UM11799

NXP Wi-Fi and Bluetooth Demo Applications for RW61x

Rev. 10.0 — 24 March 2025 User manual

Document information

Information	Content
Keywords	Wireless MCU RW610/RW612, RW61x EVK board, MCUXpresso SDK, RTOS image
Abstract	Provides step-by-step guidance to configure, compile, debug, flash and run the Wi-Fi and Bluetooth sample applications available in the MCUXpresso SDK. It also covers IDE configurations and the required tool set-up.

1 About this document

1.1 Purpose and scope

This document provides the steps to configure, compile, debug, flash, and run the Wi-Fi and Bluetooth sample applications available in the MCUXpresso SDK. It also covers IDE configurations and required tool setup.

1.2 Considerations

The RW61x is powered by FreeRTOS and features integrated Wi-Fi 6, Bluetooth Low Energy, and 802.15.4 radios. This document does not include wireless information, RW61x product information, hardware interconnection, board settings, bring-up, IDE setup, nor SDK download. These items are covered in [3]. The user must have RW61x platform-related IDE and tools installed before going through the given demo process.

2 Tool setup

2.1 Serial console tool setup

The serial console tool is used to read out the demo application logs on the computer connected to RW61x EVK board.

- Download and install the terminal emulator software such as Minicom (Linux or Mac OS) or Tera Term (Windows)
- Use a micro USB-to-USB cable to connect RW61x EVK board to the host computer running on Linux, Mac OS, or Windows.
- Open a terminal emulator program like Minicom or Tera term.
- For Minicom use following command and configure the below settings for serial console access:

```
# Minicom -s
Serial Port Setup:
- /dev/ttyACMX serial port
- 115200 baud rate - 8 data bits  - No parity
- One stop bit
- No flow control
```

Before running the Bluetooth demo application, update the serial console configuration so there is no extra spacing.

For Tera Term:

- Go to Setup > Terminal
- Look for the New line section
- Set the **Receive** to **Auto**

For Minicom:

- To open the Help menu, press **Ctrl + A**, and then press **Z**
- To add a carriage return, press the **U** key

2.2 Wireshark tool setup

The Wireshark tool is required to analyze the Wi-Fi sniffer logs. Download and install Wireshark tool for Windows and Mac OS [12].

Steps to install Wireshark tool on a computer running Linux Ubuntu:

```
sudo add-apt-repository ppa:wireshark-dev/stable
sudo apt update
sudo apt install wireshark
```

2.3 IPerf remote host setup

Remote host setup for OS-Windows:

To complete the setup:

- Download IPerf version 2.1.9 [9].

To run the iPerf:

- Use the command prompt and type the path where IPerf is downloaded

```
> cd C:\Users\XXXX\Downloads
```

- Run the appropriate command from Table 1.

Table 1. iPerf commands for Windows Remote Host

Functionality	Command
TCP server	iperf.exe -s -i 1
UDP server	iperf.exe -s -u -i 1
TCP client	iperf.exe -c <server_ip> -i 1 -t 60
UDP client	iperf.exe -c <server_ip> -u -i 1 -t 60

Note: *The default TCP/UDP port used for the server to listen on, or for the client to connect to, is 5001. The port can also be configured through the "-p" option and should be the same for both client and server.*

Remote host setup for OS-Linux

To complete the setup:

- Download Debian package of iPerf 2.1.9 for Ubuntu 16.04 [9]

```
$ sudo wget https://iperf.fr/download/ubuntu/iperf_2.1.9+dfsg1\-2_amd64.deb
```

- Install the package using one of the commands below.

```
$ sudo dpkg -i iperf_2.1.9+dfsg1-2_amd64.deb
```

OR

```
$ sudo apt install /path/to/package/iperf_2.1.9+dfsg1-2_amd64.deb
```

Note: *Iperf 2.1.9 is used for the demonstration.*

- Run the suitable command from the following table.

Table 2. iPerf commands for Linux remote host

Functionality	Command
TCP server	iperf -s -i 1
UDP server	iperf -s -u -i 1
TCP client	iperf -c <server_ip> -i 1 -t 60
UDP client	iperf -c <server_ip> -u -i 1 -t 60

Remote host setup for cell phones

To run iPerf:

- Download the iPerf application like Magic iPerf, or HE.NET Network Tools
- Open the application and select iperf2.
- Run the appropriate command from Table 2

Table 3. iPerf commands for cell phone remote host

Functionality	Command
TCP server	-s -i 1
UDP server	-s -u -i 1
TCP client	-c <server_ip> -i 1 -t 60
UDP client	-c <server_ip> -u -i 1 -t 60

2.4 IPv4 and IPv6 tool setup

The IPv4 or IPv6s tool is used to send or receive data via TCP or UDP connection to interact with wifi_ipv4_ipv6_echo sample application (Section 4.5).

Remote host setup

- **ncat** - Recommended tool that supports both IPv4 and IPv6. ncat is part of nmap tools [11].
- **nc (netcat)** - Similar to ncat. Anti virus applications tend to tag ncat as virus, so it may be available for use on a PC or laptop.
- **echotool** - Supports only IPv4 and only for Windows [8].

Zone index (zone ID)

- On Windows, the zone index is a number. You can get it from the output of the `ipconfig` command
- On Linux, the zone index is an interface name
- To connect to board with address FE80::12:13FF:FE10:1511
  - Over interface *21* on your Windows machine, specify the address as FE80::12:13FF:FE10:1511%21
  - Over interface *eth* on your Linux or Mac machine, specify the address as FE80::12:13FF:FE10:1511%eth0

Note: *The demo has only one single interface. Do not append the zone ID to any address typed to the demo terminal.*

2.5 J-Link commander setup

J-Link commander is a command line tool used with J-Link to:

- Verify the installation of the USB driver
- Check the connection to the target CPU
- Run an analysis of the target system

J-Link commander is included in *J-Link Software and Documentation Package*, with other applications such as the J-Link GDB server. The package is available for download at segger.com/jlink-software.html, and can be installed for Windows, Linux, and Mac OS.

Command to install J-link software on a computer running Linux Ubuntu:

```
sudo dpkg -i Jlink_Linux_V766d_x86_64.deb
```

Note: *To work with RW61x, additional patches are needed for the tools. Refer to the section RW61x product image setup in the user manual reference UM11798.*

3 Running a demo

This section shows how to run a demo for Wi-Fi or Bluetooth LE using MCUXpresso IDE, Arm GCC, IAR IDE, or Keil IDE.

Note: *The following examples are used to run a Wi-Fi demo. The same steps apply to run a Bluetooth LE demo.*

3.1 Run a demo using MCUXpresso IDE

This section describes the setups to import, configure, build, debug, and run the demo example through MCUXpresso IDE. MCUXpresso IDE version v11.6.0 is used in the following demo steps.

3.1.1 Import the project

Step 1 - SDK installation

- Open MCUXpresso IDE
- Locate the *Installed SDKs* tab at the bottom of the central window
- Drag and drop the SDK into the *Installed SDKs* tab (Figure 1)

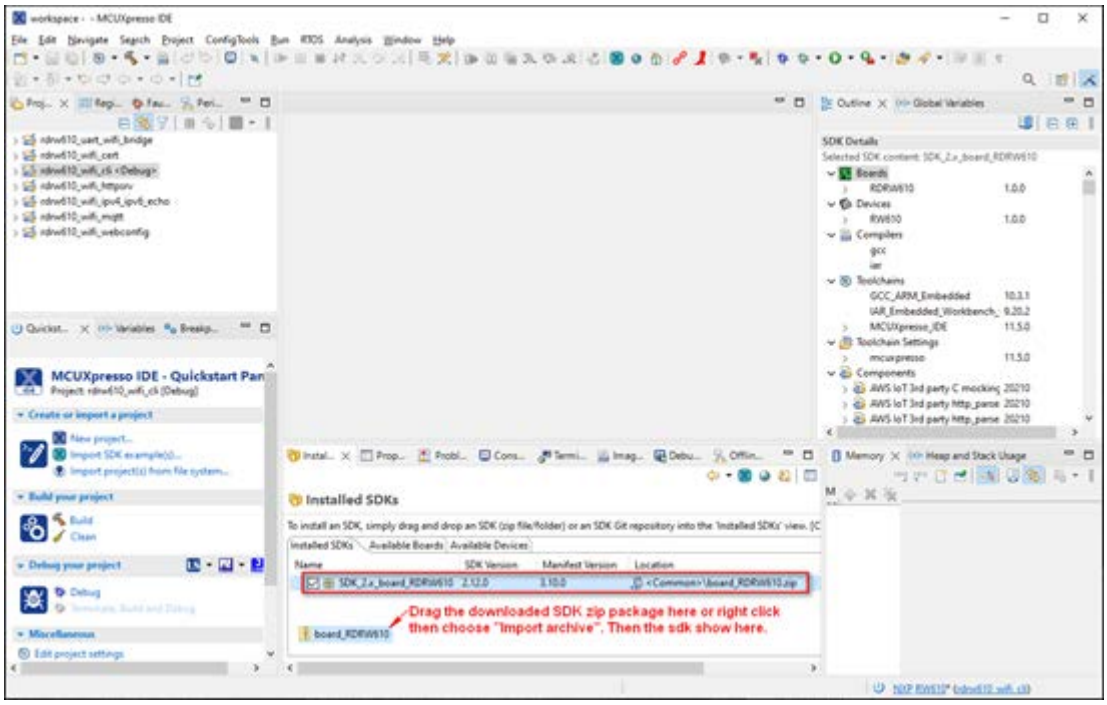


Figure 1. Drag and drop the downloaded SDK into Installed SDK tab

- Click **OK** on the pop-up window (Figure 2)

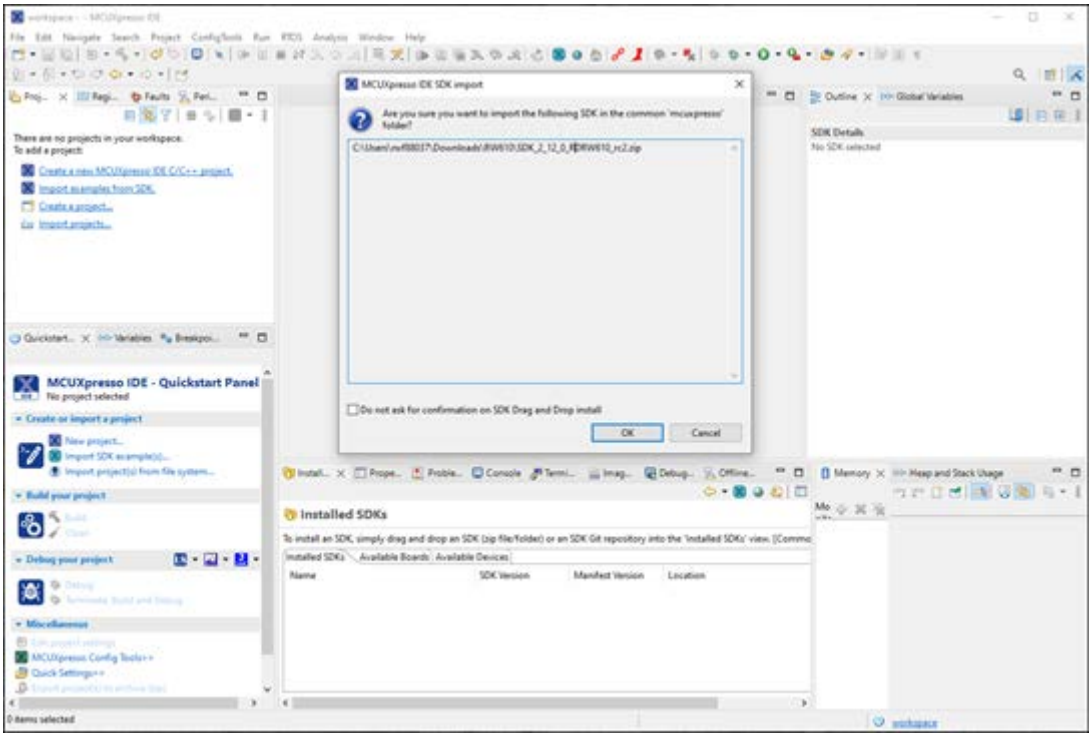


Figure 2. Confirm the SDK installation

Step 2 - Import an example

- Go to the *Quickstart* panel and select the option *Import SDK examples* (Figure 3)

