stats <mac_addr> [-v] [-f filename]`

Example	The command below logs the statistics of the packets exchanged between the Access Point (wifi0) and a Station with MAC address 00:1C:2b:10:19:1a into the file named “stats”. `#./onebox_util wifi0 print_station_stats 00:1C:2b:10:19:1a –v stats`
Select Antenna	
Description	This command is used to select one of the two RF ports connecting to antennas. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it is used to select between the two. In case **Antenna Diversity** feature is enabled, this ioctl will not have any effect. Antenna selection will happen automatically at the firmware level. **Note:** This ioctl is redundant, refer to Section 16 for futher details. The functionality of the ioclt is intact, however it might be removed in the future to reduce redundancy.
Default Value	2
Input Parameters	Base Interface (string like rpine0) Integer value mapped as follows: 2 – Select RF_OUT_2/Integrated Antenna 3 – Select RF_OUT_1/U.FL Connector
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> ant_sel <value>`
Example	The command below selects the U.FL connector in case of modules with integrated antenna and the RF_OUT_1 pin the case of module without integrated antenna. `# ./onebox_util rpine0 ant_sel 3`
Enable Background Scan and Set Parameters (only in Client mode)	
Description	This command is used to enable background scan and set the relevant parameters. Refer to the section on Background Scan Parameters for more details on each

	parameter.
Default Value	2
Input Parameters	Base Interface (string like rpine0) Background Scan Threshold RSSI Tolerance Threshold Periodicity Active Scan Duration Passive Scan Duration Two Probe Enable Number of Background Scan Channels Channels to Scan[16]
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> set_bgscan_params <bgscan_threshold> <rssi_tolerance_threshold> <periodicity> <active_scan_duration> <passive_scan_duration> <two_probe_enable> <num_of_bgscan_channels> <channels_to_scan>`
Example	The command below enables Background Scan with a scan threshold of 10, RSSI tolerance threshold of 10, periodicity of 3 seconds, active scan duration of 20 milliseconds, passive scan duration of 100 milliseconds, two-probe enabled and the channels 36, 40 and 44. `# ./onebox_util rpine0 set_bgscan_params 10 10 3 20 100 1 3 36 40 44`
Note	In order to select 11J channels 8,12,16 enter the channel number as 8J, 12J, 16J respectively. Remaining 11J channels can be selected with their channel numbers. Example: # ./onebox_util rpine0 set_bgscan_params 10 10 3 20 100 1 4 36 40 44 8J

[16] The OneBox-Mobile software supports DFS slave mode. However, DFS Channels need to be avoided till the module is certified for DFS.

Host-Triggered Background Scan (only in Client mode)	
Description	This command is used to trigger background scan without waiting for the periodicity mentioned in bgscan_parameters[17].
Default Value	-
Input Parameters	-
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> do_bgscan`
Example	The command below triggers background scan without waiting for periodicity timeout. `# ./onebox_util rpine0 do_bgscan`
Set SSID for Background Scan (only in Client mode)	
Description	This command is used to set the SSID of the Hidden Access Point (SSID not being broadcast) during Background Scan.[18]
Default Value	-
Input Parameters	Base Interface (string like rpine0) SSID (max. 32 characters)
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> bgscan_ssid <ssid>`
Example	The command below sets the SSID of a Hidden Access Point during Background Scan. `# ./onebox_util rpine0 bgscan_ssid REDPINE_AP`
Enable Power Save and Set Parameters (only in Client mode)	
Description	This command is used to enable/disable power save modes and set the required power save mode for the n-Link® module. Refer to the section on Power Save Modes, Profiles and Parameters for more details on

[17] The do_bgscan command has to be followed by set_bgscan_params command.

[18] The bgscan_ssid command has to be followed by the set_bgscan_params or do_bgscan command in order for the Probe Request to be sent with the SSID requested in the bgscan_ssid command.

	each parameter and their usage.
Default Value	-
Input Parameters	Base Interface (string like rpine0) Power Save Enable/Disable Sleep Type Transmit Threshold Receive Threshold Transmit Hysteresis Receive Hysteresis Monitor Interval Sleep Duration Listen Interval Duration Number of Beacons per Listen Interval DTIM Interval Duration Number of DTIMs Per Sleep Duration
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> set_ps_params <ps_en> <sleep_type> <tx_threshold> <rx_threshold> <tx_hysteresis> <rx_hysteresis> <monitor_interval> <sleep_duration> <listen_interval_duration> <num_beacons_per_listen_interval> <dtim_interval_duration> <num_dtims_per_sleep>`
Example	The command below enables ULP Power Save Mode for a duration of 100 ms and with a listen_interval_duration of 100ms. `# ./onebox_util rpine0 set_ps_params 1 2 0 0 0 0 0 100 100 0 0 1`
Enable UAPSD (Normal and Mimic modes) and Set Parameters	
Description	This command is used to enable the UAPSD mode and set the relevant parameters. If the Access Point does not support UAPSD, the module tries to mimic this

	mode. Refer to the section on **Power Save Modes, Profiles and Parameters** for more details.[19]
Default Value	-
Input Parameters	Base Interface (string like rpine0) UAPSD Wakeup Period in milliseconds – 0 for Transmit Based UAPSD and between 10 and 100 for Periodic UAPSD[20].
Output Parameter	None
Reset Required	No.
Usage	`# ./onebox_util <base_interface> set_uapsd_params 0xF <uapsd_wakeup_period>`
Example	The command enables UAPSD mode and sets the wakeup period as 100ms. `# ./onebox_util rpine0 set_uapsd_params 0xF 100`
Reset Adapter (only in Client mode)	
Description	This command is used to reset the Client mode virtual interface. This command can be used to change certain configurations of the Client mode and reset the VAP for the configurations to take effect.
Default Value	-
Input Parameters	Base Interface (string like rpine0) – the base interface input ensures that the Client mode VAP is reset irrespective of the actual VAP name.
Output Parameter	-
Reset Required	-
Usage	`/onebox_util <base_interface> reset_adapter`
Example	`/onebox_util rpine0 reset_adapter`
Set Beacon Interval (only in Access Point mode)	
Description	This command is used to set the Beacon Interval in milliseconds. It is recommended that this command is

[19] The set_uapsd_params command needs to be followed by the command below for the values to take effect.

```
# ./onebox_util <base_interface> reset_adapter
```

[20] Refer to the **13Power Save Modes, Profiles and Parameters** section for more details.

	given before the VAP is created.
Default Value	200
Input Parameters	Base Interface (string like rpine0) Integer value between 50 and 1000 (other values will result in default value being assigned).
Output Parameter	None
Reset Required	Yes. In order to set the beacon interval, the virtual interface has to be reset.
Usage	`# ./onebox_util <base_interface> set_beacon_intvl <beacon_intvl>`

Example	The commands given below are used to reset the Access Point and set the beacon interval to 100ms.
	`# sh remove_all.sh`
	`# sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1`
	`# ./onebox_util rpine0 set_beacon_intvl 100`
	`# ./onebox_util rpine0 create_vap wifi0 ap`
	`# ./wpa_supplicant –i wifi0 -D bsd –c wpa.conf -ddddt &`
	Note:
	Issue this command before creating any virtual Access Point interfaces.

Set WMM Parameters (only in Access Point mode)	
Description	This command is used to set the WMM parameters for specific queues. Note: This ioctl is redundant, refer to Section 16 for further details. The functionality of the ioclt is intact, however it might be removed in the future to reduce redundancy.
Default Value	Access Point:

	AIFSN	Cwmin	Cwmax	TxOp
AC_BE	3	4	6	0
AC_BG	7	4	10	0

	<table><tr><td>AC_VI</td><td>1</td><td>3</td><td>4</td><td>94</td></tr><tr><td>AC_VO</td><td>1</td><td>2</td><td>3</td><td>47</td></tr></table> Station: <table><tr><td></td><td>AIFSN</td><td>Cwmin</td><td>Cwmax</td><td>TxOp</td></tr><tr><td>AC_BE</td><td>3</td><td>4</td><td>6</td><td>0</td></tr><tr><td>AC_BG</td><td>7</td><td>4</td><td>10</td><td>0</td></tr><tr><td>AC_VI</td><td>4</td><td>3</td><td>4</td><td>94</td></tr><tr><td>AC_VO</td><td>4</td><td>2</td><td>3</td><td>47</td></tr></table>
Input Parameters	VAP Name (string like wifi0, wifi1, etc.) WMM Parameter Name (string like aifs, cwmin, cwmax, txop, acm) Integer value. The allowed values are as follows: AIFSN – 1 to 15 Cwmin – 2^n-1, where 'n' is between 1 and 4 for BE_Q and BK_Q and between 1 and 3 for VI_Q and VO_Q. Cwmax – 2^n-1, where 'n' is between 1 and 6 for BE_Q, between 1 and 10 for BK_Q and between 1 and 4 for VI_Q and VO_Q. TxOp – 0 for BE_Q, BK_Q, 94 for VI_Q and 47 for VO_Q Access Category (string, as mapped below): 1) VO_Q – Voice data packets queue 2) VI_Q – Video data packets queue 3) BK_Q – Background data packets queue 4) BE_Q – Best effort data packets queue Self or Broadcast selection (Self is for the module's Access Point VAP, Broadcast is for Clients connected to the Access Point) Update Params (integer value mapped as below): 0 – To set more WMM Parameters 1 – To update current WMM Parameters
Output Parameter	None
Reset Required	No.

Usage	`# ./onebox_util <vap_name> setwmmparams <wmm_param_name> <value> <access_category> <self/broadcast> <update_params>`
Example	The command below updates the AIFSN value for BE Access category for connected Clients. `# ./onebox_util wifi0 setwmmparams aifsn 2 BE_Q broadcast 1`
Set Country	
Description	This command is used to set the country for the n-Link® Module.[21]
Default Value	-
Input Parameters	Integer (country code) mapped as below:

REGION	COUNTRY_CODE	COUNTRY_NAME
FCC	**840**	UNITED STATES
	124	CANADA
	484	MEXICO
ETSI	**250**	FRANCE
	56	BELGIUM
	276	GERMANY
	380	ITALY
JAPAN	**392**	JAPAN
WORLD	**36**	AUSTRALIA
	356	INDIA
	364	IRAN
	458	MALAYSIA
	554	NEWZEALAND
	643	RUSSIA
	702	SINGAPORE
	710	SOUTH AFRICA

[21] Issue this command before creating any VAP interface.

Output Parameter	None
Reset Required	Yes. In order to change the country code, the virtual interface has to be reset.
Usage	`# ./onebox_util <base_interface> set_country <country_code>`

Example	The commands below reset the VAP and set the country to Singapore in Station mode.
	`# sh remove_all.sh`
	`# sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in` **Section 4.1**
	`# ./onebox_util rpine0 set_country 702`
	`# ./onebox_util rpine0 create_vap wifi0 sta sw_bmiss`
	Note:
	Issue this command before creating any interfaces.

Set External Antenna Gain	
Description	This command is used to program the gain of the external antenna for the module without antenna. The gain values are used by the module to attenuate the output transmit power so that regulatory requirements like FCC, ETSI, etc., are not violated. This command needs to be given before creating the VAP in the normal mode and before the "./transmit" command in the **Wi-Fi Performance Test** mode. In the Wi-Fi Performance Test mode, the transmission has to be stopped each time before the antenna gain values are programmed.
Default Value	0
Input Parameters	Base Interface (string like rpine0) Integer value for Antenna gain for 2.4 GHz band in dBm Integer value for Antenna gain for 5 GHz band in dBm
Output Parameter	None
Reset Required	No

Usage	`# ./onebox_util <base_interface> set_ext_ant_gain <gain_2g> <gain_5g>`

Example	The commands below set the Antenna gain values for 2.4 GHz and 5 GHz bands to 3 dBm and 5 dBm, respectively.
	`# sh remove_all.sh`
	`# sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in` **Section 4.1**
	`# ./onebox_util rpine0 set_ext_ant_gain 3 5`
	`# ./onebox_util rpine0 create_vap wifi0 sta`

Set Wake-On-Wireless LAN Parameters (only in Client Mode)	
Description	This command is used to set the Wake-On-Wireless LAN (WoWLAN) parameters in the device. The Host has to give this command each time when it enters and exits sleep state. Refer to the section on Wake-On-Wireless LAN Parameters for more details. GPIO_2 is used as a Host Wakeup Interrupt for this purpose.
Default Value	-
Input Parameters	Base Interface (string like rpine0) 48-bit Source MAC Address in hexadecimal format with colon separation. e.g., 00:23:A7:01:02:03 (valid when Unicast packet filtering from specific MAC address is enabled) Host Sleep Status WoWLAN Flags
Output Parameter	None
Reset Required	No
Usage	`# ./onebox_util <base_interface> wowlan <src_mac_addr> <host_sleep_status> <wowlan_flags>`
Example	`# ./onebox_util rpine0 wowlan 00:23:a7:0c:bb:aa 1 3`
Set RF Power Mode	
Description	This command is used to program the RF power mode

	to High, Medium and Low profiles. It has to be issued before creating the VAP. The performance of the RF is best in the High power mode.
Default Value	0 - High
Input Parameters	Integer mapped as below: 0 – High power mode 1 – Medium power mode 2 – Low power mode
Output Parameter	None
Reset Required	No
Usage	`# ./onebox_util <base_interface> set_rf_tx_rx_pwr_mode tx_value rx_value`
Example	`./onebox_util rpine0 set_rf_tx_rx_pwr_mode 0 1`
Set scan type	
Description	This command is used to select the band in which the user wants to perform the scan. Using this command user can either test in 2.4Ghz only or 5Ghz only bands.
Default Value	Both 2.4Ghz and 5Ghz bands are enabled by default. List of possible values 1 – To scan 2.4Ghz only band 2 – To scan 5Ghz only band
Input Parameters	Integer value
Output Parameter	None
Reset Required	No
Usage	`# ./onebox_util <base_interface> set_scan_type value`

Example	`./onebox_util rpine0 set_scan_type 1`
	`The above command performs scan only in 2.4Ghz band.`
	Note:
	Issue this command before creating station virtual interface.

Set Beacon Filter (Only in AP mode)	

Description	This command is used to enable beacon filtering in the firmware . All third party beacons will be filtered at the firmware after applying beacon filter Ioctl.
Default Value	0-Disabled by default.
Input Parameters	Integer as mapped below 0-Disabled beacon filtering 1-Enabled beacon filtering
Output Parameter	None
Reset Required	No
Usage	`# ./onebox_util <base_interface> set_rx_filter 0 0 0 0 <value> 0 0`

Example	`./onebox_util rpine0 set_rx_filter 0 0 0 0 1 0 0`
	`The above command does not allow beacons to be received from firmware to driver in AP mode.`
	Note:
	In the above command BIT(0,1,2,3,,5, 6) are reserved for future use. Only BIT(4) is used for beacon filtering..

Table 5: Usage of onebox util

5.6 WPS Configuration

Wi-Fi Protected Setup (WPS) is a standard for easy and secure wireless network setup and connections. Onebox-Mobile supports the following configuration methods.

- Push Button Method
- PIN Method – Enter and Generate

A WPS Configuration file is used for setting up a connection with a remote Access Point or Station. A sample WPS configuration file is given below for reference.

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
uuid=12345678-9abc-def0-1234-56789abcdef0
device_name=RSI_P2P_DEVICE
manufacturer=Redpine Signals, Inc.
model_name=M2MCombo
model_number=9113
serial_number=03
```

```
device_type=1-0050F204-1
os_version=01020300
config_methods=display push_button keypad
```

The sections below list down the steps for configuring WPS and setting up a connection in Access Point and Client modes using the methods listed above.

5.6.1 Access Point Mode

- Start the driver in Access Point mode
- Start the supplicant with the following command

```
# ./wpa_supplicant –i <vap_name> -D bsd –c <wps_conf_file> -ddddt
```

- For Push Button method:
  - Push the button on the STA
  - Enter the command below for the n-Link® Access Point
  - # ./wpa_cli –i <vap_name> wps_pbc <sta_mac_addr>22
    a. Wait for the STA to parse all the WPS Access Points.
- For Enter PIN method
  - Click on “Generate PIN” on the STA. A 4/8-digit numeric WPS PIN is generated.
  - Enter the command below for the n-Link® Access Point

    ```
    # ./wpa_cli –i <vap_name> wps_pin <sta_mac_addr> <wps_pin>
    ```
  - Wait for the STA to parse all the WPS Access Points.
- For Generate PIN method
  - Enter the command below for the n-Link® Access Point

    ```
    # ./wpa_cli –i <vap_name> wps_pin <sta_mac_addr>
    ```

    This will generate a 4/8-digit numeric WPS PIN.
  - Enter the PIN on the STA
  - Wait for the STA to parse all the WPS Access Points.

Note:

1) WPS_PIN and passphrase are different.
2) WPS connection timeout is 120 seconds
3) 3rd party Stations usually try to connect to all scanned WPS Access Points until they succeed in connecting to one of them.
4) WPS can be used along with any of the Secure modes (except WEP) and with Open mode also.

5.6.2 Client Mode

The procedure which is need to be followed in Client mode is as follows:

1. Start the driver in Client mode.

[22] This is the 3rd party Station’s MAC address. If all MAC addresses need to be allowed, the input parameter is the string “any”.

2. Start the supplicant with the following command.

