

# RS9116N Open Source Driver Technical Reference Manual

Version 1.8

May 2022

## Table of Contents

<b>1</b>	<b>Introduction to the RS9116 nLink Driver</b>	<b>5</b>
<b>2</b>	<b>Getting Started with the RS9116 nLink Driver</b>	<b>6</b>
2.1	Hardware Requirements	6
2.2	Software Requirements	6
2.3	Software Package Contents	6
<b>3</b>	<b>Compilation Steps</b>	<b>7</b>
3.1	Building the Driver from the Local Path	7
3.2	Building the Driver from Kernel Source	8
<b>4</b>	<b>Installing the nLink Driver</b>	<b>9</b>
4.1	Installation of Modules	9
4.2	Installation in Wi-Fi Only Mode	11
4.2.1	Wi-Fi Station Mode	11
4.2.2	Wi-Fi Access Point (AP) Mode	17
4.3	Installation in Bluetooth Only Modes (BT/BLE)	20
4.3.1	Bluetooth classic only mode	20
4.3.2	Bluetooth LE Only Mode	20
4.3.3	Bluetooth Classic + Bluetooth LE Mode	21
4.4	Installation in Wi-Fi + Bluetooth Classic Coexistence Mode	21
4.5	Installation in Wi-Fi + Bluetooth LE Coexistence Mode	22
4.6	Installation in Wi-Fi + Bluetooth Classic + Bluetooth LE Coexistence Mode	22
4.7	Uninstalling the Driver	23
<b>5</b>	<b>Configuration Using Wireless Tools</b>	<b>25</b>
5.1	Using the iw Wireless Tool	25
5.2	Software Rfkill	31
<b>6</b>	<b>Wi-Fi Protected Setup (WPS)</b>	<b>32</b>
6.1	STA Mode WPS Configuration	32
6.2	AP Mode WPS Push Button Configuration	33
<b>7</b>	<b>Sniffer Mode (Promiscuous Mode)</b>	<b>34</b>
<b>8</b>	<b>Background Scan &amp; Roaming</b>	<b>35</b>
8.1	Configuring Background Scan Parameters through debugfs	35
8.2	Setting Bgscan SSID through Debugfs	36
<b>9</b>	<b>Power Save</b>	<b>37</b>
9.1	Power Save Modes	37
9.2	Device Sleep Mode	37
9.3	Wakeup Procedures and Data Retrieval	37
9.4	Configuring LP Device Power Save for USB and SDIO Interface	38
9.5	Configuring ULP device power save for SDIO interface	38
9.6	Configuration of ULP GPIO Handshake and GPIO Numbers	39
9.7	Enabling Power Save	39
9.8	Configure Power Save Parameters/Profiles through debugfs Dynamically	40
9.9	USB Auto Suspend	41
<b>10</b>	<b>Steps to Configure 802.11W (PMF)</b>	<b>42</b>
10.1	Configuring and Compiling Driver for PMF in client mode	42
10.2	Configuring and Compiling Driver for PMF in AP Mode	42
<b>11</b>	<b>Antenna Selection</b>	<b>43</b>
11.1	Using antenna_sel module param	43
11.2	Using iw phy command	43
11.2.1	To Select External Antenna	43
11.2.2	To Select Internal Antenna	43
<b>12</b>	<b>Bluetooth hciconfig and hcidtool Usage</b>	<b>45</b>
12.1	BT Device Connection using Bluetoothctl	46
12.2	L2 Test Commands	47
<b>13</b>	<b>PER Driver</b>	<b>49</b>
<b>14</b>	<b>Update WLAN Region-Based Maximum Powers from Driver</b>	<b>50</b>
14.1	Gain Table Structure Format:	50

<b>15</b>	<b>Appendix A: Driver Details .....</b>	<b>53</b>
15.1	Debug Zone Prints .....	53
15.2	Version .....	53
15.3	Station Stats .....	54
15.4	SDIO stats .....	54
<b>16</b>	<b>Appendix B: Initial Configuration Parameters .....</b>	<b>56</b>
16.1	Common Configuration Parameters .....	56
16.1.1	Power Save Feature .....	56
16.1.2	BT Related Module Params .....	56
16.1.3	Miscellaneous Features .....	57
16.1.4	Developer Mode Configuration Parameters .....	57
<b>17</b>	<b>Appendix C: Hostapd Usage Guidelines .....</b>	<b>59</b>
17.1	Common Configuration Parameters .....	59
17.2	Hostapd Conf File Changes Required for ACL (Access Control List) .....	59
17.3	Enable UAPSD Advertisement in hostapd .....	59
17.4	Configure AP Mode Keep Alive Time .....	60
17.5	Configuration for Country IE .....	60
<b>18</b>	<b>Appendix D: Enterprise Security using CFG80211 .....</b>	<b>61</b>
18.1	Installation and Configuration of FREERADIUS Server .....	61
18.2	Configuring Station to Connect to an EAP Enabled AP .....	62
18.3	Configuration of AP and RADIUS Server to Use EAP Methods .....	65
18.3.1	Configuration of the AP .....	65
18.3.2	Configuring hostapd as RADIUS Server .....	66
<b>19</b>	<b>Appendix E: Kernel Configuration Needed for Driver .....</b>	<b>67</b>
19.1	SDIO Stack Options .....	67
19.1.1	Wireless Extension Tools .....	69
19.1.2	Bluetooth Stack Options .....	69
19.1.3	Kernel Compilation .....	71
<b>20</b>	<b>Appendix F: Configure and Compile Supplicant for Roaming .....</b>	<b>72</b>
20.1	Roaming .....	72
<b>21</b>	<b>Appendix G: Installation of Missing Generic Netlink Libraries .....</b>	<b>73</b>
<b>22</b>	<b>Appendix H: Using the Bluetooth Manager .....</b>	<b>74</b>
<b>23</b>	<b>Appendix I: Checking Throughput .....</b>	<b>77</b>
<b>24</b>	<b>Appendix H: Steps to Connect EAP-TLS Using wpa_supplicant v 2.6 and Above .....</b>	<b>80</b>
24.1	Installing FreeRADIUS Server .....	80
24.2	Generate Certificates .....	83
<b>25</b>	<b>RS9116 n-Link OSD Software TRM Revision History .....</b>	<b>85</b>

## About this Document

This document is a preliminary version of RS9116 n-Link Open Source Driver Technical Reference Manual for Linux, provided to customers.

**Note:** This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit [www.silabs.com/about-us/inclusive-lexicon-project](http://www.silabs.com/about-us/inclusive-lexicon-project).

## 1 Introduction to the RS9116 nLink Driver

The RS9116N Open Source Driver (OSD) is a SoftMAC driver that interacts with the Linux wireless MAC layer i.e., MAC80211. The driver is a group of simple and efficient kernel modules which currently supports RS9116N chipsets and can be ported to any embedded platform in-addition to X-86 platform. It supports the following protocols:

- Wi-Fi (Client and Access Point mode)
- Bluetooth Classic
- Bluetooth Low Energy

It supports the following protocol combinations :

- WLAN STATION
- WLAN ACCESS POINT
- BT CLASSIC MODE
- BT LE MODE
- WLAN STATION + BT CLASSIC MODE
- WLAN STATION + BT LE MODE
- BT CLASSIC + BT LE MODE
- WLAN STATION + BT CLASSIC MODE + BT LE MODE
- WLAN ACCESS POINT + BT CLASSIC MODE
- WLAN ACCESS POINT + BT LE MODE
- WLAN ACCESS POINT + BT CLASSIC MODE + BT LE MODE

The following sections will guide the user on usage of the driver.

## 2 Getting Started with the RS9116 nLink Driver

This section lists the hardware and software requirements for the installation of the software and describes the steps to be followed to initialize and run the software.

### 2.1 Hardware Requirements

The hardware requirements are as follows:

- RS9116N n-Link® Module
- Laptop/PC with SDIO or USB interface or any embedded platform with Linux Board support package.

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](http://www.hwtools.net/cardreader/SDCBA_C01.html) can be used.

### 2.2 Software Requirements

The software requirements are as follows:

- Supported Linux Kernel Versions are mentioned in ReleaseNotes.pdf file – should enable the open source SDIO and USB stacks.
- DHCP Server (for Wi-Fi Access Point mode)
- Bluetooth supported commands bluetoothctl and bluetoothd must be present
- Compatible Bluetooth Host Stack, e.g., the Open Source BlueZ Stack v4.10
- wpa\_supplicant (for Wi-Fi Client mode)
- hostapd (for Wi-Fi Access Point mode)

### 2.3 Software Package Contents

The driver package is delivered in the format: **RS9116.NX0.NL.GNU.LNX.OSD.a.b.c.d.zip**, where the naming convention is as follows:

**NX0** - defines whether the package supports only Wi-Fi (N00) or Bluetooth Classic/Low Energy along with Wi-Fi (NB0).

**NL** - indicates NetLink

**GNU** - GNU License

**LNX** - Linux

**OSD** - Open Source Driver

**a.b.c.d** – identifies the software package.

The driver package contains the following files/folders:

- Readme.txt
- ReleaseNotes.pdf
- Firmware
- rsi (contains driver source code)
- scripts

The drivers can be found under each Product on our website: <https://docs.silabs.com/rs9116/#nlink-overview>.

### 3 Compilation Steps

This section describes the steps to be followed to compile the driver for different platforms. The steps are outlined below:

1. Extract the package using the following command:

```
# unzip RS9116.NX0.NL.GNU.LNX.OSD.<version>.zip
```

2. Go to the package and copy all the files present in Firmware folder to '/lib/firmware' by following the below commands.

```
# cd RS9116.NX0.NL.GNU.LNX.OSD.<version>
# cp Firmware/* /lib/firmware
```

3. There are two ways in which you can build the driver.
  - a. Build from the local path
  - b. Build from kernel source

#### 3.1 Building the Driver from the Local Path

1. Configure build flags in driver source.

```
# cd rsi
```

2. Open Makefile and configure build flags. Below are the build flags to be set based on the usage of driver. Selecting the required options shall reduce the binary size which is important for kernel modules particularly on embedded platforms.

- a. **KERNELDIR** : Provide the kernel source path here. For example on X-86 below path is used.

```
KERNELRELEASE=$(Shell uname -r)
KERNELDIR=/lib/modules/$(KERNELRELEASE)/build
```

- b. **CONFIG\_RSI\_COEX\_MODE** : Enable this flag when Wi-Fi and BT coexistence mode is used.
- c. **CONFIG\_RSI\_DEBUGFS** : Debugfs is used by driver to take dynamic configuration from user. Supported debugfs based configurations are listed in the corresponding feature sections in TRM.
- d. **CONFIG\_RSI\_BT\_ALONE** : Enable this flag when only BT EDR/ BT LE only mode is used.

3. Build the driver using make command.

```
# make
```

For embedded platforms, add the Kernel path for target platform and toolchain path as cross compilation option to the "make" command.

For example, if the target platform is iMX6 add the kernel path as below :

```
KERNELDIR=home/test/Wand/armv7-multiplatform/KERNEL
```

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-
```

## 3.2 Building the Driver from Kernel Source

1. Copy the driver 'rsi' to <kernel\_source\_path> /drivers/net/wireless. (Ex : linux-5.7.0/drivers/net/wireless/rsi )
2. Go to rsi directory and move Makefile to Makefile\_local.

```
# mv Makefile Makefile_local
```

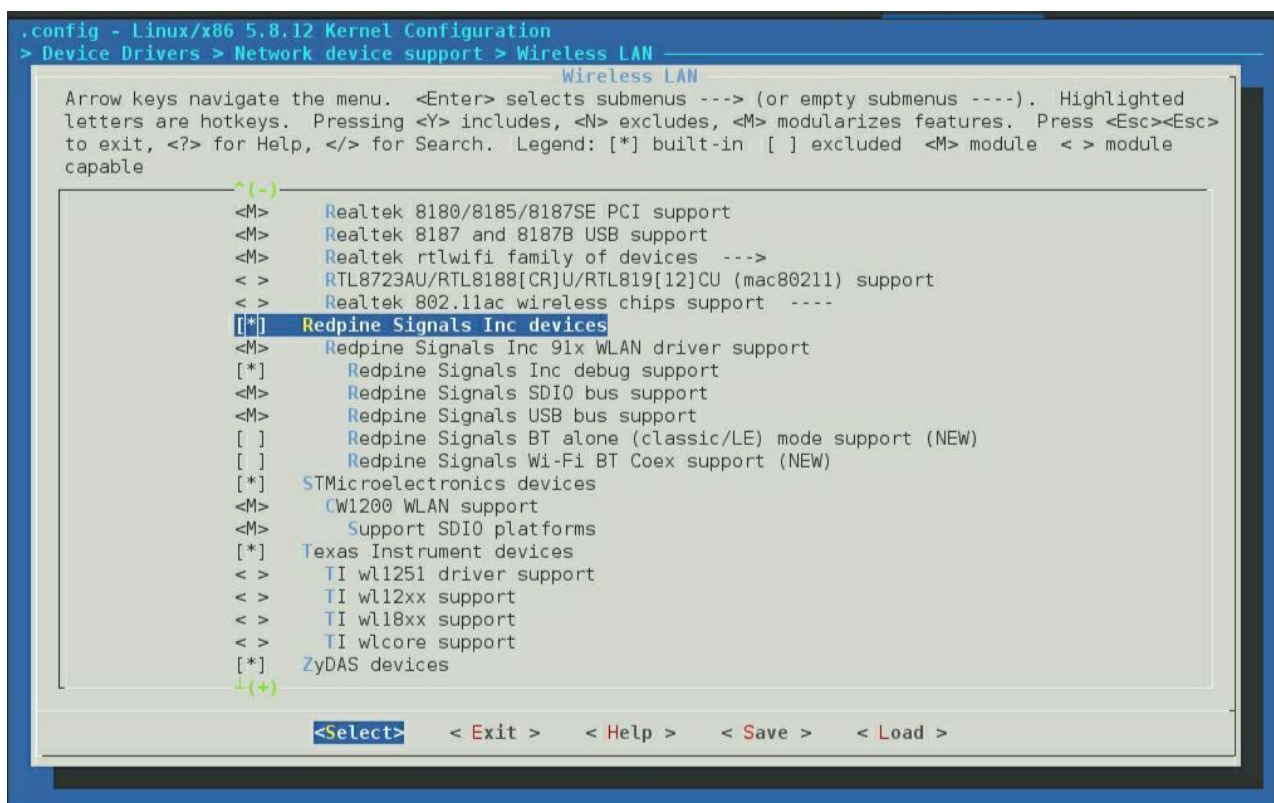
3. Move Makefile\_ker to Makefile.

```
# mv Makefile_ker Makefile
```

4. Give 'make menuconfig' from kernel source directory. (Ex : linux-5.7.0/ )

```
# make menuconfig
```

5. Go to 'Device Drivers->Network device support->Wireless LAN'.
6. Select 'Redpine Signals Inc' devices.
7. Select the SDIO/USB bus support depending on requirement. You will see the below screen with all the build options mentioned above. Select the required options.



8. Build driver by using the below commands:

```
# make M=drivers/net/wireless/rsi
```

On successful compilation, make will generate rsi\_91x.ko , rsi\_usb.ko and/or rsi\_sdio.ko according to the configuration.

In the next section, [Installing the nLink Driver](#) process is given in detailed step-by-step instructions.



## 4 Installing the nLink Driver

### 4.1 Installation of Modules

After a successful compilation, the driver generates the following modules in the rsi folder according to the configuration. They are outlined below:

- rsi\_91x.ko
- rsi\_usb.ko
- rsi\_sdio.ko

To install the driver, use the following commands:

1.) Before installing driver install the dependencies using below commands

```
# modprobe mac80211
# modprobe cfg80211
# modprobe bluetooth
```

2.) Insert rsi\_91x.ko with the required module params (configuration) as shown below:

```
# insmod rsi_91x.ko dev_oper_mode=<mode> rsi_zone_enabled=<val> ...
```

Module params are used by the driver to take initial configuration required. If not provided, default configuration is used. For most of the applications, default values of these module params will be sufficient. Supported module params with their configurable limits are explained in this TRM in respective feature sections and/or [Appendix B: Initial Configuration Parameters](#).

In the above command example, module param **rsi\_zone\_enabled** is to enable debug prints in dmesg. Default value of rsi\_zone\_enabled value is 1, which prints errors (only) in terminal. Please refer [Appendix A: Driver Details](#), to enable more debug prints.

**dev\_oper\_mode** : Device operating mode indicates the possible combination of the wireless protocols that can configured with the device.

The table below provides the operating mode details with its constraints.

S.No	Operating Mode	Protocol Support				Maximum No. of Clients for WLAN AP	Maximum No. of BT Connections	Maximum No. of BLE Connections
		STA	AP	BT EDR	BT LE			
1	1	√	X	X	X	N/A	2	3 (Can be as Main for 2 Secondary connections Or Can be as Main for one Secondary connection and can connect to other Main as Secondary)
2	1	X	√	X	X	16 clients		
3	4	X	X	√	X	N/A		
4	5	√	X	√	X	N/A		
5	6	X	√	√	X	16 clients		
6	8	X	X	X	√	N/A		
7	9	√	X	X	√	N/A		
8	10	X	√	X	√	16 clients		
9	12	X	X	√	√	N/A		
10	13	√	X	√	√	N/A		
11	14	X	√	√	√	4 clients		

If any invalid mode is passed to the module, driver returns error and exit. You can check the error message debug logs.

**Note**

For modes 4, 8 and 12 build flag CONFIG\_RSI\_BT\_ALONE should be enabled in the driver Makefile.  
For modes 5, 9, 6, 13 and 14, build flag CONFIG\_RSI\_COEX\_MODE should be enabled in the driver Makefile.

3.) For the USB interface, enter the command below:

```
# insmod rsi_usb.ko
```

4.) For the SDIO interface, enter the command below:

```
# insmod rsi_sdio.ko sdio_clock=<clk_val>
```

Note: Here “clk\_val” is 1 to 50 (in MHz's).

You can install either USB or SDIO or both depending upon the selection of the interface.

After a successful installation, a new wireless interface shall be created or WLAN and/or BT/BLE as per the dev\_oper\_mode selection.

a. If **WLAN** is selected, interface details can be verified using the commands below.

Name of the Wi-Fi interface created after successful installation of the driver can be seen using 'ifconfig' command.

```
# ifconfig -a
```

You should expect an output like the sample shown below with all other available interfaces included.

```
wlan0    flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
         inet6 fe80::8da:1aff:fe1e:dlc8 prefixlen 64 scopeid 0x20<link>
         ether 88:da:1a:1e:d1:c8 txqueuelen 1000 (Ethernet)
         RX packets: 3 bytes 372 (372.0 B)
         RX errors 0 dropped 0 overruns 0 frame 0
         TX packets: 6 bytes 696 (696.0 B)
         TX errors 0 dropped 0 overruns 0 collisions:0
```

Also, for **WLAN**, there is another utility (iw) with which you can get the interface and physical device details. For example, the command below will show the interface status and physical device number.

```
# iw dev <interface_name> info
```

The sample output for this command is shown below.

```
Interface wlan0
- ifindex 5
- wdev 0x100000001
- addr 00:23:a7:b9:ab:44
- type managed
- wiphy 1
- channel 6 (2437 MHz), width: 20 MHz (no HT), center1: 2437 MHz
```

As can be seen, in this case, phy<X> is termed as wiphy 1.

b. If **BT/BLE** is selected, interface details can be verified using below command.

Name of the BT/BLE interface created after successful installation of the driver can be seen using 'hciconfig' command.