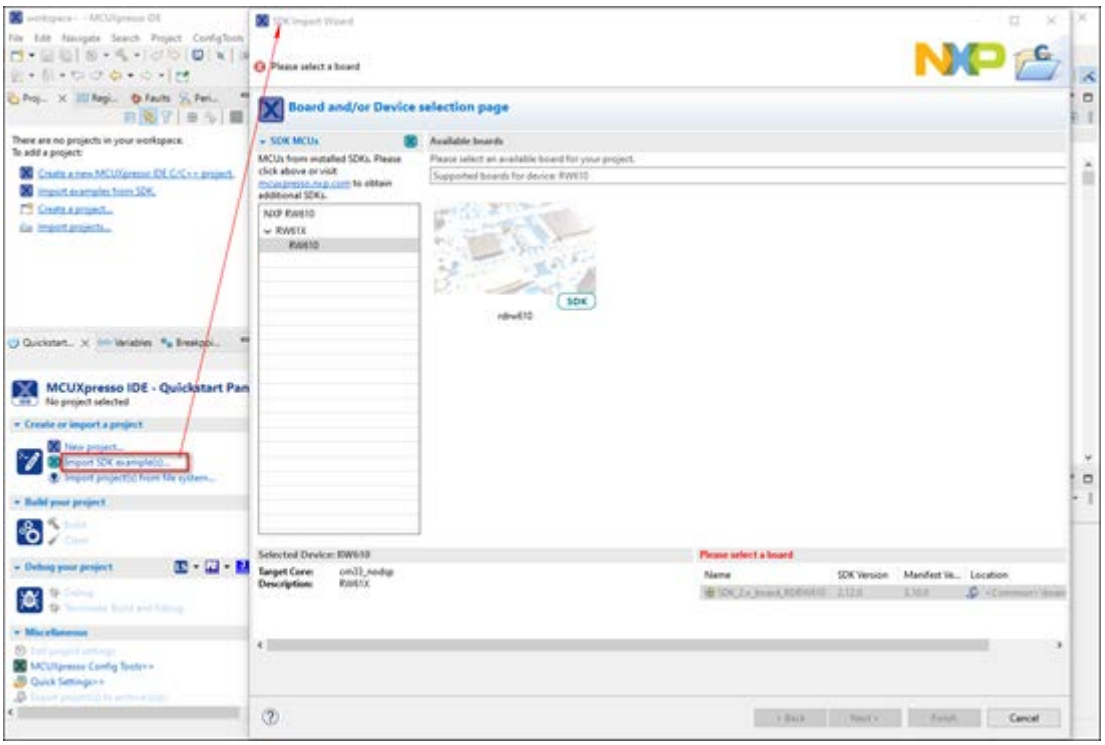


Figure 3. Import an example

Step 3 - Select the EVK board

- Select the evaluation board (Figure 4)

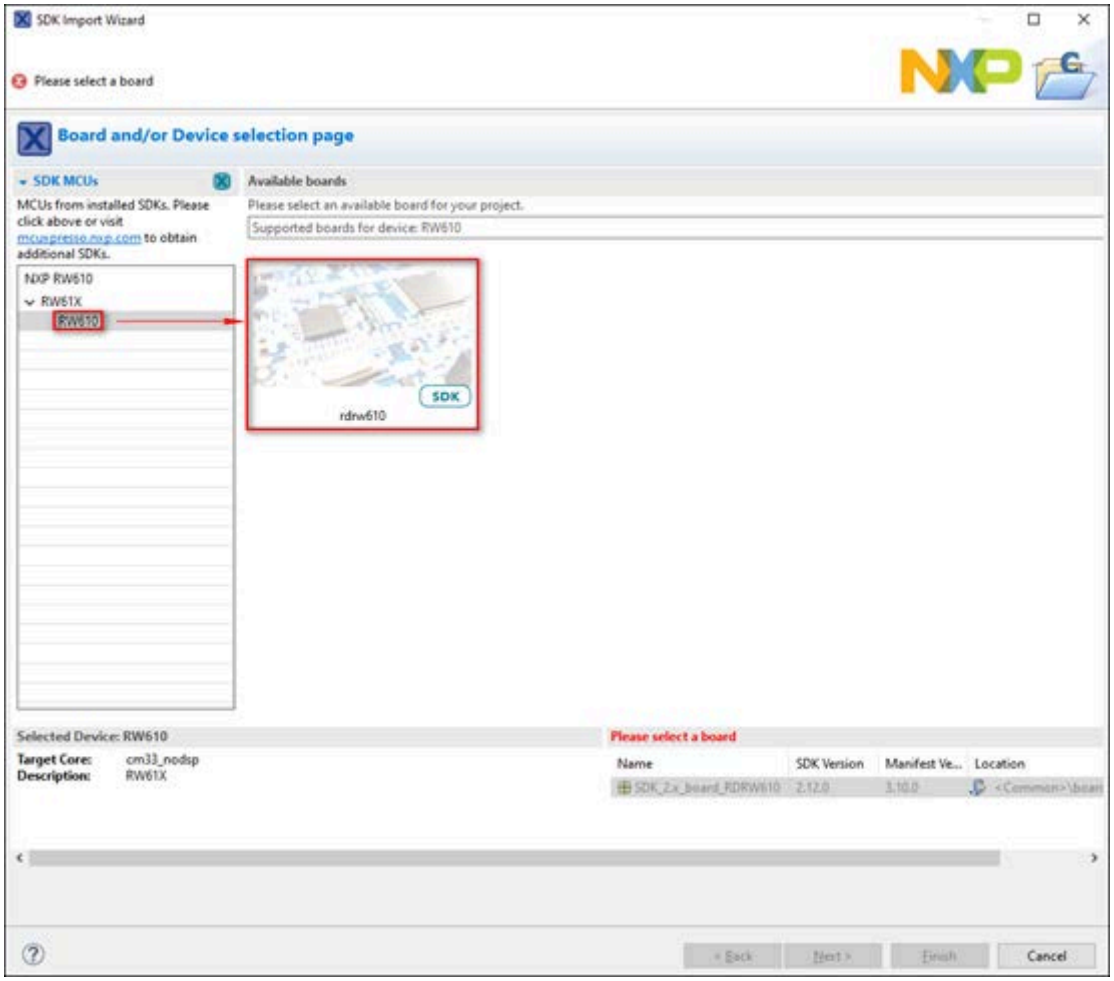


Figure 4. Select the evaluation board

Step 4 - Select a Wi-Fi or Bluetooth example and verify the default project options

- For example, select *wifi_examples* > *wifi_cli* and click the **Finish** button to import the selected example into the workspace (Figure 5)

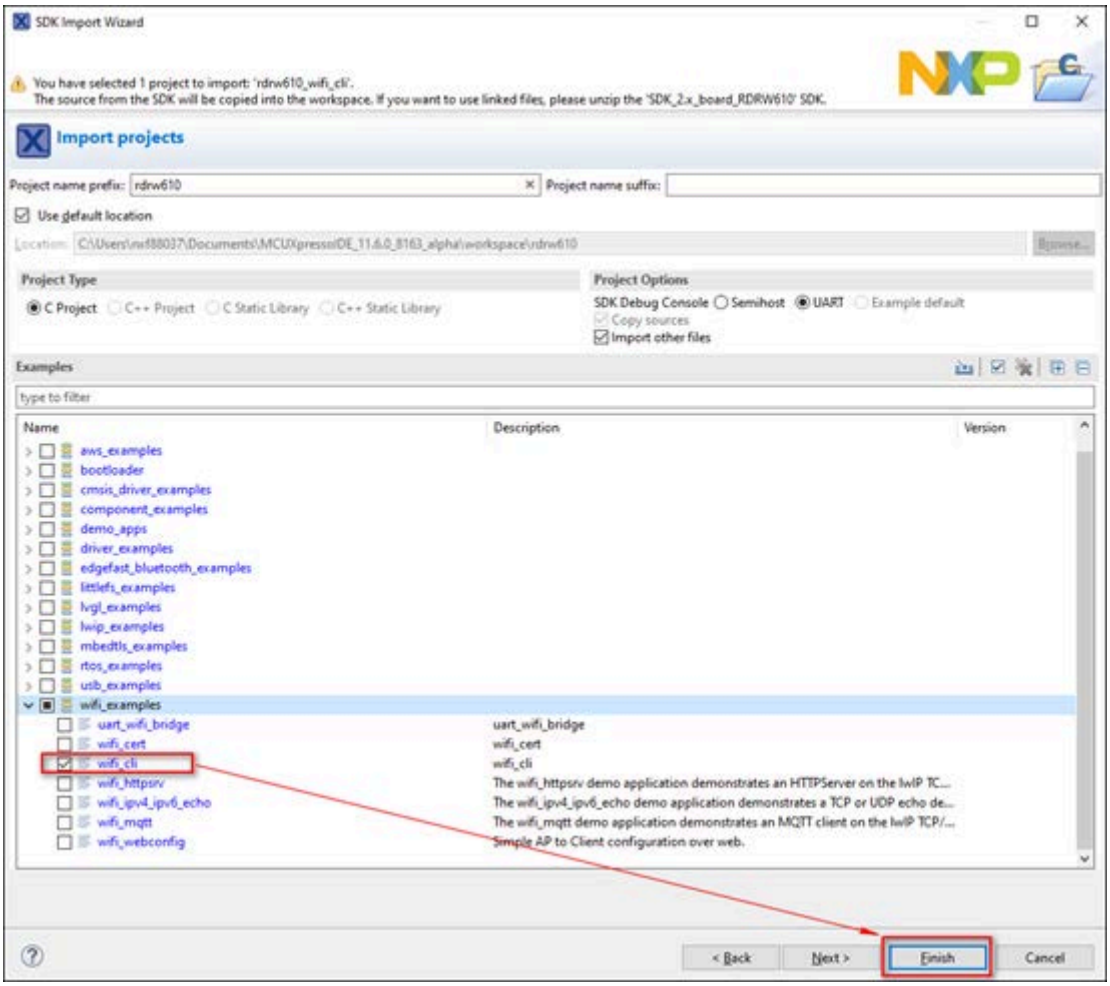


Figure 5. Import wifi_cli example

3.1.2 Build the application

To build the application:

- Go to the *Quickstart* panel and select *Build*, or select the *Build* icon in the main toolbar
- Verify the build result (success or fail) on the console window

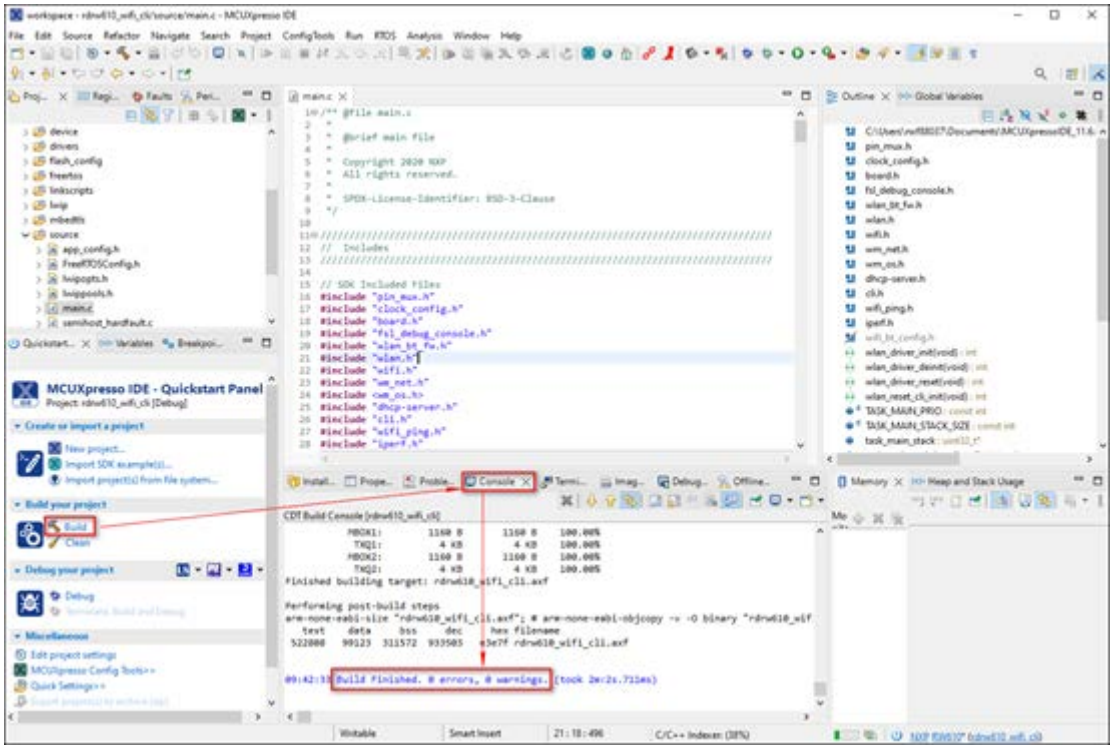


Figure 6. Select Build on Quickstart panel or in the toolbar

3.1.3 Run the application in Debug mode

To run the application in Debug mode:

- Initiate the application debug using the debug icon in the toolbar or got the *Quickstart* panel and select *Debug*
- Select the associated emulator probe for the first time and click **OK** (Figure 7)

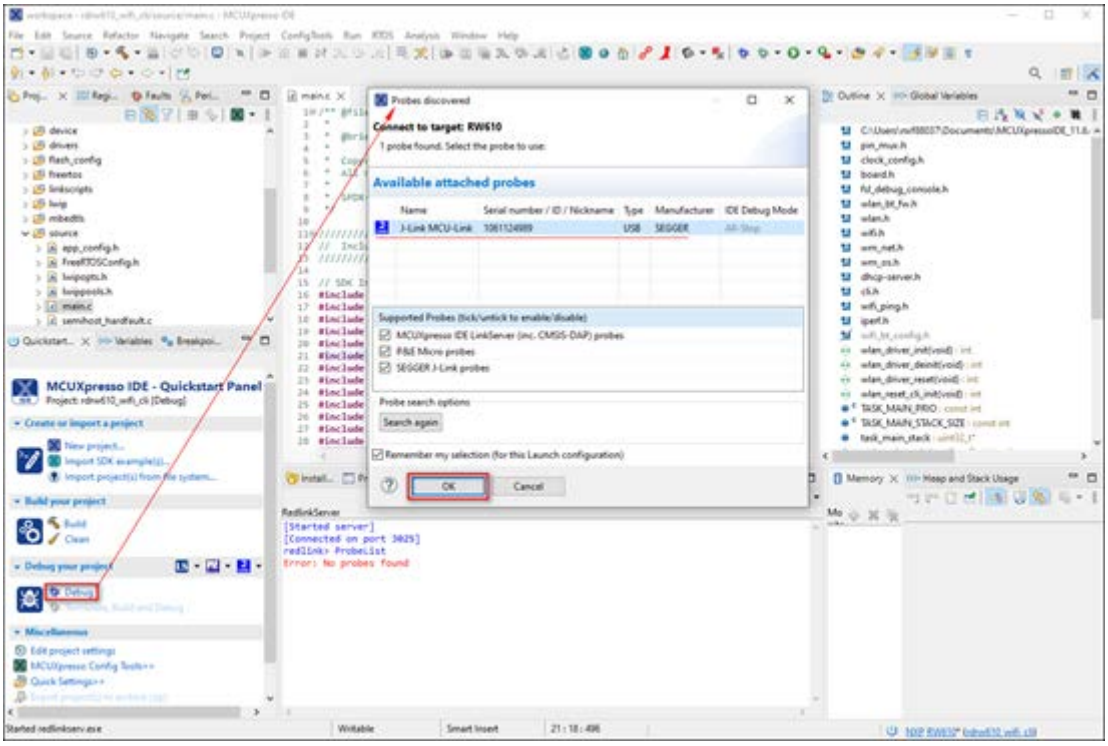


Figure 7. Select Debug mode and select the associated probe

Upon selecting the probe, the application is downloaded on the board and the program execution starts with the program counter set at the main() function (Figure 8).

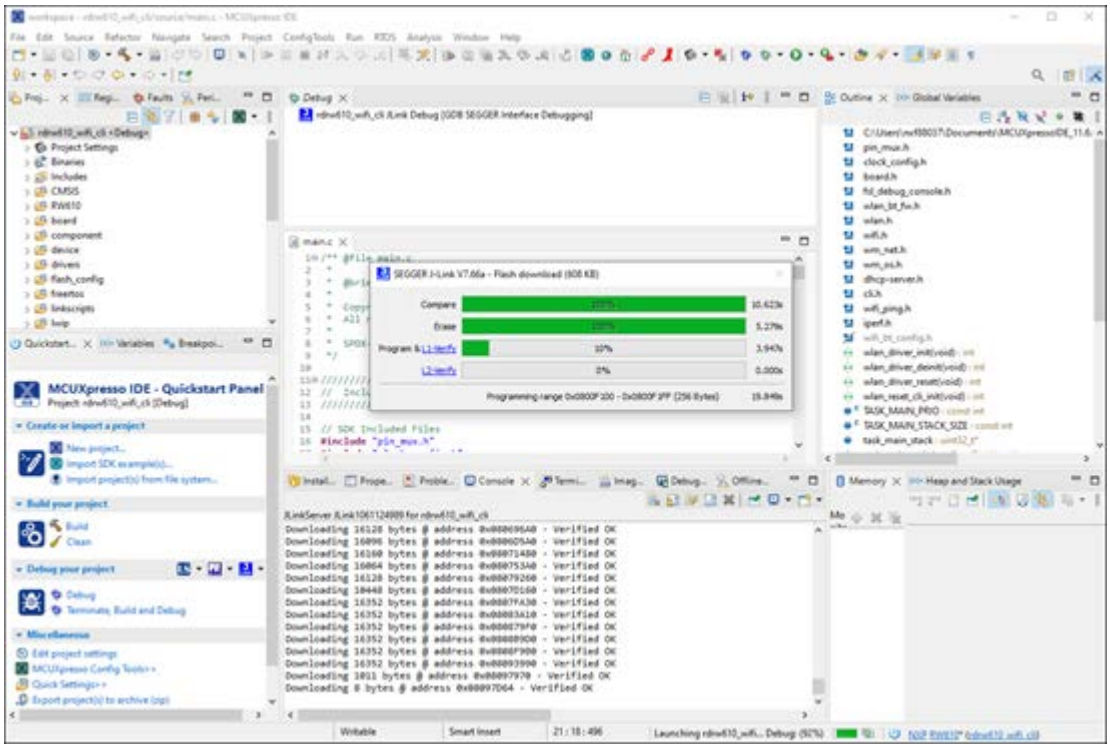


Figure 8. Application download start of program execution

- Click **Resume** to start the application
- To debug the application, use the **step into**, **step over** and **step return** buttons (Figure 9)
- To end the debugging session, use the **Terminate** button (Figure 9)

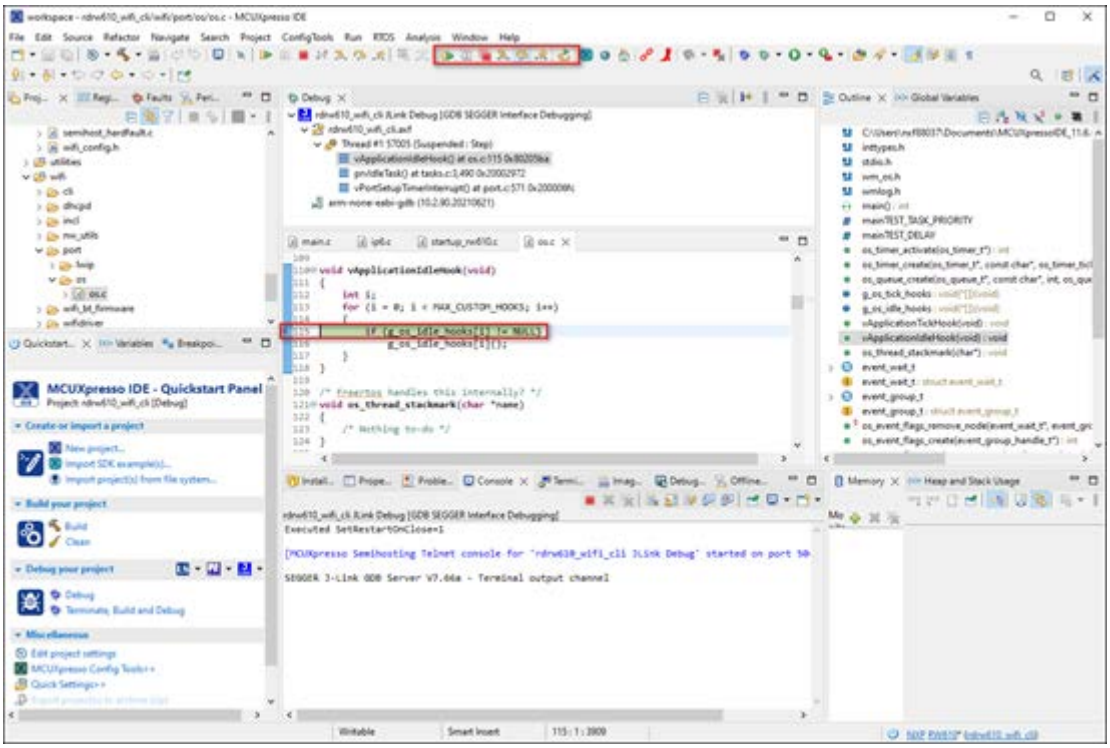


Figure 9. Run and debug the application

3.1.4 Run the application program (no debugging)

Use the following steps to flash the application program.

- To flash the required binaries, select the *GUI Flash Tool* icon in the toolbar (Figure 10)

The GUI Flash Tool can be used to flash the pre-built binary or the locally compiled binary with **.axf* or **.bin* format. The path to the locally compiled binary is *${workspace_loc}\rdwr610_wifi_cli\Debug\rdwr610_wifi_cli.axf*.

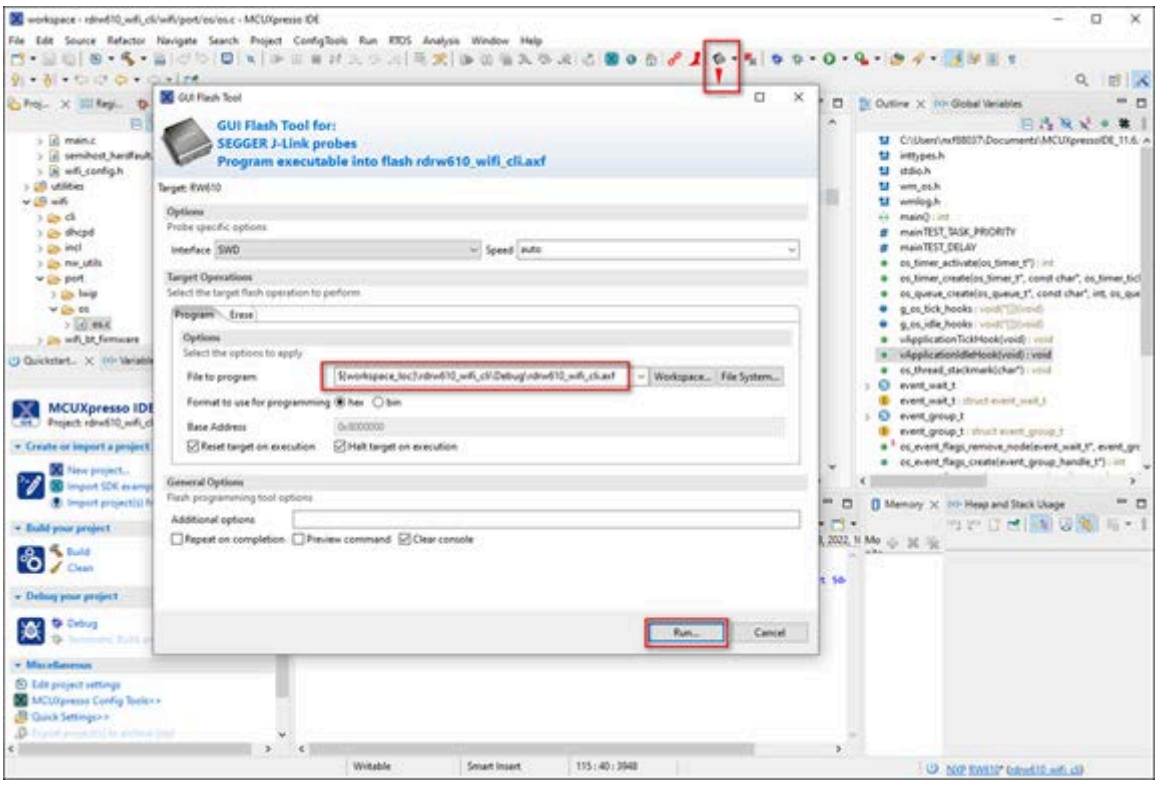


Figure 10. Using the GUI Flash tool for the pre-built binary or locally compiled binary

3.2 Run a demo using Arm® GCC

This section describes the steps to configure the command-line Arm® GCC tools to build and run demo applications. The wifi_cli application is used as an example. The same steps apply to any other example application available with the MCUXpresso SDK. The example uses Linux, one of the operating systems that Arm GCC tools support. Refer to MCUXSDKGSUG for more details on Arm GCC toolchain setup.