```
# ./wpa_supplicant –i <vap_name> -D bsd –c <wps_conf_file>
-ddddt
```

3. For Push Button method:
   - Push the button on the Access Point
   - Enter the command below for the n-Link® STA

```
# ./wpa_cli –i <vap_name> wps_pbc <bssid>23
```

   - Wait for the STA to parse all the WPS Access Points.

4. For Enter PIN method
   - Click on **"Generate PIN"** on the Access Point. A 4/8-digit numeric WPS PIN is generated.
   - Enter the command below for the n-Link® STA

```
# ./wpa_cli –i <vap_name> wps_pin <bssid> <wps_pin>
```

     a. Wait for the STA to parse all the WPS Access Points.

5. For Generate PIN method
   - Enter the command below for the n-Link® STA

```
# ./wpa_cli –i <vap_name> wps_pin <bssid>
```

   - This will generate an 8-digit numeric WPS PIN.
   - Enter the PIN on the Access Point
   - Wait for the STA to parse all the WPS Access Points.

[23] This is the Access Point's MAC address. If the BSSID is not known, the input parameter is the string "any".

6 Configuration Using CFG80211

This section explains about the usage of various IOCTL commands, which can be issued to the Onebox-Mobile™ driver operating in CFG80211 mode from user space.

6.1 Using iw Wireless Tool

'iw' is a new nl80211 based CLI configuration utility for wireless devices. It is used to set/get various parameters of a wireless network interface. This section covers the usage of 'iw' when used with the Onebox-Mobile ™ driver. For a detailed description of 'iw' tool, please refer the relevant man pages on Linux system.

Creating a virtual Interface	
Description	This command is used to create a virtual interface in the specific mode requested by user
Default Value	-
Input Parameters	<phy name > -- Phy name can be obtained using the following command `$ iw phy` In case multiple wireless interfaces are present, please refer to the NOTE given below on how to determine the phy name. <interface name> -- name of the virtual interface to be created <operating mode> -- operating mode of the virtual interface; can be either 'managed' for station mode or '__ap' for access point mode.
Output Parameter	–
Reset Required	No
Usage	`iw phy <phy name> interface add <interface name> type <operating mode>`

Example	To create a virtual interface in Access Point mode use the command given below:
	`$ iw phy phy0 interface add wifi0 type __ap`
	To create a virtual interface in Station mode use the command below:
	`$ iw phy phy0 interface add wifi0 type managed`

Scan	
Description	This command is used to scan for the Access points nearby our device.
Default Value	-
Input Parameters	Interface name on which scan has to be performed
Output Parameter	List of AP's scanned
Reset Required	No
Usage	The following command initiates a scan and displays the list of AP's scanned. `$ iw dev $interface_name scan`
Example	`$ iw dev wifi0 scan`
Connect	
Description	This command is used to connect devices to the Access points in open or WEP security mode.
Default Value	-
Input Parameters	SSID, BSSID, key_index, key of AP.
Output Parameter	None
Reset Required	No

Usage	Open mode:
	`$ iw dev $interface_name connect $SSID_NAME $BSSID.`
	WEP Security:
	`$ iw dev $interface_name $ssid_name $bssid keyid:$key_index:$key`

Example	`$ iw dev wifi0 connect REDPINE_AP 00:23:a7:00:05:55`
	The above command connects to REDPINE_AP access point in open mode
	`$ iw dev wifi0 REDPINE_AP`

	`00:23:a7:00:05:55 keys d:1:234567890` `The above command instructs our device to connect to the REDPINE_AP in wep64 mode with the key index 1 and key '234567890'.`
Disconnect	
Description	This command is used to disconnect our device from the connected network.
Default Value	-
Input Parameters	Interface name
Output Parameter	–
Reset Required	No
Usage	`iw dev $interface_name disconnect`
Example	`$ iw dev wifi0 disconnect` The above command disconnects our device from the connected Access point.
Link status	
Description	This command is used to get the connection status of our device.
Default Value	-
Input Parameters	Interface name.
Output Parameter	Connection status.
Reset Required	No
Usage	`iw dev $interface_name link`
Example	`iw dev wifi0 link`

Table 6: Usage of iw wireless tool

Note:

If there are multiple phy's, i.e there are several instances of cfg80211 being used by different modules, to determine the correct phy, run the following commands:

```
$  cat /sys/class/ieee80211/
```

This will give a list of all the phy's that are currently active.

```
$ cat /sys/class/ieee80211/phyX/macaddress
```

Where X is the number of the phy which is obtained in the previous command. The mac address that starts with "00:23:a7" is the phy that has to be used.

Generic iw commands listed below are also supported. Please refer to the man page of the utility for further information on their usage.

```
$ iw phy <phyname> info
$ iw dev <devname> info
$ iw dev <devname> del
$ iw reg get
$ iw reg set <ISO/IEC 3166-1 alpha2>
$ iw dev <devname> scan dump [-u]
$ iw phy <phyname> set name <new name>
```

The following commands are supported only in the Access Point mode:

```
$ iw dev <devname> set channel <channel> [HT20|HT40+|HT40-]
$ iw dev <devname> set freq <freq> [HT20|HT40+|HT40-]
$ iw dev <devname> station del <MAC address>
$ iw dev <devname> station get <MAC address>
```

7 Enterprise security using CFG80211

7.1 Installation and configuration of FREERADIUS Server

The following packages are required to install the freeradius server 3.09:

- libtalloc-devel
- openssl-devel

Download the freeradius tar ball and install it by following the steps below:

```
$ tar zxvf freeradius-server-3.0.9.tar.gz
$ cd freeradius_3.09
$ ./configure
$ make
$ make test
$ make install
```

Configure the freeradius server according to the given steps below:

Go to /usr/local/etc/raddb/

```
$  cd /usr/local/etc/raddb/
$  vim  radiusd.conf
```

In radiusd.conf file, change the security section from

```
security{
allow_vulnerable_openssl = no
}
```

to

```
security{
allow_vulnerable_openssl = 'CVE-2014-0160'
}
```

Now we need to edit users file, which will contain the **"identity"** and "password".

```
$ vim /usr/local/etc/raddb/users
```

Add the following line at the starting in the users file

```
test Cleartext-Password := "password"
```

As an example, **"user1"** is an identity and **"test123"** is the password that has to be entered at client side i.e. in the sta_settings.conf file.

Now we need edit **"eap"** file which contains paths to our certificates and information about the EAP- Methods supported .

```
$ vim  /usr/local/etc/raddb/mods-enabled/eap
```

Note:

If Free-radius version is below 3.x "eap", will be located in raddb folder and will be named as **"eap.conf"**.

In tls-config tls-common section, changes are made to point to our certificates which are placed in /etc/certs folder.

```
tls-config tls-common {
#private_key_password = whatever
private_key_password = Wi-Fi
#private_key_file = ${certdir}/server.pem
private_key_file = /etc/certs/wifiuser.pem
#certificate_file = ${certdir}/server.pem
certificate_file = /etc/certs/wifiuser.pem
#ca_file = ${cadir}/ca.pem
ca_file = /etc/certs/wifiuser.pem
#dh_file = ${certdir}/dh
dh_file = /etc/certs/dh
}
```

To start the Radius server, run the following command in the terminal:

```
$ radiusd -X
```

7.2 Configuration of AP and RADIUS server to use EAP methods

Hostapd is used as the RADIUS Server. The AP and the server are co-located (in the same system).

The following packages have to be installed:

1.libnl-devel

2.libsqlite3x-devel

3.openssl-devel

7.2.1 Configuration of the AP

Go to driver source folder and compile it with the following options enabled:

[*] NL80211 support

[*] HOSTAPD support

```
$ make
```

To start the device in AP mode, go to the release folder and run the following commands:

```
$ cd release
```

```
$ sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the “release” folder as per the instructions in Section 4.1
```

```
$iw phy phyX interface add wifi1 type __ap
```

Where X represents the phy number. It can be obtained by the following command:

```
$ iw list | grep phy
```

Before starting the device in AP mode, ensure that in hostapd_eap.conf the following entities are enabled:

```
ieee8021x=1
own_ip_addr=192.168.2.1  /* IP address of AP */
```

/* RADIUS authentication server */

```
auth_server_addr=127.0.0.1
auth_server_port=1812
auth_server_shared_secret=testing123 /* shared secret must be the
same as in /etc/hostapd.radius_clients file */
```

Run the following command to start the device in the AP mode:

```
$ ./hostapd  hostapd_eap.conf -ddddt  >log &
$ sh dhcp_server.sh  wifi1 , where wifi1 is the interface name.
```

7.2.2 Configuring hostapd as RADIUS server

Copy the certs folder in /etc location, which will contain the certificates, hostapd.radius_clients, hostapd.eap_user and dh files.

Go to driver folder and copy the certs folder to the /etc location in your system.

```
$ cp -rvf certs /etc/
```

Now go to hostapd_server /hostapd folder in the driver folder.

```
$ cd hostapd_server/hostapd
```

Compile it

```
$ make clean
$ make
```

Check in hostapd.conf that interface is the same as the name of AP interface name.

Example:

```
$ vim hostapd.conf
interface=wifi1 ,so that RADIUS server will listen on that
interface name.
```

Start the RADIUS server after AP had started in a new terminal.

```
$./hostapd hostapd.conf -ddddd
```

All the Credentials will be in /etc/certs/hostapd.eap_user file. A sample hostapd.eap_user file is present in the certs.tgz in the release folder.

The /etc/hostapd.radius_clients file contains IP required to communicate shared secret between AP and RADIUS server. Here it is co-located, hence it is the loop-back address.

7.2.3 Configuring Station to connect to an EAP enabled AP.

Go to Driver Folder and copy the certs folder to /etc/ in your system, as it contains all the certificates required.

```
$ cp -rvf certs /etc/
```

Go to the driver folder and compile it, ensuring that the below options are enabled in wpa_supplicant.conf file.

```
$ vim wlan/supplicant/linux/wpa_supplicant/.config
CONFIG_DRIVER_NL80211=y
CONFIG_IEEE8021X_EAPOL=y
CONFIG_EAP_MSCHAPV2=y
CONFIG_EAP_TLS=y
CONFIG_EAP_PEAP=y
CONFIG_EAP_TTLS=y
CONFIG_EAP_FAST=y
CONFIG_EAP_LEAP=y
CONFIG_PKCS12=y
CONFIG_TLS=internal
```

Ensure that in menuconfig, NL80211 support is enabled.

Compile the driver.

```
$ make
```

Go to the release folder and start the device in station mode.

```
$ cd release
$ sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1
$ service NetworkManager stop
$ iw phy phyX interface add wifi0 type managed
```

Note:

X is phy number it will vary to get it type $ iw list |grep phy.

Run the supplicant after configuring sta_settings.conf according to the required EAP method. The network blocks listed below can be used as a reference.

```
$ ./wpa_supplicant -i wifi0 -D nl80211 -c sta_settings.conf -ddddt > log &
```

To connect using EAP-PEAP method, sta_settings.conf should be as described below:

```
network={
ssid="Redpine_Signals"
key_mgmt=WPA-EAP
eap=PEAP
anonymous_identity="peapuser"
identity="test"
```

```
password="password"
}
```

To connect using EAP-TTLS method, sta_settings.conf should be described as given below:

```
network={
ssid="Redpine_Signals"
key_mgmt=WPA-EAP
eap=TTLS
anonymous_identity="ttlsuser"
identity="test"
password="password"
}
```

To connect using EAP-TLS method, sta_settings.conf should be described as given below:

```
network={
ssid="Redpine_Signals"
key_mgmt=WPA-EAP
eap=TLS
anonymous_identity="tlsuser"
identity="test"
password="password"
ca_cert="/etc/certs/wifiuser.pem"
client_cert="/etc/certs/wifiuser.pem"
private_key_passwd="Wi-Fi"
private_key="/etc/certs/wifiuser.key"
}
```

To connect using EAP-FAST method, sta_settings.conf should be described as given below:

```
network={
ssid="Redpine_Signals"
key_mgmt=WPA-EAP
eap=FAST
anonymous_identity="fastuser"
identity="test"
password="password"
phase1="fast_provisioning=1"
```

```
pac_file="/etc/p1.pac"
phase2="auth=mschapv2"
ca_cert="/etc/certs/wifiuser.pem"
private_key_passwd="wifi"
}
```

EAP-LEAP has been used when Freeradius is the RADIUS Server. This has been verified with only Cisco AP.

To connect using EAP-LEAP method, sta_settings.conf should be as described below:

```
network={
ssid="Redpine_Signals"
key_mgmt=WPA-EAP
eap=LEAP
identity="user1"
password="test123"
}
```

8 HOSTAPD and Wi-Fi Protected Setup (WPS)

This section describes how the WPS implementation in hostapd can be configured and how an external component on an AP is used to enable enrollment of client devices.

WPS uses the following terms to describe the entities participating in the network setup:

Access Point: WLAN access point

Registrar: A device that control a network and can authorize addition of new devices. This may be either in the AP ("internal Registrar") or in an external device, e.g., a laptop, ("external Registrar")

Enrollee : A device that is being authorized to use the network

It should also be noted that the AP and a client device may change roles

(i.e., AP acts as an Enrollee and client device as a Registrar) when WPS is used to configure the access point.)

8.1 Hostapd Configuration Before Compilation:

WPS component needs to be enabled in hostapd build configuration (.config)

i.e: vim host/wlan/hostapd-2.3/hostapd/.config

Ensure that the below mentioned entities are enabled in .config file

CONFIG_WPS=y
CONFIG_WPS2=y
CONFIG_WPS_UPNP=y

8.1.1 Configuration in hostapd_ccmp.conf

driver=nl80211

interface=wifi1; wifi1 is the name of the interface

WPA2-Personal configuration for the AP

```
ssid=wps-test
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
```

Default WPA passphrase for legacy (non-WPS) clients

```
wpa_passphrase=12345678
```

Enable random per-device PSK generation for WPS clients

```
wpa_psk_file=/etc/hostapd.wpa_psk
```

Note:

Check if the hostapd.wpa_psk file is present in /etc/ , if not create a new empty file naming hostapd.wpa_psk in location (/etc/).

Enable control interface for PBC/PIN entry

```
ctrl_interface=/var/run/hostapd
```

Enable internal EAP server for EAP-WSC (part of Wi-Fi Protected Setup)

```
eap_server=1
wps_state=2
ap_pin=12345670
wps_pin_requests=/var/run/hostapd_wps_pin_requests
```

8.1.2 Starting AP-mode for WPS -push button method:

$ sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1

$ is phi ; it will give phyX number

$ iw phy phyX interface add wifi1 type __ap

$ ./hostapd hostapd_ccmp.conf -ddddddt>log &

$ sh dhcp_server.sh wifi1

$ ./hostapd_cli wps_pbc

Now push wps button on station side.

At this point, the client has two minutes to complete WPS negotiation

8.1.3 Starting AP-mode for WPS -Enter-pin- method:

$ sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in **Section 4.1**

$ iw phy ; it will give phyXX number

$ iw phy phyXX interface add wifi1 type __ap

$ ./hostapd hostapd_ccmp.conf -ddddddt>log &

$ sh dhcp_server.sh wifi1

./hostapd_cli wps_pin any [wps-pin-of station]

$ ./hostapd_cli wps_pin any 12345670

8.1.4 Starting AP-mode for WPS -Generate pin- method:

$ sh wlan_enable.sh or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1

$ iw phy ; it will give phyXX number

$ iw phy phyXX interface add wifi1 type __ap

$ ./hostapd hostapd_ccmp.conf -ddddddt>log &

$ sh dhcp_server.sh wifi1

$ hostapd_cli wps_ap_pin random [timeout]
The above command generates a random AP pin number. If the optional timeout parameter is given then

the AP pin will be enabled for the specified number of seconds.
$ ./hostapd_cli wps_ap_pin random 300
The above command generates a 8digit random pin which needs to be entered at the station side using the procedure mentioned below.

Here AP acts as an Enrolee and client device as a Registrar, so ensure that below mentioned entities are enable at STATION side.

PATH: host/wlan/supplicant/linux/wpa_supplicant/.config

CONFIG_DRIVER_NL80211=y

CONFIG_WPS=y

CONFIG_WPS2=y

CONFIG_WPS_ER=y

CONFIG_WNM=y

Use the given below command to connect to the AP. Here AP pin is the pin generated randomly in section **8.1.4Starting AP-mode for WPS -Generate pin- method**: above.

$ ./wpa_cli wps_reg <AP BSSID> <AP-PIN>

8.1.5 Disable AP pin

To disable AP Pin use the command given below:

$ hostapd_cli wps_ap_pin disable

The command disables AP PIN (i.e., do not allow external Registrars to use it to learn the current AP settings or to reconfigure the AP).

8.1.6 Get the AP pin

To fetch the current AP pin use the command below:

$ hostapd_cli wps_ap_pin get

8.1.7 Set the AP pin

$ hostapd_cli wps_ap_pin set <PIN> [timeout]

Sets the AP PIN and enables it.

If the optional timeout parameter is given, the AP PIN will be enabled for the specified number of seconds

8.1.8 Get the current configuration

$ hostapd_cli get_config

The above command displays the current configuration of the AP mode.

9 Antenna Diversity

9.1 Antenna Diversity

Antenna diversity is a feature which enables the automatic selection of the antenna to use. The antenna on which packets with better RSSI values are received is selected. The RSSI monitoring happens continuously. Once enabled, this feature will persist for the entire duration of operation.

9.2 Enabling Antenna Diversity

The steps listed in this section is used to start the antenna diversity feature in Client mode only. Once it is enabled, antenna selection happens automatically:

1. Open the common_insert.sh file present in the **"release"** folder using an editor like vim.
2. Ensure that the variable RSI_ANTENNA_DIVERSITY is set as given below:
   - RSI_ANTENNA_DIVERSITY=1

10 Sniffer Mode

Below are the Steps to operate the device in Sniffer Mode.

1. Ensure that common_insert.sh present in the release folder has valid driver mode and coexistence mode.

   DRIVER_MODE=7 (Sniffer mode)

   COEX_MODE = 1 (Wi-Fi station/ Wi-Fi-Direct/Wlan-Per/Sniffer) .

2. Go to the release folder and start the driver modules using the below commands

   ```
   # sh wlan_enable.sh
   ```

3. Create the virtual interface in monitor mode.

   ```
   # ./onebox_util <base_interface>  create_vap wifi0 mon
   ```

4. Select the channel using the below command.

   ```
   # iwconfig <interface_name(wifi0)> freq <channel_number>
   ```

5. To start capturing the packets use the below command.

   ```
   # ifconfig  <interface_name (wifi0)> up
   ```

Note :

Use tcpdump or wireshark tools to observe the packets being captured by the device.

11 Monitor Mode

The Monitor Mode is one of the operating modes that can be set while creating a VAP. It enables capture of packets being transferred over a single or multiple VAPs which are operating in either Access Point or Client or P2P modes.

The order of the VAPs' creation does not matter. Once created, the **"tcpdump"** command can be used to display the packets being transferred.

- **Example Scenario 1:** Create a Client mode VAP and a Monitor mode VAP and display packets being transferred to/from the Client

```
# ./onebox_util rpine0 create_vap wifi0 sta
# ./onebox_util rpine0 create_vap wifi1 mon
# ifconfig wifi0 up
# ifconfig wifi1 up
# tcpdump –i wifi1
```

- **Example Scenario 2:** Create an Access Point mode VAP, a Client mode VAP and a Monitor mode VAP and display the packets being transferred to/from the Access Point and Client