```
# hciconfig -a
```

```
hci0:  Type: BR/EDR  Bus: USB
      BD Address: 88:DA:1A:00:00:C2  ACL MTU: 1021:3  SCO MTU: 64:3
      UP RUNNING
      RX bytes:1006 acl:0 sco:0 events:55 errors:0
      TX bytes:0 acl:0 sco:0 commands:55 errors:0
      Features: 0xbf 0xfe 0xd 0xfe 0xdb 0xff 0x5b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH SNIFF
      Link mode: SECONDARY ACCEPT
      Name: 'lapt64'
      class: 0x0c010c
      service Classes: Computer, Laptop
      HCI Version: (0x9) Revision: 0x0
      Manufacturer: internal use (65535)
```

## 4.2 Installation in Wi-Fi Only Mode

### 4.2.1 Wi-Fi Station Mode

This section provides the steps to configure Wi-Fi station mode using both wpa\_supplicant and the Network Manager CLI. Both procedures are given below. The user can choose any method.

Before installation, the user needs to stop the existing network manager and unblock WLAN from rfkill. The commands below are used to stop the network-manager on different Linux distribution.

1. For Ubuntu, use the following command.

```
# service network-manager stop
```

2. For Fedora, use the following command.

```
# service NetworkManager stop
```

3. To stop rfkill blocking WLAN, use the following command.

```
# rfkill unblock wlan (or) #rfkill unblock all
```

- For station mode connectivity, ensure that the dev\_oper\_mode is set in installation as given below and interface is detected after the installation (Refer to the [Installation of Modules](#) section).

```
dev_oper_mode = 1
```

#### 4.2.1.1 Configure Station Using WPA\_supplicant

1) Create a **sta\_settings.conf** file with the information below. Also, fill the information like ssid, psk etc corresponding to the AP you intend to connect in this file. Sample sta\_settings.conf file is available within scripts directory of release package with basic configurations required. The user may use this file and edit the information as explained below. For the details of all configurations available please refer to the open source supplicant wpa\_supplicant.conf file.

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
```

Also, add network block to the sta\_settings.conf file as per the AP security. An example network block for different security modes is listed below.

##### i. For Open (non-Secure) mode:

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=NONE
    priority=3
}
```

##### ii. 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
}
```

The pass phrase can be input either in ASCII or Hexadecimal formats:

**ASCII Format:** psk="very secret passphrase"

**Hexadecimal Format:** psk= 06b4be19da289f475aa46a33cb793029d4ab3db7a23ee92382eb0106c7

##### iii. For WPA3 security mode

To connect in WPA3, we need to compile the latest supplicant with below flags enabled in wpa\_supplicant .config file

```
CONFIG_SAE=y
CONFIG_IEEE80211W=y
```

```
pmf=2
network={
    ssid="<SSID of Access Point>"
    key_mgmt=SAE
    psk="<passphrase specified in the Access Point>"
    ieee80211w=2
}
```

WPA3 Enterprise security mode is not supported in this release.

**iv. For WPA2-EAP TLS (Enterprise mode) mode:**

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-EAP
    eap=TLS
    anonymous_identity="tlsuser"
    identity="test"
    password=<passphrase specified in the Access Point>
    ca_cert="/etc/certs/wifiuser.pem"
    client_cert="/etc/certs/wifiuser.pem"
    private_key_passwd=<private key password>
    private_key="/etc/certs/wifiuser.key"
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

In EAP-TLS user has to copy client certificates in a path and the path need to be configured in network block as given above.

**v. For WPA2-EAP PEAP (Enterprise mode) mode:**

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-EAP
    eap=PEAP
    anonymous_identity="peapuser"
    identity="test"
    password=<passphrase specified in the Access Point>
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

**vi. For WPA2-EAP TTLS (Enterprise mode) mode:**

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-EAP
    eap=TTLS
    anonymous_identity="ttlsuser"
    identity="test"
    password=<passphrase specified in the Access Point>
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

To connect to an Access Point whose SSID is not broadcast (Hidden), add the following line to the network block.

```
scan_ssid=1
```

For example:

```
network={
    ssid="<SSID of Access Point>"
    scan_ssid=1
    key_mgmt=NONE
}
```

## For Selective Scan:

To enable selective scan , Add freq\_list parameter outside the network block . freq\_list is Space-separated list of frequencies in MHz which limits the frequencies that will be scanned.

```
freq_list=2412 2437 2462
```

Example config file that will only scan on channel 1 and 36.

```
freq_list=2412 5180
network={
    ssid="<SSID of Access Point>"
    key_mgmt=NONE
}
```

2) Start the supplicant using below command:

```
# wpa_supplicant -i <interface_name> -D nl80211 -c sta_settings.conf -dddt > supp.log &
```

For example :

```
# wpa_supplicant -i wifi0 -D nl80211 -c sta_settings.conf -dddt > supp.log &
```

- "-i" option specifies the Wi-Fi interface name
- <interface\_name> - This name as listed in iw dev output (here wifi0)
- "-D" specifies the driver interface to be used. In open source driver it is nl80211.
- "-c" specifies the supplicant configuration file.
- "-d" specifies the log level of supplicant. You can append more d's to it for more detailed logs.

3) To check the scan results please use below command.

```
# wpa_cli -i <interface_name> scan_results
```

For example, above command will give scan results output as follows.

bssid	/ frequency	/ signal level	/ flags	/ ssid
50:d4:f7:1e:5a:40	2457	-21	[WPA2-PSK-CCMP] [ESS]	TP_LINK
04:79:70:72:03:e7	2412	-31	[ESS]	honor_9i

4) To check whether the connection is successful or not use below command

```
# iwconfig <interface_name>
```

For example, if the connection is successful, we will see the output below.

```
wlan0      IEEE 802.11bgn  ESSID:"Range"  Nickname:""
Mode:Managed  Frequency:2.412 GHz  Access Point: 38:A4:ED:DE:BB:06
Bit Rate:39 Mb/s   Tx-Power=16 dBm
Retry short limit:7   RTS thr:2353 B   Fragment thr:2352 B
Encryption key:off
Power Management:off
Link Quality=80/80  Signal level=-28 dBm  Noise level:0 dBm
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

If the connection is successful, then the connected Access point SSID along with the MAC address is displayed as shown above. If it is not connected to an Access point, a message **"Not Associated"** is displayed as shown below.

```
wlan0      IEEE 802.11  ESSID:off/any
Mode:Managed  Access Point: Not-Associated   Tx-Power=0 dBm
Retry short limit:7   RTS thr:off   Fragment thr:off
Encryption key:off
Power Management:off
```

5) IP for the device can be set in two ways either get IP dynamically from AP or set static IP. To obtain dynamic IP from AP, use below commands.

```
# dhclient < interface_name > -r
# dhclient < interface_name > -v
```

To set static IP to STA use below command.

```
# ifconfig <interface_name> <IP_address>
```

6) To check whether IP is assigned or not use below command.

```
# ifconfig <interface_name>
```

Output:

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.114  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::224:d7ff:fe56:54dc  prefixlen 64  scopeid 0x20<link>
ether 00:24:d7:56:54:dc  txqueuelen 1000  (Ethernet)
RX packets 31160  bytes 31082515 (29.6 MiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 23356  bytes 3367496 (3.2 MiB)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

#### 4.2.1.2 Configure station using the Network-Manager CLI (nmcli)

Below are the specific commands that can be used for connection using the Network Manager CLI(nmcli):

1. Check the network manager status (started or stopped) with below command.

For fedora,

```
# service NetworkManager status
```

For ubuntu,

```
# service network-manager status
```

2. If the network manager is inactive or not started, start it with the below command.

For fedora,

```
# service NetworkManager start
```

For ubuntu,

```
# service network-manager start
```

3. To view the currently available network connections, enter the following on command prompt:

```
# nmcli con show
```








sample output:

NAME	UUID	TYPE	DEVICE
eth0	96a5deb0-5eb0-41e1-a7ed-38fea413f9c8	802-3-ethernet	eth0
wlan0	91451385-4eb8-4080-8b82	802-11-wireless	wlan0

4. To view the list of access points, issue the below command:

```
# nmcli dev wifi list
```

Sample output is shown below.

* SSID	MODE	CHAN	RATE	SIGNAL	BARS	SECURITY
ASUS_5G	Infra	36	54 Mbit/s	100		WPA2
ASUS	Infra	11	54 Mbit/s	100		WPA2
test123	Infra	8	54 Mbit/s	32		WPA1 WPA2
cisco	Infra	1	54 Mbit/s	30		WPA1 WPA2
test	Infra	13	54 Mbit/s	25		---
Dlink	Infra	1	54 Mbit/s	0		WPA2
TP-LINK_E11946	Infra	7	54 Mbit/s	83		WPA1 WPA2

5. For connecting to an AP with WPA/WPA2 security, issue the below command:

```
# nmcli dev wifi connect ASUS password 12345678 <interface_name>
```

Here, ASUS is the AP's SSID and password is 12345678.

6. For connecting to an AP without security, issue the below command:

```
# nmcli dev wifi connect test <interface_name>
```

'test' is the SSID .

7. To know the status of the devices and the connections, issue the below command:



```
# nmcli dev status
```

Sample output:

DEVICE	TYPE	STATE	CONNECTION
wlan0	wifi	connected	my-ssid
eth0	ethernet	unavailable	--

As can be seen, the STATE corresponding to wlan0 interface shows connected.

- To Enable (to make active) a connection on interface, using nmcli, issue the below command. connection\_name can be obtained from above command.

```
# nmcli con up id <connection_name>
```

- To Disable an interface using nmcli, issue the below command:

```
# nmcli dev disconnect <interface_name>
```

#### 4.2.2 Wi-Fi Access Point (AP) Mode

This section provides steps to configure Wi-Fi AP mode using hostapd application.

- For AP mode connectivity, ensure that the dev\_oper\_mode is set in installation as given below and interface is detected after installation (Refer to above [Installation of Modules](#) section).

```
dev_oper_mode = 1
```

Follow below steps for running AP mode with hostapd application.

- Before running hostapd make sure wpa\_supplicant is not running in the background. If it is running kill wpa\_supplicant using below command

```
# killall wpa_supplicant
```

- Install hostapd.

```
# apt-get install hostapd
```

- Configure Hostapd

Create a hostapd configuration file (for eg: ap.conf) and add below:

- Create a **ap.conf** file with below information. Sample .conf files (ap\_open.conf,ap\_wpa.conf) are available within scripts directory of release package with basic configurations required. User may use this file and edit the information as explained below. For the details of all configurations available please refer open source hostapd **hostapd.conf** file.
- Set interface name:

```
interface=<interface_name>
Ex: interface=wlan0
```

- Set driver name:

```
driver=nl80211
```

- d. Set country name code in ISO/IEC 3166-1 format. This is used to set regulatory domain. Set as needed to indicate country in which device is operating. This can limit available channels and transmit power. For example, IN for India, UK for United Kingdom, US for the United States of America.

```
country_code=IN
```

- e. Set your SSID: Here, we have set 'Test\_AP' as ssid for example.

```
ssid=Test_AP
```

- f. Set operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g)

```
hw_mode=g
```

- g. Set Beacon Interval:

```
beacon_int=100
```

User may select required beacon interval within the range of 56-1000 ms. For value less than 56ms or more than 1000ms driver will return an error.

The default value is 100ms.

- h. Set channel number

```
channel=6
```

User may select required channel of operation or he may opt for Auto Channel Selection (ACS). In case of ACS, user has to follow below procedure.

Compile hostapd with below flag set in its .config file.

```
CONFIG_ACS=y
```

Also user has to add below configurations to hostapd.conf file.

```
channel=0
```

```
acs_num_scans=5 (Default Value, user may select)
```

- i. Set wpa mode to 2:

```
wpa=2
```

- j. Set your passphrase (Wi-Fi password):

```
wpa_passphrase=MyWiFiPassword
```

- k. Set key and auth options for WPA2:  
Set the key management algorithm as shown.

```
wpa_key_mgmt=WPA-PSK
```

Set cipher suites i.e., encryption algorithms:

```
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Shared Key Authentication :

```
auth_algs=1
```

Save and close the file.

TKIP stands for Temporal Key Integrity Protocol and CCMP is AES in Counter mode with CBC-MAC .

l. Start the hostapd application:

```
# hostapd ap.conf -dddt > log_file &
```

m. To check if AP mode is successfully started or not use below command :

```
# iw dev
```

For example If the AP is successfully started ,expect the below sample output i.e with SSID and channel information:

```
phy#10
Interface wlan0
  ifindex 13
  wdev 0xa00000001
  addr 88:da:1a:78:06:e4
  ssid rsi_ap_wpa
  type AP
  channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz
```

And If AP failed to start, output doesnot show SSID and channel information as shown in below sample output :

```
phy#10
Interface wlan0
  ifindex 13
  wdev 0xa00000001
  addr 88:da:1a:78:06:e4
  type managed
```

n. Run dhcp server script (In scripts folder) to assign IPs to client.

```
# sh dhcp_server.sh <interface_name>
```

dhcp\_server.sh script uses dhcpd.conf file for required configurations. User may modify this file as per the requirement.

In the scripts folder, several hostapd config files are provided to start the AP in various modes like open (ap\_open.conf), WPA/2-PSK (ap\_wpa.conf). User could use these conf files instead of creating new ones.

For other configurations of hostapd (ex. ACL Policy , Keep Alive) for AP mode please refer [Appendix C: Hostapd usage guidelines](#).

Please refer [Appendix I: Checking Throughput](#) section for measuring Wi-Fi performance through UDP/TCP protocols using iperf application.

## 4.3 Installation in Bluetooth Only Modes (BT/BLE)

### 4.3.1 Bluetooth classic only mode

This section provides steps to configure BT classic mode. BT classic mode supports only one connection.

- For BT classic usage, ensure that the `dev_oper_mode` is set in installation as given below and BT interface is detected after installation (Refer to [Installation of Modules](#) section).

```
dev_oper_mode = 4
```

- Bring hci interface up with below command.

```
# hciconfig -a hci0 up
```

- After the device is up, we can pair it with the other devices using the Bluetooth Manager application or `bluetoothctl` application. The files can also be sent and received using Bluetooth Manager. Instead of Bluetooth Manager, the device can be configured using "`hcitool`" or "`hciconfig`" as given below.
- The procedure for using `bluetoothctl` is explained in [Using the Bluetoothctl Application](#).
- The procedure for using Bluetooth Manager is explained in the section [Appendix H: Using the Bluetooth Manager](#).

- To start BT-Classic scanning, we need to give below command.

```
# hcitool -i hci0 scan
```

- Once BT-classic connectivity completed, we can use `l2test` application for connectivity status as given below.

```
# l2test -i hci0 -r (on RSI BT device side)
```

```
# l2test -i hci0 -s < BD address of RSI BT devices>(on third party BT device side)
```

### 4.3.2 Bluetooth LE Only Mode

This section provides the steps to configure Bluetooth LE mode. Bluetooth LE mode supports a maximum of 2 connections, i.e., as Main for 2 Secondary connections (or) as Main for one Secondary connection and can connect to other Main as Secondary.

- For Bluetooth LE usage, ensure that the `dev_oper_mode` is set in installation as given below and BLE interface is detected after installation (Refer to the [Installation of Modules](#) section).

```
dev_oper_mode = 8
```

- Bring hci interface up with below command.

```
# hciconfig -a hci0 up
```

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

a. Enable Advertise

```
# hciconfig -a <hciX> leadv
```

b. Disable Advertise

```
# hciconfig -a <hciX> noleadv
```

c. Initiate Scan - Below command displays the scan responses and advertising information.

```
# hcitool -i <hciX> lescan
```

d. Main 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.

e. Secondary 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 of the EVB/module, e.g., 00:23:AC:01:02:03

f. The above advertise ,scan and connect procedure can be followed for multiple slaves.

### 4.3.3 Bluetooth Classic + Bluetooth LE Mode

Ensure that the dev\_oper\_mode is set as follows:

- dev\_oper\_mode = 12

Once the installation completed and interface of Bluetooth is detected follow below procedure to install BT and BLE.

1. For Bluetooth LE protocol, follow the instructions given in [Bluetooth LE only mode](#) installation.
2. Also for Bluetooth classic protocol, follow the instructions given in [Bluetooth classic only mode](#) installation.

## 4.4 Installation in Wi-Fi + Bluetooth Classic Coexistence Mode

Ensure that the dev\_oper\_mode is set as below

- dev\_oper\_mode = 5 (Wi-Fi STA + BT)
- dev\_oper\_mode = 6 (Wi-Fi AP + BT)

Once the installation completed and interface of Wi-Fi and Bluetooth are detected follow below procedure to install Wi-Fi and BT.

1. For Wi-Fi, follow the instructions given in [Wi-Fi station mode](#) or [Wi-Fi Access Point \(AP\) mode](#) in alone mode for Wi-Fi installation.

- Also, for Bluetooth classic protocol, follow the instructions given in the [Bluetooth classic only mode](#) installation.

## 4.5 Installation in Wi-Fi + Bluetooth LE Coexistence Mode

Ensure that the dev\_oper\_mode is set as below.

- dev\_oper\_mode = 9 (Wi-Fi STA + BLE)
- dev\_oper\_mode = 10 (Wi-Fi AP + BLE)

Once the installation completed and interface of Wi-Fi and Bluetooth are detected follow below procedure to install Wi-Fi and BLE.

- For Wi-Fi, follow the instructions given in [Wi-Fi station mode](#) or [Wi-Fi Access Point \(AP\) mode](#) in alone mode for Wi-Fi installation.
- Also for Bluetooth LE protocol, follow the instructions given in [Bluetooth LE only mode](#) installation.

## 4.6 Installation in Wi-Fi + Bluetooth Classic + Bluetooth LE Coexistence Mode

Ensure that the dev\_oper\_mode is set as below

- dev\_oper\_mode = 13 ( Wi-Fi STA + BT + BLE)
- dev\_oper\_mode = 14 ( Wi-Fi AP + BT + BLE)

Once the installation completed and interface of Wi-Fi and Bluetooth are detected follow below procedure to install Wi-Fi and BLE.

- For Wi-Fi, follow the instructions given in [Wi-Fi station mode](#) or [Wi-Fi Access Point \(AP\) mode](#) in alone mode for Wi-Fi installation.
- Also for Bluetooth LE protocol, follow the instructions given in [Bluetooth LE only mode](#) installation.
- Also for Bluetooth classic protocol, follow the instructions given in [Bluetooth classic only mode](#) installation.
- In coex modes, to know the device type for BT i.e., device is supporting LE or BR/EDR use below command.

```
# hciconfig -a hci<x> features
```

### Example output 1:

For LE Opermode i.e., dev\_oper\_mode = 8

```
hci1:      Type: BR/EDR  Bus: USB
BD Address: 88:DA:1A:16:E4:4F  ACL MTU: 251:5  SCO MTU: 0:0
Features page 0: 0xbf 0xfe 0x0d 0xbe 0xfb 0xff 0x41 0x85
<3-slot packets> <5-slot packets> <encryption> <slot offset>
<timing accuracy> <role switch> <sniff mode> <RSSI>
<channel quality> <SCO link> <HV2 packets> <HV3 packets>
<u-law log> <A-law log> <CVSD> <power control>
<transparent SCO> <EDR ACL 2 Mbps> <EDR ACL 3 Mbps>
<enhanced iscan> <interlaced iscan> <interlaced pscan>
<extended SCO> <EV4 packets> <EV5 packets> <AFH cap. secondary>
<AFH class. secondary> <BR/EDR not supp.> <LE support>
<3-slot EDR ACL> <5-slot EDR ACL> <sniff subrating>
<pause encryption> <AFH cap. main> <AFH class. main>
<EDR eSCO 2 Mbps> <EDR eSCO 3 Mbps> <3-slot EDR eSCO>
<extended inquiry> <non-flush flag> <LSTO> <EPC>
<extended features>
Features page 1: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

## Example output 2:

For Classic(BT BR/EDR) Only i.e., dev\_oper\_mode = 4

```
hci1:    Type: BR/EDR  Bus: USB
BD Address: 88:DA:1A:16:E4:4F  ACL MTU: 1021:3  SCO MTU: 64:3
Features page 0: 0xbf 0xfe 0x0d 0xfe 0x9b 0xff 0x59 0x87
<3-slot packets> <5-slot packets> <encryption> <slot offset>
<timing accuracy> <role switch> <sniff mode> <RSSI>
<channel quality> <SCO link> <HV2 packets> <HV3 packets>
<u-law log> <A-law log> <CVSD> <power control>
<transparent SCO> <EDR ACL 2 Mbps> <EDR ACL 3 Mbps>
<enhanced iscan> <interlaced iscan> <interlaced pscan>
<inquiry with RSSI> <extended SCO> <EV4 packets> <EV5 packets>
<AFH cap. secondary> <AFH class. secondary> <3-slot EDR ACL>
<5-slot EDR ACL> <sniff subrating> <pause encryption>
<AFH cap. main> <AFH class. main> <EDR eSCO 2 Mbps>
<EDR eSCO 3 Mbps> <3-slot EDR eSCO> <extended inquiry>
<simple pairing> <encapsulated PDU> <non-flush flag> <LSTO>
<inquiry TX power> <EPC> <extended features>
Features page 1: 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Features page 1: 0x30 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

## Example output 3:

For the Classic and LE i.e., dev\_oper\_mode = 12

```
hci1:    Type: BR/EDR  Bus: USB
BD Address: 88:DA:1A:16:E4:4F  ACL MTU: 1021:3  SCO MTU: 64:3
Features page 0: 0xbf 0xfe 0x0d 0xfe 0xdb 0xff 0x5b 0x87
<3-slot packets> <5-slot packets> <encryption> <slot offset>
<timing accuracy> <role switch> <sniff mode> <RSSI>
<channel quality> <SCO link> <HV2 packets> <HV3 packets>
<u-law log> <A-law log> <CVSD> <power control>
<transparent SCO> <EDR ACL 2 Mbps> <EDR ACL 3 Mbps>
<enhanced iscan> <interlaced iscan> <interlaced pscan>
<inquiry with RSSI> <extended SCO> <EV4 packets> <EV5 packets>
<AFH cap. secondary> <AFH class. secondary> <LE support>
<3-slot EDR ACL> <5-slot EDR ACL> <sniff subrating>
<pause encryption> <AFH cap. main> <AFH class. main>
<EDR eSCO 2 Mbps> <EDR eSCO 3 Mbps> <3-slot EDR eSCO>
<extended inquiry> <LE and BR/EDR> <simple pairing>
<encapsulated PDU> <non-flush flag> <LSTO> <inquiry TX power>
<EPC> <extended features>
Features page 1: 0x03 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Features page 2: 0x30 0x00 0x00 0x00 0x00 0x00 0x00 0x0
```

## 4.7 Uninstalling the Driver

To uninstall the driver, follow the procedure below.