3.2.1 Install ARM® GCC toolchain

In this section, the following steps are given to install toolchain:

- Download the toolchain for Linux x86_64 system from the Link (package *Linux x86_64 tarball*).
- Create a directory at the location of your choice, for example */home* or */usr/bin*:

```
$ mkdir toolchain-dir
```

- Copy the downloaded toolchain package to the created directory and extract the downloaded toolchain.

```
$ cp <download_path>/gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2 toolchain-
dir/
$ cd toolchain-dir/
$ tar -xf gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2
```

- Export the ARMGCC_DIR variable using the following command:

```
$ export ARMGCC_DIR=<absolute-path>/toolchain-dir/gcc-arm-none-eabi-10-2020-q4-major/
```

- Add the toolchain path to the **PATH** environment variable using the command:

```
$ export PATH=$PATH:<absolute-path>/toolchain-dir/ gcc-arm-none-eabi-10-2020-q4-major/
bin/
```

- Download and install *cmake* (source and binary distribution) for Linux system [7].
  Or use `sudo apt-get install cmake` for the installation.
- Extract the source distribution and copy it to the */usr/share/* directory

```
$ tar -zxf cmake-3.19.1.tar.gz
$ sudo cp -rf cmake-3.19.1 /usr/share/cmake-3.19
```

- Extract the binary distribution and copy the binaries to the */usr/bin/* directory

```
$ tar -zxf cmake-3.19.1-Linux-x86_64.tar.gz
$ sudo cp cmake-3.19.1-Linux-x86_64/bin/* /usr/bin/
```

3.2.2 Build the application

This section provides the steps to build the application using the Arm GCC toolchain:

- Go to the *armgcc* directory of the application

```
$ cd <SDK-top-dir>/boards/rdrw610/wifi_examples/wifi_cli/armgcc/
```

- Build the binary

```
$ sh build_flash_debug.sh
[100%] Linking C executable flash_debug/wifi_cli.elf
[100%] Built target wifi_cli.elf
```

The application image *sdk20-app.bin* is auto generated.

```
$ ls ./flash_debug
sdk20-app.bin  wifi_cli.elf
```

Note: *Refer to MCUXSDKGSUG for details on how to debug the application using GDB.*

3.2.3 Flash the application program (no debugging)

This section provides the steps to flash the binary on the RW61x EVK board:

- Connect the board to the Windows host system. Open J-Link commander and connect to RW61x.

```
J-Link>con
Device>RW610
TIF>S
Speed><Enter>
```

- Flash the application image *sdk20-app.bin* to RW61x EVK FlexSPI NOR flash.

```
J-Link>loadbin sdk20_app.bin,0x08000000
```

Where `0x08000000` is the NOR Flash base address.

- Reset RW61x EVK board power .
- To access the device using the serial console, refer to section Section 2.1.

```
=========================
wifi cli demo
=========================================
Initialize CLI
=========================================
Initialize WLAN Driver
=========================================
MAC Address: 00:13:43:7F:9C:9F
[net] Initialized TCP/IP networking stack
=========================================
app_cb: WLAN: received event 10
=========================================
app_cb: WLAN initialized
=========================================
WLAN CLIs are initialized
=========================================
```

Note: *Refer to Section 4.1.2 to view the output on the console once the application is executed.*

3.3 Run a demo with IAR IDE

This section provides the steps to open, configure, build, debug, and run the demo example using IAR Embedded Workbench IDE. The instructions and illustrations refer to IAR version 9.10.2.

3.3.1 Open the project workspace

To open the wifi_cli project available in the SDK, double-click the project workspace file named *wifi_cli.eww* stored at the following location:

<install_dir>\boards\rdrw610\wifi_examples\wifi_cli\iar\wifi_cli.eww

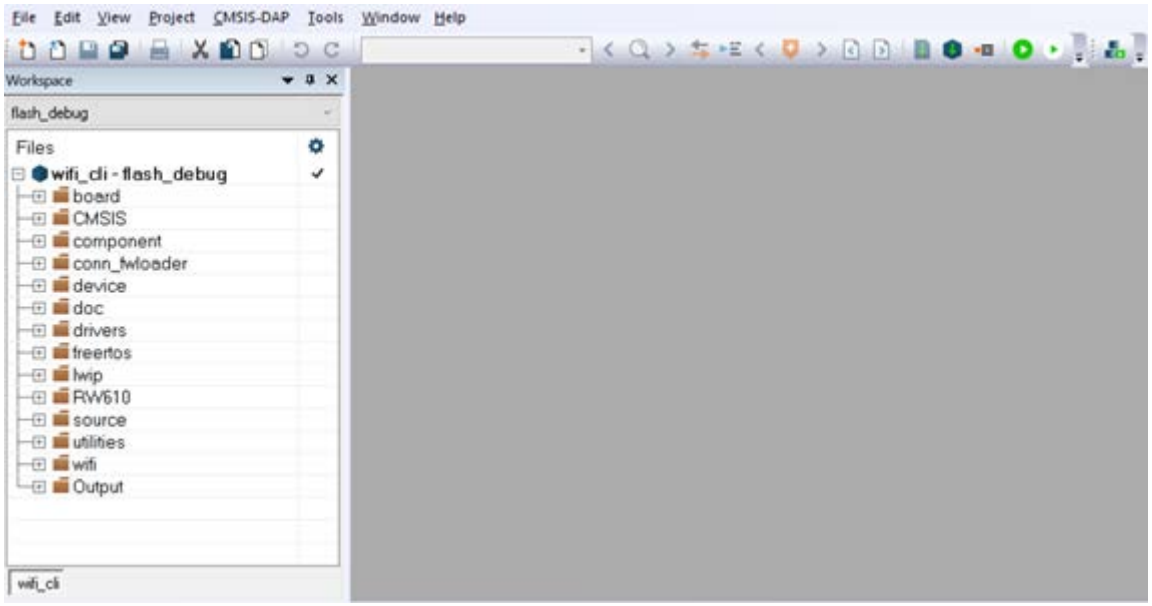


Figure 11. Open the project in IAR

3.3.2 Project settings

By default, the project is configured to use the `WIFI_BOARD_AW_RW610` in *app_config.h* from the source code directory.

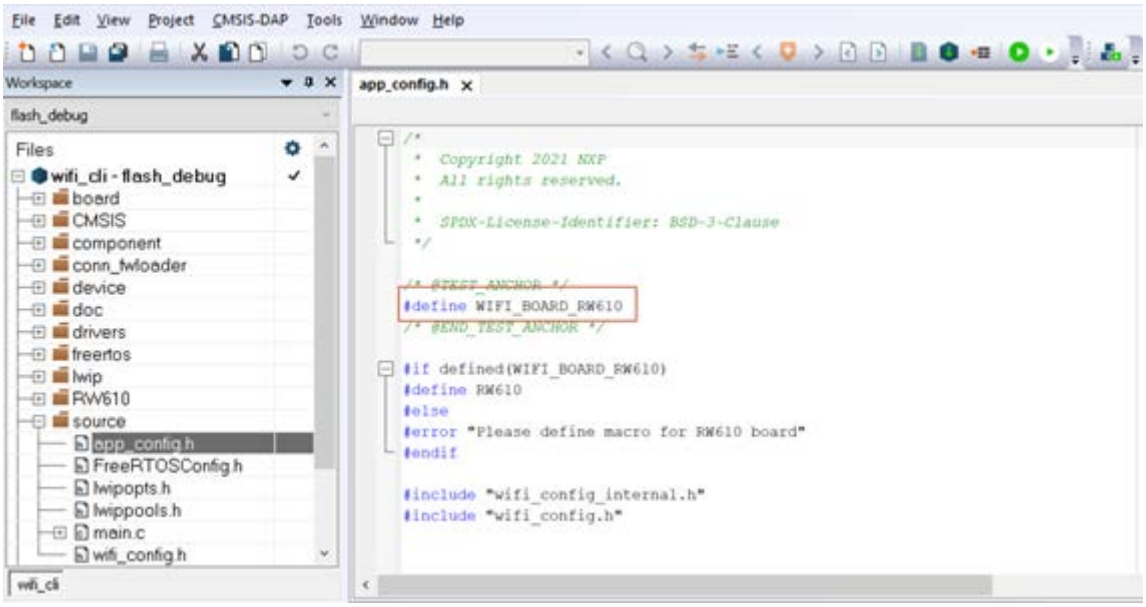


Figure 12. Wi-Fi board selection in IAR

3.3.3 Build the application

To build the *wifi_cli* application:

- Press the **Make** icon as illustrated below.

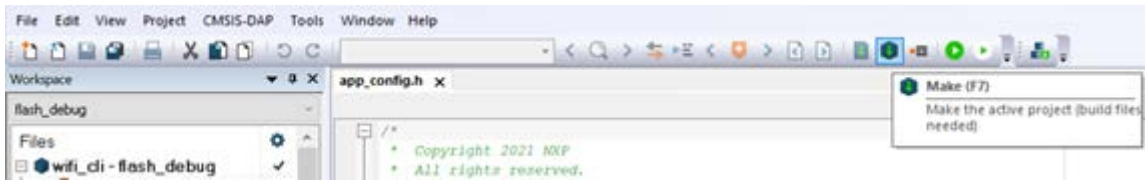


Figure 13. Application build in IAR

The details of the Build procedure are displayed in the **Messages** window of the **Build** tab.

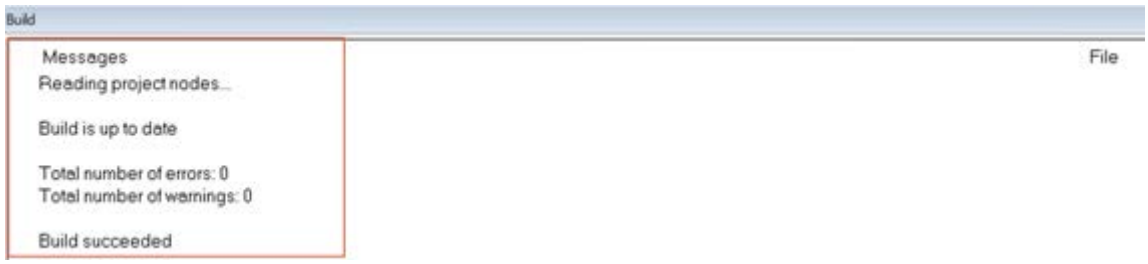


Figure 14. Build message in IAR

3.3.4 Run the application in Debug mode

The following steps describe how to run the application in Debug mode.

The default debugger is **CMSIS-DAP**. However, if **CMSIS-DAP** is not selected, use the drop-down list to select it and press **OK**.

The selection of the debugger is a one-time configuration step that is not required for incremental debug.