```
# ./onebox_util rpine0 create_vap wifi0 ap
# ./onebox_util rpine0 create_vap wifi1 sta
# ./onebox_util rpine0 create_vap wifi2 mon
# ifconfig wifi0 up
# ifconfig wifi1 up
# ifconfig wifi2 up
# tcpdump –i wifi2
```

Note:

Difference between Sniffer and Monitor modes

Monitor mode displays the packets being transferred to/from the device which is configured in different operating modes like Access_point or Client.

Sniffer mode displays all the packets on air depending on the channel and band width configured and displays them using wire shark tool.

12 Background Scan Parameters

This section describes the parameters for the Background scan commands that can be sent to the n-Link Client using the onebox_util program.

- <bgscan_threshold>: The Background scan threshold is referred to as the RSSI Upper Threshold. At every background scan interval (configured via <periodicity>), the n-Link® module decides whether or not to initiate a background scan based on the connected Access Point's RSSI. The module initiates a background scan if the RSSI of the connected Access Point is below this threshold. The input value should be the absolute value in dBm.
- <rssi_tolerance_threshold>: If the difference between the current RSSI value of the connected Access Point and the RSSI value of the Access Point from the previous background scan is greater than the RSSI Tolerance Threshold, then the module performs a background scan. Assigning a large value to this field will eliminate this method of triggering background scans.
- <periodicity>: This parameter specifies the interval between the background scans. The unit of this field is seconds. Setting the value of this field as 0 will disable background scans.
- <active_scan_duration>: This parameter determines the duration of the active scan in each channel during the Background scan process. The recommended value for this parameter is 20ms for quicker Background scan operation and uninterrupted throughput. The maximum allowed value for this parameter is 255ms.
- <passive_scan_duration>: This parameter determines the duration of the passive scan in each DFS channel. If an active scan is enabled in a DFS channel and a beacon or probe response is received during that period, the module converts the passive scan into an active scan and waits through the duration specified by the <active_scan_duration> parameter. The recommended value for this parameter is 100ms, to ensure receiving beacons in a channel, if any, during a passive scan. Active scan in DFS channel can be enabled through Background scan probe request. Active scanning will be performed only if channel switch IE (Information Element) is not present in the received beacon or probe response packets. The maximum allowed value for this parameter is 255ms.
- <two_probe_enable>: If this feature is enabled, the Client sends two probe requests to the Access Point. This is useful when scanning is carried out in channels with high traffic. The valid values are
  - 0 – Disable
  - 1 – Enable
- <num_of_bgscan_channels>: Number of Background scan channels. The n-Link® module supports up to 24 channels.
- <channels_to_scan>: The list of channels in which Background scan has to be performed.

13 Power Save Modes, Profiles and Parameters

The Power Save modes and parameters are valid only for the Client mode. By default, the module's power save is disabled.

13.1 Power Save Modes

The module broadly supports two types of power save modes:

- **Low Power (LP) Mode:** The PHY (RF and Baseband) and LMAC sections are powered off but the UMAC and Host Interface sections of the module are powered on and fed a low frequency clock. The module responds to commands/requests from the Host processor immediately in this mode.
- **Ultra-low Power (ULP) Mode:** A majority of the module is powered off except for a small section which has a timer and interrupt logic for waking up the module. The module cannot respond to the Host processor's commands/requests unless it wakes up because of timeout or because of an interrupt asserted by Host processor. The sleep entry/exit procedures in this mode are indicated to the Host processor either through a packet based or signal based handshake. This mode is supported only for SDIO host interface.

13.2 Power Save Profiles

For each of the above power save modes, the module supports multiple power save profiles. These are explained below:

- **Deep Sleep:** The module is in deep sleep mode when it is not connected to an Access Point. The duration of the Deep Sleep is defined by the <sleep_duration> parameter of the set_ps_params command. For LP mode, a value of 0 for the <sleep_duration> parameter programs the module to be in Deep Sleep mode indefinitely till it is woken up by the Host processor via the host interface. The value of 0 is invalid for ULP mode and should not be used.
- **Connected Power Save:** In the connected state, the module can operate in Traffic Based Power Save Profile (PSP) or Fast PSP. These profiles are used by the module to decide when to enter and exit from power save modes on the fly. They have to be selected based on the performance and power consumption requirements of the end product.
  - **Traffic Based PSP**: This profile is dependent on the `<tx_threshold>` and `<rx_threshold>` parameters, which indicate transmit and receive throughput thresholds beyond which the module exits power save mode and below which the module enters power save mode. The `<tx_hysteresis>` and `<rx_hysteresis>` parameters are also used in this profile. This profile is enabled when non-zero values are assigned to the `<tx_threshold>` and `<rx_threshold>` parameters along with the `<monitor_interval>` parameter.
  - **Fast PSP:** This profile is a variant of the Traffic Based PSP which exits power save mode even for a single packet and enters the power save mode if no packet is transferred for the <monitor_interval> amount of duration. This profile is enabled independently for the Transmit and Receive directions if the <tx_threshold> and <rx_threshold> parameters are assigned zero, respectively, while assigning a non-zero value to the <monitor_interval> parameter.

13.3 Wakeup Procedures and Data Retrieval

When in power save mode, the module wakes up at periodic intervals or due to certain events (like pending transmit packets from the Host). At every wake up, the module has to poll the Access Point

and check whether there are any pending Rx packets destined for the module. The module uses different protocols to retrieve data from the Access Point based on the protocol supported by the Access Point. These data retrieval methods (protocol-based) are used to further classify the power save profiles described in the previous section into Max PSP, Periodic UAPSD and Transmit based UAPSD.

The MAX PSP and UAPSD modes are explained below:

- **Max PSP:** In this mode, the module wakes up at the end of sleep period (Listen or DTIM interval) and retrieves pending Rx packets from the Access Point by sending a PS-POLL packet. It also transmits any packets received from the Host processor and then goes back to sleep. The parameters listed below are used by the module to decide the period of sleep during power save, in the same order of priority:
  a. <listen_interval_duration>
  b. <dtim_interval_duration>
  c. <num_beacons_per_listen_interval>
  d. <num_dtims_per_sleep>
- **Periodic UAPSD:** This mode is enabled by the set_uapsd_params command and if the <uapsd_wakeup_period> parameter is assigned a non-zero value. For this mode, the wakeup period can be assigned a value between 10 and 100 milliseconds. If supported by the Access Point, in this mode, the module wakes up at the end of each sleep period and transmits pending data or a QoS Null packet to retrieve the data from the Access Point. The sleep period is governed by the parameters set using the set_ps_params command (see the list under Max PSP above) and also set_uapsd_params command. The sleep period is the minimum of the values programmed using the above two commands. If the Access Point does not support UAPSD, the module tries to mimic this mode by waking up at the end of the sleep period and transmits pending data and a PS_POLL packet to retrieve the data from the Access Point.
- **Transmit based UAPSD:** If <uapsd_wakeup_period> parameter is set to 0 in the set_uapsd_params command, the Transmit based UAPSD mode is enabled. In ULP mode, the Transmit based UAPSD mode can be used only when the signal-based handshake is enabled (and not in packet-based handshake mode). In this mode, the module wakes up from sleep when the Host sends a packet to be transmitted and then retrieves the pending packets from the Access Point by transmitting the packet. The module also wakes up if there is no packet transmitted for the sleep duration programmed in the set_ps_params command. If the Access Point does not support UAPSD, the module mimics this mode by waking up whenever there is packet to be transmitted, transmits the packet and then retrieves the pending data from the Access Point by sending a PS_POLL packet.

13.4 Power Save Parameters

The input parameters of the set_ps_params command are explained below.

- <ps_en>: This parameter is used to enable (1) or disable (0) power save mode.
- <sleep_type>: This parameter is used to select the sleep mode between LP (1) and ULP (2) modes.
- <tx_threshold>: If a non-zero value is assigned, this parameter is used to set a threshold for the Transmit throughput computed during the <monitor_interval> period so that module can decide to enter (throughput ≤ threshold) or exit (throughput > threshold) the power save mode. The value is in Mbps and minimum value is 0 Mbps.

- <rx_threshold>: If a non-zero value is assigned, this parameter is used to set a threshold for the Receive throughput computed during the <monitor_interval> period so that module can decide to enter (throughput ≤ threshold) or exit (throughput > threshold) the power save mode. The value is in Mbps and minimum value is 0 Mbps.
- <tx_hysteresis>: The decision to enter or exit power save mode based on the Transmit throughput alone can result in frequent switching between the power save and non-power save modes. If this is not beneficial, the <tx_hysteresis> parameter can used to make the module re-enter the power save mode only when the throughput falls below the difference between the <tx_threshold> and <tx_hysteresis> values. The value is in Mbps and minimum value is 0 Mbps. This parameter should be assigned a value which is less than the value assigned to the <tx_threshold> parameter.
- <rx_hysteresis>: The decision to enter or exit power save mode based on the Receive throughput which alone can result in frequent switching between the power save and non-power save modes. If this is not beneficial, the <rx_hysteresis> parameter can be used to make the module re-enter the power save mode only when the throughput falls below the difference between the <rx_threshold> and <rx_hysteresis> values. The value is in Mbps and minimum value is 0 Mbps. This parameter should be assigned a value which is less than the value assigned to the <rx_threshold> parameter.
- <monitor_interval>: This parameter is the duration (in milliseconds) over which the Transmit and Receive throughputs are computed to compare with the <tx_threshold>, <rx_threshold>, <tx_hysteresis> and <rx_hysteresis> values. The maximum value of this parameter is 30000 ms (30 seconds).
- <sleep_duration>: This parameter is the duration (in milliseconds) for which the module sleeps in the Deep Sleep mode. For LP mode, a value of 0 for the <sleep_duration> parameter programs the module to be in Deep Sleep mode indefinitely till it is woken up by the Host processor via the host interface. The value of 0 is invalid for ULP mode and should not be used. The maximum value for this parameter can be 65535.
- <listen_interval_duration>: This parameter is the duration (in milliseconds) for which the module sleeps in the connected state power save modes. If a non-zero value is assigned to this parameter it takes precedence over the other sleep duration parameters that follow (<num_beacons_per_listen_interval>, <dtim_interval_duration>, <num_dtims_per_sleep>). The maximum duration for which the device supports sleep is 4095 times the duration of the beacon interval considering the listen interval parameters of the access point. The maximum value for this parameter can be 65535, but the duration should be decided factoring in the beacon interval of the access point. This parameter is considered only after the module is connected to the access point. For example, if the beacon interval of the AP is 100ms and listen interval of AP is 8 beacons, then the maximum time the device can sleep without any data loss is 800 ms (8 * 100). Hence, the listen_interval_duration can be up to 800ms.
- <num_beacons_per_listen_interval>: This parameter is the number of beacon intervals for which the module sleeps in the connected state power save modes. Here, the device will wake up for the nth beacon, where n is the listen interval value programmed by the user. If a non-zero value is assigned to this parameter it takes precedence over the other sleep duration parameters that follow (<dtim_interval_duration>, <num_dtims_per_sleep>). This parameter is used only when the above parameter is assigned 0. The maximum value for this parameter is 4095. The value for this parameter also has to be chosen keeping in mind the listen interval of the access point. . This parameter is considered only after the module is connected to the access point.

- <dtim_interval_duration>: This parameter is the duration (in milliseconds) for which the module sleeps in the connected state power save modes. The device will wake up for the nearest DTIM beacon after the time which the user has programmed has expired. This parameter can be used when DTIM information is not available. If a non-zero value is assigned to this parameter it takes precedence over the other sleep duration parameter that follows (<num_dtims_per_sleep>). This parameter is used only when the above parameters are assigned 0. The maximum value for this parameter can be 10000ms. This parameter is considered only after the module is connected to the access point.
- <num_dtims_per_sleep>: This parameter is the number of DTIM intervals for which the module sleeps in the connected state power save modes. This parameter has least priority compared to the ones above and is used only if the above parameters are assigned 0. The maximum value for this parameter is 10. This parameter is considered only after the module is connected to the access point.

Note:

The LP and ULP Power Save modes are supported with SDIO interface. USB interface supports only LP Power Save mode.

14 Wi-Fi Performance Test ioctl Usage

The OneBox-Mobile software provides applications to test Transmit and Receive performance of the module. The Band of operation of the module needs to be configured before performing any tests.

Note:

Open the common_insert.sh file present in the **"release"** folder using an editor like vim. Ensure that the DRIVER_MODE is set as below:

```
DRIVER_MODE = 2
```

Run the following command to install the Driver in Performance Test mode:

`# sh wlan_enable.sh` or wlan_bt_insert.sh or wlan_zigb_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1

Next, follow the instructions below to run the Transmit and Receive tests.

14.1 WIFI Transmit Tests

The **"transmit"** utility, present in the "release" folder allows the configuration of the following parameters and starts the transmission of packets.

- Transmit Power
- Transmit Data Rate
- Packet Length
- Transmit Mode
- External PA Enable/Disable[24]
- Rate Flags like Short GI, Greenfield, etc.
- Enable/Disable Aggregation
- Number of packets to be transmitted in Burst Mode
- Delay between packets in Burst Mode
- Regulatory Domain

14.1.1 Transmit Command Usage

The command usage is explained below.

```
# ./transmit <tp> <r> <l> <m> <c> <p> <f> <a> <n> <d> <rd>
```

`<tp>`: Transmit power in dBm for controlling transmit power. To set the transmit power value, enter a value either between -7 and18. If a value of 127 is entered, the packet will be transmitted at the maximum power from the Transmit power table in the module.

`<r>`: Transmit Data Rate. To set the transmit data rate, select a value from 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54, mcs0, mcs1, mcs2, mcs3, mcs4, mcs5, mcs6 and mcs7.

`<l>`: Transmit packet length in bytes. Enter a value between 24 and 1536 when aggregation is not enabled and between 24 and 30000 when aggregation is enabled.

`<m>`: Transmit mode. Enter 0 for Burst mode and 1 for Continuous mode.

[24] This is not supported in the current release.

<c>: Transmit channel number[25]. The following table maps the channel numbers to the center frequencies for 20MHz and 40MHz bandwidth modes in 2.4 GHz and 5 GHz bands.

Band (GHz)	Bandwidth (MHz)	Channel Number	Center Frequency (MHz)
2.4	20	1	2412
2.4	20	2	2417
2.4	20	3	2422
2.4	20	4	2427
2.4	20	5	2432
2.4	20	6	2437
2.4	20	7	2442
2.4	20	8	2447
2.4	20	9	2452
2.4	20	10	2457
2.4	20	11	2462
2.4	20	12	2467
2.4	20	13	2472
2.4	40	3	2422
2.4	40	4	2427
2.4	40	5	2432
2.4	40	6	2437
2.4	40	7	2442
2.4	40	8	2447
2.4	40	9	2452
2.4	40	10	2457
2.4	40	11	2462
4.9	20	184	4920
4.9	20	188	4940

[25] On-air testing in DFS channels should be avoided till the module is certified for DFS. Cabled tests can be run in these channels.

4.9	20	192	4960
4.9	20	196	4980
5	20	8	5040
5	20	12	5060
5	20	16	5080
5	20	36	5180
5	20	40	5200
5	20	44	5220
5	20	48	5240
5	20	52	5260
5	20	56	5280
5	20	60	5300
5	20	64	5320
5	20	100	5500
5	20	104	5520
5	20	108	5540
5	20	112	5560
5	20	116	5580
5	20	120	5600
5	20	124	5620
5	20	128	5640
5	20	132	5660
5	20	136	5680
5	20	140	5700
5	20	149	5745
5	20	153	5765
5	20	157	5785
5	20	161	5805

5	20	165	5825
5	40	38	5190
5	40	42	5210
5	40	46	5230
5	40	50	5250
5	40	54	5270
5	40	58	5290
5	40	62	5310
5	40	102	5510
5	40	106	5530
5	40	110	5550
5	40	114	5570
5	40	118	5590
5	40	122	5610
5	40	126	5630
5	40	130	5650
5	40	134	5670
5	40	138	5690
5	40	151	5755
5	40	155	5775
5	40	159	5795
5	40	163	5815

Table 7: Channel Numbers and Corresponding Center Frequencies

<p>: Enable/Disable External PA. This parameter is not supported in the current release.

<f>: Rate Flags. This parameter is used to enable/disable Short GI and Greenfield and also to set the channel width of the transmitted packets. The table below explains the flags that can be enabled and disabled. Multiple flags can be set at a time.

Bit	Description

0	Short GI 0 – Disable Short GI 1 – Enable Short GI
1	Greenfield transmission 0 – Disable Greenfield transmission 1 – Enable Greenfield transmission
[4:2]	Operating bandwidth of the channel (3 bits) 0 – 20MHz 2 (Bit 3 is set) – Upper 20MHz of 40MHz 4 (Bit 4 is set) – Lower 20MHz of 40MHz 6 (Bits 3 and 4 are set) – Full 40MHz

Table 8: Rate Flags for Transmit Tests

<a>: Enable/Disable Aggregation. Enter 0 to disable aggregation and 1 to enable aggregation. The packet length is divided into chunks of size 1792 bytes and aggregated. This parameter applies only to the Burst mode transmission and is ignored in the case of Continuous mode of transmission.

<n>: Number of packets to be transmitted in Burst mode. The transmission stops after the number of packets specified by this parameter are transmitted in the Burst mode. If this value is 0, then the transmission will not stop until the user gives the "./transmit 0" command to stop the transmissions. This parameter is ignored in the case of Continuous mode of transmission.

<d>: Delay between packets in Burst mode. This parameter is used to introduce a delay between any two packets. The delay has to be specified in microseconds. If this value is 0, then the packets will be transmitted without any delay. This parameter is ignored in the case of Continuous mode of transmission.

Note:

After the transmission starts, the following commands need to be given to stop the transmissions.

```
# ./transmit 0
```

<rd>: Regulatory Domain. Refer the table below for the mapping of values to the regulatory domains.

Input Value	Regulatory Domain
0	US (FCC)
1	Europe (ETSI)
2	Japan(JP)
255	World Domain

Table 9: Regulatory Domain Input in Transmit Tests

Examples:

```
# ./transmit 2 5.5 750 1 11 0 1 0 0 0 0
```

The above command starts continuous transmission with the following configuration:

Transmit gain – 2dbm

Data rate – 5.5Mbps

Packet Length – 750 bytes

Transmit mode – 1 (continuous mode).

Channel number – 11

External PA – disabled

Rate flags – 1 (Short GI is enabled with 20MHz Channel width)

Aggregation – disabled (ignored in continuous mode)

Number of packets to be transmitted – 0 (ignored in continuous mode)

Delay between the packets – 0 (ignored in continuous mode)

```
# ./transmit 12 36 1000 0 6 0 25 0 0 1000 0 0
```

The above command starts burst mode transmission with the following configuration:

Transmit gain – 12dBm

Data rate – 36Mbps

Packet Length – 1000 bytes

Transmit mode – 0 (Burst mode).

Channel number – 6

External PA – disabled

Rate flags – 25 (Short GI with Full 40MHz Channel width)

Aggregation – disabled

Number of packets to be transmitted – 1000

Delay between the packets – 0

14.2 WIFI Receive Tests

The **"receive"** utility, present in the "release" folder, can be invoked for displaying the following information

- Total number of CRC PASS packets
- Total number of CRC FAIL packets and
- Total number of FALSE CCAs

Receive Command Usage

```
# ./receive <filename> <channel_number> <start/stop>
<channel_width>
```

<filename>: Name of the file into which the statistics will be logged, in addition to being displayed on the console.

<channel_number>[26]: Channel number in which the statistics need to be logged. Refer to the **Table 7: Channel Numbers and Corresponding Center Frequencies** for more details.

<start/stop>: Parameter to start or stop logging the statistics. Enter 0 to start logging and 1 to stop logging.

<channel_width>: Operating bandwidth of the channel. Refer to the table below.

Value	Channel Width
0	20MHz
2	Upper 20MHz of 40MHz
4	Lower 20MHz of 40MHz
6	Full 40MHz

Table 10: Channel Width Values

Examples:

```
# ./receive stats 6 0 0
```

The above command starts the receive utility and logs statistics with the following parameters.

Filename – stats

Channel number – 6

Channel Width – 20MHz

The test utility displays the following information:

- Total number of packets received with correct CRC.
- Total number of packets received with incorrect CRC.
- Total number of False CCA's received.