If wpa supplicant method is used to connect in STA mode, kill wpa supplicant using below command.

```
# killall wpa_supplicant
```

To kill hostapd application, use the below command.

```
# killall hostapd
```

To remove the driver, use the commands below.

```
# rmmod rsi_usb  
# rmmod rsi_sdio  
# rmmod rsi_91x
```

After uninstalling the driver, the created wireless interface disappears.



## 5 Configuration Using Wireless Tools

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

### 5.1 Using the 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 driver. For a detailed description of 'iw' tool, please refer to the relevant man pages on Linux system. The list of supported commands via "iw" tool are listed below.

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 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.
Example	\$ iw dev wifi0 connect TEST_AP 00:23:a7:00:05:55 The above command connects to TEST_AP access point in open mode
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
Interface Info	
Description	This command is used to get information about the device .
Default Value	-
Input Parameters	Interface name.
Output Parameter	Interface mac address, type, operating mode etc.
Reset Required	No
Usage	iw dev <interface_name> info
Example	iw dev wifi0 info
Station Dump	
Description	This command is used to station statistic information such as the amount of tx/rx bytes, the last TX bitrate (including MCS rate)
Default Value	-
Input Parameters	Interface name.
Output Parameter	Connected Stations/AP mac address,tx bytes, rx bytes, signal level etc,. will be displayed.
Reset Required	No
Usage	iw dev <interface_name> station dump
Example	iw dev wifi0 station dump

### Set Power save mode

Description	This command is used to set power save mode on/off in station mode.
Default Value	-
Input Parameters	Interface name.
Output Parameter	No
Reset Required	No
Usage	<code>iw dev &lt;interface_name&gt; set power_save &lt;on   off&gt;</code>
Example	<code>iw dev wifi0 set power_save &lt;on   off&gt;</code>

### Get Power save mode

Description	This command is used to get power save mode on/off in station mode.
Default Value	-
Input Parameters	Interface name.
Output Parameter	Shows whether power save mode is on   off in station mode
Reset Required	No
Usage	<code>iw dev &lt;interface_name&gt; get power_save</code>
Example	<code>iw dev wifi0 get power_save</code>

Set Rate	
Description	This command is used to fix data rate.
Default Value	-
Input Parameters	Interface name, rate
Output Parameter	-
Reset Required	No
Usage	<p>iw dev &lt;interface_name&gt; set bitrates legacy-&lt;2.4 5&gt; &lt;legacy rate in Mbps&gt;</p> <p>For mcs rates, in iw versions below 3.14 use:</p> <p>iw dev &lt;interface_name&gt; set bitrates mcs-&lt;2.4 5&gt; &lt;mcs rate in Mbps&gt;</p> <p>In iw versions above 3.14 use:</p> <p>iw dev &lt;interface_name&gt; set bitrates ht-mcs-&lt;2.4 5&gt; &lt;mcs rate in Mbps&gt;</p> <p>Above command(s) with no rate specified is used to set things to normal (auto rate).</p> <p>Ex: iw dev &lt;interface_name&gt; set bitrates mcs-&lt;2.4 5&gt;</p> <p>OR</p> <p>iw dev &lt;interface_name&gt; set bitrates ht-mcs-&lt;2.4 5&gt;</p> <p><b>Note:</b> 1. Driver supports only single rate to be set using above commands. Multiple rate setting is not supported.</p> <p>2. For kernel version greater than 4.13.16 setting bitrate legacy will take only basic rates.</p> <p>3. iw version can be obtained using below command</p> <p># iw --version</p>
Example	<p>iw dev wlan0 set bitrates legacy-2.4 12</p> <p>iw dev wlan0 set bitrates mcs-2.4 1</p> <p>iw dev wlan0 set bitrates ht-mcs-5 4</p>
Set country code	
Description	This command is used to set the country code.
Default Value	-
Input Parameters	country_code
Output Parameter	-
Reset Required	-
Usage	iw reg set <country_code>
Example	<p>iw reg set IN (For India)</p> <p>iw reg set JP (For Japan)</p> <p>iw reg set GE (For Germany)</p>

### Get country code

Description	This command is used to get the country code.
Default Value	-
Input Parameters	-
Output Parameter	country domain
Reset Required	-
Usage	iw reg get
Example	iw reg get output : country IN: DFS-JP (2402 - 2482 @ 40), (N/A, 20) (5170 - 5250 @ 80), (N/A, 20) (5250 - 5330 @ 80), (N/A, 20) , DFS (5735 - 5835 @ 80), (N/A, 20)

### Set Tx Power

Description	This command is used to set the Tx Power.
Default Value	-
Input Parameters	tx power value
Output Parameter	-
Reset Required	-
Usage	iwconfig <interface_name> txpower <NmW NdBm off auto>
Example	iwconfig wlan0 txpower 11

To understand the list of channels allowed in the current regulatory domain, please check the below link.  
[https://en.wikipedia.org/wiki/List\\_of\\_WLAN\\_channels](https://en.wikipedia.org/wiki/List_of_WLAN_channels)

## Regulatory mapping

Mapping of country code to regulatory region code for Caracalla

Sr.No	Country	Country code	Region code
1	Australia	AU	ETSI
2	Austria	AT	ETSI
3	Belgium	BE	ETSI
4	Brazil	BR	WORLD
5	Canada	CA	FCC
6	Chile	CL	WORLD
7	China	CN	WORLD
8	Colombia	CO	FCC
9	Czech Republic	CZ	ETSI
10	Denmark	DK	ETSI
11	Finland	FI	ETSI
12	France	FR	ETSI
13	Germany	DE	ETSI
14	Hong Kong	HK	WORLD
15	India	IN	WORLD
16	Indonesia	ID	WORLD
17	Ireland	IE	ETSI
18	Israel	IL	ETSI
19	Italy	IT	ETSI
20	Japan	JP	TELEC
21	Republic of Korea	KR	WORLD
22	Luxembourg	LU	ETSI
23	Malaysia	MY	WORLD
24	Mexico	MX	FCC
25	Morocco	MA	WORLD
26	Netherlands	NL	ETSI

### Note:

If there are multiple phy's, i.e., there are several instances of cfg80211 being used by different modules, then 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's which are obtained from the previous command. The module MAC address (xx:xx:xx:xx:xx:xx) has to be used in the field 'macaddress'.

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> 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 commands that are supported only in the Access Point mode are as follows:

```
$ iw dev <devname> set channel <channel> [HT20]
```

```
$ iw dev <devname> set freq <freq> [HT20]
```

```
$ iw dev <devname> station del <MAC address>
```

```
$ iw dev <devname> station get <MAC address>
```

## 5.2 Software Rfkill

The driver has support for RFKill command. As a pre-requisite, please install the rfkill package.

1) In order to list out the wireless interfaces in the system, use the command given below.

```
# rfkill list
```

2) To block the 9116 Wi-Fi interface, use the command given below:

```
# rfkill block <interface_number_listed_in_rfkill_list>
```

3) To unblock the 9116 Wi-Fi interface, use the command given below:

```
# rfkill unblock <interface_number_listed_in_rfkill_list>
```

## 6 Wi-Fi Protected Setup (WPS)

Wi-Fi Protected Setup (WPS) is a standard for easy and secure wireless network setup and connections. The OSD driver supports the following configuration method:

- Push Button Method

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

**Access Point:** WLAN access point

**Registrar:** A device that controls 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.)

### 6.1 STA Mode WPS Configuration

A WPS Configuration file is used for setting up a connection with a remote Access Point. 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=9116
serial_number=03
device_type=1-0050F204-1
os_version=01020300
config_methods=display push_button keypad
```

The steps for configuring WPS in Client mode are as follows:

1. Start the driver in Client mode.
2. Start the supplicant by entering the following command without any network block.

```
# wpa_supplicant -i <vap_name> -D nl80211 -c <wps_conf_file> -dddt
```

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> -p <path of ctrl sockets> wps_pbc <bssid>
```

This is the Access Point's MAC address. If the BSSID is not known, the input parameter will be the string named "any"

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



## 6.2 AP Mode WPS Push Button Configuration

WPS support is not enabled in default Hostapd application. Below configuration needs to be enabled in hostapd build configuration (.config) and compile hostapd.

```
CONFIG_WPS=y  
CONFIG_WPS_UPNP=y
```

Below changes needs to be enabled in hostapd configuration file .

```
wpa_key_mgmt=WPA-PSK  
wpa_psk_file=/etc/hostapd.psk  
eap_server=1  
wps_state=2  
ap_setup_locked=1  
uuid=12345678-9abc-def0-1234-56789abcdef0  
device_name=RS9116_n-Link  
model_name=OneBox-Mobile  
serial_number=000000000001  
device_type=1-0050F204-1  
os_version=01020300  
config_methods=display push_button keypad
```

The steps for configuring WPS in AP mode are as follows:

1. Start the driver in AP mode.
2. Start the Hostapd with configuration file as input (ex: ap.conf) as given below:

```
# hostapd ap.conf -dddt > log_file &
```

3. For Push button enter the below command , This pbc mode lasts up to two minutes, within two minutes a client needs to be connected else AP will move out of WPS-PBC.

```
# hostapdcli wps_pbc
```

4. For station in WPS-PBC user can follow the steps in [STA mode WPS](#) configuration and try connecting with AP .

## 7 Sniffer Mode (Promiscuous Mode)

The Steps for operating the device in Sniffer Mode are outlined below.

1) Install the driver using the below commands

```
# insmod rsi_91x.ko rsi_zone_enabled=1 dev_oper_mode=1 driver_mode_value=7
```

2) According to the interface usb or sdio

```
# insmod rsi_usb.ko
OR
# insmod rsi_sdio.ko
```

3) Make sure that interface is Down if not use below command to down the interface

```
# ifconfig <interface_name> down
```

4) Change the default interface to monitor using below command

```
# iwconfig < interface_name> mode monitor
OR
# iw dev <interface_name> set type monitor
```

5) Make the interface up using below command.

```
# ifconfig <interface_name> up
```

6) Set the channel in which you want to capture the on air packets

```
# iwconfig <interface_name> channel < channel no>
OR
# iw dev <interface_name> set channel <channel no>
```

7) Use any network packet analysis tool to see captured packets

```
# wireshark &
OR
# tcpdump &
```

## 8 Background Scan & Roaming

Background scanning and roaming can be verified using wpa\_supplicant.

It is recommended to use supplicant version greater than 2.6 for better performance in roaming.

To use this facility, user needs to ensure the flag **CONFIG\_BGSCAN\_SIMPLE** is enabled in the supplicant build configuration file (.config).

This will enable building BGSCAN SIMPLE module which is responsible for requesting background scans for the purpose of roaming within ESS. If this option is not enabled, rebuild wpa\_supplicant binary with this option.

'bgscan' parameters use the following format:

```
bgscan="simple:<short_bgscan_intrvl_in_secs>:<signal_strength_thrshld>:<long_bgscan_intrvl_in_secs>"
```

This line should be present either inside a network block or outside of all network blocks based on the requirement.

Eg: bgscan="simple: 30:-45:300"

If user does not require bgscan he has to disable bgscan in the supplicant config and should not include above configurations in the wpa\_supplicant.conf file. This cannot be done if the user is connected through network manager.

### 8.1 Configuring Background Scan Parameters through debugfs

For Bgscan, f/w requires some of the parameters to be configured. Default values are configured if user doesn't configure them through debugfs. Below commands are used to configure bgscan params,.

1) To verify the bgscan status and parameters

```
# cat /sys/kernel/debug/phy<X>/bgscan
```

2) To enable background scan and configure its parameters from debugfs:

```
# echo 1 10 10 20 20 100 1 3 1 6 11 > /sys/kernel/debug/phy<X>/bgscan
```

The input parameters of the background scan command are explained below.

- **<background\_enable>** : To enable the background scan.
- **<bgscan\_threshold>** : The Background scan threshold is referred to as the RSSI Upper Threshold. At every background scan interval, the n-Link® module decides whether to initiate 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. During a passive scan, if any beacon is received in a channel, then the recommended value for this parameter will be 100ms. The 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
  1. 0 – Disable
  2. 1 – Enable
- **<num\_of\_bgscan\_channels>**: Specifies the 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.
  - 3) To disable background scan which is happening in channels provided through debugfs.

```
# echo 0 > /sys/kernel/debug/phy<X>/bgscan
```

- 4) For checking the list of bgscan channels configured to device use below command. This will display the list of bgscan channels configured to device with DFS indication also.

```
# cat /sys/kernel/debug/phy<X>/bgscan
```

## 8.2 Setting Bgscan SSID through Debugfs

If Background scan is happening, then user can set bgscan ssid to send two probe request one without SSID and other one with SSID for that we need to follow below steps to set bgscan SSID through debugfs.

- 1) Set the bgscan SSID using below command

```
# echo <bgscan_ssid_name> >/sys/kernel/debug/phy<X>/bgscan_ssid
```

- 2) Check whether the bgscan ssid is getting set or not using below command

```
# cat /sys/kernel/debug/phy<X>/bgscan_ssid
```

- 3) After setting bgscan\_ssid give below command to update bgscan params with two probe enabled.

```
# echo 1 10 10 30 20 70 1 6 1 2 3 4 5 6 >/sys/kernel/debug/phy<X>/bgscan
```

- 4) Follow the instruction present in above section to update the bgscan parameters.
- 5) Check the sniffer capture we will observe two probe request one with SSID and other one without SSID.

## 9 Power Save

### 9.1 Power Save Modes

The RS9116 modules broadly support two types of power save modes. They are outlined below:

- **Low Power (LP) Mode:** The PHY (RF and Baseband) and LMAC sections are powered off but the Host Interface section of the module is 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 interrupts logic for waking up the module. The module cannot respond to the Host processor's commands/requests unless and until it gets wake 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.

Out of two ULP Handshake Modes (signal (GPIO) based or Packet (message) based), GPIO based mode is more effective in power save. If target platform does not have free/spare GPIO, they can use message based mode.

In Wi-Fi, only Client (Station) mode supports power save. By default, the module will be in power save disable state, user has to enable it explicitly.

### 9.2 Device Sleep Mode

For each of the above power save modes, the module supports following sleep modes. They are outlined below:

- **Deep Sleep:** The module is in deep sleep mode when it is not connected to an Access Point. The Deep Sleep is defined by the <deep\_sleep\_wakeup\_period> parameter of the PS command.
- **Connected Sleep:** In the connected state, the module can operate in Max PSP or UAPSD. 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.
- **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.

### 9.3 Wakeup Procedures and Data Retrieval

Device acting as station 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\_dtim\_per\_sleep>

- **UAPSD:** Two types of UAPSD power save modes are supported as explained below. For using UAPSD, <enabled\_uapsd> module param has to be set to 1. Also <max\_sp\_len> module param needs to be configured as required while installing the modules. In addition to these configurations, <uapsd\_wakeup\_period> can be configured using debugfs ps\_params command to select one of the two variants of UAPSD as explained below.
  - i. **Periodic UAPSD:** This mode is enabled by default if power save enabled by iw dev command (as given below) and takes <uapsd\_wakeup\_period> as 100. This can be configured using the ps\_params command with the non zero value in <uapsd\_wakeup\_period> parameter. For this mode, the wakeup period can be assigned with a value ranging between 10 and 100 milliseconds. If it is supported by the Access Point, then in this mode, the module wakes up at the end of each sleep period and transmits pending data or a QoS Null packet in order to retrieve the data from the Access Point. The sleep period is governed by the parameter set in the ps\_params command in addition to the <uapsd\_wakeup\_period>. The sleep period has the minimum of the values programmed using the above command. 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.
  - ii. **Transmit based UAPSD:** If <uapsd\_wakeup\_period> parameter is set to 0 in the ps\_params command, then 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 ps\_params command. If the Access Point does not support UAPSD, the module mimics this mode by waking up whenever there is a packet to be transmitted. It generally transmits the packet and then retrieves the pending data from the Access Point by sending a PS\_POLL packet.

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

## 9.4 Configuring LP Device Power Save for USB and SDIO Interface

Install the driver as follow to enable LP power save.

```
# insmod rsi_91x.ko rsi_zone_enabled=1 dev_oper_mode=<value> ps_sleep_type=1
```

For USB interface :

```
# insmod rsi_usb.ko
```

For SDIO interface :

```
# insmod rsi_sdio.ko
```

## 9.5 Configuring ULP device power save for SDIO interface

Install the driver as follow to enable ULP power save

```
# insmod rsi_91x.ko rsi_zone_enabled=1 dev_oper_mode=<value> ps_sleep_type=2  
ulp_handshake_mode=<value1>
```

For No handshake : ulp\_handshake\_mode = 0

For GPIO based handshake : ulp\_handshake\_mode = 1

For Packet based handshake : ulp\_handshake\_mode = 2

```
# insmod rsi_sdio.ko
```

## 9.6 Configuration of ULP GPIO Handshake and GPIO Numbers

For GPIO handshake driver requires two GPIO pins. These pins need to be configured by the user as module params.

Install the driver as follow to enable GPIO handshake

```
# insmod rsi_91x.ko ps_sleep_type=<value> ulp_handshake_mode=1 ulp_gpio_read=X
ulp_gpio_write=Y
```

1) For LP power save : ps\_sleep\_type = 1

For ULP power save : ps\_sleep\_type = 2

ULP GPIO handshake supported for sdio interface only.

- Here 'X' and 'Y' are the platform GPIO's need to used for GPIO handshake.
- Platform GPIO 'X' should be connected with UULP3 /UULP0 in the rsi\_EVB. (Particular PIN can be selected using sleep\_ind\_gpio\_sel module param)
- Platform GPIO 'Y' should be connected with UULP2 in the rsi\_EVB.

## 9.7 Enabling Power Save

Power save can be enabled or disabled through command line using iw commands. By default 802.11 default power save is enabled if Coex mode is enabled. UAPSD is enabled based on AP's UAPSD configuration.