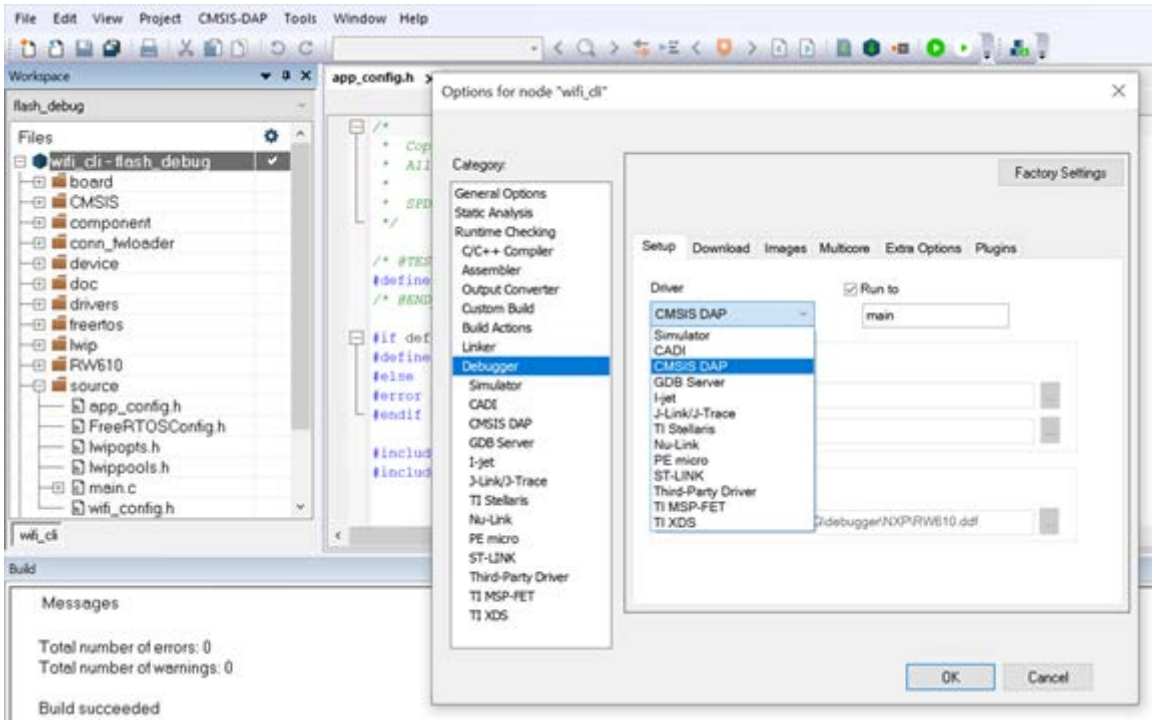


Figure 15. Debugger selection in IAR

- To initiate the application debug, press the **Download and Debug** icon on the toolbar.

Figure 16. Initiate Debug in IAR

The **Download and Debug** button is used to download the application to the target and set the program counter to the main( ) function of the application.

- Press **Go** to start the application.
- To debug the application, use the **Step Into**, **Step over** and **Step return** icons.
- To stop the debugging session, press the **Stop Debugging** icon.

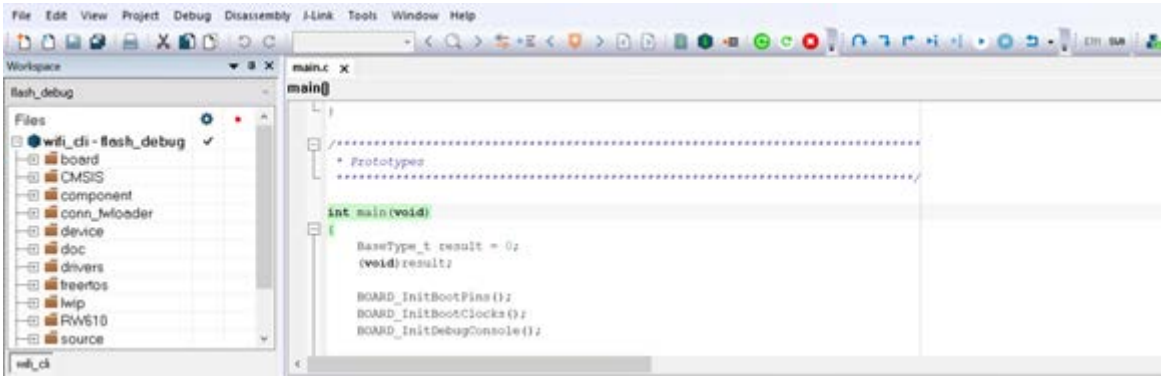


Figure 17. Application debugging in IAR

3.3.5 Flash the application program (no debugging)

To flash the application program:

- Go to **Project > Download** to flash the binary file.

The **Download** menu provides the commands to flash the pre-built binary file and to erase the memory.

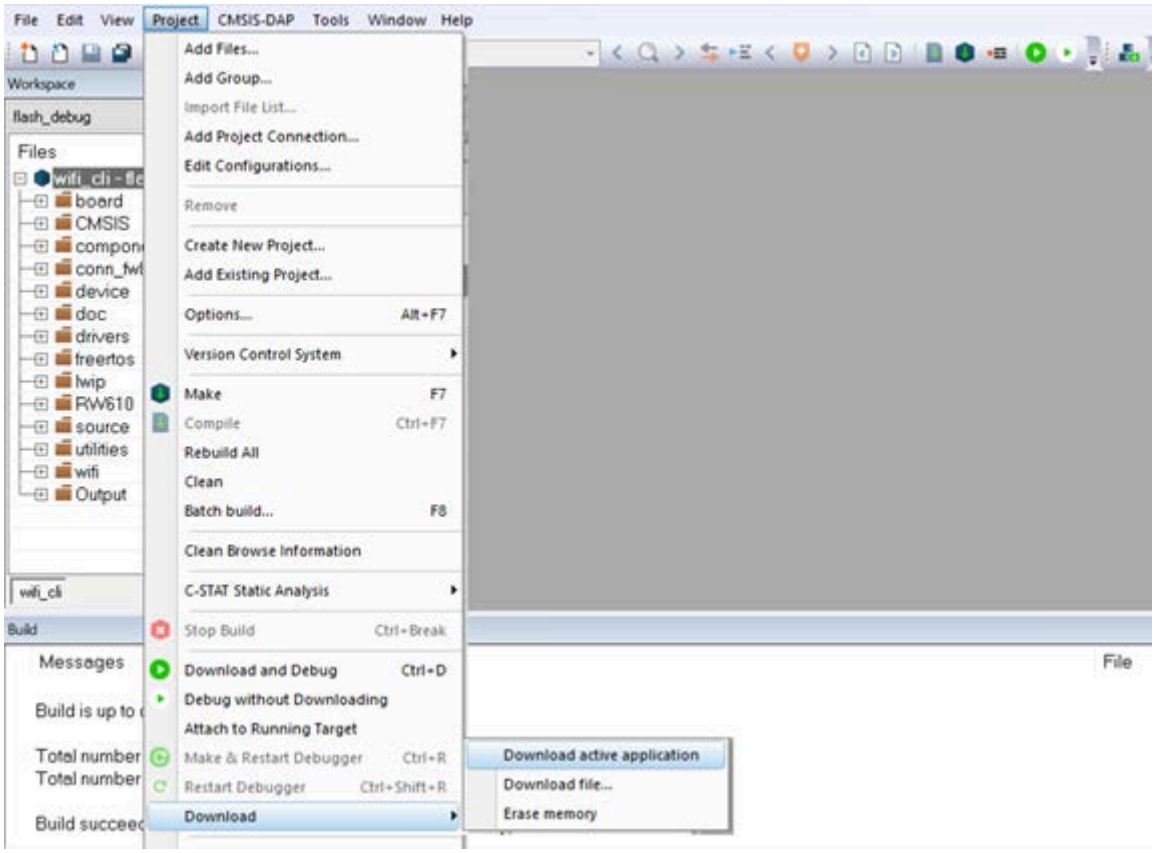


Figure 18. Binary flashing in IAR

Note: *Refer to Section 4.1.2 to view the output on the console once the application is executed.*

3.4 Run a demo using Keil MDK/µVision

This section details the steps to open, configure, build, debug, and run a demo example using Keil IDE. The Keil version used in this document is v5.38.

3.4.1 Install CMSIS device pack

Following the installation of the MDK tools, install the CMSIS device packs so you can use the debug functionality on your device. The CMSIS device packs include the memory map information, register definitions and flash programming algorithms. The following steps install the CMSIS pack for RW612.

- Download RW612_DFP file from NXP website
- Double click the downloaded file to install RW612 software pack
- When the installation is complete, click on the **Pack Installer** icon in the toolbar. RW612 can be found in the Devices tab. The DFP is listed in the **Packs** tab and displayed as up to date in the **Action** column.

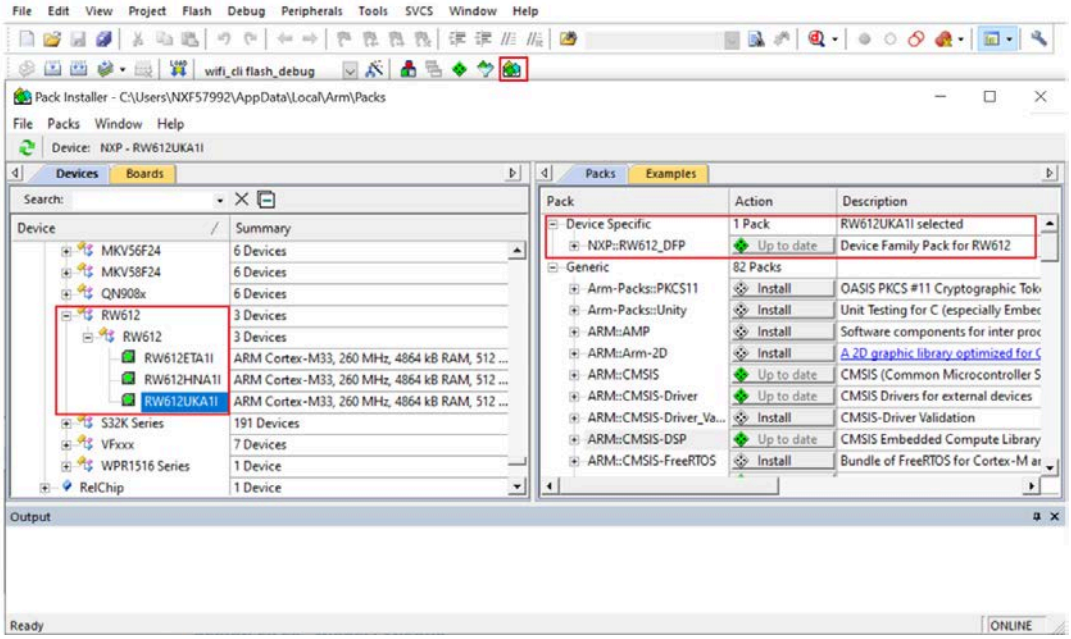


Figure 19. Pack Installer in Keil

3.4.2 Open the project workspace

To open the wifi_cli project, double-click the project workspace file *wifi_cli.uvprojx* located at:

<install_dir>\boards\rdrw61x\wifi_examples\wifi_cli\mdk\wifi_cli.uvprojx

Note: *For a multi-project, use wifi_cli.uvmpw instead of wifi_cli.uvprojx.*

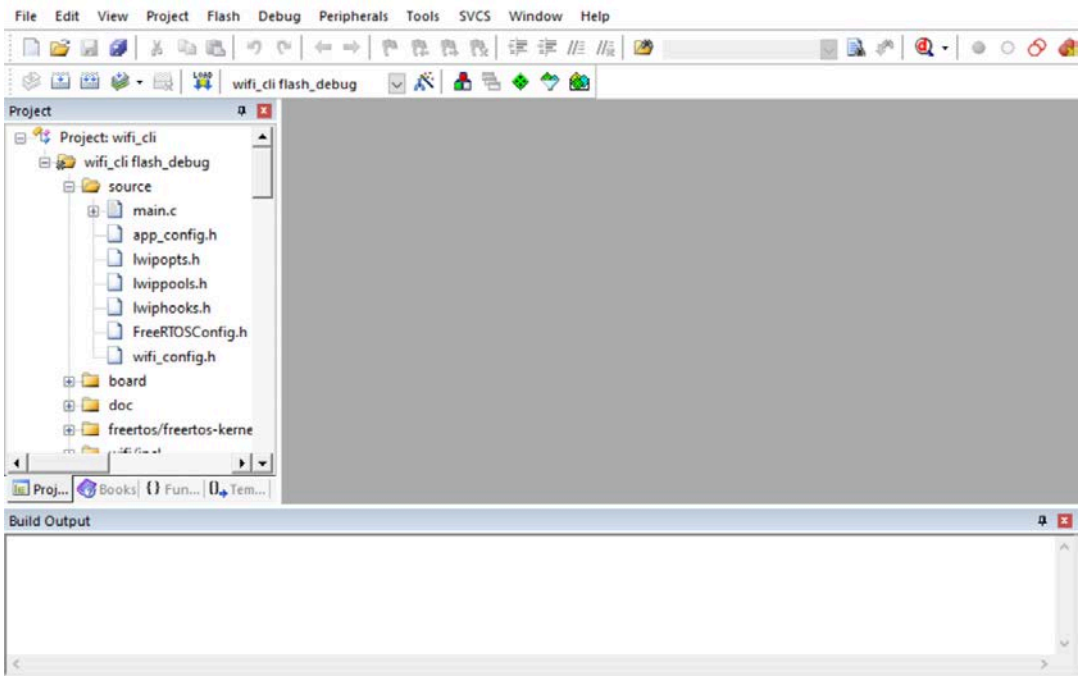


Figure 20. Open the project in Keil

3.4.3 Project settings

By default, the project is configured to use the WIFI_BOARD_RW610 in *app_config.h* from the source code directory.