```
# ./receive stats 6 1 0
```

The above command will stop the receive application

14.3 Continuous Wave (CW) mode

The Continuous Wave mode is used to transmit a single tone – either a sine wave or a cosine wave.

Command Usage

```
./onebox_util <base_interface> cw_mode <channel> <start/stop>
<type>
```

[26] On-air testing in DFS channels should be avoided till the module is certified for DFS. Cabled tests can be run in these channels.

<base_interface>: Base Interface (string like rpine0)

<channel_number>: Channel number in which the transmission has to be done. Please refer to the **Table 7: Channel Numbers and Corresponding Center Frequencies** for a mapping between the channel numbers and the center frequencies.

<start/stop>: Parameter to start or stop the transmission. Enter 0 to start transmission and 2 to stop transmission.
In order to start transmission for 11J 20MHz channels, enter 1.

<type>: Parameter to select from among the different types of waves to be transmitted.
Enter 2 for Single Tone of 5MHz.
Enter 5 for DC tone.

The transmit power for the CW mode transmission is set using the “transmit” utility. The “transmit” command has to be first issued to start transmission at the required transmit power level and then called again to stop the transmission before giving the “onebox_util” command to start the CW transmission.

The Antenna direction in CW mode will be reverse direction.

The user can select the appropriate antenna by using the following command.

- # ./onebox_util <base_interface> ant_sel <value>
- <value = 2> – Select RF_OUT_2/Integrated Antenna
- <value = 3 > – Select RF_OUT_1/U.FL Connector

Examples

```
# ./transmit 2 5.5 750 1 11 0 1 0 0 0 0
# ./transmit 0
# ./onebox_util rpine0 cw_mode 6 0 2
```

The above command starts continuous wave transmission with the following configuration.

Channel number – 6

Type – Single tone

Transmit Power – 2dBm

```
# ./onebox_util rpine0 cw_mode 6 2 2
```

The above command stops continuous wave transmission.

```
# ./onebox_util rpine0 cw_mode 184 1 2
```

The above command is to start transmission in 184(11J) channel. The corresponding command to stop is,

./onebox_util rpine0 cw_mode 184 2 2

15 Wake-On-Wireless LAN Parameters

The parameters listed below for the Wake-On-Wireless LAN are valid only in Client mode. The <hw_bmiss> parameter needs to be given as an input during VAP creation to be able to use the WoWLAN feature – refer to the section on Configuring Using onebox_util for details on VAP creation.

- <base_interface>: Base Interface (string like rpine0)
- <src_mac_addr>: This parameter is the 48-bit Source MAC address in hexadecimal format with colon separation, which is used to filter the Unicast packets received by the device. This parameter is valid only when bit 2 of the <wowlan_flags> parameter is set to '1'.
- <host_sleep_status>: This parameter informs the device whether the Host is entering sleep state ("1") or exiting sleep state ("0"). The device will toggle the GPIO_2 (Host Wakeup Interrupt) only when the Host indicates that it is entering to sleep state.
- <wowlan_flags>: This parameter is a bitmap used to program the device to wake up the Host based on the type of packets received by it. It is a 16-bit value as explained in the table below. The Host can program multiple bits to "1" at the same time to enable wakeup on different types of events.

Bit [15:0]	Description
[15:4]	Reserved.
3	Wake up Host when EAPOL packets are received by the device
2	Wake up Host when Unicast packets from a specific MAC address (specified by <src_mac_addr> are received by the device
1	Wake up Host when Unicast packets are received by the device
0	Wake up Host for any packet received by the device

Table 11: WoWLAN Flags

Note:

In case of multiple devices connected to a host then use the appropriate interface name to issue ioctls on base interface device using onebox utility.

If USB device0 is connected to host then rpine0 will be created to device0. When USB device1 is connected, rpine1 gets created. Now in order to issue ioctl's on rpine1 device rpine0 should be replaced by rpine1 in the commands explained above.

16 Bluetooth hcitool and hciconfig Usage

The hcitool and hciconfig commands are used to control and configure parameters for the Bluetooth interface. The HCI commands explained here are the most frequently used commands. For other HCI commands please refer the Bluetooth specification, Volume 2 Part E, Chapter7 from www.bluetooth.org.

Reset	
Description	This command is used to issue a soft reset to the Bluetooth module
Default Value	-
Input Parameters	None
Output Parameter	None
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x03 0x03`
Read Local Version Information	
Description	This command is used to read the local version information
Default Value	-
Input Parameters	None
Output Parameter	HCI version HCI revision LMP version Manufacturer name LMP subversion
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x04 0x01`
Read Local Supported Commands	
Description	This command is used to read the local controller supported HCI commands.
Default Value	-
Input Parameters	None

Output Parameter	List of supported commands (64 bytes of bit field)
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x04 0x02`
Get Local BD Address	
Description	This command is used to get the local BD Address
Default Value	-
Input Parameters	None
Output Parameter	6 Byte BD Address
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x04 0x09`
Start Inquiry	
Description	This command is used to start the Inquiry process
Default Value	
Input Parameters	LAP (3 Bytes): (0x9E8B00 – 0x9E8B3F) Inquiry duration: (0x01 to 0x30 -> 1.28 to 61.44 Seconds) Number of responses: (0x01 – 0xFF)
Output Parameter	None.
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x01 0x01 <LAP> <duration> <no_of_responses>`
Write Local Name	
Description	This command is used to Set the local device name
Default Value	
Input Parameters	Name of the device.

Output Parameter	None.
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x03 0x13 <name>`

Table 12: Bluetooth hcitool and hciconfig usage

16.1 Bluetooth Power Save Commands

Vendor-specific HCI Commands are used to configure the device in the power save mode. The module supports Low Power (LP) and Ultra-Low Power (ULP) modes. These are explained in more detail in the Power Save Modes section of WLAN ioctl Usage Guide. The LP and ULP modes are supported with the SDIO interface while only the LP mode is supported in USB mode.

Vendor Specific Power Save	
Description	This command is used to enable/disable the power save mode of the device and also set the sleep duration in Standby mode.
Default Value	-
Input Parameters	**Sleep Enable:** 0x01 - Sleep enable 0x00 - Sleep disable **Sleep Mode:** 0x01 – LP (Low Power) mode 0x02 – ULP (Ultra Low Power) mode Sleep Duration in Standby mode (in msec) : (Range 0x00 – 0xFF)
Output Parameter	None
Reset Required	No.
Usage	`hcitool -i <hciX> cmd 0x3F 0x0003 <sleep enable/disable> <sleep mode> <sleep duration>`

16.2 Bluetooth Performance Test ioctl Usage

The OneBox-Mobile software provides applications to test Transmit and Receive performance of the module.

Note:

Open the common_insert.sh file present in the **"release"** folder using an editor like vim. Ensure that the DRIVER_MODE and COEX_MODE is set as below:

```
DRIVER_MODE = 2

COEX_MODE = 4(for BT Classic)

COEX_MODE = 8(for BT LE)

Ensure that only Bluetooth is selected in menuconfig before using
bt_enable.sh command
```

Run the following command to install the Driver in Performance Test mode:

`# sh bt_enable.sh` or wlan_bt_insert.sh or onebox_insert.sh script present in the "release" folder as per the instructions in Section 4.1

Next, follow the instructions below to run the Transmit and Receive tests.

16.2.1 BT Transmit Tests

The "bt_transmit" utility, present in the "release" folder, allows the configuration of the following parameters and starts the transmission of packets.

- Device Address
- Packet Type
- Packet Length
- BR/EDR Mode
- Receive Channel Index
- Transmit Channel Index
- Link Type
- Scrambler Seed
- Number of Packets
- Payload Type
- Classic/LE Mode
- LE Channel Type
- Transmit Power
- Transmit Mode
- Hopping Type
- Antenna Select

16.2.1.1 BT Transmit Command Usage

The command usage is explained below.

```
./bt_transmit <dev_addr> <pkt_type> <pkt_length> <br_edr_mode>
<rx_channel_index> <tx_channel_index> <link_type> <scrambler_seed>
<no_of_packets> <payload_type> <classic_le_mode> <le_channel_type>
<tx_power> <tx_mode> <hopping_type> <ant_sel>
```

`<dev_addr>`: Device address. It is a 48-bit address in hexadecimal format , e.g., 0023A7010203.

<pkt_type>: Type of the packet to be transmitted, as per the Bluetooth standard.

<pkt_length>: Length of the packet, in bytes, to be transmitted.

<br_edr_mode>: Decides whether the transmission has to happen in Basic Rate or Enhanced Data Rate in Classic mode. It is invalid in LE mode.

'1' – Basic data Rate (1Mbps)

'2' or '3' – Enhanced Data Rate(2 Mbps or 3 Mbps)

<rx_channel_index>: Receive channel index, as per the Bluetooth standard.

<tx_channel_index>: Transmit channel index, as per the Bluetooth standard.

<link_type>: Link Type – ACL, SCO, eSCO. Valid only in the Classic mode and invalid in LE mode.

- '0' – SCO
- '1' – ACL
- '2' – eSCO

<scrambler_seed>: Initial seed to be used for whitening. It should be set to '0' to disable whitening.

<no_of_packets>: Number of packets to be transmitted. It is valid only when the <tx_mode> is set to Burst mode (0).

<payload_type>: Type of payload to be transmitted.

- '0' – Payload consists of all zeros.
- '1' – Payload consists of all 0xFF's.
- '2' – Payload consists of all 0x55's
- '3' – Payload consists of all 0xF0's.
- '4' – Payload consists of PN9 sequence.

<classic_le_mode>: Choose between Bluetooth Classic and LE modes for the packet transmission.

- '1' – Classic mode
- '2' – LE Mode

<le_channel_type>: Channel type in LE mode. It is invalid in Classic mode

- '0' – Advertising channel
- '1' – Data channel

<tx_power>: Transmit power (in dBm) to be used by the module. The value should be between 0 and 18.

<tx_mode>: Choose between Burst and Continuous modes of transmission.

- '0' – Burst mode
- '1' – Continuous mode

<hopping_type>: Choose the hopping pattern.

- '0' – No hopping
- '1' – Fixed hopping
- '2' – Random hopping

<ant_sel>: Select one of the two RF ports. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it is used to select between the two.

- '2' – RF_OUT_2/Antenna
- '3' – RF_OUT_1/U.FL

Note:

After the transmission starts, the following command can be given to stop the transmission.

```
./bt_transmit 0
```

Example

```
./bt_transmit 0023a7010203 15 1021 3 10 10 1 0 0 1 1 0 10 0 0 2
```

The above command starts transmitting 3DH5 packets in burst mode with no hopping at 3 Mbps with the following configuration.

Device address – 00:23:A7:01:02:03

Packet type – 0xF

Packet length – 1021 bytes

BR/EDR mode – 3 (EDR mode with 3 Mbps)

Rx channel index – 10

Tx channel index – 10

Link type – 1(ACL)

Scrambler seed – 0 (Disable whitening)

No of packets – 0(Since tx_mode is burst)

Payload type – 1(Payload consists of all 0xFF's)

Classic/LE mode – 1(Classic mode)

LE channel type – 0(Invalid in Classic mode)

Tx power – 10(10dBm)

Tx mode – 0(Burst mode)

Hopping type – 0(no hopping)

Antenna select – 2(RF_OUT_2/Antenna)

Refer to the table below for more details.

Standard Packet	pkt_ type	br_edr_mode	classic/le Mode	Packet Length	Link type
DM1	3	1	1	0-17	1
DH1	4	1	1	0-27	1
DH3	11	1	1	0-183	1
DM3	10	1	1	0-121	1

DH5	15	1	1	0-339	1
DM5	14	1	1	0-224	1
2-DH1	4	2	1	0-54	1
2-DH3	10	2	1	0-367	1
2-DH5	14	2	1	0-679	1
3-DH1	8	3	1	0-83	1
3-DH3	11	3	1	0-552	1
3-DH5	15	3	1	0-1021	1
Any Value	Any Value	1	2	0-37	Any Value

Table 13: BT Packet lengths

16.2.1.2 BT Receive Tests

The Receive tests can be performed by using either of the two commands – “bt_receive” and “bt_util”.

The “bt_receive” utility, present in the “release” folder, allows the configuration of the parameters below.

- Device Address
- Link Type
- Packet Type
- Packet Length
- Scrambler Seed
- BR/EDR Mode
- Receive Channel Index
- Transmit Channel Index
- Classic/LE Mode
- LE Channel Type
- Hopping Type
- Antenna Select

The “bt_util” utility, present in the “release” folder can be used to collect the receive statistics.

Command Usage

The “bt_receive” command usage is explained below.

```
./bt_receive <dev_addr> <link_type> <pkt_type> <pkt_length>
<scrambler_seed> <br_edr_mode> <rx_channel_index>
```

<tx_channel_index> <classic_le_mode> <le_channel_type> <hopping_type> <ant_sel>

The parameters for the “bt_receive” command have the same definition as the ones for the “bt_transmit” command.

Note:

After the reception starts using bt_receive, the following command can be given to stop the reception.

./bt_receive 0

The “bt_util” command usage is explained below.

./bt_util bt_stats <filename>

The <filename> parameter above is the file to which the statistics are saved. The following statistics are returned for every second.

crc_pass: The number of CRC passed packets received in the past 1 second

crc_fail: The number of CRC failed packets received in the past 1 second

id_pkt_rcvd: The number of ID packets received in the past 1 second

rssi: The RSSI value of the last received packet

Note:

After the reception starts using bt_util, to stop the reception, quit the bt_util process using Ctrl+C

Example

./bt_receive 0023a7010203 1 15 1021 0 3 10 10 1 1 0 2

The above command starts receiving 3DH5 packets with no hopping at 3 Mbps with the following configuration

Device address – 00:23:A7:01:02:03

Link type – 1(ACL)

Packet type – 0xF

Packet length – 1021 bytes

Scrambler seed – 0(Disable whitening)

BR/EDR mode – 3 (EDR mode with 3 Mbps)

Rx channel index – 10

Tx channel index – 10

Classic/LE mode – 1(Classic mode)

LE channel type – 1(Invalid in Classic mode)

Hopping type – 0(no hopping)

Antenna select – 2(RF_OUT_2/Antenna)

16.2.1.3 Continuous Wave Transmit Mode

The "bt_util" command is used to configure the device to transmit a continuous wave. The following parameters can be configured.

1) Channel Index
2) Start/Stop
3) Antenna Select

Command Usage

The command usage is explained below.

```
./bt_util cw_mode <channel_index> <start/stop> <ant_sel>
```

<channel_index>: Channel index, as per the Bluetooth standard.

<start/stop>: Start or Stop the Continuous Wave mode transmission.

- '0' – start the cw mode transmission
- '2' – stop the cw mode transmission

<ant_sel>: Select one of the two RF ports. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it is used to select between the two.

- '2' – RF_OUT_1/U.FL
- '3' – RF_OUT_2/Antenna

Example

```
./bt_util cw_mode 10 0 3
```

The above command starts continuous wave transmission with the following configuration

Channel index – 10

Start -0 (starts the transmission)

Antennal Select – 3(RF_OUT_2/Antenna)

16.2.1.4 Hopping Tests

Command Usage

The "bt_util" command is used to configure the device to transmit in the particular channels. The following parameters can be configured.

1) Start Channel
2) End Channel

Example

./bt_util afh_map 10 30

The above command starts transmitting only in the particular channels between 10 and 30.

Note:-

The above configuration when you already kept the device in transmit burst mode and random hopping enabled.For more details to configure the device in the transmit burst mode please refer to the section **16.2.1BT Transmit Tests**

17 ZigBee Performance Test Application Usage

performance of the module.

Note:

Open the common_insert.sh file present in the "release" folder using an editor like vim. Ensure that the DRIVER_MODE and COEX_MODE is set as below:

```
DRIVER_MODE = 2
```

```
COEX_MODE = 16 (for ZigBee)
```

Run the following command to install the Driver in Performance Test mode:

```
# sh zigb_enable.sh or wlan_zigb_insert.sh or onebox_insert.sh
script present in the "release" folder as per the instructions in
```
Section 4.1

Next, follow the instructions below to run the Transmit and Receive tests.

17.1 ZigBee Transmit Tests

The "zb_transmit" utility, present in the "release" folder, allows the configuration of the following parameters and starts the transmission of packets.

- Transmit Power
- Packet Length
- Transmit Mode
- Channel Index
- Number of Packets
- Delay

17.1.1 Zb_transmit Command Usage

The **"zb_transmit"** command usage is explained below.

```
./zb_transmit <tx_power> <pkt_length> <tx_mode> <channel_index>
<no_of_packets> <delay>
```

<tx_power>: This is the transmit power (in dBm) to be used by the module. The value should be between 0 and 18.

<pkt_length>: This is the length of the packet, in bytes, to be transmitted. Valid range for packet length is [6-127]

<tx_mode>: This parameter is used to choose between Burst and Continuous modes of transmission.

- '0' – Burst mode
- '1' – Continuous mode

<channel_index>: This parameter indicates the channel index, as per the ZigBee standard.

<no_of_packets>: This is the number of packets to be transmitted. This is valid only when the <tx_mode> is set to Burst mode (0).

<delay>[27]: Delay between packets in Burst mode. This parameter is used to introduce a delay between any two packets. The delay has to be specified in microseconds. If this value is 0, then the packets will be transmitted without any delay. This parameter is ignored in the case of Continuous mode of transmission. Receive Tests

In order to stop the transmit, the user must issue the following command,

- ./zb_transmit 0

The **"zb_util"** utility, present in the "release" folder, allows the configuration of the channel and collection of the receive statistics in that channel.

17.1.2 Zb_util Command Usage

The **"zb_util"** command usage is explained below. It has to be issued twice – first to set the channel and then to start/stop the collection of statistics. The statistics are reported once every second.

```
./zb_util set_channel <channel_index>
```

```
./zb_util zb_stats <filename>
```

<channel_index>: This parameter indicates the channel index, as per the ZigBee standard.

<filename>: This parameter indicates the file to which the statistics are saved.

The following statistics are returned every second.

crc_pass: The number of packets received which passed CRC check.

crc_fail: The number of packets received which failed CRC check.

rssi: The RSSI value of the last received packet.

17.1.2.1 Continuous Wave Transmit Mode

The **"zb_util"** command is used to configure the device to transmit a continuous wave. The following parameters can be configured.

- Channel Index
- Start/Stop
- Antenna Select

Command Usage

The command usage is explained below.