Following are the commands used in power save configuration.

1) Enable the power save:

```
# iw dev <interface_name> set power_save on
```

2) Disable power save:

```
# iw dev < interface_name> set power_save off
```

3) Check the power save status:

```
# iw dev <interface_name> get power_save
```

Here interface\_name will vary from host to host we can get that interface name with below command's

```
# iw dev
```

In case of BT coexistence with Wi-Fi, we need to give bt power save command.

```
# hcitool -I hci0 cmd <vendor command> <power save related> <power save ON/OFF>
<ULP(0x02)/LP(0x01) power save> <sleep duration>
```



```
Example:hcitool -i hci0 cmd 0x3f 0x0003 0x01 0x02 0xff
```

## 9.8 Configure Power Save Parameters/Profiles through debugfs Dynamically

Driver supports dynamic configuration of power save type and profile parameters using debugfs as explained below.

To update power save parameter use below command

```
# echo <sleep_type> <tx_threshold> <rx_threshold> <tx_hysteresis> <rx_hysteresis> <monitor_interval>  
<listen_interval_duration> <num_beacons_per_listen_interval> <dtim_interval_duration>  
<num_dtim_per_sleep> <deep_sleep_wakeup_period> <uapsd_wakeup_period> >/sys/kernel/debug/phy<X>/  
ps_params
```

The input parameters of the power save command are explained below.

- **<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 the module can decide to enter (throughput ≤ threshold) or exit (throughput > threshold) the power save mode. The value is in Mbps and Supported TX threshold is 0 to 10Mbps
- **<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 the module can decide to enter (throughput ≤ threshold) or exit (throughput > threshold) the power save mode. The value is in Mbps and Supported RX threshold is 0 to 10Mbps
- **<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 be 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 specifies 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).
- **<listen\_interval\_duration>**: This parameter specifies 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\_dtim\_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 the deciding factor 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

Note: Listen interval duration greater than one sec (> 1sec ) is not supported

- **<num\_beacons\_per\_listen\_interval>**: This parameter specifies 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 to 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 specifies 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 expires. This parameter can be used when DTIM information is not available. If a non-zero value is assigned to this parameter, then 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 specifies 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 to 0. The maximum value for this parameter is 10. This parameter is considered only after the module is connected to the access point.
- **<deep\_sleep\_wakeup\_period>:** This parameter specifies 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.
- **<uapsd\_wakeup\_period>:** This parameter specifies the duration (in milliseconds) for which the module sleeps after connection if the AP supports uapsd. For value 10-100 milliseconds it will work as periodic uapsd and for value 0 it will work as transmit based uapsd.

## 9.9 USB Auto Suspend

USB auto suspend is enabled by default and we can see “Killing URB’s” print periodically in dmesg log

1) To check if it is enabled or not use below command

```
# cat /sys/bus/usb/devices/2-4/power/control
```

Note: ‘2-4’ varies from device to device, you can find this in the dmesg log, while we insert the device as below, “usb 2-4: New USB device found, idVendor=1618, idProduct=9116”.

If the value is :

- auto : auto suspend is enabled
- on : auto suspend is disabled

2) To modify the value use below command

```
# echo auto > /sys/bus/usb/devices/2-4/power/control
```

3) To check the auto suspend period with below command

```
# cat /sys/bus/usb/devices/2-4/power/autosuspend_delay_ms
```

4) The value is 2000ms by default, we can change it with below command

```
# echo value > /sys/bus/usb/devices/2-4/power/autosuspend_delay_ms
```

## 10 Steps to Configure 802.11W (PMF)

### 10.1 Configuring and Compiling Driver for PMF in client mode

1. Enable CONFIG\_IEEE80211W=y in wpa\_supplicant .config.
2. Enable WPA-PSK-SHA256 as key\_mgmt in network block in supplicant sta\_settings.conf
  - a. pmf=1/2, PMF is enabled/required correspondingly .

```
pmf=2
network = {
    ssid="SSID of Access Point"
    pairwise=CCMP
    group=CCMP
    key_mgmt=WPA-PSK-SHA256
    psk="12345678"
    proto=WPA2
    priority=1
}
```

3. Configure AP as MFP Capable/Required.

### 10.2 Configuring and Compiling Driver for PMF in AP Mode

1. Enable CONFIG\_IEEE80211W=y in hostapd .config
2. Enable WPA-PSK-SHA256 as key\_mgmt in hostapd\_ccmp.conf
3. Make sure below options are enabled apart from your configuration.

```
# This field is a bit field that can be used to enable WPA (IEEE 802.11i/D3.0)
# and/or WPA2 (full IEEE 802.11i/RSN):
# bit0 = WPA
# bit1 = IEEE 802.11i/RSN (WPA2) (dot11RSNAEnabled)
wpa=2

# ieee80211w: Whether management frame protection (MFP) is enabled
# 0 = disabled (default)
# 1 = optional
# 2 = required
ieee80211w=2

wpa_key_mgmt=WPA-PSK-SHA256
group_mgmt_cipher=AES-128-CMAC
```

## 11 Antenna Selection

RS911x module supports two antennas i.e., internal and external antennas. If hardware supports, user can select particular antenna using two ways.

1. Using antenna\_sel module param
2. Using iw phy command

### 11.1 Using antenna\_sel module param

user can configure antenna\_sel module param while inserting the driver with the following command.

```
# insmod rsi_91x.ko antenna_sel=2/3
```

**antenna\_sel=2** to select Internal antenna

**antenna\_sel=3** to select External antenna

**Note:** This is one time configurable parameter and supports in all oper modes

### 11.2 Using iw phy command

#### 11.2.1 To Select External Antenna

Follow the steps given below:

- 1) Make the interface down (deactivate the interface)

```
# ifconfig <interface_name> down
```

- 2) Set the antenna

```
# iw phy <phyX> set antenna 1 0
```

- 3) Make the interface up (activate the interface)

```
# ifconfig <interface_name> up
```

By default internal antenna is configured.

#### 11.2.2 To Select Internal Antenna

Follow the steps given below:

- 1) Make the interface down (deactivation)

```
# ifconfig <interface_name> down
```

## 2) Set the antenna

```
# iw phy <phyX> set antenna 0 0
```

## 3) Make the interface up (activation)

```
# ifconfig <interface_name> up
```

**Note:** This command doesn't support in BT/LE alone cases

## 12 Bluetooth hciconfig and hcitool Usage

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

<b>Reset</b>	
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
<b>Read Local Version Information</b>	
Description	This command is used to read the local version information
Default Value	-
Input Parameters	None
Output Parameter	We have to give "btmon" in separate terminal before giving this command. So, the below parameters will be available. HCI version HCI revision LMP version Manufacturer name LMP subversion
Reset Required	No.
Usage	hcitool -i <hciX> cmd 0x04 0x01
<b>Read Local Supported Commands</b>	
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
<b>Get Local BD Address</b>	
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

<b>Start Inquiry</b>	
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>
<b>Write Local Name</b>	
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>  For example: name = ABCD → to set as Local Name hcitool -i <hciX> cmd 0x03 0x13 0x41 0x42 0x43 0x44  And Read Local Name using hcitool -i <hciX> cmd 0x03 0x14  and check the name in "btmon" window
<b>Sniff Mode command</b>	
Description	This command is used to keep BT device in Sniff Mode
Default value	None
Input Parameter	Connection Handle -0x1 Sniff max interval -0x0190 (250 msec) Sniff min interval – 0x0190(250 msec) Sniff attempt – 0x0005(6.25 msec) Sniff timeout – 0x0002 (2.50 msec)
Output parameter	None
Reset Required	No
Usage	hcitool -i hci<x> cmd 0x02 0x0003 <Connetion handle > < Sniff Max Interval> <Sniff Min Interval> <sniff attempt> <sniff timeout>
Example	hcitool -i hci0 cmd 0x02 0x0003 0x01 0x00 0x90 0x01 0x90 0x01 0x05 0x00 0x02 0x00

## 12.1 BT Device Connection using Bluetoothctl

Below are the set of commands and steps to be followed to successfully pair 9116 BT device with a third-party BT dongle/Mobile phone. Please make sure bluez-tools are installed on the system where the 9116 driver is installed.

Use the following commands to pair / trust / connect to another Bluetooth device.

1. bluetoothctl -a (upon executing this command user will enter into a Bluetooth mode console, and user needs enter the following commands)
2. power on
3. agent on
4. scan on (when this command is entered, you will see the scan list of other Bluetooth devices on the console)
5. scan off
6. pair <Target BD address> (it will ask for a key confirmation)
7. trust <Target BD address>
8. Connect <Target <BD address>
9. After successful connection, user can see the connection status on the command line as well as using the command "hcitool -l hciX con", here X can be 0,1,2,3..

Example:

```
# bluetoothctl -a
[bluetooth]# power on
[bluetooth]# scan on
[bluetooth]# pair 64:CC:2E:9C:23:BA
[bluetooth]# connect 64:CC:2E:9C:23:BA
Attempting to connect to 64:CC:2E:9C:23:BA
[bluetooth]# disconnect CC:2E:9C:23:BA
Attempting to connect to 64:CC:2E:9C:23:BA
[CHG] Device 64:CC:2E:9C:23:BA Connected: yes
[CHG] Device 64:CC:2E:9C:23:BA Modalias: bluetooth:v001Dp1200d1436
[CHG] Device 64:CC:2E:9C:23:BA UUIDs: 00001103-0000-1000-8000-00805f9b34fb
[CHG] Device 64:CC:2E:9C:23:BA UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device 64:CC:2E:9C:23:BA UUIDs: 00001106-0000-1000-8000-00805f9b34fb
[CHG] Device 64:CC:2E:9C:23:BA UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device 64:CC:2E:9C:23:BA UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
```

Now, user can do data transfer by transferring a file to the target device and run some tests like l2test, l2ping etc..

## 12.2 L2 Test Commands

- 1.) The below commands is used for data transfer for throughput.  
For Tx use below command:

```
# l2test -i <hciX> -s <Mac_addr_of_Rx_device>
```

For Rx use below command:

```
# l2test -i <hciX> -r
```

Note : If the receiver needs specific bytes of data to be received, need to use this command

```
# l2test -i <hciX> -r -b <n>
```

Where b-> bytes and n-> number of bytes to be in Tx/Rx.

Example 1 : If the Rx needs to be 672 bytes, the command should be as follows :

```
# l2test -i <hciX> -r -b 672
```

Example 2 : If the Rx needs to be 10 MB, the command should be as follows :

```
# l2test -i <hciX> -r -b 10000000
```

2.) For EDR (3 mbps) , Install the driver as follows :

```
# insmod rsi_91x.ko rsi_zone_enabled=1 dev_oper_mode=4 BT_BDR_MODE=0 (3Mbps)
```

```
# insmod rsi_sdio.ko
```

For Tx use below command:

```
# l2test -i <hciX> -s <Remote_BD_ADDR> -l 1014 -o 1014
```

For Rx use below command:

```
# l2test -i <hciX> -b 100000 -l 1014 -o 1014
```

3.) The below commands is used for Poll-Null data transfer.

For Tx use below command:

```
# l2test -i <hciX> -n <MAC_addr_of_Rx_device>
```

For Rx use below command:

```
# l2test -i <hciX> -r
```



## 13 PER Driver

The RS9116 Proprietary driver software provides application to test Transmit and Receive Performance of RS9116 Module in PER test mode. The GUI interface is provided to ease the user effort in evaluating the product.

Current release of Open Source Driver does not support PER, there by we recommend you to use Proprietary driver to validate PER with current firmware.

For this please follow below steps.

1. Download Proprietary driver from below link.  
[https://siliconlabs.force.com/SL\\_CommunitiesLogin?ec=302&startURL=%2Fs%2Flicense-agreement%3FName%3DRS9116.NB0.NL.GENR.LNX](https://siliconlabs.force.com/SL_CommunitiesLogin?ec=302&startURL=%2Fs%2Flicense-agreement%3FName%3DRS9116.NB0.NL.GENR.LNX)
2. Unzip the downloaded software package.
3. Copy firmware binary files (pmemdata & pmemdata\_wlan\_bt\_classic) files from **RS9116.NB0.NL.GNU.LNX.OSD.2.0.0.000X/Firmware** directory of Open Source Driver to the **RS9116.NB0.NL.GENR.LNX.1.2.24.0013/source/host/release/firmware** directory of downloaded software package.
4. Follow steps given in the PER\_GUI\_USER\_GUIDE\_v1.x.pdf ( found in **RS9116.NB0.NL.GENR.LNX.1.2.24.0013/Docs** directory) for evaluation of PER tests.

## 14 Update WLAN Region-Based Maximum Powers from Driver

Internally firmware maintains two tables for Maximum powers: Worldwide table & Region based table. Worldwide table is populated by firmware with Max power values that chip can transmit that meets target specs like EVM. Region based table has default gain value set.

1)When certifying with user antenna, Region has to be set to Worldwide and sweep the power from 0 to 21dBm. Arrive at max power level that is passing certification especially band-edge.

2)These FCC/ETSI/TELEC/KCC Max power level should be loaded in end-to-end mode via WLAN User Gain table. This has to be called done every boot-up since this information is not saved inside flash. Region based user gain table sent by application is copied onto Region based table. SoC uses this table in FCC/ETSI/TELEC/KCC to limit power and not to violate allowed limits.

For Worldwide region firmware uses Worldwide table for Tx. For other regions(FCC/ETSI/TELEC/KCC), Firmware uses min value out of Worldwide & Region based table for Tx. Also, there will be part to part variation across chips and offsets are estimated during manufacturing flow which will be applied as correction factor during normal mode of operation.

This frame has to be used by customers who has done FCC/ETSI/TELEC/KCC certification with their own antenna. All other customers should not use this. Inappropriate use of this frame may result in violation of FCC/ETSI/TELEC/KCC or any certifications and Silicon labs is not liable for that.

Following are the steps need to be followed to program region-based maximum powers to the WLAN device.

1. Compile and insert the driver by following [Compilation Step](#) and [Installing n-link driver](#) sections.
2. Go to the rsi/release folder.
3. Open and update maximum powers in wlan\_gain\_table.txt file. Follow below **Gain table structure** format while updating the file.
4. Now give below commands.

```
#./onebox_util rpine0 update_wlan_gain_table
```

With the above commands, gain tables programming will start and a successful programming should show below output in dmesg logs.

```
*****Successfully completed programming n gain tables*****
```

Here, 'n' is the number of structures given in wlan\_gain\_table.txt file.

### 14.1 Gain Table Structure Format:

1. Each value should be separated by ',' and space characters.
2. Multi-line comments not supported. (Eg. /\* comment \*/)
3. A space before comment is needed, for in-line comments. (Eg. <Data> //<Comment>)
4. Hexa-decimal numbers not supported.
5. No space between value and ',' character.
6. Supported table names and their maximum supported size are  
For 2GHz 20MHz --- \_RS9116\_HP\_MODE\_REGION\_BASED\_RATE\_VS\_MAXPOWER\_SINGLE\_BAND\_NONHPM    MAX\_LENGTH = 128 bytes  
For 5GHz 20MHz --- \_RS9113\_RS8111\_5G\_HP\_MODE\_REGION\_BASED\_RATE\_VS\_MAXPOWER\_NONHPM    MAX\_LENGTH = 64 bytes

NOTE: No support for other table names.

7. Gain table Format for 2G Band: (Each entry of the table is 1 byte)

In 2Ghz, Max Gain/Power obtained from certification should be doubled and loaded.

```

<TABLE NAME>[] = {
<NO.of Regions>,
<REGION NAME 1>, <CHANNEL_CODE_2G>,
<CHANNEL NUMBER 1>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
<CHANNEL NUMBER 2>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
.
.
<CHANNEL NUMBER m-1>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
<CHANNEL NUMBER m>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
<REGION NAME 2>, <CHANNEL_CODE_2G>,
<CHANNEL NUMBER 1>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
<CHANNEL NUMBER 2>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
.
.
<CHANNEL NUMBER m-1>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
<CHANNEL NUMBER m>, <2 * MAX POWER FOR b RATE>, <2 * MAX POWER FOR g RATE>, <2 * MAX POWER FOR n RATE>,
<REGION NAME 3>, <CHANNEL_CODE_2G>,
.
.
<REGION NAME y>, <CHANNEL_CODE_2G>,
};

```

Gain table Format for 5G Band: (Each entry of the table is 1 byte)

In 5Ghz, Max Gain/Power obtained from certification should be loaded.

```

<TABLE NAME>[] = {
<NO.of Regions>,
<REGION NAME 1>, <CHANNEL_CODE_5G>,
<CHANNEL NUMBER IN BAND 1 IF ANY>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<BAND_NUMBER 1>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<CHANNEL NUMBER IN BAND 2 IF ANY>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<BAND_NUMBER 2>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<CHANNEL NUMBER IN BAND 3 IF ANY>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<BAND_NUMBER 3>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<CHANNEL NUMBER IN BAND 4 IF ANY>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
<BAND_NUMBER 4>, <MAX POWER FOR 11a RATE>, <MAX POWER FOR n RATE>,
.
.
<REGION NAME y>, <CHANNEL_CODE_5G>,
};

```

#### 8. Supported Region names:

FCC, ETSI,TELEC, KCC

#### 9. <CHANNEL\_CODE\_2G> is a 8 bit value which is encoded as:

If Tx powers of all the channels are same, then use CHANNEL\_CODE\_2G as 17. In this case, mention channel number as 255.

If Tx power is not same for all channels, then indicate CHANNEL\_CODE\_2G as no-of channels. And specify tx power values for all the channels indicated.

#### 10. <CHANNEL\_CODE\_5G> is a 8 bit value encoded as number of rows in a region for 5G band.

##### a. 5G is divided into 4 sub bands:

- band 1: channel number <= 48
- band 2: channel number > 48 and channel number <= 64
- band 3: channel number > 64 and channel number <= 144
- band 4: channel number > 144

b. If any channel in a band has different set of power values, specify the channel number followed by power values.

- c. If all the channels in a band 1 has same power values, specify the band number as 1 followed by power value.
- d. If all the channels in a band 2 has same power values, specify the band number as 2 followed by power value.
- e. If all the channels in a band 3 has same power values, specify the band number as 3 followed by power value.
- f. If all the channels in a band 4 has same power values, specify the band number as 4 followed by power value.

#### Examples:

```

_RS9116_HP_MODE_REGION_BASED_RATE_VS_MAXPOWER_SINGLE_BAND_NONHPM[] = {
3, //NUM_OF_REGIONS
FCC, 13, //NUM_OF_CHANNELS
// rate, 11b, 11g, 11n
1, 34, 20, 20,
2, 34, 28, 28,
3, 34, 32, 32,
4, 34, 36, 36,
5, 34, 38, 38,
6, 34, 40, 40,
7, 34, 38, 38,
8, 34, 36, 36,
9, 34, 32, 32,

```

```

    10, 34, 32, 32,
    11, 34, 24, 24,
    12, 34, 16, 24,
    13, 34, 12, 12,
    TELEC, 17,
    255, 20, 16, 16,
    KCC, 17,
    255, 26, 20, 20,
}; //}}}

_RS9113_RS8111_5G_HP_MODE_REGION_BASED_RATE_VS_MAXPOWER_NONHPM[] = {
2,
FCC, 6,
    1, 9, 10, //band 1
    2, 8, 9, //band 2
    100, 4, 4, //band 3
    3, 6, 8, //band 3
    149, 3, 3, //band 4
    4, 6, 7, //band 4
TELEC, 4,
    1, 9, 10, //band 1
    2, 8, 10, //band 2
    3, 6, 8, //band 3
    4, 6, 7, //band 4
};

```

## 15 Appendix A: Driver Details

### 15.1 Debug Zone Prints

Driver provides an option to enable debug prints (which can be observed in dmesg) using two ways.

1. By assigning module param while installing the driver
2. Through debugfs after installing the driver.

Both are explained below.

1. Assigning module param:

```
# insmod rsi_91x.ko rsi_zone_enabled = 2
```

Here rsi\_zone enables is bitmap which is explained below.