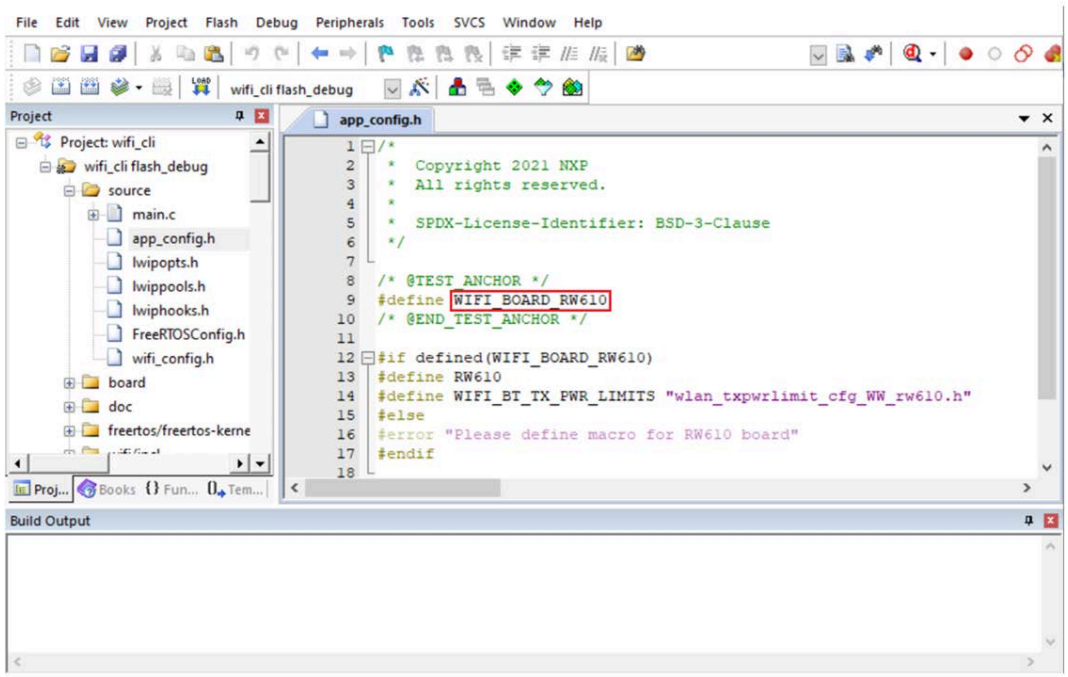


Figure 21. Wi-Fi board selection in Keil

3.4.4 Build the application

To build the application:

- Click the Build or Rebuild icons

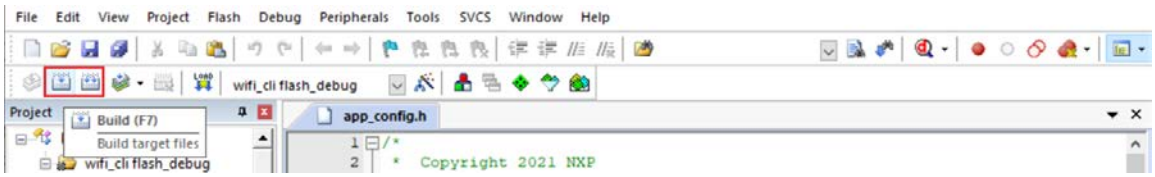


Figure 22. Build and Rebuild icons in Keil

- Verify the build progress in the Build Output window.

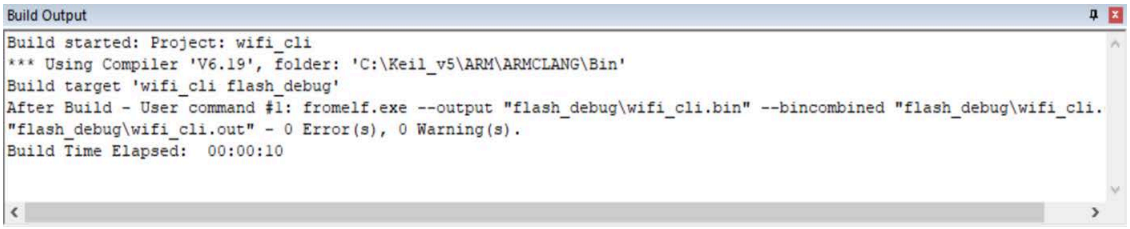


Figure 23. Build output window in Keil

3.4.5 Run the application in debug mode

The default debugger is CMSIS-DAP. If CMSIS-DAP is not selected: use the Options icon in the toolbar, open the Debug tab, select the debugger in the drop-down list, and press OK.

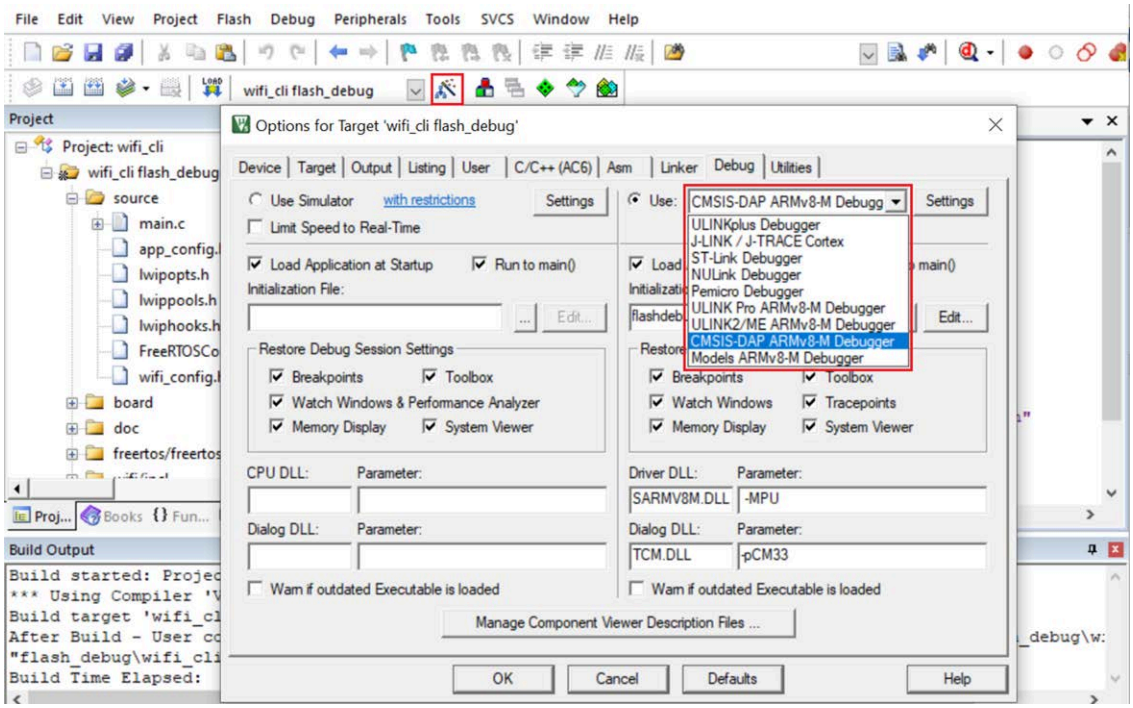


Figure 24. Select the debugger in Keil

To start the application debug:

- Click the **LOAD** icon to download the application on the board

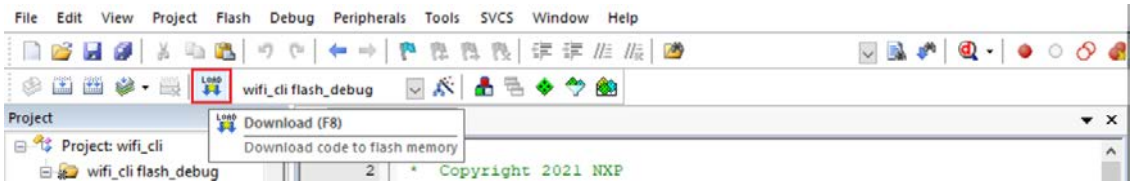


Figure 25. Load the application

- Click the **Start/Stop Debug Session** icon in the toolbar

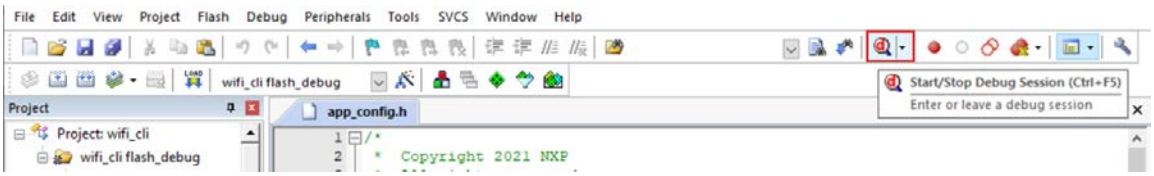


Figure 26. Start the debug session in Keil

- Click the **Start/Stop Debug Session** icon to set the program counter to the `main()` function of the application

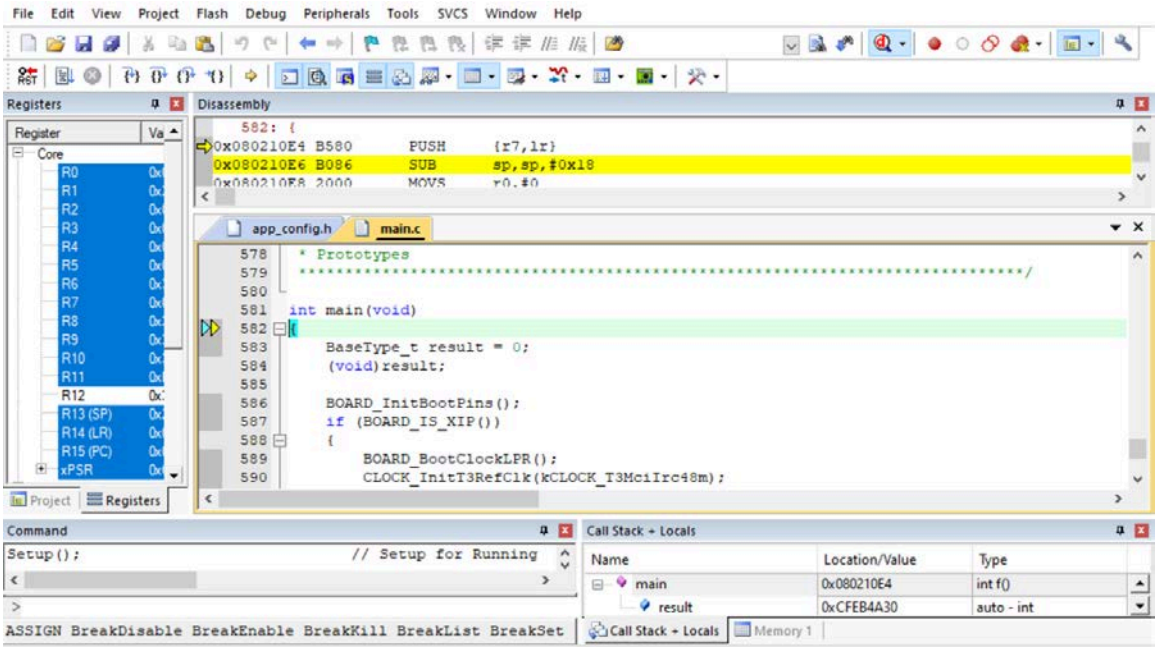


Figure 27. Set the program counter in Keil

- Press **Run** to start the application. Use **Step**, **Step Over**, **Step Out**, and **Run to Cursor Line** icons in the toolbar to debug the application.
- To end the debugging session, click the **Stop** icon

Figure 28. Application debugging features in Keil

3.4.6 Flash the application program (no debugging)

To flash the application program:

- Click the **Download** icon in the toolbar to flash the required binary file
- Refer to Section 4.1.2.1 to view the output on the console once the application is executed.