```
./zb_util cw_mode <channel_index> <start/stop> <ant_sel>
```

<channel_index>: Channel index, as per the zigbee standard.

<start/stop>: To Start or Stop the Continuous Wave mode transmission.

- '0' – start the cw mode transmission
- '2' – stop the cw mode transmission

<ant_sel>: Select one of the two RF ports. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it is used to select between the two.

- '2' – RF_OUT_1/U.FL
- '3' – RF_OUT_2/Antenna

[27] This parameter is currently not supported and should be set to 0.

Example

```
./zb_util cw_mode 26 0 2
```

The above command starts continuous wave transmission with the following configuration

- Channel index – 26
- 0 – Start(starts the transmission)
- Antennal Select – 2(RF_OUT_1/U.FL)

18 Appendix A: Configuration of Kernels 3.13 to 4.1.15

To ensure that the OneBox-Mobile software works on kernel versions 3.13 to 4.1.15, some configuration changes might be needed. These are explained in this section. Super user permissions are needed to make these changes.

18.1 SDIO Stack Options

If SDIO is the interface to the Host processor, it has to be ensured that the SDIO stack related modules are compiled in the kernel. If the SDIO stack modules are not present, follow the steps below to enable SDIO support in the kernel.

1. Navigate to the Linux kernel source folder. This is usually in `/usr/src/kernels/Linux-<kernel-version>`
2. Execute the '`make menuconfig`' command to open the Kernel Configuration menu.
3. Scroll down to the "Device Drivers --->" option and hit Enter.
4. In the new menu, scroll down to the "MMC/SD/SDIO card support ---->" option and press 'M' to modularize the "MMC/SD/SDIO card support" feature and hit Enter.
5. In the new menu, press 'M' to modularize the following options:
   - MMC block device driver
   - Secure Digital Host Controller Interface support
   - SDHCI support on PCI bus
6. Hit the Tab key to select Exit and hit Enter. Repeat this till you are asked whether you want to save the configuration. Select "Yes" and hit Enter. If the above options are already selected, the menuconfig screen will exit immediately.

18.2 Wireless Extension Tools

Wireless Extension tools like 'iwconfig' and 'iwpriv' are required for configuring the OneBox-Mobile software. Make sure the wireless extensions are enabled in the Linux kernel configuration file.

18.3 Bluetooth Stack Options

If Bluetooth is required, it has to be ensured that the Bluetooth modules are compiled in the kernel. If the Bluetooth modules are not present, follow the steps below to enable Bluetooth support in the kernel.

1. Navigate to the Linux kernel source folder. This is usually in `/usr/src/kernels/linux-<kernel-version>`
2. Execute the '`make menuconfig`' command to open the Kernel Configuration menu.
3. Scroll down to "Networking support --->" and hit Enter.
4. In the new menu, scroll down to the "Bluetooth subsystem support ---->" option and press 'M' to modularize the "Bluetooth subsystem support" feature and hit Enter.
5. In the new menu, press 'M' to modularize the following options:
   - RFCOMM Protocol support (enable the "RFCOMM TTY support" feature under this).
   - BNEP Protocol support (enable the "Multicast filter support" and "Broadcast filter support" features under this).
   - CMTP Protocol support
   - HIDP Protocol support

6. Hit the Tab key to select Exit and hit Enter. Repeat this till you are asked whether you want to save the configuration. Select “Yes” and hit Enter. If the above options are already selected, the menuconfig screen will exit immediately.

18.4 Kernel Compilation

Execute the steps listed below to compile the kernel with the above options enabled.

1. Navigate to the kernel source folder.
2. Execute the “make” command.
3. Execute the “make modules_install” command.
4. Execute the “make install” command. This ensures that the customized kernel is installed and the boot loader is updated appropriately.
5. Reboot the system to boot with the customized kernel.

19 Appendix B: Binary Files for Embedded Platforms

Redpine offers pre-built binary files of the OneBox-Mobile software to enable customers to evaluate the software on specific embedded processor platforms. The platforms supported for the current release are listed below:

- Freescale i.MX6
- Atmel ATSAM9G45 and AT91SAM9M10

The sections below explain the usage of the binaries on these platforms and also how to generate the binaries in case the OneBox-Mobile software source is available.

19.1 Freescale i.MX6

19.1.1 Hardware Requirements

- RS9113 Evaluation Kit. The contents are as follows:
  - RS9113 Module Evaluation Board
  - USB-to-microUSB Cable
  - SDIO Adaptor Cable
  - SPI Adaptor Cable
  - USB Pen Drive
- **i.MX 6SoloLite Evaluation Kit.** The kit contents are as follows:
  - Board: MCIMX6SLEVK
  - Cables: Micro USB-B-2-USB-Type A male, V2.0
  - Power supply: 100/240 V input, 5 V, 2.4 A output W/AC adaptor
  - Two SD cards: Programmed Android™
- Linux PC with Serial-to-USB drivers installed – this will be used to communicate with the i.MX6 platform.

19.1.2 Software Requirements

- Toolchain, BSP and Ubuntu Linux OS package for i.MX6 - Kernel version 3.0.35.
- OneBox-Mobile Software Release package

19.1.3 Hardware Setup

1. Connect the i.MX6 board to the Linux PC using the USB-to-microUSB cable – the cable has to be connected to port J26 (microUSB) of the board.
2. Connect the Redpine Evaluation Board (EVB) to the i.MX6 board using the SDIO adaptor or USB-to-microUSB cable (both included in the Redpine Evaluation Kit), depending on which Host Interface is needed.
   - i.MX6 + Redpine EVB with USB: Connect USB cable to J10 (USB) port of i.MX6
   - i.MX6 + Redpine EVB with SDIO: Connect SDIO Adapter to SD3 port of i.MX6
3. Preparing the MMC Card: An SD/MMC memory card is required to transfer the bootloader and kernel images for initializing the partition table and copy the root file system. This is included in the i.MX6 Evaluation Kit, but is programmed for Android OS. Refer to the i.MX_6SoloLite_EVK_Linux_User's_Guide.pdf document provided by Freescale as part of the **L3.0.35_4.1.0_LINUX_MMDOCS** documentation package, to prepare the SD/MMC card for Linux OS with kernel version 3.0.35.

19.1.4 Cross Compile and Copy OneBox-Mobile Software

If the OneBox-Mobile software's source is available, follow the steps listed in the **Compiling the Driver** section to cross compile the OneBox-Mobile software for i.MX6.

Assign the DEF_KERNEL_DIR variable in the Makefile as follows (assuming the kernel source is available in the "/lib/modules" folder):

```
DEF_KERNEL_DIR := /lib/modules/linux-3.0.35_SOLOLITE_hw
```

The "make" command for the i.MX6 is as follows, assuming the toolchain is present in the "/toolchain/opt/freescale" folder:

```
# make ARCH=arm
CROSS_COMPILE=/toolchain/opt/freescale/FWIOCUA0R1M1P1/TOOLS/cross/b
in/arm-mv5sft-linux-gnueabi-
```

Next, plugin the SD/MMC card to the PC and execute the commands given below to copy the pre-built binaries or the binaries generated above to the SD/MMC card.

```
# sudo mount /dev/sdb1 /mnt

# mkdir -p /mnt/home/rsi

# cp -r <path to OneBox-Mobile package>/host/release /mnt/home/rsi

# umount /mnt
```

Plugin the SD/MMC card into the i.MX6 board and follow the boot procedure. Once the bootup and login are completed, go to the /home/rsi/release folder and follow the procedure explained in the **Installing the Drive**r section.

19.2 Free scale i.MX53

19.2.1 Hardware Requirements

- RS9113 Evaluation Kit. The contents are as follows:
  - RS9113 Module Evaluation Board
  - USB-to-microUSB Cable
  - SDIO Adaptor Cable
  - SPI Adaptor Cable
  - USB Pen Drive
- **IMX53QSB: i.MX53 Quick Start Board**. The kit contents are as follows:
  - i.MX53-QUICK START Board
  - microSD Card preloaded with Ubuntu Demonstration Software
  - USB Cable (Standard-A to Micro-B connectors)
  - 5V/2.0A Power Supply
  - Quick Start Guide
  - Documentation DVD
- Linux PC with Serial port – this will be used to communicate with the processor platform.
- Serial RS232 Cable

19.2.2 Software Requirements

- Toolchain, BSP and Linux OS package for i.MX6 - Kernel version 2.6.35.

- OneBox-Mobile Software Release package
- minicom/GTKTerm on the Linux PC

19.2.3 Hardware Setup

1. Connect the i.MX53 board to the Linux PC using the Serial RS232 cable.
2. Connect the Redpine Evaluation Board (EVB) to the i.MX53 board using the SDIO adaptor or USB-to-microUSB cable (both included in the Redpine Evaluation Kit), depending on which Host Interface is needed.
3. Open a serial terminal program like minicom or GTKTerm and configure it with the following settings:
   - Baud Rate: 115200
   - Data bits: 8
   - Stop bits: 1
   - Parity: None
   - Flow Control:
4. Preparing the MMC Card: An SD/MMC memory card is required to transfer the bootloader and kernel images for initializing the partition table and copy the root file system. This is included in the i.MX53 Evaluation Kit. Refer to the i.MX53_EVK_Linux_BSP_UserGuide.pdf document provided by Freescale as part of the **IMX53_1109_LINUXDOCS_BUNDLE** documentation package, to prepare the SD/MMC card for Linux OS with kernel version 2.6.35.

19.2.4 Cross Compile and Copy OneBox-Mobile Software

If the OneBox-Mobile software's source is available, follow the steps listed in the **Compiling the Driver** section to cross compile the OneBox-Mobile software for i.MX53.

Assign the DEF_KERNEL_DIR variable in the Makefile as follows (assuming the kernel source is available in the "/lib/modules" folder):

```
DEF_KERNEL_DIR := /lib/modules/linux-2.6.35.3
```

The "make" command for the i.MX53 is as follows:

```
# make ARCH=arm
CROSS_COMPILE=/toolchain/opt/freescale/usr/local/gcc-4.4.4-glibc-
2.11.1-multilib-1.0/arm-fsl-linux-gnueabi/bin/arm-none-linux-
gnueabi-
```

Next, plugin the SD/MMC card to the PC and execute the commands given below to copy the pre-built binaries or the binaries generated above to the SD/MMC card.

```
# sudo mount /dev/sdb1 /mnt

# mkdir –p /mnt/home/rsi

# cp –r <path to OneBox-Mobile package>/host/release /mnt/home/rsi

# umount /mnt
```

Plugin the SD/MMC card into the i.MX53 board and follow the boot procedure. Once the bootup and login are completed, go to the /home/rsi/release folder and follow the procedure explained in the **Installing the Driver** section.

19.3 Atmel AT91SAM9G45 and AT91SAM9M10[28]

19.3.1 Hardware Requirements

- RS9113 Evaluation Kit. The contents are as follows:
  - RS9113 Module Evaluation Board
  - USB-to-microUSB Cable
  - SDIO Adaptor Cable
  - SPI Adaptor Cable
  - USB Pen Drive
- **SAM9M10-G45-EK - ARM926-based eMPU Eval** Kit. The kit contents are as follows:
  - Board: SAM9M10-G45-EK
  - Cables: One micro A/B-type USB cable, One serial RS232 cable, One RJ45 crossed cable
  - Power supply: Universal input AC/DC power supply, One 3V Lithium Battery type CR1225
- Linux PC with Serial port – this will be used to communicate with the processor platform.

19.3.2 Software Requirements

- Toolchain, BSP and Ubuntu Linux OS package for AT91SAM9G45 and AT91SAM9M10 - Kernel version 2.6.30
- OneBox-Mobile Software Release package
- minicom/GTKTerm on the Linux PC

19.3.3 Hardware Setup

1. Connect the Atmel board to the Linux PC using the Serial RS232 cable.
2. Connect the Redpine Evaluation Board (EVB) to the processor board using the SDIO adaptor or USB-to-microUSB cable (both included in the Redpine Evaluation Kit), depending on which Host Interface is needed.
3. Power up the processor board.
4. Open a serial terminal program like minicom or GTKTerm and configure it with the following settings:
   - Baud Rate: 115200
   - Data bits: 8
   - Stop bits: 1
   - Parity: None
   - Flow Control:
5. Connect the RJ45 cable between the PC and the board.
6. Follow the instructions given at **http://www.at91.com/linux4sam/bin/view/Linux4SAM/GettingStarted** to setup the board with the Linux OS kernel version 2.6.30.

[28] The Linux kernel version used on the Atmel AT91SAM9G45/M10 is 2.6.30. This is used to verify only the Wi-Fi mode. Bluetooth and ZigBee drivers are not compatible with this kernel version.

19.3.4 Cross Compile and Copy OneBox-Mobile Software

If the OneBox-Mobile software's source is available, follow the steps listed in the **Compiling the Driver** section to cross compile the OneBox-Mobile software for the Atmel processor.

Assign the DEF_KERNEL_DIR variable in the Makefile as follows (assuming the kernel source is available in the "/lib/modules" folder):

```
DEF_KERNEL_DIR := /lib/modules/linux-2.6.30
```

The **"make"** command for the AT91SAM9G45/M10 is as follows, assuming the toolchain is present in the "/toolchain/opt/atmel" folder:

```
# make ARCH=arm CROSS_COMPILE=/toolchain/opt/atmel/arm-
2007q1/bin/arm-none-linux-gnueabi-
```

Follow the given steps to copy the pre-built binaries or the binaries generate above to the Atmel processor platform.

1. Ensure that the Linux PC and the Atmel platform are in the same subnet. The IP of the processor platform can be assigned using the minicom/GTKTerm terminal.
2. # ifconfig <vap_name> <ip_address>
3. Example: `ifconfig eth0 192.168.1.24`
4. Power cycle the board.
5. Login as "root". There is no password required for the default credentials unless there has been a change done by the user.
6. Create a folder called "rsi" in the "/home" folder.
7. Copy the OneBox-Mobile binaries using the command below.
8. # scp –r release/ root@192.168.1.24:/home/rsi
9. Follow the procedure explained in the Installing the Driver section to start using the OneBox-Mobile software.

20 Appendix C: Using the Bluetooth Manager

The steps given below explains about the usage of the Bluetooth Manager in Fedora Core 18 on an x86 platform for pairing Bluetooth devices and transferring files.

1. Once the Bluetooth modules have been installed using wlan_bt_insert.sh or onebox_insert.sh script present in the “release” folder as per the instructions in Section 4.1hit the “Windows” button on the keyboard. You will see Bluetooth symbol at the bottom-right corner of the screen, as show in the image below.

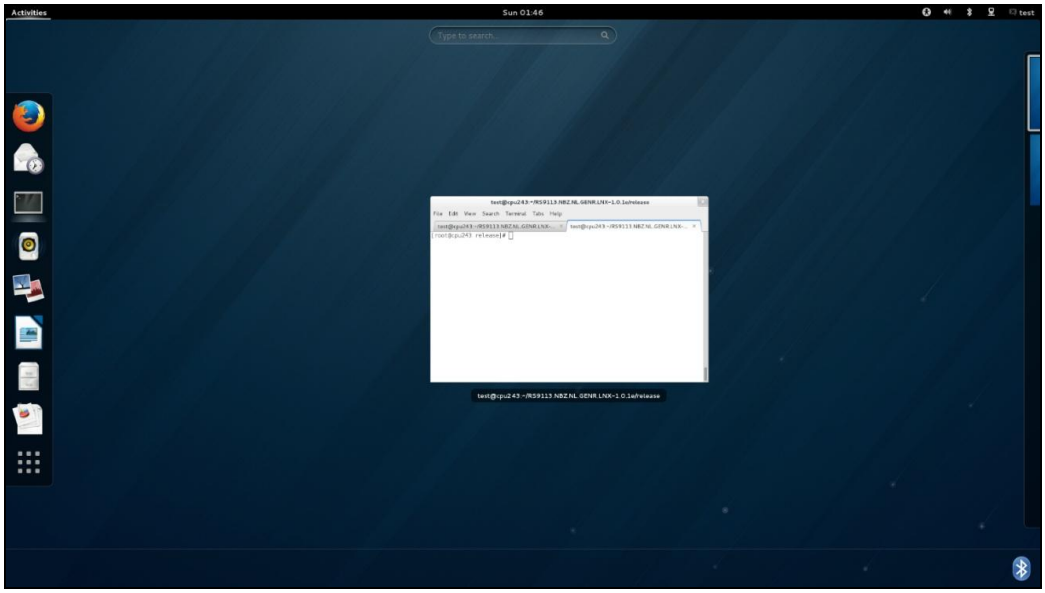

Figure 20-1: Invoking Bluetooth Manager

2. This will open the Bluetooth Manager as shown in the image below.

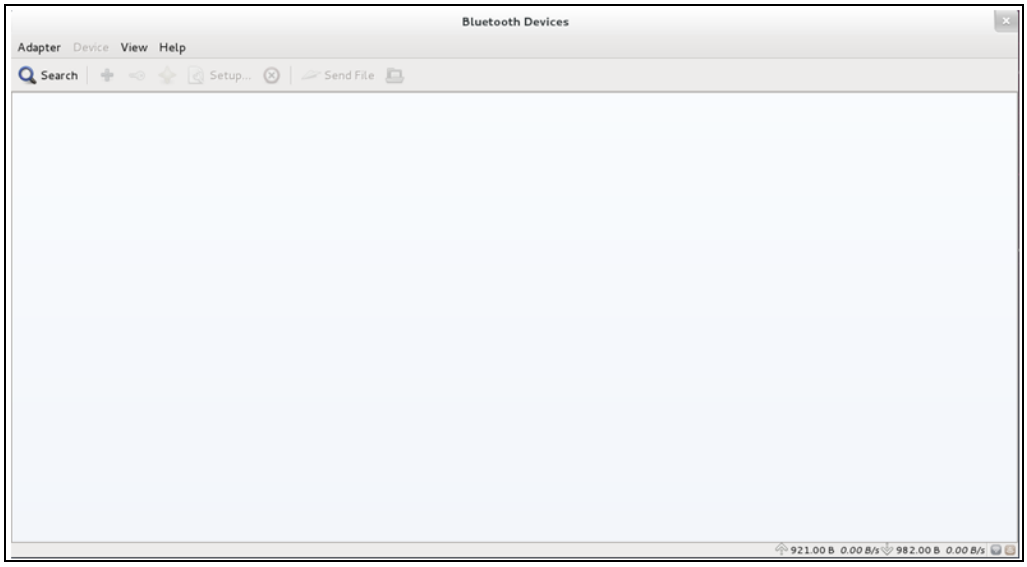

Figure 20-2: Bluetooth Manager Basic Window

3. Click on Search to start inquiry.

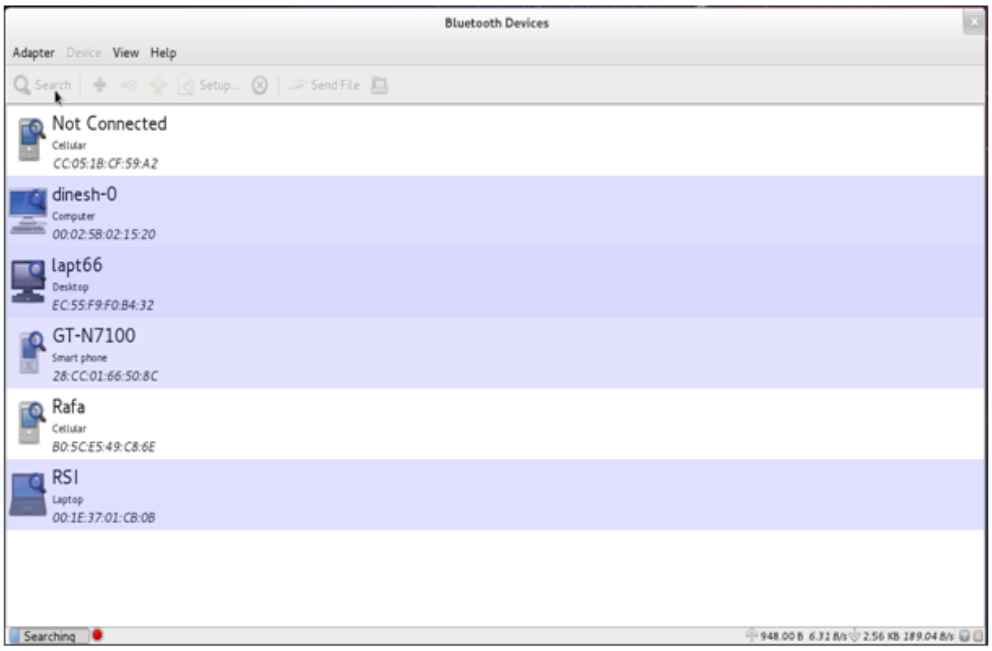

Figure 20-3: Click on Search to inquire

4. Select particular device, like your smartphone, right click and select Pair to pair with that device.

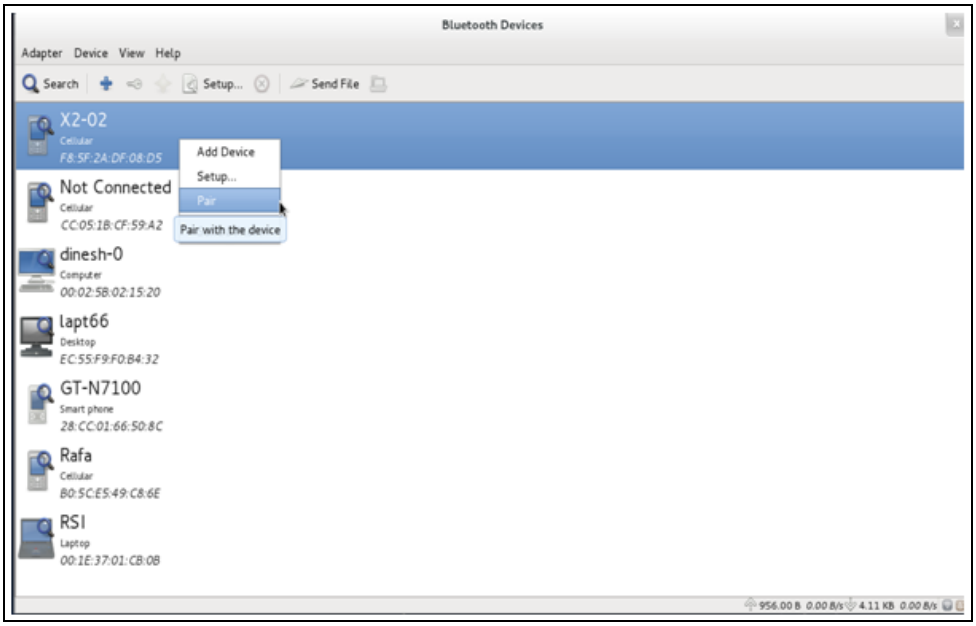

Figure 20-4: Pairing with a Device

5. After successfully pairing with the device, right-click on the device and select “Send a file” to send data to the device. You will be presented with a dialog box to select the file that you wish to send.

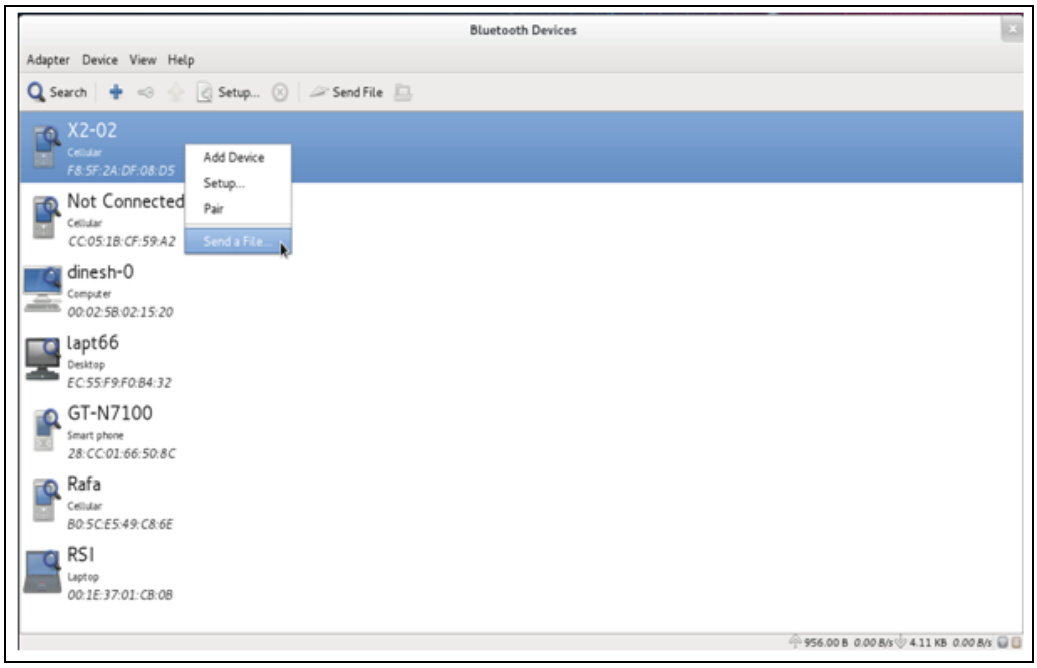

Figure 20-5: Send a File to a Device

21 Appendix D: Porting Driver to Android 4.4.3

This section describes the steps to be followed to port the RS9113 n-Link® driver to Android 4.4.3. The **i.MX 6SoloLite Evaluation** Kit is used as a reference platform in this section.

Note:

This section assumes the user has a direct Internet connection. In case you are behind a proxy firewall, contact your System Administrator for help on downloading the packages.

21.1 Requirements

1. PC with 8GB RAM, 16GB swap space
2. 160 GB Hard disk space
3. Ubuntu 12.04 LTS 64-bit Operating System with the GNU Make, awk and sed packages included
4. RS9113 Evaluation Kit. The contents are as follows:
   - RS9113 Module Evaluation Board
   - USB-to-microUSB Cable
   - SDIO Adaptor Cable
   - SPI Adaptor Cable
   - USB Pen Drive
5. i.MX 6SoloLite Evaluation Kit. The kit contents are as follows:
   - Board: MCIMX6SLEVK
   - Cables: Micro USB-B-2-USB-Type A male, V2.0
   - Power supply: 100/240 V input, 5 V, 2.4 A output W/AC adaptor
   - Two SD cards: Programmed Android™

21.2 Resolving Dependencies

The following dependencies need to be resolved before the Android Source Code can be compiled.

1. Download and install the Oracle JDK 6u45 package from **http://download.oracle.com/otn/java/jdk/6u45-b06/jdk-6u45-linux-x64-rpm.bin**
2. Run the commands below to download and install the remaining dependencies

```
$ sudo apt-get install git gnupg flex bison gperf \
build-essential zip curl libc6-dev \
libncurses5-dev:i386 x11proto-core-dev \
libx11-dev:i386 libreadline6-dev:i386 \
libgl1-mesa-glx:i386 libgl1-mesa-dev \
g++-multilib mingw32 tofrodos python-markdown \
libxml2-utils xsltproc zlib1g-dev:i386

$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 \
      /usr/lib/i386-linux-gnu/libGL.so
```

```
$ sudo apt-get install uuid uuid-dev
$ sudo apt-get install zlib1g-dev liblz-dev
$ sudo apt-get install liblzo2-2 liblzo2-dev
$ sudo add-apt-repository ppa:git-core/ppa
$ sudo apt-get install lzop
$ sudo apt-get update
$ sudo apt-get install git-core curl
$ sudo apt-get install u-boot-tools
```

21.3 Downloading Android Source Code and Patches

21.3.1 Downloading Android Source Code

The Android source code is maintained as more than 100 gits in the Android repository (android.googlesource.com).

Run the commands below to download the Android source code from Google repo.

```
$ cd ~
$ mkdir myandroid
$ mkdir bin
$ cd myandroid
$ curl http://commondatastorage.googleapis.com/git-repo-
downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ ~/bin/repo init -u \
      https://android.googlesource.com/platform/manifest -b \
      android-4.4.3_r1
$ ~/bin/repo sync
```

Note:

The last command starts the download of the source code and can take several hours to complete, depending on the Internet connection.

21.3.2 Downloading Android Kernel

The Android Kernel needs to be downloaded from the Freescale website. Run the commands below to download the kernel.

```
$ cd myandroid
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git \
      kernel_imx
$ cd kernel_imx
```

```
$ git checkout kk4.4.3_2.0.0-ga
```

Note:

The “git clone” command can take between 30 to 60 minutes to complete, depending on the Internet connection.

21.3.3 Downloading i.MX6 Bootloader

The i.MX6 U-boot bootloader needs to be downloaded from the Freescale Open Source git. Run the commands below to download the U-boot bootloader for i.MX6.

```
$ cd myandroid/bootable
$ cd bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git \
      uboot-imx
$ cd uboot-imx
$ git checkout kk4.4.3_2.0.0-ga
```

21.3.4 Download and Unpack i.MX6 Android Release Package

Download the Android 4.4.3 Release Package for i.MX6 (IMX6_KK443_200_ANDROID_SOURCE_BSP) from the link below (login is required):

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=IMXANDROID&fpsp=1&tab=Design_Tools_Tab

Once downloaded, run the commands below to unpack the Release Package to the “/opt” folder.

```
$ cp android_KK4.4.3_2.0.0-ga_core_source.tar.gz /opt
$ tar xzvf android_KK4.4.3_2.0.0-ga_core_source.tar.gz
$ cd android_KK4.4.3_2.0.0-ga_core_source/code/
$ tar xzvf KK4.4.3_2.0.0-ga.tar.gz
```

21.4 Applying Patches on Android Source Code

Run the commands below to apply the patches on the downloaded Android Source Code so that it is compiled for the i.MX6.

```
$ cd ~/myandroid
$ source /opt/android_KK4.4.3_2.0.0-\
      ga_core_source/code/KK4.4.3_2.0.0-ga/and_patch.sh
$ help
```

The output of the “help” command above should show that the “c_patch” function is available.

```
$ c_patch /opt/android_KK4.4.3_2.0.0-\
      ga_core_source/code/KK4.4.3_2.0.0-ga \
```

```
imx_KK4.4.3_2.0.0-ga
```

Here, "/opt/android_KK4.4.3_2.0.0-ga_source/code/KK4.4.3_2.0.0-ga" is the location of the patches – this folder is created when you unzip the release package.

"imx_KK4.4.3_2.0.0-ga" is the branch which will be created automatically for you to hold all patches (only in those existing Google gits). You can choose a branch name different from "imx_KK4.4.3_2.0.0-ga" too.

If everything is OK, **"c_patch"** will generate the following output to indicate the successful patch:

```
Success: Now you can build the Android code for FSL i.MX platform
```

21.5 Building the Android Source Code

After applying all i.MX patches, build the U-Boot, kernel, and Android images by using the commands listed below.

```
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch evk_6sl-user
$ make 2>&1 | tee build_evk_6sl_android.log
```

Note:

The last command can take several hours to complete, depending on the hardware configuration of the PC.

21.6 Cross Compiling the RS9113 n-Link® Driver

The process of cross compiling the RS9113 n-Link® driver for Android 4.4.3 on the i.MX 6SoloLite Evaluation Kit is explained in the steps listed below.

1. Modify the path assigned to the “DEF_KERNEL_DIR” variable in the Makefile in the “RS9113.NXX.NL.GEN.LNX.x.y.z/source/host” folder:

   ```
   # cd RS9113.NXX.NL.GEN.LNX.x.y.z/source/host
   # vim Makefile
   ```

   The DEF_KERNEL_DIR variable has to be assigned “/root/myandroid/kernel_imx/” as shown below:

   ```
   DEF_KERNEL_DIR:= /root/myandroid/kernel_imx
   ```

2. Next, use the “make” command given below to start the menuconfig utility.

   ```
   $ make ARCH=arm CROSS_COMPILE= \
      ~/myandroid/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6\
      /bin/arm-eabi
   ```

3. The “make” command starts the menuconfig utility. Select the operating system as Android. Refer to the steps listed in the section on Compiling the Driver for details on how to proceed with the compilation.

4. After successful compilation of the driver, copy the release folder into /root/myandroid/out/target/product/evk6sl/system/bin/
5. Run the commands below to integrate the driver's binaries into the Android image.

```
$ cd /root/myandroid/
$ source build/envsetup.sh
$ lunch evk_6sl-user
$ make 2>&1 | tee build_evk_6sl_android.log
```

21.7 RS9113 n-Link® Driver Integration with Android

This section lists the changes needed to integrate the RS9113 n-Link® Driver with Android.

wifi.c

The wifi.c file is present in the following path:

`/root/myandroid/hardware/libhardware_legacy/wifi/`

1. Add the following global declarations:

```
FILE *fp;
char out_iface[20];
char out[200];
```

2. Add the following lines at the start of the wifi.c file:

```
#define DRIVER_MODULE_WLAN "/system/bin/release/wlan.ko"
#define DRIVER_MODULE_WLAN_WEP "/system/bin/release/wlan_wep.ko"
#define DRIVER_MODULE_WLAN_TKIP
"/system/bin/release/wlan_tkip.ko"
#define DRIVER_MODULE_WLAN_CCMP
"/system/bin/release/wlan_ccmp.ko"
#define DRIVER_MODULE_WLAN_ACL "/system/bin/release/wlan_acl.ko"
#define DRIVER_MODULE_WLAN_XAUTH
"/system/bin/release/wlan_xauth.ko"
#define DRIVER_MODULE_WLAN_SCAN_STA
"/system/bin/release/wlan_scan_sta.ko"
#define DRIVER_MODULE_ONEBOX_NONGPL
"/system/bin/release/onebox_nongpl.ko"
#define DRIVER_MODULE_ONEBOX_GPL
"/system/bin/release/onebox_gpl.ko"
#define DRIVER_MODULE_ONEBOX_WLAN_NONGPL
"/system/bin/release/onebox_wlan_nongpl.ko"
#define DRIVER_MODULE_ONEBOX_WLAN_GPL
"/system/bin/release/onebox_wlan_gpl.ko"
#define DRIVER_MODULE_WLAN_NAME "wlan"
#define DRIVER_MODULE_WLAN_WEP_NAME  "wlan_wep"
#define DRIVER_MODULE_WLAN_TKIP_NAME "wlan_tkip"
```

```
#define DRIVER_MODULE_WLAN_CCMP_NAME "wlan_ccmp"
#define DRIVER_MODULE_WLAN_ACL_NAME "wlan_acl"
#define DRIVER_MODULE_WLAN_XAUTH_NAME "wlan_xauth"
#define DRIVER_MODULE_WLAN_SCAN_STA_NAME "wlan_scan_sta"
#define DRIVER_MODULE_ONEBOX_NONGPL_NAME "onebox_nongpl"
#define DRIVER_MODULE_ONEBOX_GPL_NAME "onebox_gpl"
#define DRIVER_MODULE_ONEBOX_WLAN_NONGPL_NAME \
   "onebox_wlan_nongpl"
#define DRIVER_MODULE_ONEBOX_WLAN_GPL_NAME \
   "onebox_wlan_gpl"
#define WIFI_TEST_INTERFACE "wlan0"
#define WIFI_DRIVER_MODULE_ARG "driver_mode=1 \
   firmware_path=/system/bin/release/firmware/ \
   onebox_zone_enabled=0x1  coex_mode=1 ta_aggr=4 \
   skip_fw_load=0 fw_load_mode=1"
```

3. Comment the following lines:

```
//#ifndef WIFI_DRIVER_MODULE_ARG
//#define WIFI_DRIVER_MODULE_ARG                    ""
//#endif
//#ifndef WIFI_FIRMWARE_LOADER
//#define WIFI_FIRMWARE_LOADER              ""
//#endif
//#define WIFI_TEST_INTERFACE               "sta"
//#ifndef WIFI_DRIVER_FW_PATH_STA
//#define WIFI_DRIVER_FW_PATH_STA           NULL
//#endif
//#ifndef WIFI_DRIVER_FW_PATH_AP
//#define WIFI_DRIVER_FW_PATH_AP       NULL
//#endif
//#ifndef WIFI_DRIVER_FW_PATH_P2P
//#define WIFI_DRIVER_FW_PATH_P2P           NULL
//#endif
//#ifndef WIFI_DRIVER_FW_PATH_PARAM
//#define WIFI_DRIVER_FW_PATH_PARAM
   "/sys/module/wlan/parameters/fwpath"
//#endif
```

4. Change the following line to "`IFNAME=wlan0`":

```
static const char IFNAME[] = "IFNAME=";
```

5. Comment the following line:

```
static const char FIRMWARE_LOADER[] = WIFI_FIRMWARE_LOADER;
```

6. Comment all code in the "`wifi_load_driver()`" function and add the following lines:

```
If (is_wifi_driver_loaded()) {
    return 0;
}
ALOGE("Loading WiFi driver here\n");
if(insmod(DRIVER_MODULE_ONEBOX_NONGPL , \
   WI-FI_DRIVER_MODULE_ARG ) < 0)
   return -1;
ALOGE("*****onebox_nongpl.ko inserted\n");
if(insmod(DRIVER_MODULE_ONEBOX_GPL,"") < 0)
   return -1;
if (insmod(DRIVER_MODULE_WLAN,"") < 0)
   return -1;
ALOGE("*****wlan.ko inserted\n");
if(insmod(DRIVER_MODULE_WLAN_WEP,"") < 0)
   return -1;
ALOGE("****wlan_wep.ko inserted\n");
if(insmod(DRIVER_MODULE_WLAN_TKIP,"") < 0)
   return -1;
ALOGE("*****wlan_tkip.ko inserted\n");
if(insmod(DRIVER_MODULE_WLAN_CCMP,"") < 0)
   return -1;
ALOGE("*****wlan_ccmp.ko inserted");
if(insmod(DRIVER_MODULE_WLAN_ACL,"") < 0)
   return -1;
ALOGE("********wlan_acl.ko inserted\n");
if(insmod(DRIVER_MODULE_WLAN_XAUTH,"") < 0)
   return -1;
ALOGE("*****wlan_xauth.ko inserted\n");
if(insmod(DRIVER_MODULE_WLAN_SCAN_STA,"") < 0)
   return -1;
ALOGE("***wlan_scan_sta.ko inserted\n");
if(insmod(DRIVER_MODULE_ONEBOX_WLAN_NONGPL,"") < 0)
   return -1;
```

```
if(insmod(DRIVER_MODULE_ONEBOX_WLAN_GPL,"") < 0)
   return -1;
ALOGE("***********onebox_gpl.ko\n********* ALL ko inserted\
   successfully\n");
property_set(DRIVER_PROP_NAME, "ok");
return 0;
```

7. Comment the lines after "`#else`" in the "`wifi_unload_driver()`" function and add the following lines:

Note:

Indentation has been removed in the code below to make it readable in this document.