2. Through debugfs :

```
# echo <value> > /sys/kernel/debug/phy<X>/debug_zone
```

rsi\_zone\_enabled or <value> are bitmapped values as explained below.

BIT	ZONE
BIT(0)	ERROR ZONE
BIT(1)	INFO ZONE
BIT(2)	INIT ZONE
BIT(3)	MGMT TX ZONE
BIT(4)	MGMT RX ZONE
BIT(5)	DATA TX ZONE
BIT(6)	DATA RX ZONE
BIT(7)	FSM ZONE
BIT(8)	ISR ZONE
BIT(9)	INT MGMT ZONE
BIT(10)	MGMT DEBUG ZONE

For example if the user wants to enable both INFO ZONE and ERROR ZONE then he has to enable BIT(0) & BIT(1). So rsi\_zone\_enabled value will be 3.

### 15.2 Version

User can check the Driver and LMAC version by giving the below command.

```
# cat /sys/kernel/debug/phy<X>/version
```

User can expect the sample result below:

```
Driver : RS911X.NB0.NL.GNU.LNX.OSD.2.0.1
LMAC   : 1610.2.0.0.0009
```

## 15.3 Station Stats

User can check the driver status as well as management and data packets stats by giving below command.

```
# cat /sys/kernel/debug/phy<X>/stats
```

User can expect the sample result below:

```
==> RSI STA DRIVER STATUS <==
DRIVER_FSM_STATE: (9)

total_mgmt_pkt_send : 114
total_mgmt_pkt_queued : 0
total_mgmt_pkt_freed : 0
total_data_vo_pkt_send:      4      total_data_vo_pkt_send:      0
total_data_vo_pkt_send:      4      total_data_vo_pkt_send:      0
total_data_vi_pkt_send:      0      total_data_vo_pkt_send:      0
total_data_vo_pkt_send:      0      total_data_vo_pkt_send:      0
total_data_be_pkt_send:    230      total_data_vo_pkt_send:      0
total_data_vo_pkt_send:    230      total_data_vo_pkt_send:      0
total_data_bk_pkt_send:      0      total_data_vo_pkt_send:      0
total_data_vo_pkt_send:      0
```

Here the DRIVER\_FSM\_STATE value is mentioned in the table below:

DRIVER_FSM_STATE	Value
FSM_FW_NOT_LOADED	0
FSM_CARD_NOT_READY	1
FSM_COMMON_DEV_PARAMS_SENT	2
FSM_BOOT_PARAMS_SENT	3
FSM_EEPROM_READ_MAC_ADDR	4
FSM_EEPROM_READ_RF_TYPE	5
FSM_RESET_MAC_SENT	6
FSM_RADIO_CAPS_SENT	7
FSM_BB_RF_PROG_SENT	8
FSM_MAC_INIT_DONE	9

## 15.4 SDIO stats

User can check the the sdio related info like number of interrupts received or buffer full status using the below command :

```
# cat /sys/kernel/debug/phy<X>/sdio_stats
```

User can expect the sample result below:

```
total_sdio_interrupts: 1855
sdio_msdu_pending_intr_count: 1890
sdio_buff_full_count: 0
sdio_buff_semi_full_count: 0
sdio_unknown_intr_count: 0
BUFFER FULL STATUS : 0
SEMI BUFFER FULL STATUS : 0
MGMT BUFFER FULL STATUS : 0
BUFFER FULL COUNTER : 0
BUFFER SEMI FULL COUNTER : 0
MGMT BUFFER FULL COUNTER : 0
```

## 16 Appendix B: Initial Configuration Parameters

Module parameters available in driver are mentioned below. They can be configured using the below syntax. If the module parameter is not configured, then the respective default value will be reflected.

```
# insmod rsi_91x.ko [module_param = <value>],[module_param = <value>],...[module_param = <value>]
```

Replace 'module\_param' with module parameter to be configured and <value> with the corresponding value that needs to be assigned.

### 16.1 Common Configuration Parameters

Module parameters in this section are grouped according to the corresponding feature.

#### 16.1.1 Power Save Feature

1. **enabled\_uapsd** : This command is used to enable the U-APSD power save mode and set the relevant parameters. If the Access Point does not support U-APSD power save, the module tries to mimic this mode.

0 - Disable U-APSD mode

1 - Enable U-APSD mode

Default value of enabled\_uapsd = 0.

2. **max\_sp\_len** : U-APSD Service Period Length- This field indicates number of packets delivered by AP to station after receiving one trigger frame. This field value ranges between 0-3 as described below.

0-All buffered packets will be delivered. (Default)

1-Two buffered packets will be delivered.

2-Four buffered packets will be delivered.

3-Six buffered packets will be delivered.

Note: This parameter is valid only when enabled\_uapsd is set.

3. **lp\_handshake\_mode**: Low power mode handshake type selection.

0 - No handshake Mode (Default)

1 - GPIO Handshake Mode

#### 16.1.2 BT Related Module Params

1. **bt\_rf\_type**: This variable is used to select the BT RF TYPE which the module has to operate. The following are the possible values:

0 - EXTERNAL RF

1 - INTERNAL RF (Default)

For example, bt\_rf\_type = 1 sets bt rf type to Internal RF.

2. **ble\_tx\_pwr\_inx** :This module param is used to select the BLE\_TX\_PWR index value. Default Value for BLE Tx Power Index is 30. The following are the possible values:

Range for the BLE Tx LP Chain Power Index is 1 - 63 (0, 32 are invalid)

Range for the BLE Tx HP Chain Power Index is 64 - 76

For example, you can set the value as shown here.

BLE\_TX\_PWR\_INX=0x1e

3. **ble\_power\_save\_options**: This variable is used to select the BLE\_PWR\_SAVE\_OPTIONS mode value.

The following are the possible values.

BLE\_DUTY\_CYCLING      BIT(0)



BLR\_DUTY\_CYCLING BIT(1)

BLE\_PWR\_SAVE\_4X\_MODE BIT(2)

Default value is 2 i.e BIT(1)- BLR\_DUTY\_CYCLING is set.

4. **ble\_roles**: The following are the possible values for ble\_roles.

BIT[3:0] - Number of Central Roles Allowed

BIT[7:4] - Number of Peripheral Roles Allowed

Default value is set to 19.

5. **bt\_bdr\_mode**: BDR mode in classic.

BIT(0) - BDR only selection

BIT(1) - BDR in LP chain selection

Default value is bt\_bdr\_mode = 0.

6. **bt\_rf\_tx\_power\_mode**:

Default value is 0.

7. **bt\_rf\_rx\_power\_mode**:

Default value is 0.

8. **bt\_feature\_bitmap**: The bt\_feature map can be assigned a value as explained below.

BIT[0] - For Enabling role switch from host set this bit to 1

BIT[1] - For Enabling sniff from host set this bit to 1

BIT[5] - For Enabling BT Spoof MAC address i.e to use HARDCODE\_MAC\_ADDR in BT this bit should be set to

1

BIT[7:3] - Reserved

Default value is 0.

### 16.1.3 Miscellaneous Features

1. **driver\_mode\_value**: It is used to enable sniffer mode support. Default value is 1.

1 - End to end mode

7 - Sniffer mode

2. **sdio\_clock**: It is used to set sdio clock while installing the driver (only for sdio interface). The range is 1- 50MHZ.

Default value is 50MHZ.

3. **ulp\_gpio\_read**: In GPIO handshake it is used to configure Host GPIO read pin. It will vary form platform to platform.

Default value is 0xFF.

4. **ulp\_gpio\_write**: In GPIO handshake it is use to configure Host GPIO write pin. It will vary form platform to platform.

Default value is 0xFF.

### 16.1.4 Developer Mode Configuration Parameters

1. **power\_save\_opt**: Module parameter to configure Power Save options.

0 - Disable Duty Cycling & Undestined Packet Drop

1 - Enable Duty Cycling

2 - Enable Undestined Packet Drop

3 - Enable Duty Cycling & Undestined Packet Drop (Default)

2. **standby\_assoc\_chain\_sel** : LP/HP Chain Selection in standby associated mode

0 - HP Chain Enabled

1 - LP Chain Enabled(Default)

3. **feature\_bitmap\_9116** : Explanation

For 3.3V keep FEATURE\_BITMAP\_9116=0

For 1.8V keep FEATURE\_BITMAP\_9116=2

For 3.3V, 5Mhz BW keep FEATURE\_BITMAP\_9116=32 (For testing purposes only)

Default value is 0.

#### 4. **lmac\_bcon\_drop** : LMAC BEACON DROP Feature Options

0 - Disable LMAC BEACON DROP Feature

1 - Enable LMAC BEACON DROP Feature

Default value for LMAC BEACON DROP Feature option is 1 - Enable LMAC BEACON DROP Feature.

#### 5. **anchor\_point\_gap** :

Default value is 1.

6. **sleep\_clk\_source\_sel** : Sleep clock source selection has the following possible values with each representing a different source.

0 - Use RC clock as sleep clock

1 - Use 32KHz clock from external XTAL OSCILLATOR

2 - Use 32KHz bypass clock on UULP\_GPIO\_3

3 - Use 32KHz bypass clock on UULP\_GPIO\_4

Default value is 0.

7. **sleep\_ind\_gpio\_sel** : sleep indication from device to host.

0 - UULP\_GPIO\_3

1 - UULP\_GPIO\_0

Default value is 0.

8. **host\_intf\_on\_demand** : Host Interface on Demand Feature has the following possible values.

0 - Disable Host Interface on Demand Feature

1 - Enable Host Interface on Demand Feature

Default value for Host Interface on Demand Feature Options is 0, which indicates that Host Interface on Demand Feature is disabled.

9. **ext\_opt** : Extended Options

0 - NA

Default value for Extended options is 0.

## 17 Appendix C: Hostapd Usage Guidelines

This page lists some of the configurations of hostapd to support user in selecting options.

### 17.1 Common Configuration Parameters

- **Hidden ssid:** To disable ssid broadcast in beacons for AP using hostapd, use following variable in hostapd.conf file.

```
ignore_broadcast_ssid=0
```

- **DTIM Interval:** To set dtim interval in beacons for AP using hostapd, use following variable in hostapd.conf file.

```
dtim_period=5
```

- **SHORT GI:** To enable Short GI using hostapd following params must be enabled in hostapd.conf file.

```
ht_capab=[SHORT-GI-20]
```

- **Beacon Interval:** To set beacon interval for AP using hostapd, use following variable in hostapd.conf file.

```
beacon_int=200
```

**Note:** Range of Beacon interval value is 56 msec to 1000msec.

### 17.2 Hostapd Conf File Changes Required for ACL (Access Control List)

1) Following configuration should be done in hostapd configuration file.

Enable macaddr\_acl according to your choice

```
macaddr_acl = 0/1/2
```

- For macaddr\_acl = 0

```
deny_mac_file = /etc/hostapd.deny
```

- For macaddr\_acl = 1

```
accept_mac_file = /etc/hostapd.accept
```

- For macaddr\_acl = 2

- It will use external RADIUS server.

2) Follow the steps in [Wi-Fi Access Point \(AP\) mode](#).

### 17.3 Enable UAPSD Advertisement in hostapd

Following configuration should be done in hostapd configuration file.

```
uapsd_advertisement_enabled = 1/0
```

- 1 - Enable UAPSD
- 0 - Disable UAPSD

## 17.4 Configure AP Mode Keep Alive Time

Following configuration in hostapd configuration file is useful in configuring AP keep alive time, after which AP sends a Qos null to confirm if client (STA) is still connected.

```
ap_max_inactivity = 300
```

If this is not enabled the default value is taken as 300 seconds or User can configure the time.

## 17.5 Configuration for Country IE

Following configuration in hostapd configuration file needs to be done to get country information element in beacon.

```
country_code=US  
ieee80211d=1
```

country\_code is used to set the regulatory domain. It should be set to indicate the country in which the device is operating. For more country codes refer section [Configuration Using wireless tools](#).

ieee80211d is enabled to advertise the country\_code, if not enabled default 0 value is used.

## 18 Appendix D: Enterprise Security using CFG80211

### 18.1 Installation and Configuration of FREERADIUS Server

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

- libtalloc-devel
- openssl-devel

The steps for downloading as well as installing the freeradius tar ball are as follows:

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

Configure the freeradius server as per the given steps below:

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 to edit “eap” file which contains the paths consisting of certificates and information about the EAP-Methods supported.

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

If Free-radius version is below 3.x “eap”, it 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
```

For openssl versions of range 1.0.2 release - 1.0.2h release (or) in range 1.0.1 - 1.0.1t release (or) in range 1.1.0 - 1.1.0a release

edit radiusd.conf file

```
# vim /usr/local/etc/raddb/radiusd.conf
```

and change 'allow\_vulnerable\_openssl' to yes or CVE-2016-6304

```
allow_vulnerable_openssl =yes  
(or)  
allow_vulnerable_openssl = 'CVE-2016-6304'
```

(here CVE-2016-6304 is openssl vulnerability ID which radius server will allow)

## 18.2 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
```

Compile the driver.

```
# make
```

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

```
# insmod rsi_91x.ko and insmod rsi_usb.ko or insmod rsi_sdio.ko as per the instructions  
mentioned in Section  
# service NetworkManager stop  
# iw phy phyX interface add wifi0 type managed
```

X is the 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 -dddt > log &
```

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

```
network={  
    ssid="SSID of Access Point"  
    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 below:

```
network={
    ssid="SSID of Access Point"
    key_mgmt=WPA-EAP
    eap=TLS
    anonymous_identity="ttlsuser"
    identity="test"
    password="password"
}
```

To connect using EAP-TLS method, **sta\_settings.conf** should be described as below:

```
network={
    ssid="SSID of Access Point"
    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 below:

```
network={
    ssid="SSID of Access Point"
    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 described as below:

```
network={
    ssid="SSID of Access Point"
    key_mgmt=WPA-EAP
    eap=LEAP
    identity="user1"
    password="test123"
}
```

To connect using EAP-LEAP for CCX, **sta\_settings.conf** should be described as below:



```
network={
    ssid="SSID of Access Point"
    key_mgmt=WPA-CKM
    eap=LEAP
    identity="user1"
    password="test123"
    pairwise=TKIP
    group=TKIP
    proto= WPA2 WPA
    scan_ssid=1
    priority=2
}
```

```
# radiusd -X
```

## 18.3 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 which must be installed are as follows:

- libnl-devel
- libsqlite3x-devel
- openssl-devel

### 18.3.1 Configuration of the AP

1. For AP mode connectivity, ensure that the `dev_oper_mode` is set in installation as given below and interface is detected after installation (Refer to above [Installation of Modules](#) section).

```
dev_oper_mode = 1
```

2. Go to scripts folder and Ensure that all basic configurations such as interface, ssid, country\_code, beacon\_int, channel etc. are updated in `hostapd_eap.conf` as mentioned in [Wi-Fi Access Point \(AP\) Mode](#) section.
3. Before starting the AP in EAP 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/certs/hostapd.radius_clients file */
```

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

```
$ hostapd hostapd_eap.conf -ddddd >log &

$ sh dhcp_server.sh wifil , where wifil is the interface name
```

### 18.3.2 Configuring hostapd as RADIUS Server

The steps for configuring hostapd as RADIUS server are as follows:

Copy the certs folder available in scripts folder to /etc location, which will contain the certificates, hostapd.radius\_clients, hostapd.eap\_user and dh files.

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

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

Check whether the interface in hostapd\_server.conf is same or not as the name of AP interface name.

#### Example

```
$ vim hostapd_server.conf
```

interface = wifi1 ,so that RADIUS server will listen on that interface name.

Start the RADIUS server in a new terminal after starting AP.

```
$ hostapd hostapd_server.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 folder in the scripts folder.

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

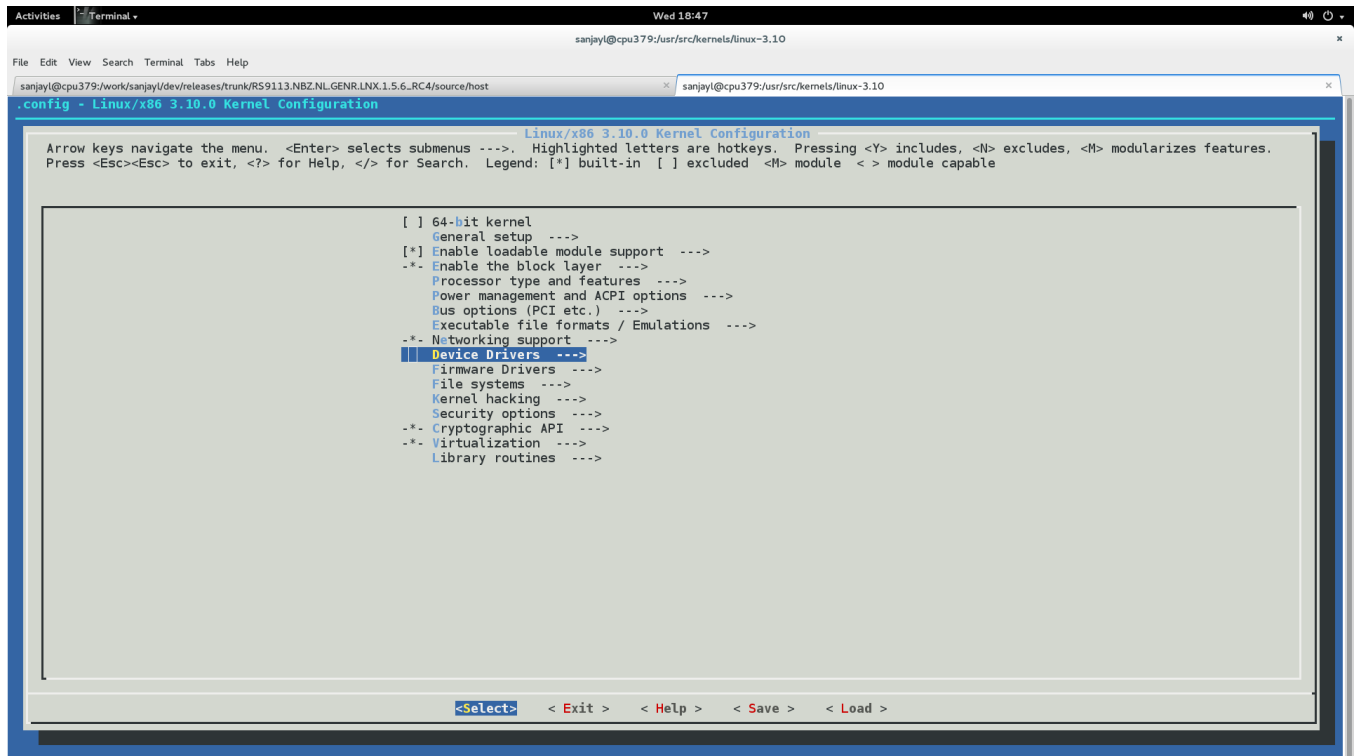
## 19 Appendix E: Kernel Configuration Needed for Driver

To ensure that the driver software works on every platform, some kernel configurations might be needed. These are explained in this section. Super user permissions are needed to make these changes.