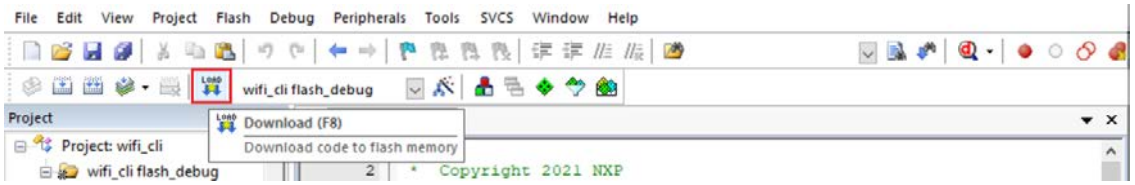


Figure 29. Flashing a binary file in Keil

4 Wi-Fi sample applications

This section describes the Wi-Fi example applications that are available in the SDK, and the steps to configure, compile, debug, flash, and execute these examples.

4.1 wifi_cli sample application

This section describes the *wifi_cli* application. wifi_cli is used to demonstrate the CLI support to handle and enable Wi-Fi configuration to:

- Scan the visible access points
- Create and configure the access point
- Connect with the access point
- Check the throughput performance using iPerf measurement tool

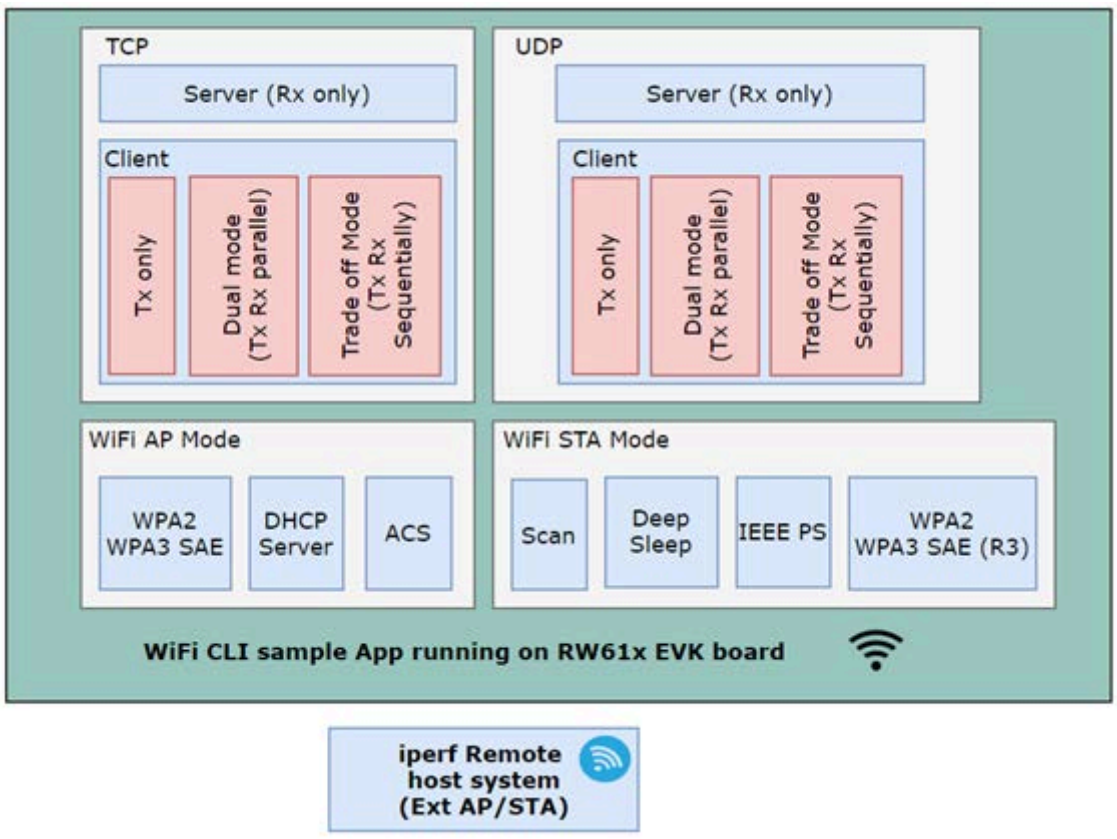


Figure 30. wifi_cli sample application components

Wi-Fi and iPerf features:

Table 4. Sample application features

Features	Details
Wi-Fi	Wi-Fi Mobile AP mode Wi-Fi Station mode Wi-Fi Scan Wi-Fi Roaming Wi-Fi TX Power Limit Wi-Fi Regulatory Domain/Operating Class/Country Wi-Fi Power Save (IEEEPS, WMMPS, WNMPS, Deep Sleep) Wi-Fi Security (WPA2/WPA3) Wi-Fi ED MAC Wi-Fi Net Monitor Host Sleep
IPerf	TCP Client and Server TCP Client dual mode (TX and RX in simultaneous) TCP Client trade-off mode (TX and RX individual) UDP Client and Server UDP Client dual mode (TX and RX in simultaneous) UDP Client trade-off mode (TX and RX individual)

4.1.1 Flash the Wi-Fi firmware

RW61x application and Wi-Fi firmware binary are stored in different partitions of FlexSPI NOR flash. The application reads Wi-Fi firmware during initialization and downloads it to RW61x internal Wi-Fi MCU to run. This section describes the steps to flash Wi-Fi firmware with SEGGER J-Link tool.

- Open J-Link commander on Windows and connect RW61x chip

```
J-Link>con
Device>RW610
TIF>S
Speed><Enter>
```

- Flash Wi-Fi firmware

The path to Wi-Fi secure firmware binary is:

${SDK}\components\conn_fwloader\fw_bin\rw61x_sb_wifi_v1.bin for A1 version of RW61x.

${SDK}\components\conn_fwloader\fw_bin\rw61x_sb_wifi_v2.bin for A2 version of RW61x.

```
J-Link>loadbin rw61x_sb_wifi.bin_v<version number>,0x08400000
```

Note: *Wi-Fi firmware must be flashed once unless it is erased. It is stored at a given address (Figure 31). Ensure that the Wi-Fi firmware is flashed before running any Wi-Fi demo application.*

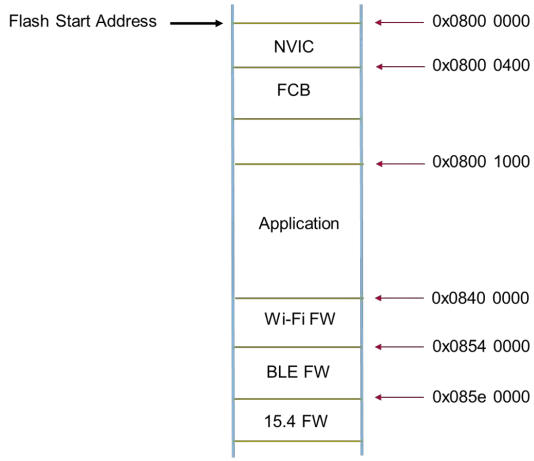


Figure 31. RW61x FlexSPI flash layout

4.1.2 wifi_cli application execution

4.1.2.1 Start-up logs

The following logs show on the console once RW61x EVK board is up and running and the console shows that Wi-Fi is ready for the operations. This section describes the available Wi-Fi commands. Press Enter for the command prompt.

```
========================
wifi cli demo
========================================
Initialize CLI
========================================
Initialize WLAN Driver
========================================
MAC Address: 00:13:43:7F:9C:9F
[net] Initialized TCP/IP networking stack
========================================
app_cb: WLAN: received event 10
========================================
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
CLIs Available:
========================================

help
clear
wlan-version
wlan-mac
wlan-thread-info
wlan-net-stats
wlan-set-mac <MAC_Address>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-connect-opt <profile_name> ...
wlan-reassociate
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-set-ps-cfg <null_pkt_interval>
wlan-deep-sleep-ps <0/1>
wlan-get-beacon-interval
wlan-wnm-ps <0/1> <sleep_interval>
wlan-set-max-clients-count <max clients count>
wlan-rts <sta/uap> <rts threshold>
wlan-frag <sta/uap> <fragment threshold>
wlan-host-11k-enable <0/1>
wlan-host-11k-neighbor-req [ssid <ssid>]
wlan-host-11v-bss-trans-query <0..16>
wlan-mbo-enable <0/1>
wlan-mbo-nonprefer-ch <ch0> <Preference0: 0/1/255> <ch1> <Preference1: 0/1/255>
wlan-sta-filter  <filter mode> [<mac address list>]
wlan-get-log <sta/uap> <ext>
wlan-tx-pert <0/1> <STA/UAP> <p> <r> <n>
wlan-roaming <0/1> <rssi_threshold>
wlan-multi-mef <ping/arp/multicast/del> [<action>]
wlan-wakeup-condition <mef/wowlan wake_up_conds>
wlan-auto-host-sleep <enable> <mode> <rtc_timer> <periodic>
wlan-send-hostcmd
wlan-ext-coex-uwb
```

```
wlan-set-uap-hidden-ssid <0/1/2>
wlan-eu-crypto-rc4 <EncDec>
wlan-eu-crypto-aes-wrap <EncDec>
wlan-eu-crypto-aes-ecb <EncDec>
wlan-eu-crypto-ccmp-128 <EncDec>
wlan-eu-crypto-ccmp-256 <EncDec>
wlan-eu-crypto-gcmp-128 <EncDec>
wlan-eu-crypto-gcmp-256 <EncDec>
wlan-mem-access <memory_address> [<value>]
wlan-eu-validation <value>
wlan-set-antcfg <ant_mode> <evaluate_time> <evaluate_mode>
wlan-get-antcfg
wlan-scan-channel-gap <channel_gap_value>
wlan-wmm-stat <bss_type>
wlan-reset
wlan-set-regioncode <region-code>
wlan-get-regioncode
wlan-11d-enable <sta/uap> <0/1>
wlan-uap-set-ecsa-cfg <block_tx> <oper_class> <new_channel> <switch_count> <bandwidth>
wlan-csi-cfg
wlan-set-csi-param-header  <csi_enable> <head_id> <tail_id> <chip_id> <band_config> <channel>
 <csi_monitor_enable> <ra4us>
wlan-set-csi-filter <opt> <macaddr> <pkt_type> <type> <flag>
wlan-txrx-histogram <action> <enable>
wlan-subscribe-event <action> <type> <value> <freq>
wlan-reg-access <type> <offset> [value]
wlan-uapsd-enable <uapsd_enable>
wlan-uapsd-qosinfo <qos_info>
wlan-uapsd-sleep-period <sleep_period>
wlan-tx-ampdu-prot-mode <mode>
wlan-rssi-low-threshold <threshold_value>
wlan-rx-abort-cfg
wlan-set-rx-abort-cfg-ext enable <enable> margin <margin> ceil <ceil_thresh> floor <floor_thresh>
wlan-get-rx-abort-cfg-ext
wlan-cck-desense-cfg
wlan-net-monitor-cfg
wlan-set-monitor-filter <opt> <macaddr>
wlan-set-monitor-param <action> <monitor_activity> <filter_flags> <radio_type> <chan_number>
wlan-set-tsp-cfg <enable> <backoff> <highThreshold> <lowThreshold> <dutycycstep> <dutycycmin>
 <highthrtemp> <lowthrtemp>
wlan-get-tsp-cfg
wlan-get-signal
wlan-set-ips <option>
wlan-set-debug-htc <count> <vht> <he> <rxNss> <channelWidth> <ulMuDisable> <txNSTS> <erSuDisable>
 <erSuDisable> <erSuDisable>
wlan-enable-disable-htc <option>
wlan-set-su <0/1>
wlan-set-forceRTS <0/1>
wlan-set-mmsf <enable> <Density> <MMSF>
wlan-get-mmsf
wlan-get-turbo-mode <STA/UAP>
wlan-set-turbo-mode <STA/UAP> <mode>
wlan-set-multiple-dtim <value>
wlan-cloud-keep-alive <start/stop/reset>
wlan_tcp_client dst_ip <dst_ip> src_port <src_port> dst_port <dst_port>
wlan-set-country <country_code_str>
wlan-set-country-ie-ignore <0/1>
wlan-single-ant-duty-cycle <enable/disable> [<Ieee154Duration> <TotalDuration>]
wlan-dual-ant-duty-cycle <enable/disable> [<Ieee154Duration> <TotalDuration>
 <Ieee154FarRangeDuration>]
wlan-sta-inactivityto <n> <m> <l> [k] [j]
wlan-get-temperature
wlan-auto-null-tx <start/stop>
wlan-detect-ant <detect_mode> <ant_port_count> channel <channel> ...
wlan-get-txpwrlimit <subband>
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting> <autoTx_set>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
wlan-get-pmfcfg
wlan-uap-get-pmfcfg
wlan-set-ed-mac-mode <interface> <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
wlan-get-ed-mac-mode <interface>
wlan-set-tx-omi <interface> <tx-omi> <tx-option> <num_data_pkts>
```

```
wlan-set-toltime <value>
wlan-set-rutxpwrlimit
wlan-11ax-cfg <11ax_cfg>
wlan-11ax-bcast-twt <bcast_twt_cfg>
wlan-11ax-twt-setup <twt_cfg>
wlan-11ax-twt-teardown <twt_cfg>
wlan-11ax-twt-report <twt_report_get>
wlan-get-tsfinfo <format-type>
wlan-set-clocksync <mode> <role> <gpio_pin> <gpio_level> <pulse width>
wlan-suspend <power mode>
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6 address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
========================================
```