```
ALOGE("Beginning to unload driver here!!\n");
if(rmmod(DRIVER_MODULE_ONEBOX_WLAN_GPL_NAME)==0){
ALOGE("********onebox_gpl.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_ONEBOX_WLAN_NONGPL_NAME)==0){
ALOGE("*******onebox_nongpl.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_SCAN_STA_NAME)==0){
ALOGE("********wlan_s can_sta.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_XAUTH_NAME)==0){
ALOGE("*****wlan_xauth.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_ACL_NAME)==0){
ALOGE("****wlan_acl.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_CCMP_NAME)==0){
ALOGE("******wlan_ccmp.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_TKIP_NAME)==0){
ALOGE ("******wlan_tkip.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_WEP_NAME)==0){
ALOGE("*****wlan_wep.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_WLAN_NAME)==0){
ALOGE("*****wlan.ko rmmod success\n all ko rmmod \
   success\n");
if(rmmod(DRIVER_MODULE_ONEBOX_GPL_NAME)==0){
ALOGE("*****onebox_gpl.ko rmmod success\n");
if(rmmod(DRIVER_MODULE_ONEBOX_NONGPL_NAME)==0){
ALOGE("*****onebox_nongpl.ko rmmod success\n");
property_set(DRIVER_PROP_NAME,"unloaded");
return 0;
```

```
}}}}}}}}}}}
else{
ALOGE("******** *****ko rmmod failed\n");
return -1;
}
return -1;
```

8. Add the following line in beginning of the "`wifi_start_supplicant (int p2p_supported)`" and "`wifi_stop_supplicant (int p2p_supported)`" functions:

```
p2p_supported = 0;
```

9. Add the following line in the "`wifi_wait_on_socket (char *buf, size_t buflen)`" function:

```
char new_buf[5000];
```

10. Add the following lines in the "`wifi_wait_on_socket (char *buf, size_t buflen)`" function after the comments below:

```
/* where N is the message level in numerical form (0=VERBOSE,
1=DEBUG,
* etc.) and XXX is the event name. The level information is not
useful
* to us, so strip it off.
*/
ALOGE("Received the following from the \
       supplicant: %s\n", buf);
strncpy(new_buf, IFNAME, IFNAMELEN);
strcpy(&new_buf[IFNAMELEN], buf);
strcpy(buf, new_buf);
```

11. Add the following lines under the switch case "`WIFI_GET_FW_PATH_STA`" in the "`wifi_get_fw_path (int fw_type)`" function:

```
fp = popen("/system/bin/onebox_util rpine0 create_vap \
         wlan0 sta sw_bmiss", "r");
if(fp==NULL)
   ALOGE("Failed to run command\n" );
while (fgets(out, sizeof(out)-1, fp) != NULL);
ALOGE("Vap Creation Output:\n%s\n",out);
usleep(1000000);
pclose(fp);
fp = popen("busybox ifconfig -a | grep \"00:23:A7\" | grep\
    \"wlan\" | busybox awk '{printf $1}'", "r");
if(fp==NULL)
```

```
    ALOGE("Failed to run command\n" );
  while (fgets(out_iface, sizeof(out_iface)-1, fp) != NULL);
  ALOGE("****Inside STA wifi.c %s out_iface is %s", \
    __func__,out_iface);
  pclose(fp);
```

Comment the following line at the end of the switch case

```
return NULL;
```

12. Return "NULL" instead of "WIFI_DRIVER_FW_PATH_P2P" under the switch case "WIFI_GET_FW_PATH_P2P" in the "wifi_get_fw_path (int fw_type)" function.
13. Comment the following line under the switch case "WIFI_GET_FW_PATH_AP"

```
return WIFI_DRIVER_FW_PATH_AP;
```

14. Comment all lines in the "wifi_change_fw_path (const char *fwpath)" function except the "return" statement. Add "ret=0" under the declarations.

Boardconfig.mk

The Boardconfig.mk file is present in the following path:

```
/root/myandroid/device/fsl/evk_6sl
```

Modify the Boardconfig.mk file as per the information below:

```
BOARD_WLAN_VENDOR=REDPINE
TARGET_BOOTLOADER_BOARD_NAME := EVK
PRODUCT_MODEL := EVK_MX6SL
BOARD_WLAN_DEVICE             := RSI
BOARD_WLAN_VENDOR             := REDPINE
WPA_SUPPLICANT_VERSION        := VER_0_8_X
BOARD_WPA_SUPPLICANT_DRIVER   := BSD
#BOARD_HOSTAPD_DRIVER         := BSD
#BOARD_HOSTAPD_PRIVATE_LIB_QCOM             :=
lib_driver_cmd_qcwcn
#BOARD_HOSTAPD_PRIVATE_LIB_RTL              := lib_driver_cmd_rtl
#BOARD_WPA_SUPPLICANT_PRIVATE_LIB_QCOM      :=
lib_driver_cmd_qcwcn
#BOARD_WPA_SUPPLICANT_PRIVATE_LIB_RTL       := lib_driver_cmd_rtl
#for redpine vendor
ifeq ($(BOARD_WLAN_VENDOR),REDPINE)
#BOARD_HOSTAPD_PRIVATE_LIB                 := private_lib_driver_cmd
BOARD_WPA_SUPPLICANT_PRIVATE_LIB          := private_lib_driver_cmd
WPA_SUPPLICANT_VERSION                    := VER_0_8_X
#HOSTAPD_VERSION                          := VER_0_8_X
```

```
#WIFI_DRIVER_MODULE_PATH                    :=
"/system/lib/modules/iwlagn.ko"

#WIFI_DRIVER_MODULE_NAME                    := "iwlagn"

#WIFI_DRIVER_MODULE_PATH                    ?= auto

endif
```

Please note the lines other than those mentioned above should be as it is in the original file.

init.rc

The init.rc file is present in the following path:

```
/root/myandroid/out/target/product/evk_6sl/root
```

1. Add the following lines:

```
#Android socket changes

   mkdir /system/etc/wifi 0770 wifi wifi

   chmod 0770 /system/etc/wifi

   chmod 0660 /system/etc/wifi/wpa_supplicant.conf

   chown wifi wifi /system/etc/wifi/wpa_supplicant.conf

   #wpa_supplicant control socket for android wifi.c (android
   private socket)

   mkdir /data/misc/wifi 0770 wifi wifi

   mkdir /data/misc/wifi/sockets 0770 wifi wifi

   chmod 0770 /data/misc/wifi

   chmod 0660 /data/misc/wifi/wpa_supplicant.conf

   chown wifi wifi /data/misc/wifi

   chown wifi wifi /data/misc/wifi/wpa_supplicant.conf

#Endof Android socket changes
```

2. Comment the following line:

```
setprop.wifi.ap.interface wlan0
```

3. Add the following lines at the end of the init.rc file:

```
service wpa_supplicant /system/bin/wpa_supplicant -iwlan0 \
   -Dbsd -c /system/etc/wifi/wpa_supplicant.conf -dd

socket wpa_wlan0 dgram 660 wifi wifi

group system wifi inet

disabled

oneshot
```

WifiStateMachine.java

The WifiStateMachine.java file is present in the following path

```
/root/myandroid/frameworks/base/wifi/java/android/net/wifi/
```

1. Comment the following lines:

```
//mP2pSupported=mContext.getPackageManager().hasSystemFeature(Pa
ckageManager.FEATURE_WIFI_DIRECT);
```

2. Add the following line in the function "WifiStateMachine"

```
mP2pSupported = false;
```

WifiP2pServive.java

The WifiP2pService.java file is present in the following path:

```
/root/myandroid/frameworks/base/wifi/java/android/net/wifi/p2p
```

1. Comment the following lines:

```
//mP2pSupported=mContext.getPackageManager().hasSystemFeature(Pa
ckageManager.FEATURE_WIFI_DIRECT);
```

2. Add the following line in the function "WifiP2pService"

```
mP2pSupported = false;
```

WifiApConfigStore.java

The WifiApconfigStore.java file is present in the following path:

```
/root/myandroid/frameworks/base/wifi/java/android/net/wifi/
```

1. Add the following lines under the "import" section:

```
import  java.io.FileWriter;
```

2. Edit the code in the "writeApConfiguration" function as indicated below:

```
DataOutputStream out = null;
FileWriter fw = null;
try {
   out = new DataOutputStream(new BufferedOutputStream \
         (new FileOutputStream(AP_CONFIG_FILE)));
   fw = new FileWriter(AP_CONFIG_FILE);//Edited by RT added
   // out.writeInt(AP_CONFIG_FILE_VERSION);
   // out.writeUTF(config.SSID);
   // int authType = config.getAuthType();
   //out.writeInt(authType);
   //if(authType != KeyMgmt.NONE) {
   //    out.writeUTF(config.preSharedKey);
   //}
   int authType = config.getAuthType();
   fw.write("update_config=1\n");
   fw.write("ctrl_interface=DIR=/data/misc/wifi/hostapd \
         GROUP=wifi\n");
   fw.write("ap_scan=2\n");
   //fw.write("uuid=12345678-9abc-def0-1234-\
         56789abcdef0\n");
```

```
    //fw.write("device_name=RSI_ANDROID_D EVICE\n" );
    //fw.write("manufacturer=Redpine Signals INC\n" );
    //fw.write("model_number=9113\n");
    //fw.write("manufacturer=Redpine Signals INC\n" );
    //fw.write("model_number=9113\n");
    //fw.write("serial_number=03\n");
    //fw.write("device_type=1-0050F204-1\n");
    //fw.write("os_version=01020300\n");
    //fw.write("config_methods=display push_button \
          keypad\n");
    //fw.write("wps_cred_processing=0\n");
    if(authType != KeyMgmt.NONE) {
       fw.write("network={\n");
       fw.write("ssid=" + "\"" + config.SSID + "\"");
       fw.write("\nfrequency=2437\n");
       fw.write("proto=WPA2 WPA\n");
       fw.write("key_mgmt=WPA-PSK\n");
       fw.write("pairwise=TKIP CCMP\n");
       fw.write("group=TKIP CCMP\n");
       fw.write("psk=" + "\"" + config.preSharedKey + "\"");
       fw.write("\nmode=2\n");
       fw.write("}\n");
    }
    else {
       fw.write("network={\n");
       fw.write("ssid=" + "\"" + config.SSID + "\"");
       fw.write("\nfrequency=2437\n");
       fw.write("key_mgmt=NONE\n");
       fw.write("mode=2\n");
       fw.write("}\n");
    }
} catch (IOException e) {
    Log.e(TAG, "Error writing hotspot configuration" + e);
    } finally {
       if (out != null) {
          try {
```

```
                out.close();
                fw.close(); //edited by RT added
             } catch (IOException e) {}
          }
       }
    }
```

3. Comment the following lines:

```
out.writeInt(AP_CONFIG_FILE_VERSION);
out.writeUTF(config.SSID);
int authType = config.getAuthType();
out.writeInt(authType);
if(authType != KeyMgmt.NONE) {
    out.writeUTF(config.preSharedKey);
}
```

NetworkManagementService.java

The NetworkManagementService.java file is present in the following path:

```
/root/myandroid/frameworks/base/services/java/com/android/server
```

1. Comment the line under the “startAccessPoint()” function as shown below:

```
catch (NativeDaemonConnectorException e) {
   // throw e.rethrowAsParcelableException();
}
```

2. Comment the line under “stopAccessPoint()” function as shown below:

```
catch (NativeDaemonConnectorException e) {
   // throw e.rethrowAsParcelableException();
}
```

SoftapController.cpp

The SoftapController.cpp file is present in the following path:

```
/root/myandroid/system/netd
```

1. Add the following global declarations:

```
static const char SOFTAP_CONF_FILE[] =
"/data/misc/wifi/softap.conf";
static const char driver_name[] = "bsd";
```

2. Edit the function “SoftapController::startSoftap()“ as shown below:

```
pid_t pid = 1;
char driver_vendor[PROPERTY_VALUE_MAX] = {'\0'};
int ret = 0,err;
int mSock;
if (mPid) {
```

```
    ALOGE("SoftAP is already running");
    return ResponseCode::SoftapStatusResult;
}
if (mSock < 0) {
    ALOGE("Softap startap - failed to open socket");
    return -1;
}
if ((pid = fork()) < 0) {
    ALOGE("fork failed (%s)", strerror(errno));
    return ResponseCode::ServiceStartFailed;
    return -1;
}
if (!pid) {
    ensure_entropy_file_exists();
#if 0
    if ((property_get(DRIVER_VENDOR_NAME, driver_vendor, NULL)
          && (strcmp(driver_vendor, "realtek") == 0))?
       (execl(HOSTAPD_BIN_FILE_RTL, HOSTAPD_BIN_FILE,
             "-e", WIFI_ENTROPY_FILE,
             HOSTAPD_CONF_FILE, (char *) NULL)):
       (execl(HOSTAPD_BIN_FILE, HOSTAPD_BIN_FILE,
             "-e", WIFI_ENTROPY_FILE,
             HOSTAPD_CONF_FILE, (char *) NULL)
       )) {
       ALOGE("execl failed (%s)", strerror(errno));
    }
    ALOGE("SoftAP failed to start");
    return ResponseCode::ServiceStartFailed;
#endif
    err = chown(SOFTAP_CONF_FILE,AID_WIFI,AID_WIFI);
    err = chmod(SOFTAP_CONF_FILE,0777);
    err = \
       chown("/data/misc/wifi/softap.conf",AID_WIFI,AID_WIFI);
    err = chmod("/data/misc/wifi/softap.conf",0777);
    if (execl("/system/bin/hostapd", "/system/bin/hostapd", "-
      iwlan0", "-D", driver_name, "-e", WIFI_ENTROPY_FILE, "-c",
      SOFTAP_CONF_FILE, "-dddddt",(char *) NULL)) {
```

```
            ALOGE("execl failed (%s)", strerror(errno));
        }
        // system("/system/bin/hostapd -iwlan0 -Dbsd -c \
            /data/misc/wifi/hostapd.conf -ddddt &");
        ALOGE("Should never get here!");
        return -1;
        } else {
            usleep(2000000);
            system("iwconfig wlan0 freq 6");
            mPid = pid;
            ALOGD("SoftAP started successfully");
            usleep(AP_BSS_START_DELAY);
        }
        //return ResponseCode::SoftapStatusResult;
        return ret;
    }
```

3. Edit the function “SoftapController::fwReloadSoftap (int argc, char *argv[])” as shown below:

```
int ret=0, i = 0;
char *fwpath = NULL;
char *iface;
int mSock;
if (mSock < 0) {
    ALOGE("Softap fwrealod - failed to open socket");
    return -1;
}
if (argc < 4) {
    ALOGE("SoftAP fwreload is missing arguments. Please use:
        softap <wlan iface>        <AP|P2P|STA>");
    // return ResponseCode::CommandSyntaxError;
    return -1;
}
iface = argv[2];
if (strcmp(argv[3], "AP") == 0) {
    fwpath = (char *)wifi_get_fw_path(WIFI_GET_FW_PATH_AP);
} else if (strcmp(argv[3], "P2P") == 0) {
    fwpath = (char *)wifi_get_fw_path(WIFI_GET_FW_PATH_P2P);
```

```
} else if (strcmp(argv[3], "STA") == 0) {
   fwpath = (char *)wifi_get_fw_path(WIFI_GET_FW_PATH_STA);
}
#if 0
if (!fwpath)
   return ResponseCode::CommandParameterError;
if (wifi_change_fw_path((const char *)fwpath)) {
   ALOGE("Softap fwReload failed");
   return ResponseCode::OperationFailed;
}
else {
   ALOGD("Softap fwReload - Ok");
}
#endif
return ret;
```

WifiMonitor.java

The WifiMonitor.java file is present in the following path:

`/root/myandroid/frameworks/base/wifi/java/android/net/wifi/`

Edit some of the lines in "`public void run()`" function as shown below:

1. Change "`space`" to "`12`" in the line below:
   `iface = eventStr.substring(7,12);`
2. Change "`iface`" to "`wlan0`" in the line appearing exactly below the line:
   `iface = eventStr.substring(7,12);`
   `m = mWifiMonitorSingleton.getMonitor("wlan0");`
3. Change "`space+1`" to "`15`" in the line below:
   `eventStr = eventStr.substring(15);`

Supplicant changes

The supplicant folder is present in the path /root/myandroid/external/wpa_supplicant_8

Android.mk

The Android.mk file is present in the following path:

`/root/myandroid/external/wpa_supplicant_8/wpa_supplicant`

Remove or comment the following lines in this file

```
ifeq ($(BOARD_WLAN_DEVICE), bcmdhd)
L_CFLAGS += -DANDROID_P2P
L_CFLAGS += -DP2P_CONCURRENT_SEARCH_DELAY=0
endif
```

```
ifeq ($(BOARD_WLAN_DEVICE),$(filter $(BOARD_WLAN_DEVICE), qcwcn
UNITE))
L_CFLAGS += -DANDROID_P2P
endif
ifeq ($(BOARD_WLAN_DEVICE), mrvl)
L_CFLAGS += -DANDROID_P2P
endif
ifdef CONFIG_P2P
OBJS += p2p_supplicant.c
OBJS += src/p2p/p2p.c
OBJS += src/p2p/p2p_utils.c
OBJS += src/p2p/p2p_parse.c
OBJS += src/p2p/p2p_build.c
OBJS += src/p2p/p2p_go_neg.c
OBJS += src/p2p/p2p_sd.c
OBJS += src/p2p/p2p_pd.c
OBJS += src/p2p/p2p_invitation.c
OBJS += src/p2p/p2p_dev_disc.c
OBJS += src/p2p/p2p_group.c
OBJS += src/ap/p2p_hostapd.c
L_CFLAGS += -DCONFIG_P2P
NEED_GAS=y
NEED_OFFCHANNEL=y
NEED_80211_COMMON=y
CONFIG_WPS=y
CONFIG_AP=y
ifdef CONFIG_P2P_STRICT
L_CFLAGS += -DCONFIG_P2P_STRICT
endif
endif
ifdef CONFIG_WIFI_DISPLAY
L_CFLAGS += -DCONFIG_WIFI_DISPLAY
OBJS += wifi_display.c
endif
ifdef CONFIG_P2P
DBUS_OBJS += dbus/dbus_new_handlers_p2p.c
endif
```

config.c

The config.c file is present in the following path:

```
/root/myandroid/external/wpa_supplicant_8/wpa_supplicant
```

Add "4" after "NONE" in the "`wpa_config_parse_key_mgmt()`" as shown below:

```
else if (os_strncmp(start, "NONE",4) == 0)
```

ctrl_iface.c

The ctrl_iface.c file is present in the following path:

```
/root/myandroid/external/wpa_supplicant_8/wpa_supplicant
```

1. Edit the "`print_bss_info()`" function as shown below:

```
// if (mask & WPA_BSS_MASK_DELIM) {
#ifdef ANDROID
if (1){
   ret = os_snprintf(pos, end - pos, "====\n");
   if (ret < 0 || ret >= end - pos)
      return 0;
   pos += ret;
}
#endif
// }
```

2. Comment the following line in the "`wpa_supplicant_ctrl_iface_bss()`" function:

```
//p2p_set_country(p2p, country);
```

3. Add the following lines in the "`wpa_supplicant_ctrl_iface_process()`" function :

```
char  new_buf[100];
int length = 0;
```

4. Add the following lines in the "`wpa_supplicant_ctrl_iface_process()`" function before the "os_memcpy(reply, "OK\n", 3);" line:

```
strncpy(new_buf, &buf[13], strlen(buf) - 13);
new_buf[strlen(buf)-13] = '\0';
strncpy(buf, new_buf, strlen(new_buf));
length = strlen(new_buf);
buf[length] = '\0';
```

5. Edit a line in the "`wpa_supplicant_ctrl_iface_process()`" function as shown below:

   Original line:

```
else if (os_strncmp(buf, "ADD_NETWORK") == 0)
```

   Modified line:

```
else if (os_strncmp(buf, "ADD_NETWORK", 11) == 0)
```

6. Comment the following line in the "`wpa_supplicant_ctrl_iface_process()`" function after the below line:

   if (os_strncasecmp(buf + 7, "P2P_DISABLE", 11) == 0)

   `//wpas_p2p_stop_find(wpa_s);`

7. In "`wpa_supplicant_ctrl_iface_process()`" function add the line after } else if (os_strcmp(buf, "SCAN") == 0 ||

   `os_strncmp(buf, "SCAN", 4) == 0 ||`

defconfig

The defconfig file is present in the following path:

`/root/myandroid/external/wpa_supplicant_8/wpa_supplicant`

1. Ensure that the line below is uncommented:

   `CONFIG_DRIVER_BSD=y`

2. Ensure that the lines below are commented:

   `#CONFIG_DRIVER_WEXT=y`

   `#CONFIG_DRIVER_NL80211=y`

WPA Supplicant Makefile

The WPA Supplicant Makefile is present in the following path:

`/root/myandroid/external/wpa_supplicant_8/wpa_supplicant/`

1. Comment or remove the following lines in this file:

```
ifdef CONFIG_P2P
OBJS += p2p_supplicant.o
OBJS += ../src/p2p/p2p.o
OBJS += ../src/p2p/p2p_utils.o
OBJS += ../src/p2p/p2p_parse.o
OBJS += ../src/p2p/p2p_build.o
OBJS += ../src/p2p/p2p_go_neg.o
OBJS += ../src/p2p/p2p_sd.o
OBJS += ../src/p2p/p2p_pd.o
OBJS += ../src/p2p/p2p_invitation.o
OBJS += ../src/p2p/p2p_dev_disc.o
OBJS += ../src/p2p/p2p_group.o
OBJS += ../src/ap/p2p_hostapd.o
CFLAGS += -DCONFIG_P2P
NEED_GAS=y
NEED_OFFCHANNEL=y
NEED_80211_COMMON=y
CONFIG_WPS=y
```

```
CONFIG_AP=y
ifdef CONFIG_P2P_STRICT
CFLAGS += -DCONFIG_P2P_STRICT
endif
endif
ifdef CONFIG_WIFI_DISPLAY
CFLAGS += -DCONFIG_WIFI_DISPLAY
OBJS += wifi_display.o
endif
```

2. Comment the following lines after the "`ifdef CONFIG_WPS`"
   `DBUS_OBJS += dbus/dbus_new_handlers_wps.o`" line:

```
ifdef CONFIG_P2P
DBUS_OBJS += dbus/dbus_new_handlers_p2p.o
endif
```

ctrl_iface_ap.c

The ctrl_iface_ap.c is present in the following path:

`/root/myandroid/external/wpa_suppplicant_8/wpa_suppplicant/src/ap`

Comment the following lines in the "`hostapd_ctrl_iface_sta_mib()`" function:

```
//res = hostapd_p2p_get_mib_sta(hapd, sta, buf + len, buflen -
len);
//if (res >= 0)
 //len += res;
```

driver_bsd.c

The driver_bsd.c file is present in the following path:

`/root/myandroid/external/wpa_supplicant_8/wpa_supplicant/src/drivers/`

1. Copy the file driver_bsd.c file from the "RS9113.NXX.NL.GEN.LNX.x.y.z/host/wlan/supplicant/src/drivers/" folder to the "root/myandroid/external/wpa_supplicant_8/wpa_supplicant/src/drivers/" folder.
2. Comment of the include directive in the copied driver_bsd.c file as shown below:
   `//#include <sys/sysctl.h>`
3. Comment all the lines in the "`rtbuf_len()`" function and add the following line:
   `size_t len = 2048;`
4. In the "`wpa_driver_bsd_associate()`" function
   Edit the lines as shown below:

   - `privacy = !(params->pairwise_suite == 0 &&`
     `params->group_suite == 0 &&`
     `params->key_mgmt_suite == 2 &&`

```
params->wpa_ie_len == 0);
```

- Comment the following switch case line in this function

```
case IEEE80211_MODE_P2P_GO:
```

- Change the following line

```
if ((params->mode == IEEE80211_MODE_AP) || (params->mode
== IEEE80211_MODE_P2P_GO)) {
```

to "`if ((params->mode == IEEE80211_MODE_AP)) {`"

5. Comment the following lines in the "`wpa_driver_bsd_scan()`" function:

```
if (wpa_driver_bsd_set_wpa(drv, 1) < 0) {
   wpa_printf(MSG_ERROR, "%s: failed to set wpa: %s", __func__,
      strerror(errno));
   return -1;
}
```

6. Comment the following lines in the "`wpa_driver_bsd_get_scan_results2()`" function:

```
if (buff == NULL)
   return NULL;
```

7. Comment the following line (present three times) in the "`wpa_driver_bsd_get_scan_results2()`" function:

```
//os_free(buf);
```

8. Comment the following line (present twice) in the "`wpa_driver_bsd_capa()`" function:

```
//os_free(devcaps);
```

priv_netlink.h

The priv_netlink.h file is present in the following path:

```
/root/myandroid/external/wpa_supplicant_8/wpa_supplicant/src/driver
s/
```

Comment the following lines in this file:

```
//struct sockaddr_nl
//{
//    sa_family_t nl_family;
//    unsigned short nl_pad;
//    u32 nl_pid;
//    u32 nl_groups;
//};


//struct nlmsghdr
//{
//    u32 nlmsg_len;
```

```
//    u16 nlmsg_type;
//    u16 nlmsg_flags;
//    u32 nlmsg_seq;
//    u32 nlmsg_pid;
//};
```

wpa_cli.c

The wpa_cli.c file is present in the following path:

```
/root/myandroid/external/wpa_supplicant_8/wpa_supplicant
```

9. Edit the following line in this file:

```
#define CONFIG_CTRL_IFACE_DIR "wlan0"
```

21.8 Compiling onebox_util for Android

The steps below explain the process for compiling the "`onebox_util`" program for Android.

1. Create a folder named "`onebox-utils`" in the /root/myandroid/external/ folder.
2. Create an Android.mk file with the following content in the /root/myandroid/external/onebox-utils folder:

```
LOCAL_PATH:= $(call my-dir)
################## build onebox_util #################
include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := onebox_util.c
#LOCAL_CFLAGS += -Wstrict-prototypes -Wmissing-prototypes -Wshadow
-Wpointer-arith -#Wcast-qual -Winline -MMD -fPIC
EXTRA_CFLAGS += -MMD -O2 -Wall -g
LOCAL_MODULE:= onebox_util
LOCAL_STATIC_LIBRARIES := libc
#LOCAL_FORCE_STATIC_EX ECUTABLE := true
LOCAL_MODULE_PATH := $(TARGET_OUT_OPTIONAL_EXECUTABLES )
# install to system/xbin
#LOCAL_UNSTRIPPED_PATH := $(TARGET_ROOT_OUT_UNSTRIPPED)
#LOCAL_MODULE_TAGS := eng user
include $(BUILD_EXECUTABLE)
```

3. Copy Makefile_onebox_util.h and onebox_util.c from the RS9113.NXX.NL.GEN.LNX.x.y.z/host/utils folder to the /root/myandroid/external/onebox-util folder.
4. Add the following line in the "`get_driver_state()`" function in the onebox_util.c file:

```
fp = popen("cat /proc/rpine0/stats | grep -i FSM_ | busybox cut
   -f2 -d ' ' | busybox cut -f1 -d '('","r");
```

5. Run the commands below to compile the files and create a binary for onebox_util in the /root/myandroid/out/target/product/evk6sl/obj/EXECUTABLES/onebox_util_intermediates folder.

```
$ cd /root/myandroid/
$ source build/envsetup.sh
$ lunch evk_6sl-user
$ cd /root/myandroid/external/onebox-util/
$ mm
```

6. Run the commands below to ensure that the onebox_util program appears in /system/bin path of the file system:

```
$ cd ~/myandroid/
$ make snod
```

21.9 Flashing the Android Image into SD Card

Once the RS9113 n-Link® driver is integrated with Android and the Android image is built, it needs to be flashed into the SD Card. Delete all partitions on the SD card and run the commands given below.

```
$ cd /root/myandroid
$ sudo chmod +x ./device/fsl/common/tools/fsl-sdcard-\
      partition.sh
$ cd /root/myandroid/out/target/product/evk6sl/
$ sudo /root/myandroid/./device/fsl/common/tools/fsl-\
      sdcard-partition.sh -f imx6sl /dev/sdb
```

22 Common Configuration Parameters

The common_insert.sh script is used to configure parameters at boot time. The parameters with their usage and input values are described below.

22.1 RF Power Mode parameter

The RF Power Mode parameter is used to set the power mode at which the RF operates. It is applicable for each protocol. By default, it is set to high power TX and high power RX. The following are the possible configurable values:

- 0x00 - For Both TX and RX High Power
- 0x11 - For Both TX and RX Medium Power
- 0x22 - For Both TX and RX LOW Power
- 0x10 - For High Power TX and Medium RX Power
- 0x20 - For High Power TX and LOW RX Power
- 0x01 - For Medium TX and RX High Power
- 0x21 - For Medium Power TX and LOW RX Power
- 0x02 - For Low Power TX and RX High Power
- 0x12 - For LOW Power TX and Medium RX Power

WLAN_RF_PWR_MODE is used to set the rf power mode for WLAN protocol.

BT_RF_PWR_MODE is used to set the rf power mode for Bluetooth protocol.

ZB_RF_PWR_MODE is used to set the rf power mode for Zigbee protocol.

Example:

WLAN_RF_PWR_MODE=0x00

The above sets high TX and high RX power for WLAN.

BT_RF_PWR_MODE=0x00

The above sets high TX and high RX power for Bluetooth.

ZIGB_RF_PWR_MODE=0x00

The above sets high TX and high RX power for Zigbee.

22.2 Country selection

This parameter is used to set the module in a specific country. This is set commonly across all protocols. The following country codes are applicable.

- 0 - World Domain
- 840 - US Domain Maps to US Region
- 276 - Germany Maps to EU Region
- 392 - Japan Maps to Japan Region

Example:

SET_COUNTRY_CODE=0

The above sets the module in the world domain.

22.3 Antenna selection

This variable is used to select the antenna to be used. The following are the possible values:

- 2 – Select internal antenna
- 3 – Select external antenna

Example:

ANT_SEL_VALUE=2

The above line selects the internal antenna. The Operation starts on this antenna.

Note:

If antenna diversity selection feature is also enabled, initial operation will start on the antenna selected. Antenna diversity operation will continue as expected.

22.4 COEX Mode selection

This variable is used to select the Coex mode in which the module has to operate. The following are the possible values:

- 1 - WLAN STATION /WIFI-Direct/WLAN PER
- 2 - WLAN ACCESS POINT (including multiple APs on different vaps)
- 3 - WLAN ACCESS POINT + STATION MODE (on multiple vaps)
- 4 - BT CLASSIC MODE/BT CLASSIC PER MODE
- 5 - WLAN STATION + BT CLASSIC MODE
- 6 - WLAN ACCESS POINT + BT CLASSIC MODE
- 8 - BT LE MODE /BT LE PER MODE
- 9 - WLAN STATION + BT LE MODE
- 12 - BT CLASSIC + BT LE MODE
- 13- WLAN STATION + BT CLASSIC MODE+ BT LE MODE
- 14 - WLAN ACCESS POINT + BT CLASSIC MODE+ BT LE MODE
- 16 - ZIGBEE MODE/ ZIGBEE PER MODE
- 17 - WLAN STATION + ZIGBEE
- 32 - ZIGBEE COORDINATOR
- 48 - ZIGBEE ROUTER

Example:

COEX_MODE=3

The above line sets the module to operate in WLAN AP + STA concurrent mode.

Appendix E : Installation of Missing Generic Netlink Libraries

Note:

libnl CFlags should be enabled with CONFIG_LIBNL32=y in supplicant and hostpad .config file [The above configuration settings should be set to “y” in case NL80211 is used]. Make sure the NL80211 support and Hostapd support are enabled in the menuconfig during compilation.

1. Create a directory in the location where Tool chain and BSP are present

   # mkdir build

2. Download the libnl 3.2.xx.tar.gz[Referring 3.2.27.tar.gz as an example here] library and extract it in the build directory.

   # cd build

   # tar xvf 3.2.27.tar.gz

3. Configure the libnl library for target platfrom

   # CC=/path to the toochain/bin/arm-linux-gnueabihf-gcc

   # ./configure --host=arm-linux-gnueabihf -prefix=/<complete path to build directory>/

   Here headers will be installed in ${prefix}/include/libnl3.

4. Make and install the libraries in the destination directory or else they will be installed in /usr/local/lib and /usr/local/include/libnl folders of host machine by default.
5. Follow the below example:

   # make DESTDIR=$(arm-cortex_a8-linux-gnueabihf-gcc -print -/<path to build directory>/build/)

6. Exporting the path for build directory in the command line

   #export LDFLAGS='-L/<path to build directory>/lib/libnl'

   OR

Add these flags in the supplicant and hostapd config files under CONFIG_DRIVER_NL80211= y variable.

CFLAGS += -I/<path to build directory>/include/libnl3

Ex: LIBS += -L/<path to build directory>/lib/libnl

LIBS : Contains a list of additional libraries to pass to the linker command.

Revision History

S.No.	Version No.	Date	Changes
1.	1.1.0	Mar'15	1) Merged the RS9113 n-Link Driver Installation Guide and Driver User Manual and created a new document called Software Technical Reference Manual. 2) Assigned version as per version of the n-Link Software Release. 3) Added new ioctls for Wi-Fi. 4) Added new ioctls for Wi-Fi Test mode 5) Added Bluetooth HCI commands. 6) Added information on usage of OneBox-Mobile software on reference platforms from Freescale and Atmel.
2.	1.1.1	Apr'15	No changes in this document. Version updated as per Release.
3.	1.1.2	May'15	1) Changed the name of Section 5.10 to "Wi-Fi Performance Test Usage Guide" 2) Added Regulatory Domain as an input to the "transmit" command in Section 5.10. 3) Added channels 12 and 13 in the list of channels in Section 5.10.
4.	1.1.3	Jun'15	1) Added the "Set External Antenna Gain" and "Set Wake-On-Wireless LAN" ioctl commands in Section 5.5. 2) Updated the list of channels in 5 GHz band in Section 5.10. 3) Added Section on Porting of driver for Android 4.4.3 on i.MX 6SoloLite Evaluation Kit.
5.	1.2.0	Jul'15	1) Added HCI Command for the BT/BTLE power-save. 2) Added BT Performance Test ioctl Usage Guide. 3) Updated the Android 4.4.3 compilation changes. . 4) Updated ZigBee switch APP command 5) Added ZigBee Performance Test ioctl Usage Guide. 6) Added ioctl to program RF power mode.
6.	1.2.1	Sep'15	1) Added the command to change the USB suspend time in the "Enable Power Save and Set Parameters" IOCTL section.
7.	1.2.2	Sep'15	No changes in this document. Version updated as per Release.

S.No.	Version No.	Date	Changes
8.	1.2.3	Sep'15	No changes in this document. Version updated as per Release.
9.	1.2.4	Oct'15	No changes in this document. Version updated as per Release.
10.	1.2.6	Nov'15	1) Added CFG80211 support changes. 2) Enterprise security with Hostapd usage 3) Usage of WPS with Hostapd 4) Added Antenna diversity usage 5) Added common configuration parameters for RF Power Mode, Country selection, Antenna selection and updated COEX mode usage
11.	1.3.0	Dec'15	1) Compilation and Installation of Driver sections are updated In Section 3 and Section 4. 2) Commands have been updated in accordance with changes mentioned above. 3) Added Section 4.5 for WIFI Access point + BT Classic and BT LE mode. 4) Made some corrections, added examples and a note for stopping receive and transmit in BT Performance Evaluation mode in Section 6.2. 5) Added Wifi Debug Zone ioctl usage in WiFi IOCTL usage section 5.2.
12.	1.4.0	Feb'16	1) Added Section 4.2 for Enabling required Protocol(s). 2) Added Section 4.3 for Disabling required Protocol(s). 3) Compilation of Driver section is updated In Section 3 . 4) Modified Note in section 4.1. 5) Removed the insert scripts and added corresponding enable scripts for protocols.
13.	1.4.1	Apr'16	Updated the regulatory domain values PER transmit in the Table 8.
14.	1.4.2	May'16	1)Set mode ioctl removed in section 5.2 2) set_country ioctl corrected in section 5.5 3) RF port input corrected in cw_mode ioctl in section 6.2.1.5

S.No.	Version No.	Date	Changes
			4) Hopping tests added in section 6.2.1.7 5) removed compact_wireless_extensions script in section 8.2 6) Operating mode for ZigBee coordinator and router are added in section 16.4.
15.	1.4.3	May'16	1) Cw mode command start/stop values added in section 6.2.1.5 2) Set_mgmt_rate ioctl added in section 5.2
16.	1.4.3	June'16	1) Added required information to ioctl's(essid, beacon_interval, set bandwidth, set DTIM Period, set Hidden ssid, Set Beacon interval, set scan type). 2) Renamed sections Transmit Command Usage 5.11.1.1, 5.11.1.3, subsections under 6.2, 7.1,. 3) Removed sub headings under section 11.7
17.	1.5.0	June'16	1) Cw mode command added in section 11.1.2.1 2)Added Beacon filter ioctl in section 5.5 3) Added coexmode 13 in section 4.7 4) Added Appendix E and sections 4.4.5, 4.4.6.
18.	1.5.0	Aug`16	1) Updated the Table of Contents. There was a mistake with the heading after the Antenna Diversity Section. 2) Added CW mode configuration for 11J channels 3) Added 11J channels in Table 7 4) Added Background scan Configurability fpr 11J channels 5)