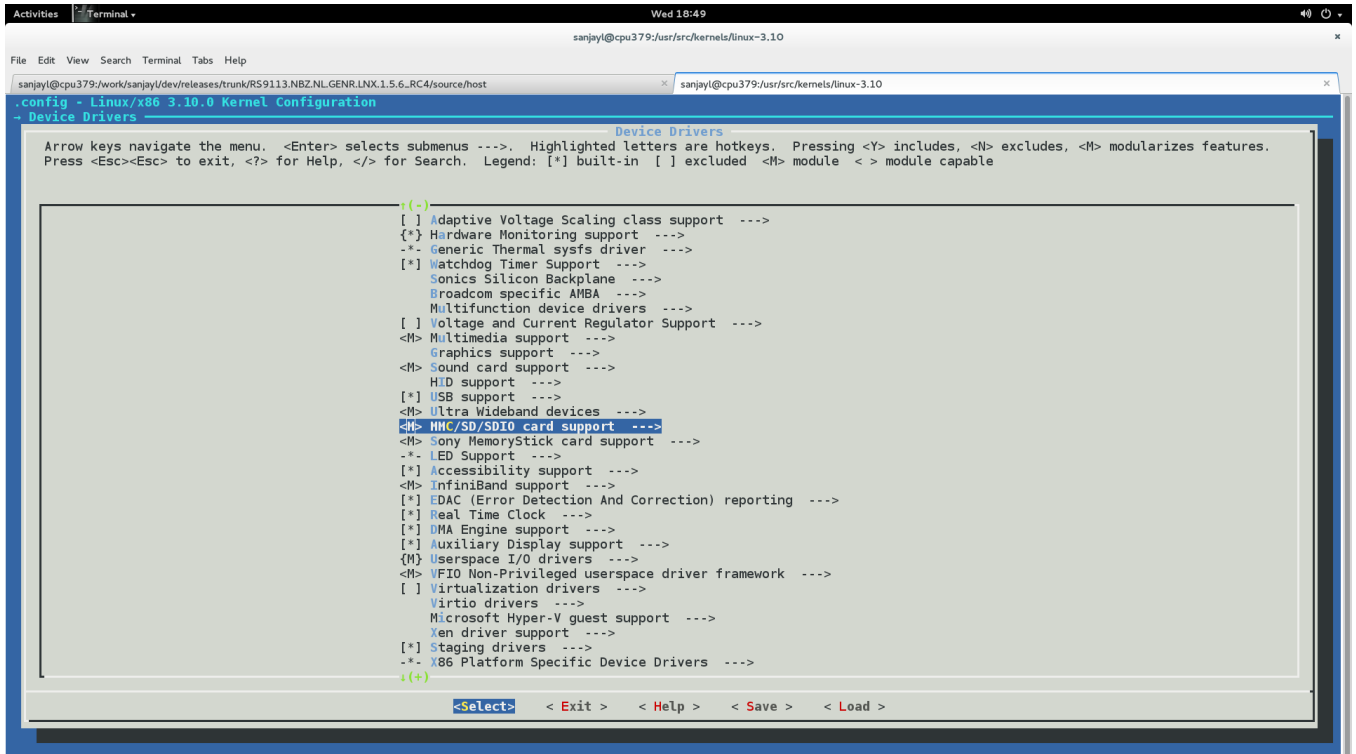
### 19.1 SDIO Stack Options

If SDIO is the interface to the Host processor, it must 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.



```

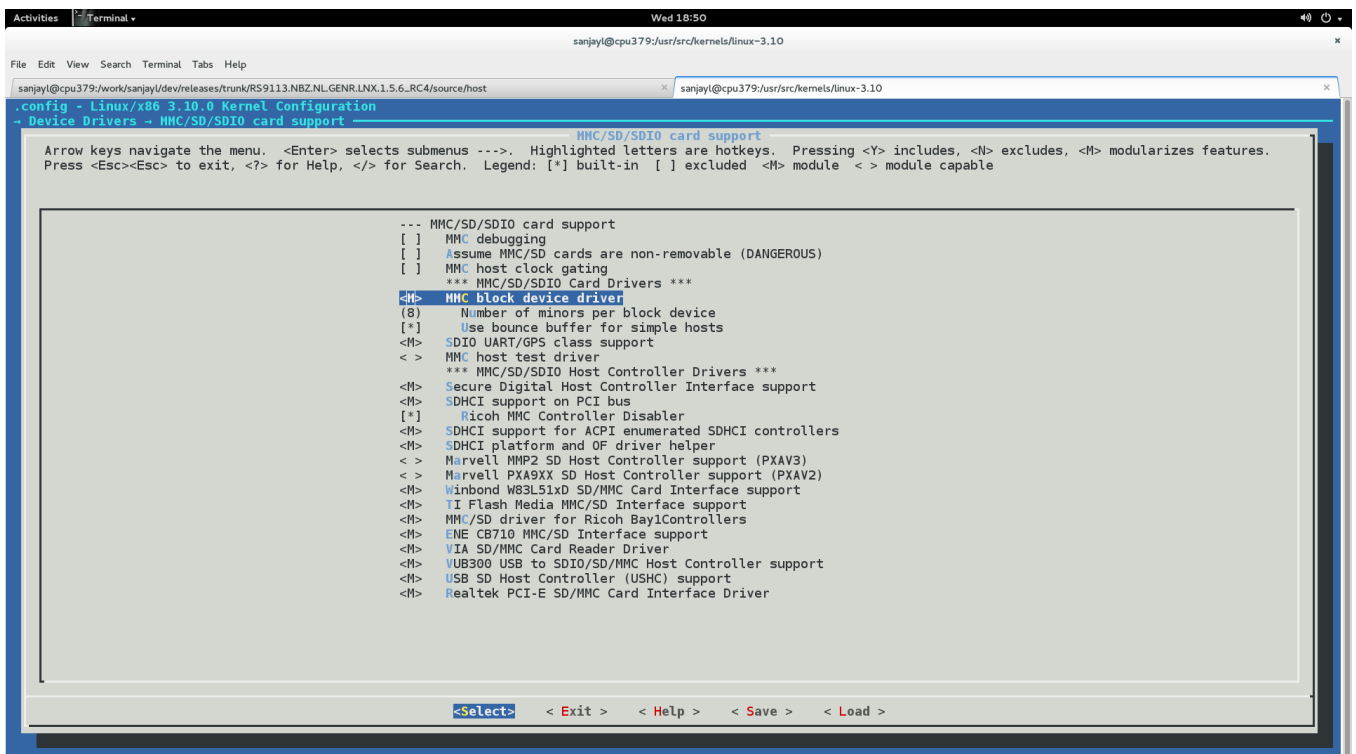
config - Linux/x86 3.10.0 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features.
Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module <> module capable

[*] Adaptive Voltage Scaling class support ---
[*] Hardware Monitoring support ---
[*] Generic Thermal sysfs driver ---
[*] Watchdog Timer Support ---
[*] Sonics Silicon Backplane ---
[*] Broadcom specific AMBA ---
[*] Multifunction device drivers ---
<M> Voltage and Current Regulator Support ---
<M> Multimedia support ---
<M> Graphics support ---
<M> Sound card support ---
[*] HID support ---
[*] USB support ---
<M> Ultra Wideband devices ---
<M> MMC/SD/SDIO card support ---
<M> Sony MemoryStick card support ---
[*] LED Support ---
[*] Accessibility support ---
<M> InfiniBand support ---
[*] I2C (Error Detection And Correction) reporting ---
[*] Real Time Clock ---
[*] DMA Engine support ---
[*] Auxiliary Display support ---
<M> Userspace I/O drivers ---
<M> VFIO Non-Privileged userspace driver framework ---
[*] Virtualization drivers ---
[*] Virtio drivers ---
[*] Microsoft Hyper-V guest support ---
[*] Xen driver support ---
[*] Staging drivers ---
[*] X86 Platform Specific Device Drivers ---

<Select> < Exit > < Help > < Save > < Load >
  
```

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



```

config - Linux/x86 3.10.0 Kernel Configuration
Device Drivers -- MMC/SD/SDIO card support
MMC/SD/SDIO card support
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features.
Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module <> module capable

--- MMC/SD/SDIO card support
[ ] MMC debugging
[ ] Assume MMC/SD cards are non-removable (DANGEROUS)
[ ] MMC host clock gating
*** MMC/SD/SDIO card Drivers ***
<M> MMC block device driver
(8) Number of minors per block device
[*] Use bounce buffer for simple hosts
<M> SDIO UART/GPS class support
< > MMC host test driver
*** MMC/SD/SDIO Host Controller Drivers ***
<M> Secure Digital Host Controller Interface support
<M> SDHCI support on PCI bus
[*] Ricoh MMC Controller Disabler
<M> SDHCI support for ACPI enumerated SDHCI controllers
<M> SDHCI platform and OF driver helper
< > Marvell MMP2 SD Host Controller support (PXA93)
< > Marvell PXA9XX SD Host Controller support (PXA92)
<M> Winbond W83L51xD SD/MMC Card Interface support
<M> TI Flash Media MMC/SD Interface support
<M> MMC/SD driver for Ricoh Bay1Controllers
<M> ENE CB710 MMC/SD Interface support
<M> VIA SD/MMC Card Reader Driver
<M> VUB300 USB to SDIO/SD/MMC Host Controller support
<M> USB SD Host Controller (USHC) support
<M> Realtek PCI-E SD/MMC Card Interface Driver

<Select> < Exit > < Help > < Save > < Load >
  
```

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

### 19.1.1 Wireless Extension Tools

Wireless Extension tools like '**iwconfig**' and '**iwpriv**' are required for configuring the driver. Make sure that the wireless extensions are enabled in the Linux kernel configuration file.

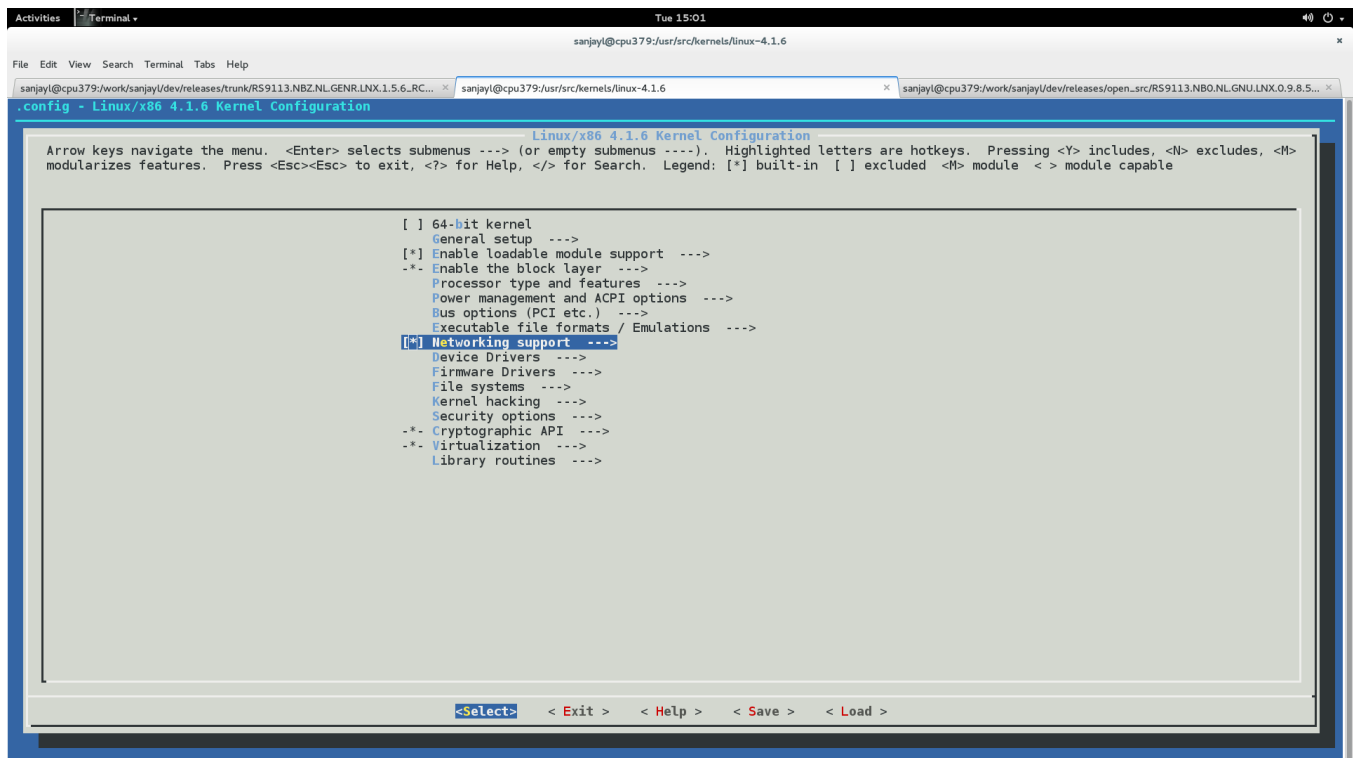
User needs to enable below options in kernel configuration file, re-compile the kernel and cross compile the driver.

```
CONFIG_WIRELESS
CONFIG_WIRELESS_EXT
CONFIG_WEXT_PRIV
CONFIG_WEXT_SPY
CONFIG_WEXT_PROC
CONFIG_WEXT_CORE
CONFIG_HOSTAP
```

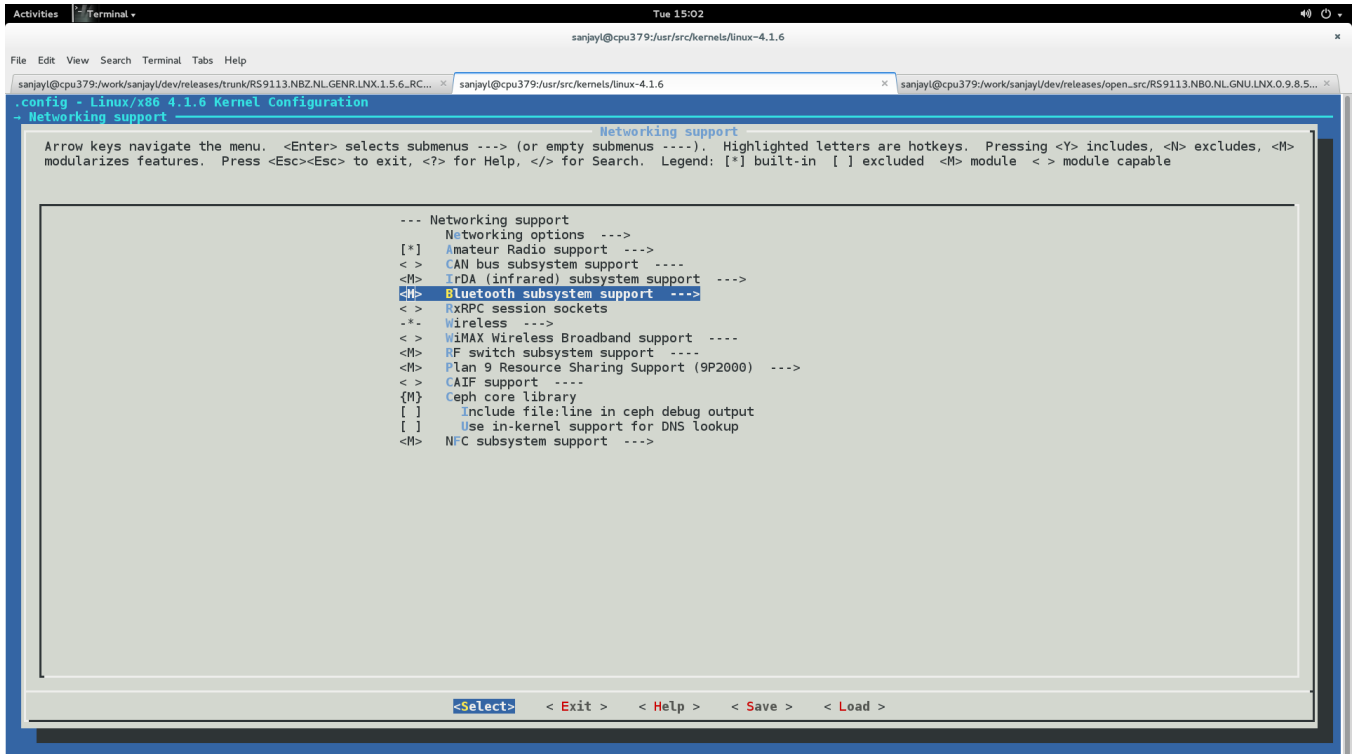
### 19.1.2 Bluetooth Stack Options

If Bluetooth is required, it must 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.



Terminal window showing the Linux Kernel Configuration menu. The menu is titled "Networking support" and lists various options. The "Bluetooth subsystem support" option is highlighted with a blue background.

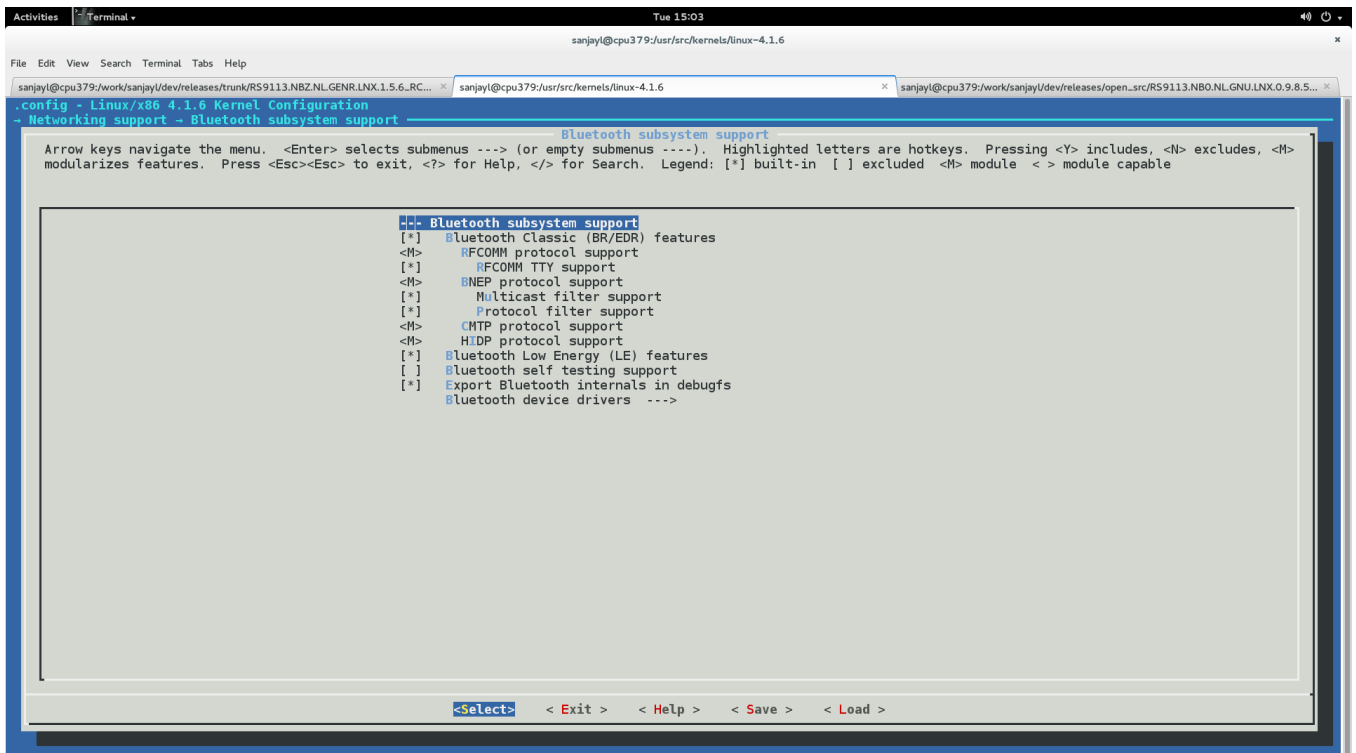
```

--- Networking support
    Networking options --->
[*]  Amateur Radio support --->
< > CAN bus subsystem support ----
<M> IrDA (infrared) subsystem support --->
<M> Bluetooth subsystem support --->
< > L2CAP session sockets
-- Wireless --->
< > WiMAX Wireless Broadband support ----
<M> RF switch subsystem support ----
<M> Plan 9 Resource Sharing Support (9P2000) --->
< > CAIF support ----
{M} Ceph core library
[ ] Include file:line in ceph debug output
[ ] Use in-kernel support for DNS lookup
<M> NFC subsystem support --->
  
```

At the bottom of the menu, there are navigation options: **<Select>**, **<Exit>**, **<Help>**, **<Save>**, and **<Load>**.

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



Terminal window showing the Linux Kernel Configuration menu. The menu is titled "Bluetooth subsystem support" and lists various options. The "Bluetooth subsystem support" option is highlighted with a blue background.

```

Bluetooth subsystem support
[*] Bluetooth Classic (BR/EDR) features
<M> RFCOMM protocol support
[*] RFCOMM TTY support
<M> BNEP protocol support
[*] Multicast filter support
[*] Protocol filter support
<M> CMTP protocol support
<M> HIDP protocol support
[*] Bluetooth Low Energy (LE) features
[ ] Bluetooth self testing support
[*] Export Bluetooth internals in debugfs
Bluetooth device drivers --->
  
```

At the bottom of the menu, there are navigation options: **<Select>**, **<Exit>**, **<Help>**, **<Save>**, and **<Load>**.

6. Hit the Tab key to select Exit and hit Enter. Repeat this till you are asked whether you want to save the configuration.

7. Select "Yes" and hit Enter. If the above options are already selected, the menuconfig screen will exit immediately.

### 19.1.3 Kernel Compilation

The steps used for Kernel Compilation are as follows:

1. Navigate to the kernel source folder.
2. Execute the "**make**" command.
3. Execute the "**make modules**" command.
4. Execute the "**make modules\_install**" command.
5. Execute the "**make install**" command. This ensures that the customized kernel is installed, and the boot loader is updated appropriately.
6. Reboot the system to boot up with the customized kernel.

## 20 Appendix F: Configure and Compile Supplicant for Roaming

Download the supplicant from [https://w1.fi/wpa\\_supplicant/](https://w1.fi/wpa_supplicant/)

Extract the supplicant using the following command.

```
# tar xvf wpa_supplicant-2.6.tar.gz
# cd wpa_supplicant-2.6/wpa_supplicant
# cp defconfig .config
```

Make sure the following parameters are enabled in the supplicant configuration file (.config)

```
CONFIG_DRIVER_NL80211=y
CONFIG_BGSCAN_SIMPLE=y
NL80211_CMD_ROAM=y
CONFIG_LIBNL20=y
CONFIG_LIBNL32=y
CONFIG_WPS2=y
CONFIG_BGSCAN=y
```

Save the configuration file and exit.

Compile the supplicant using "make" command in the following path.

```
# cd wpa_supplicant-2.6/wpa_supplicant
# make clean
# make
```

After successful compilation the supplicant executable will be found in the same path. Copy the supplicant executable to the driver release folder.

```
# cp wpa_supplicant RS9116.NB0.NL.GNU.LNX.OSD.a.b.c.d/rsi.
```

### 20.1 Roaming

Configure Connection quality monitoring (cqm) rssi and hysteresis using iw command.

To know more about iw tool, refer to the section [Configuration Using Wireless Tools](#).

```
# iw dev <devname> cqm rssi <threshold|off> [<hysteresis>]
Set connection quality monitor RSSI threshold.
```

Example:

```
# iw dev wlan0 cqm rssi -45 4
```

To know more about Background Scan and Set Parameters configuration, refer to the section [Background Scan & Roaming](#).



## 21 Appendix G: Installation of Missing Generic Netlink Libraries

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].

If libnl libraries are not installed in the platform, then follow any of the two methods given below.

1. Required libnl libraries can be installed through yum/dnf/apt-get/ using below command.

```
# yum install libnl*
```

2. Alternately, if you are unable to install libnl libraries using above method, follow below process where we download libnl source package and compile.

- a. Create a directory in the location where Tool chain and BSP are present.

```
# mkdir build
```

- b. 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
```

- c. Configure the libnl library for target platform.

```
# CC=/path to the toolchain/bin/arm-linux-gnueabi-gcc
# ./configure --host=arm-linux-gnueabi --prefix=/<complete path to build directory>/
```

Here headers will be installed in \${prefix}/include/libnl3.

- d. 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.

Follow the example given below:

```
# make DESTDIR=$(arm-cortex_a8-linux-gnueabi-gcc -print -/<path to build
directory>/build/)
# make install
```

- e. Exporting the path for build directory in the command line or add these flags in the supplicant and hostpad config files under CONFIG\_DRIVER\_NL80211= y variable.

```
# export LDFLAGS='-L/<path to build directory>/lib/libnl'
```

OR

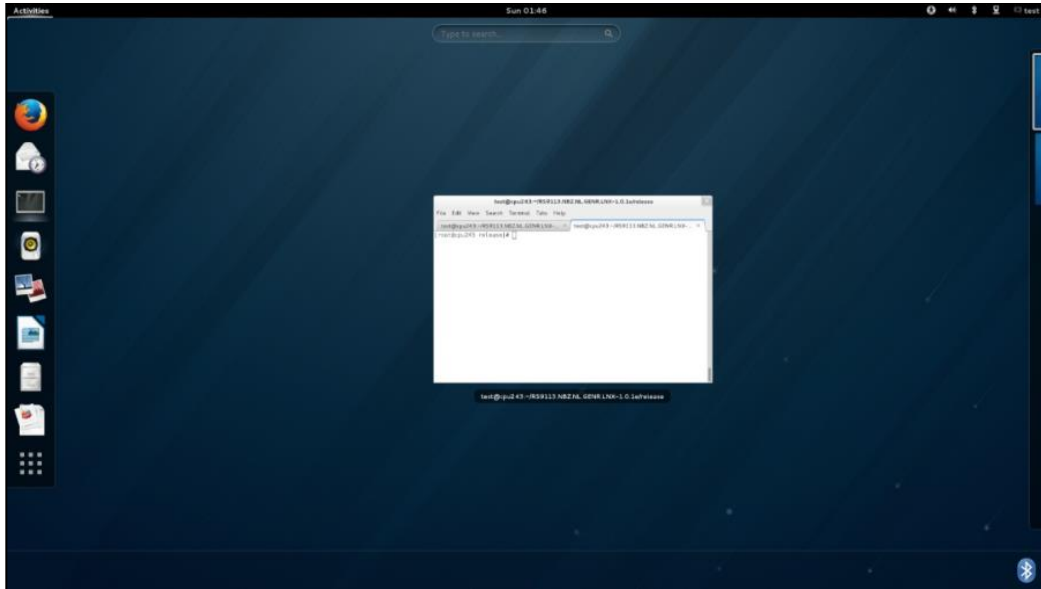
- f. Add these flags in the supplicant and hostpad .config files under CONFIG\_DRIVER\_NL80211= y variable.

```
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.
```

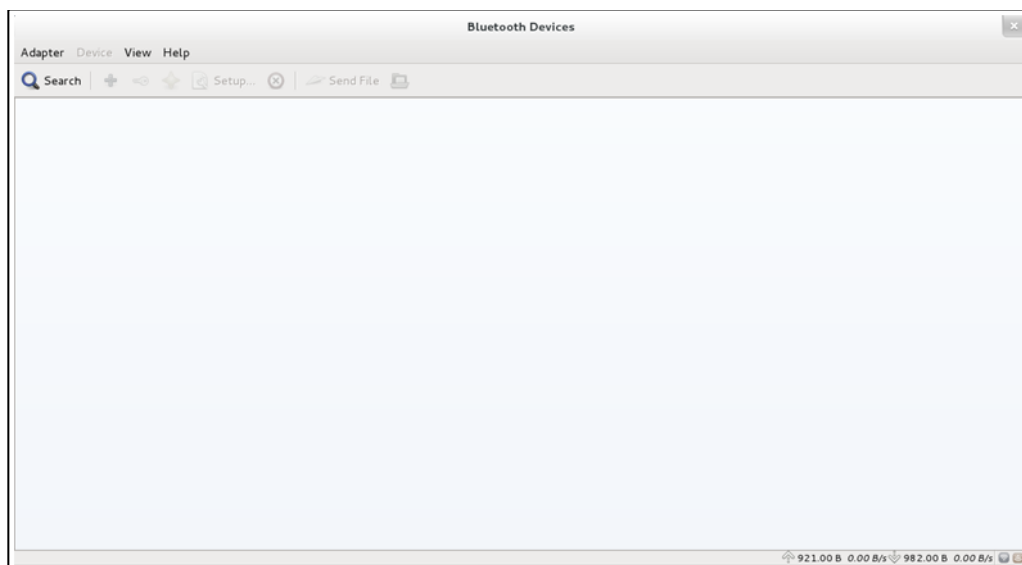
## 22 Appendix H: Using the Bluetooth Manager

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

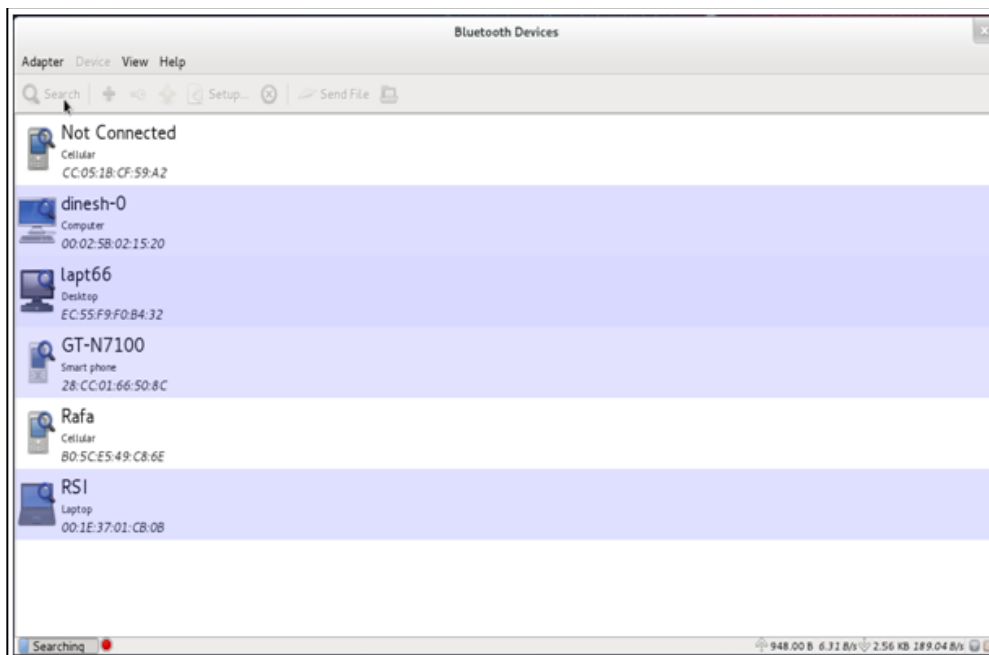
1. Once the Bluetooth modules have been installed as mentioned in section [Installation in Bluetooth only modes \(BT/BLE\)](#), hit the "Windows" button on the keyboard. You will see Bluetooth symbol at the bottom-right corner of the screen, as shown in the given below figure.



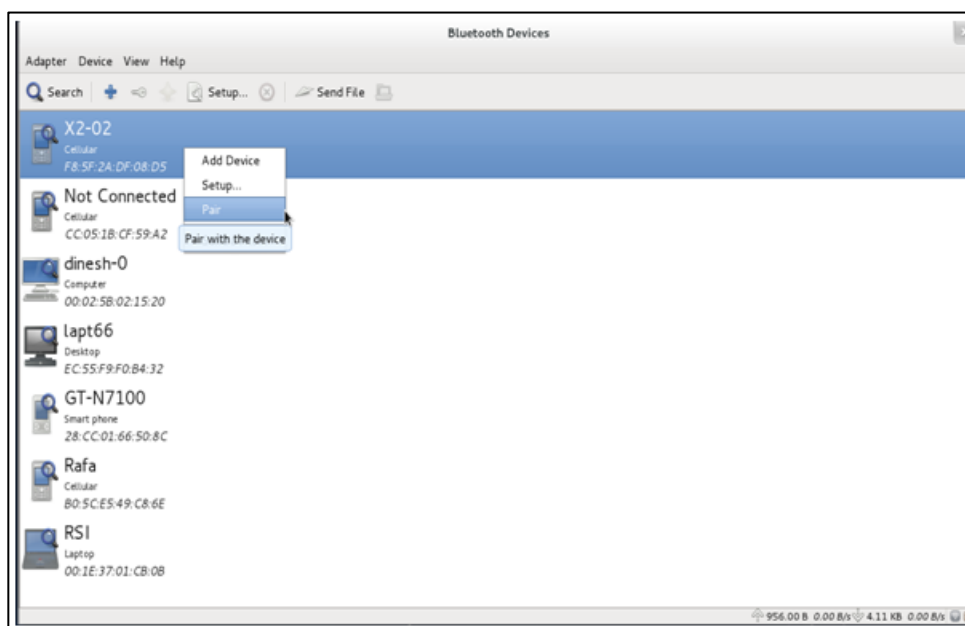
2. This will open the Bluetooth Manager as shown in the figure below:



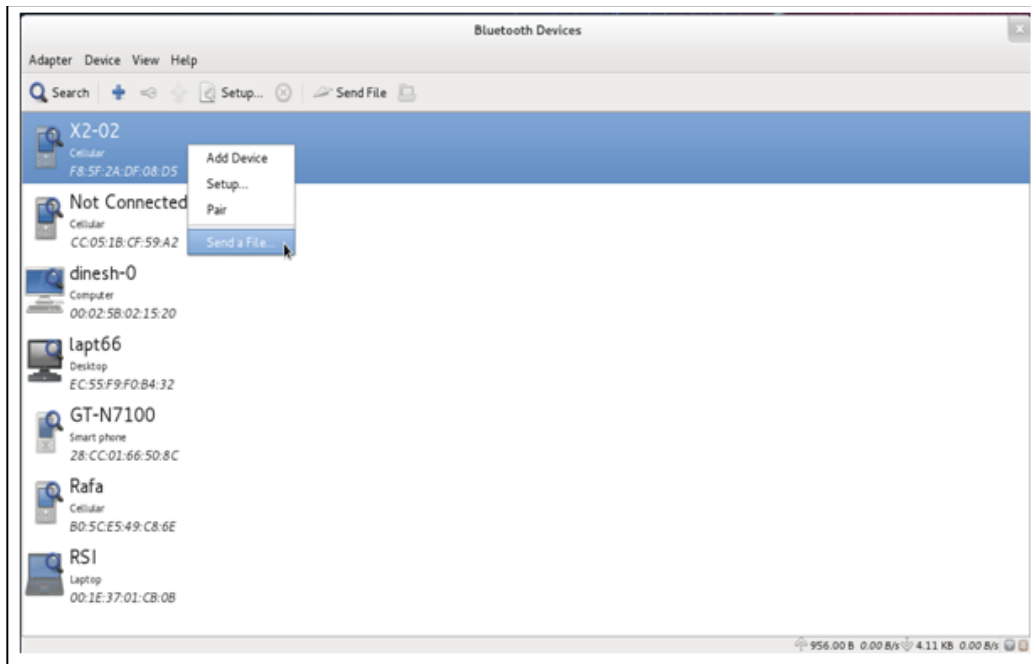
3. Click on **Search** to start inquiry.



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



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

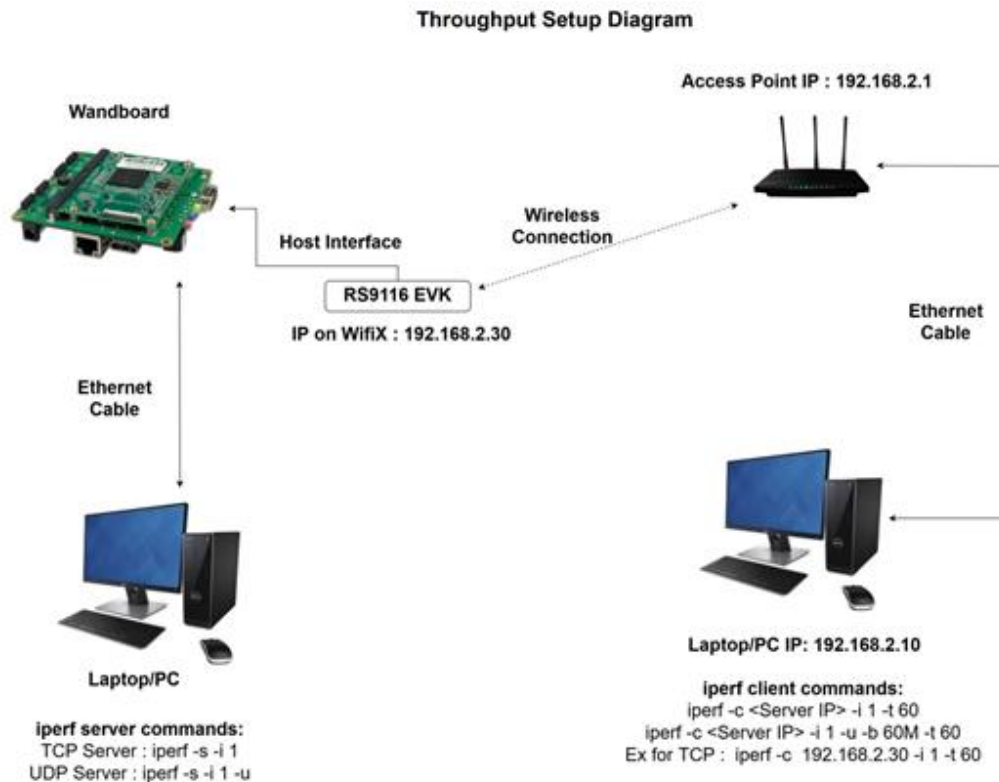


## 23 Appendix I: Checking Throughput

You can measure Wi-Fi performance through UDP/TCP protocols by using the iperf application. If you wish to evaluate the throughputs in Wi-Fi Client mode, you will need to connect a second PC/Laptop to the Access Point. Download and install the iperf application from <https://iperf.fr> on the second PC/Laptop.

Please note the following points for this evaluation:

1. To evaluate the module for throughput performance, it's recommended that you connect the second PC/Laptop to the Access Point over Ethernet.



**Figure 1: Throughput Measurement Setup Diagram**

2. It is recommended that you choose a Wi-Fi channel which has less interference, preferably in the 5GHz band if the EVB/module is a Dual band module, to observe optimal throughput of the module.
3. Wireless throughput vary with the environment of the test setup – distance, obstacles, type of obstacles, etc.
4. The throughput is also dependent on whether all the Wi-Fi components in the test support only 20MHz bandwidth.
5. To check Transmit throughput, first run the command below on the 2<sup>nd</sup> PC/Laptop:

```
# iperf -s -u -i 1
```

This command starts an iperf Server which is listening for data. It prints the received throughput at an interval of 1 second. The <-u>option in the above command is for UDP traffic. If it is not mentioned, then the traffic is TCP.

6. Next, run the command below on the PC/Laptop connected to the EVB/module.

```
# iperf -c <IP address of 2nd PC> -u -i 1 -b 100M -t 120
```

This command starts an iperf Client which starts transmitting data to the Server. The <-u> and <-b 100m> options in the above command are for UDP traffic. If they are not mentioned, then the traffic is TCP.

7. To check Receive throughput, run the Server command first on the PC/Laptop connected to the EVB/module, followed by the client command on the 2<sup>nd</sup> PC/Laptop.

Example of running throughput:

TCP Server:

```
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 192.168.0.14 port 5001 connected with 192.168.0.2 port 51568
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0- 1.0 sec  5.19 MBytes 43.5 Mbits/sec
[ 4] 1.0- 2.0 sec  5.63 MBytes 47.2 Mbits/sec
[ 4] 2.0- 3.0 sec  5.77 MBytes 48.4 Mbits/sec
[ 4] 3.0- 4.0 sec  5.94 MBytes 49.8 Mbits/sec
[ 4] 4.0- 5.0 sec  5.92 MBytes 49.6 Mbits/sec
[ 4] 5.0- 6.0 sec  5.88 MBytes 49.3 Mbits/sec
[ 4] 6.0- 7.0 sec  5.93 MBytes 49.7 Mbits/sec
[ 4] 7.0- 8.0 sec  5.86 MBytes 49.1 Mbits/sec
[ 4] 8.0- 9.0 sec  5.92 MBytes 49.6 Mbits/sec
[ 4] 9.0-10.0 sec  5.83 MBytes 48.9 Mbits/sec
[ 4] 0.0-10.2 sec  59.2 MBytes 48.6 Mbits/sec
```

TCP Client:

```
-----
Client connecting to 192.168.0.2, TCP port 5001
TCP window size: 43.8 KByte (default)
-----
[ 3] local 192.168.0.14 port 45964 connected with 192.168.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 1.0 sec  5.25 MBytes 44.0 Mbits/sec
[ 3] 1.0- 2.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 2.0- 3.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 3.0- 4.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 4.0- 5.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 5.0- 6.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 6.0- 7.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 7.0- 8.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 8.0- 9.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 9.0-10.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 0.0-10.0 sec  50.9 MBytes 42.5 Mbits/sec
```

UDP Server:

```
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 3] local 192.168.0.14 port 5001 connected with 192.168.0.2 port 36565
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagrams
[ 3] 0.0- 1.0 sec  6.80 MBytes 57.1 Mbits/sec 0.043 ms    0/ 4852 (0%)
[ 3] 1.0- 2.0 sec  6.48 MBytes 54.4 Mbits/sec 0.050 ms    0/ 4622 (0%)
[ 3] 2.0- 3.0 sec  6.60 MBytes 55.3 Mbits/sec 0.083 ms    0/ 4706 (0%)
[ 3] 3.0- 4.0 sec  6.62 MBytes 55.6 Mbits/sec 0.049 ms    0/ 4724 (0%)
[ 3] 4.0- 5.0 sec  6.87 MBytes 57.6 Mbits/sec 0.043 ms    0/ 4900 (0%)
[ 3] 5.0- 6.0 sec  6.50 MBytes 54.6 Mbits/sec 0.043 ms    0/ 4640 (0%)
[ 3] 6.0- 7.0 sec  6.86 MBytes 57.6 Mbits/sec 0.066 ms    0/ 4896 (0%)
[ 3] 7.0- 8.0 sec  6.62 MBytes 55.5 Mbits/sec 0.060 ms   108/ 4831 (2.2%)
[ 3] 8.0- 9.0 sec  6.83 MBytes 57.3 Mbits/sec 0.085 ms   223/ 5095 (4.4%)
[ 3] 9.0-10.0 sec  6.84 MBytes 57.4 Mbits/sec 0.053 ms   233/ 5113 (4.6%)
[ 3] 0.0-10.5 sec  70.6 MBytes 56.3 Mbits/sec 0.063 ms  672/51020 (1.3%)
```

UDP Client:

```

-----
Client connecting to 192.168.0.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 3] local 192.168.0.126 port 50346 connected with 192.168.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 1.0 sec    6.32 MBytes 53.0 Mbits/sec
[ 3] 1.0- 2.0 sec    6.23 MBytes 52.2 Mbits/sec
[ 3] 2.0- 3.0 sec    6.28 MBytes 52.7 Mbits/sec
[ 3] 3.0- 4.0 sec    6.34 MBytes 53.2 Mbits/sec
[ 3] 4.0- 5.0 sec    6.30 MBytes 52.9 Mbits/sec
[ 3] 5.0- 6.0 sec    6.37 MBytes 53.4 Mbits/sec
[ 3] 6.0- 7.0 sec    6.01 MBytes 50.4 Mbits/sec
[ 3] 7.0- 8.0 sec    6.00 MBytes 50.3 Mbits/sec
[ 3] 8.0- 9.0 sec    5.94 MBytes 49.9 Mbits/sec
[ 3] 9.0-10.0 sec    6.26 MBytes 52.5 Mbits/sec
[ 3] 0.0-10.0 sec    62.1 MBytes 52.1 Mbits/sec
[ 3] Sent 44263 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec    62.1 MBytes 51.8 Mbits/sec  0.186 ms  0/44262 (0%)
[ 3] 0.0-10.0 sec    1 datagrams received out-of-order

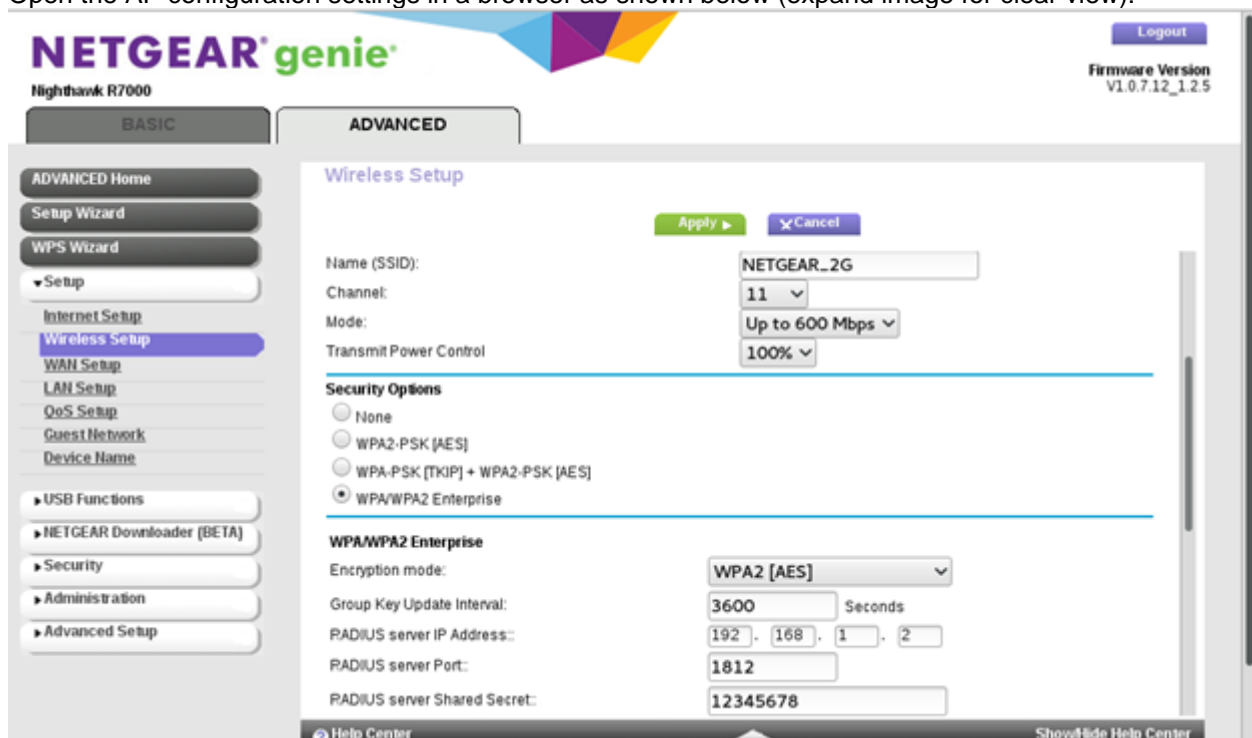
```



## 24 Appendix H: Steps to Connect EAP-TLS Using wpa\_supplicant v 2.6 and Above

### 24.1 Installing FreeRADIUS Server

1. Download FreeRADIUS Server from the link below:  
Link: <https://freeradius.org/>  
**Note:** Try to download the latest release.
2. After downloading, the file name looks like this  
freeradius-server-release\_x\_x\_x.tar.gz
3. Untar the above file  
tar -xvf freeradius-server-release\_x\_x\_x.tar.gz
4. Enter the directory after untar and give following command:  
cd freeradius-server-release\_x\_x\_x/  
./configure
5. It will check the necessary prerequisites for the installation. You may get errors like libtalloc not found. Install them with:  
dnf install libtalloc\*
6. After a successful execution of configure with no errors, give the make command; make install
7. Connect the radius server installed machine (laptop or PC installed with radius server) to the target AP with a LAN cable.
8. Get the IP address from the AP by giving:  
dhclient <interface name> -r  
dhclient <interface name> -v
9. Open the AP configuration settings in a browser as shown below (expand image for clear view):



Refer to the image above to configure the AP settings.

**Note:** Block contains 'radius server shared secret', Enter the shared Secret. Which is used to configure secret in client.conf as explained below.

Enter IP Address which is obtained by dhclient

Radius server Port number is 1812

10. After a successful configuration, go to directory: cd /usr/local/etc/raddb
11. Configure two files:  
**clients.conf** [Here, client in the sense, AP (Wi-Fi Router), which require Radius server].  
**users** [Here, users in the sense, clients (mobile phones or station cards), which are going to connect to the above mentioned AP].
12. Open file **/usr/local/etc/raddb/clients.conf**, Configure network block as:  
client private-network-1 {



```
ipaddr = <IP of AP>/24    // example : 192.168.1.1/24
secret = <shared secret> // shared secret configured in AP
}
```

Save and close the file.

Example:

```
# All IPv6 Site-local clients
#client sitelocal_ipv6 {
#   ipv6addr    = fe80::/16
#   secret      = testing123
#}

#client example.org {
#   ipaddr      = radius.example.org
#   secret      = testing123
#}

# You can now specify one secret for a network of clients.
# When a client request comes in, the BEST match is chosen.
# i.e. The entry from the smallest possible network.
#
client private-network-1 {
    ipaddr    = 192.168.1.1/24
    secret     = 12345678
}

#client private-network-2 {
#   ipaddr      = 198.51.100.0/24
#   secret      = testing123-2
#}

#####
#
# Per-socket client lists. The configuration entries are exactly
# the same as above, but they are nested inside of a section.
#
# You can have as many per-socket client lists as you have "listen"
# sections, or you can re-use a list among multiple "listen" sections.
#
# Un-comment this section, and edit a "listen" section to add:
# "clients = per_socket_clients". That IP address/port combination
# will then accept ONLY the clients listed in this section.
#
#clients per_socket_clients {
#   client socket_client {
#       ipaddr = 192.0.2.4
```

13. Open the **/usr/local/etc/raddb/users** file and configure identity in the file as given below

<identity> Cleartext-password :=<password for client>

Example : user1 Cleartext-password := "12345678"

```
# get any attributes in addition to the ones listed here.
#
#steve  Cleartext-Password := "testing"
#       Service-Type = Framed-User,
#       Framed-Protocol = PPP,
#       Framed-IP-Address = 172.16.3.33,
#       Framed-IP-Netmask = 255.255.255.0,
#       Framed-Routing = Broadcast-Listen,
#       Framed-Filter-Id = "std.ppp",
#       Framed-MTU = 1500,
#       Framed-Compression = Van-Jacobson-TCP-IP
#
# The canonical testing user which is in most of the
# examples.
#
user1  Cleartext-Password := "12345678"
#       Reply-Message := "Hello, %{User-Name}"
#
#
# This is an entry for a user with a space in their name.
# Note the double quotes surrounding the name. If you have
# users with spaces in their names, you must also change
# the "filter_username" policy to allow spaces.
#
# See raddb/policy.d/filter, filter_username {} section.
#
#"John Doe"  Cleartext-Password := "hello"
#             Reply-Message = "Hello, %{User-Name}"
#
# Dial user back and telnet to the default host for that port
```

**Note:** Above **'user1'** is the identity, **12345678** is the password to be entered in the station settings file or supplicant configuration file.

Save and close the file.

14. Generate certificates using the process given in the section below and copy certs to /etc folder. Configure the network block for EAP-TLS in Supplicant configuration file as given below . Here we need to configure the SSID of AP, identity,password,path for ca\_cert, client\_cert,private\_key, and private\_key\_passwd.These certificates can be generated using the process given below.

```
#Enable this block for EAP-TLSnetwork={
    ssid="<SSID>"
    key_mgmt=WPA-EAP
    eap=TLS
    anonymous_identity="tlsuser"
    identity="<user>"
    password="<password>"
    ca_cert="/etc/certs/ca.pem"
    client_cert="/etc/certs/client.pem"
    private_key_passwd="whatever"
    private_key="/etc/certs/client.key"
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

Example:

```
#Enable this block for EAP-TLS
network={
    ssid="NETGEAR_2G"
    key_mgmt=WPA-EAP
    eap=TLS
    anonymous_identity="tlsuser"
    identity="user1"
    password="12345678"
    ca_cert="/etc/certs/ca.pem"
    client_cert="/etc/certs/client.pem"
    private_key_passwd="whatever"
    private_key="/etc/certs/client.key"
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

15. Configure the certificate in the server by editing the following entries in the **/usr/local/etc/raddb/mods-enabled/eap** file.

```
tls-config tls-common {
    private_key_password = whatever
```

```
private_key_file = /etc/certs/server.key
certificate_file = /etc/certs/server.pem
ca_file = /etc/certs/ca.pem
```

Example:

```
# authenticate via EAP-TLS! This is likely not what you want.
#
tls-config tls-common {
    private_key_password = whatever
    private_key_file = /etc/certs/server.key

    # If Private key & Certificate are located in
    # the same file, then private_key_file &
    # certificate_file must contain the same file
    # name.
    #
    # If ca file (below) is not used, then the
    # certificate_file below SHOULD also include all of
    # the intermediate CA certificates used to sign the
    # server certificate, but NOT the root CA.
    #
    # Including the ROOT CA certificate is not useful and
    # merely inflates the exchanged data volume during
    # the TLS negotiation.
    #
    # When using "ca_file" or "ca_dir", the
    # "certificate_file" should contain only
    # "server.pem". And then you may (or may not) need
    # to set "auto_chain", depending on your version of
    # OpenSSL.
    #
    # In short, SSL / TLS certificates are complex.
    # There are many versions of software, each of which
    # behave slightly differently. It is impossible to
    # give advice which will work everywhere. Instead,
    # we give general guidelines.
    #
    certificate_file = /etc/certs/server.pem

    # Trusted Root CA list
    #
    # This file can contain multiple CA certificates.
    # ALL of the CA's in this list will be trusted to
    # issue client certificates for authentication.
    #
    # In general, you should use self-signed
    # certificates for 802.1x (EAP) authentication.
    # In that case, this CA file should contain
    # *one* CA certificate.
    #
    ca_file = /etc/certs/ca.pem

    # OpenSSL will automatically create certificate chains,
```

16. Start the Radius server using the command below.

**radiusd -X**

17. Connect the STA by running supplicant. STA should get connected to the AP if all the configurations are proper.

## 24.2 Generate Certificates

1. Go to the **/usr/local/etc/raddb/certs** folder
2. Remove all the previous certs using the command below:  
\$ rm -f \*.pem \*.der \*.csr \*.crt \*.key \*.p12 serial\* index.txt\*
3. **Generate the CA certificate**

\$ vi ca.cnf

The user can edit "input\_password" and "output\_password" fields to be the password for the CA certificate. default\_days and default\_crt\_days are the limits for validity of the certificates. The user can modify as per the requirement.

\$ make ca.pem

This step creates the CA certificate.

\$ make ca.der

This step creates the DER format of the self-signed certificate, which can be imported into Windows.

4. **Generate the SERVER CERTIFICATE**

The following steps will let you create a server certificate for use with TLS-based EAP methods, such as EAP-TLS, PEAP, and TTLS.

\$ vi server.cnf

User can edit the "input\_password" and "output\_password" fields to be the password for the server certificate.

default\_days and default\_crt\_days are the limits for validity of the certificates. The user can modify as per the requirement.

```
$ make server.pem
```

This step creates the server certificate.

```
$ make server.csr
```

Ensure that the certificate contains the XP extensions needed by Microsoft clients.

#### 5. **Generate the CLIENT CERTIFICATE**

Client certificates are used by EAP-TLS, and optionally by EAP TTLS and PEAP. The following steps outline how to create a client certificate that is signed by the server certificate created above. You will have to have the password for the server certificate in the "input\_password" and "output\_password" fields of the server.cnf file.

```
$ vi client.cnf
```

The user can edit the "input\_password" and "output\_password" fields to be the password for the client certificate. You will have to give these passwords to the end user who will be using the certificates.

default\_days and default\_crt\_days are the limits for validity of the certificates. The user can modify as per the requirement.

```
$ make client.pem
```

This step creates the client certificate.

#### 6. **Note:**

i) User can configure input\_password and output\_password in all three cnf files.

ii) default\_days and default\_crt\_days are the limits for validity of the certificates. The user can modify as per the requirement

#### 7. After creating all the certs, copy them to the /etc folder:

```
$ cp certs/ /etc -rf
```

## 25 RS9116 n-Link OSD Software TRM Revision History

Revision No.	Version No.	Date	Author	Changes
1	0.1	15th Feb	Fariya	Initial version
2	0.2	18th Mar	Prameela Rani	<ol style="list-style-type: none"> <li>1. Modified chapter 1 -Modified the content as well as added section 1.1 and 1.2.</li> <li>2. Modified chapter 2-Package</li> <li>3. Modified the whole chapter 3-Compilation &amp; Installation / Un-installation instructions</li> <li>4. Modified the content of chapter 4-Device Configuration commands as well as content of section 4.1 and 4.2</li> <li>5. Removed chapter 8</li> </ol>
			Venkanna	Added section 4.7-Wake on WLAN (WOWLAN) – BT Connectivity under chapter 4 3
3	0.5	3rd Oct, 2017	Prameela Rani	ZiGB basic details (build, install) added
4	0.6	5th Oct, 2017	Prameela Rani	ZiGB usage details added
5	0.7	26th Oct, 2017	Prameela Rani	<ol style="list-style-type: none"> <li>1. WPS, P2P sections added</li> <li>2. Regulatory mapping table added for Caracalla.</li> </ol>
6	0.8	10th May 2018	Siva Rebbagondla	Removed rsi_zigb.ko insertion and deletion sections, as zigb modules added to rsi_91x
7	1.0	25th Oct 2018	Siva Rebbagondla	Added sdio_clock module_param and RS9116_flash mode details
8	1.1	13th Mar 2019	Ganapathi Raju	Added Antenna Diversity in RSI_STA mode.
9	1.3	24th Dec 2019	Amol Hanwate	<ol style="list-style-type: none"> <li>1. Added Sniffer mode</li> <li>2. Added ACS with Hostapd</li> <li>3. Added dev_oper_mode 12 details</li> <li>4. Removed Concurrent mode</li> <li>5. Added Power save section</li> </ol>
10	1.4	3rd April 2020	Amol Hanwate	<ol style="list-style-type: none"> <li>1. Added how to configure 11W</li> <li>2. Added BT sniff mode command</li> </ol>
11	1.5	13th Aug 2020	Ganapathi Raju Amol Hanwate	<ol style="list-style-type: none"> <li>1. Updated power save configuration and added GPIO handshake configurations.</li> <li>2. Updated BG scan configuration.</li> </ol>
12	1.6	Sep 2020	Pooja Labde Vandhana Gunti	<ol style="list-style-type: none"> <li>1. Added changes as per OSD 2.0 release supported features.</li> <li>2. Updated document format and structure</li> </ol>
13	1.7	Oct 2020	Pooja Labde Vandhana Gunti	<ol style="list-style-type: none"> <li>1. Added <a href="#">Appendix I: Checking Throughput</a> section.</li> <li>2. Reviewed and updated compilation and installation sections.</li> </ol>
14	1.8	May 2022	Shivandam Gude	<ol style="list-style-type: none"> <li>1. Added <a href="#">Appendix H: Steps to connect EAP-TLS using wpa_supplicant v2.6 and above</a> section.</li> <li>2. Added antenna_sel module param in <a href="#">Antenna Selection</a> section.</li> </ol>

Revision No.	Version No.	Date	Author	Changes
				<ol style="list-style-type: none"><li>Added Beacon Interval details in <a href="#">Appendix C: Hostapd Usage Guidelines</a> section.</li><li>Added Selective scan details in <a href="#">Installation in Wi-Fi Only Mode</a> section.</li><li>Added <a href="#">Update WLAN Region-Based Maximum Powers from Driver</a> section.</li><li>Added Fast PSP power save profile in <a href="#">Device Sleep Mode</a> section.</li><li>Added <a href="#">Configuration of AP and RADIUS Server to Use EAP Methods</a> section.</li></ol>



# Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



**IoT Portfolio**  
[www.silabs.com/iot](http://www.silabs.com/iot)



**SW/HW**  
[www.silabs.com/simplicity](http://www.silabs.com/simplicity)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support & Community**  
[www.silabs.com/community](http://www.silabs.com/community)

## Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

**Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit [www.silabs.com/about-us/inclusive-lexicon-project](http://www.silabs.com/about-us/inclusive-lexicon-project)**

## Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals®, WiSeConnect, n-Link, ThreadArch®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, Precision32®, Simplicity Studio®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.  
400 West Cesar Chavez  
Austin, TX 78701  
USA

[www.silabs.com](http://www.silabs.com)