4.1.2.2 Help command

The help command is used to get the list of commands available in the *wifi_cli* sample application.

```
# help
help
clear
wlan-version
wlan-mac
wlan-thread-info
wlan-net-stats
wlan-set-mac <MAC_Address>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-connect-opt <profile_name> ...
wlan-reassociate
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-set-ps-cfg <null_pkt_interval>
wlan-deep-sleep-ps <0/1>
wlan-get-beacon-interval
wlan-wnm-ps <0/1> <sleep_interval>
wlan-set-max-clients-count <max clients count>
wlan-rts <sta/uap> <rts threshold>
wlan-frag <sta/uap> <fragment threshold>
wlan-host-11k-enable <0/1>
wlan-host-11k-neighbor-req [ssid <ssid>]
wlan-host-11v-bss-trans-query <0..16>
wlan-mbo-enable <0/1>
wlan-mbo-nonprefer-ch <ch0> <Preference0: 0/1/255> <ch1> <Preference1: 0/1/255>
wlan-sta-filter  <filter mode> [<mac address list>]
wlan-get-log <sta/uap> <ext>
wlan-tx-pert <0/1> <STA/UAP> <p> <r> <n>
wlan-roaming <0/1> <rssi_threshold>
wlan-multi-mef <ping/arp/multicast/del> [<action>]
wlan-wakeup-condition <mef/wowlan wake_up_conds>
wlan-auto-host-sleep <enable> <mode> <rtc_timer> <periodic>
wlan-send-hostcmd
wlan-ext-coex-uwb
wlan-set-uap-hidden-ssid <0/1/2>
wlan-eu-crypto-rc4 <EncDec>
wlan-eu-crypto-aes-wrap <EncDec>
wlan-eu-crypto-aes-ecb <EncDec>
wlan-eu-crypto-ccmp-128 <EncDec>
wlan-eu-crypto-ccmp-256 <EncDec>
wlan-eu-crypto-gcmp-128 <EncDec>
wlan-eu-crypto-gcmp-256 <EncDec>
wlan-mem-access <memory_address> [<value>]
wlan-eu-validation <value>
wlan-set-antcfg <ant_mode> <evaluate_time> <evaluate_mode>
wlan-get-antcfg
wlan-scan-channel-gap <channel_gap_value>
wlan-wmm-stat <bss_type>
wlan-reset
wlan-set-regioncode <region-code>
wlan-get-regioncode
wlan-11d-enable <sta/uap> <0/1>
wlan-uap-set-ecsa-cfg <block_tx> <oper_class> <new_channel> <switch_count> <bandwidth>
wlan-csi-cfg
wlan-set-csi-param-header  <csi_enable> <head_id> <tail_id> <chip_id> <band_config> <channel>
 <csi_monitor_enable> <ra4us>
wlan-set-csi-filter <opt> <macaddr> <pkt_type> <type> <flag>
wlan-txrx-histogram <action> <enable>
```

```
wlan-subscribe-event <action> <type> <value> <freq>
wlan-reg-access <type> <offset> [value]
wlan-uapsd-enable <uapsd_enable>
wlan-uapsd-qosinfo <qos_info>
wlan-uapsd-sleep-period <sleep_period>
wlan-tx-ampdu-prot-mode <mode>
wlan-rssi-low-threshold <threshold_value>
wlan-rx-abort-cfg
wlan-set-rx-abort-cfg-ext enable <enable> margin <margin> ceil <ceil_thresh> floor <floor_thresh>
wlan-get-rx-abort-cfg-ext
wlan-cck-desense-cfg
wlan-net-monitor-cfg
wlan-set-monitor-filter <opt> <macaddr>
wlan-set-monitor-param <action> <monitor_activity> <filter_flags> <radio_type> <chan_number>
wlan-set-tsp-cfg <enable> <backoff> <highThreshold> <lowThreshold> <dutycycstep> <dutycycmin>
 <highthrtemp> <lowthrtemp>
wlan-get-tsp-cfg
wlan-get-signal
wlan-set-ips <option>
wlan-set-debug-htc <count> <vht> <he> <rxNss> <channelWidth> <ulMuDisable> <txNSTS> <erSuDisable>
 <erSuDisable> <erSuDisable>
wlan-enable-disable-htc <option>
wlan-set-su <0/1>
wlan-set-forceRTS <0/1>
wlan-set-mmsf <enable> <Density> <MMSF>
wlan-get-mmsf
wlan-get-turbo-mode <STA/UAP>
wlan-set-turbo-mode <STA/UAP> <mode>
wlan-set-multiple-dtim <value>
wlan-cloud-keep-alive <start/stop/reset>
wlan_tcp_client dst_ip <dst_ip> src_port <src_port> dst_port <dst_port>
wlan-set-country <country_code_str>
wlan-set-country-ie-ignore <0/1>
wlan-single-ant-duty-cycle <enable/disable> [<Ieee154Duration> <TotalDuration>]
wlan-dual-ant-duty-cycle <enable/disable> [<Ieee154Duration> <TotalDuration>
 <Ieee154FarRangeDuration>]
wlan-sta-inactivityto <n> <m> <l> [k] [j]
wlan-get-temperature
wlan-auto-null-tx <start/stop>
wlan-detect-ant <detect_mode> <ant_port_count> channel <channel> ...
wlan-get-txpwrlimit <subband>
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting> <autoTx_set>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
wlan-get-pmfcfg
wlan-uap-get-pmfcfg
wlan-set-ed-mac-mode <interface> <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
wlan-get-ed-mac-mode <interface>
wlan-set-tx-omi <interface> <tx-omi> <tx-option> <num_data_pkts>
wlan-set-toltime <value>
wlan-set-rutxpwrlimit
wlan-11ax-cfg <11ax_cfg>
wlan-11ax-bcast-twt <bcast_twt_cfg>
wlan-11ax-twt-setup <twt_cfg>
wlan-11ax-twt-teardown <twt_cfg>
wlan-11ax-twt-report <twt_report_get>
wlan-get-tsfinfo <format-type>
wlan-set-clocksync <mode> <role> <gpio_pin> <gpio_level> <pulse width>
wlan-suspend <power mode>
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6 address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
```

4.1.2.3 Scan command

The scan command is used to scan the visible access points.

```
# wlan-scan
Scan scheduled...

# 1 network found:
38:E6:0A:C6:1A:EC  "nxp" Infra
        channel: 11
        rssi: -57 dBm
        security: WPA2
        WMM: YES

# wlan-scan-opt ssid nxp
Scan for ssid "nxp" scheduled...

# 1 network found:
38:E6:0A:C6:1A:EC  "nxp" Infra
        channel: 11
        rssi: -54 dBm
        security: WPA2
        WMM: YES
```

4.1.2.4 Add a network profile

Before adding a network profile for station (STA) and mobile AP (uAP) modes, check the command usage below.

```
# wlan-add
```

For station interface

- For DHCP IP address assignment:

```
 wlan-add <profile_name> ssid <ssid> [wpa2 <psk/psk-sha256> <secret>] [mfpc <1> mfpr <0>]
```

Note: *If using WPA2 security, set the PMF configuration if necessary as mentioned above.*

```
wlan-add <profile_name> ssid <ssid> [wpa3 sae <secret> [pwe <0/1/2>] mfpc <1> mfpr
 <0/1>]
```

Note: *If using WPA3 SAE security, always set the PMF configuration.*

```
wlan-add <profile_name> ssid <ssid> [wpa2 psk/psk-sha256 <secret> wpa3 sae <secret>]
 [mfpc <1> mfpr <0>]
```

Note: *If using WPA2/WPA3 mixed security, set the PMF configuration as mentioned above.*

- For static IP assignment:

```
wlan-add <profile_name> ssid <ssid>
ip:<ip_addr>,<gateway_ip>,<netmask>
[bssid <bssid>] [channel <channel number>]
[wpa2 <psk/psk-sha256> <secret>] [wpa3 sae <secret>] [mfpc <0/1> mfpr <0/1>]
```

For Micro-AP interface

```
wlan-add <profile_name> ssid <ssid>
ip:<ip_addr>,<gateway_ip>,<netmask>
role uap [bssid <bssid>]
[channel <channelnumber>]
[wpa2 <psk/psk-sha256> <secret>]/[wpa3 sae <secret> [pwe <0/1/2>] [tr <0/1>]]
[mfpc <0/1>] [mfpr <0/1>]
```

If setting `dtim`:

- The value of `dtim` is an integer.
- The default value is 10.

Note: *if UAP bandwidth is set to 80 MHz, setting the channel value greater than or equal to 36 is mandatory.*

```
[capa <11ax/11ac/11n/legacy>]
```

If `Set channel` value is 0, set `acs_band` to 0 1.

```
0: 2.4GHz channel 1: 5GHz channel - Not support to select dual band
```

4.1.2.5 Station mode (connect to AP)

WPA2 security

Use the following command to add the network profile to configure the device in station mode. Provide any profile name as well as use your AP SSID and passphrase in the argument as shown below:

```
# wlan-add abc ssid nxp wpa2 psk 1234567890
Added "abc"
```

Connect to the AP network using the saved network profile:

```
# wlan-connect abc
Connecting to network...
Use 'wlan-stat' for current connection status.
========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [nxp], IP = [192.168.43.35]
```

Note: *Once connected to the AP, the console output shows that the Client is connected to the AP with* `ssid = [nxp]` *and* `IP address = [192.168.43.35]` *from AP.*

WPA3 security

Use the following command to add the network profile to configure the device in station mode. Provide any profile name as well as use your AP SSID and passphrase in the argument as shown below:

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 mfpc 1 mfpr 1
Added "nxp_test_1"
```

Connect to the AP network using the saved network profile:

```
# wlan-connect nxp_test_1
Connecting to network...
Use 'wlan-stat' for current connection status.

========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [WPA3_AP], IP = [192.168.10.2]
```

Note: *Once connected to the AP, the console output shows that the Client is connected to AP with* `ssid = [WPA3_AP]` *and the* `IP address = [192.168.10.2]` *from the AP. For WPA3 R3, this configuration also works.*

4.1.2.6 Wpa2 station disconnection (from AP)

Disconnect from the AP network profile:

```
# wlan-disconnect
========================================
app_cb: WLAN: received event 9
========================================
app_cb: disconnected
```

Remove the saved network profile:

```
# wlan-remove abc
Removed "abc"
```

WPA3 security

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 mfpc 1 mfpr 1
Added "nxp_test_1"
```

Connect to the AP network using the saved network profile:

```
# wlan-connect nxp_test_1
Connecting to network...
Use 'wlan-stat' for current connection status.
========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [WPA3_AP], IP = [192.168.10.2]
```

Note: *Once connected to the AP, the console output shows the Client successfully connected to AP with* `SSID = [WPA3_AP]` *and* `IP = [192.168.10.2]` *from AP. For WPA3 R3, the above configuration also works.*

4.1.2.7 Wpa3 station disconnection (from AP)

Disconnect from the AP network profile:

```
# wlan-disconnect
========================================
app_cb: WLAN: received event 9
========================================
app_cb: disconnected
```

Remove the saved network profile:

```
# wlan-remove nxp_test_1
Removed "nxp_test_1"
```

4.1.2.8 Start the mobile AP

Use the following command to add the network profile to configure the device in AP mode. Use your AP SSID, IP details, role, channel, and security (passphrase if applicable) in the argument as shown below.

WPA2

```
# wlan-add xyz ssid NXPAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 6
 wpa2 psk 12345678
 Added "xyz"
```

WPA3

```
wlan-add xyz ssid NXPAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 6
 wpa3 sae 12345678 mfpc 1 mfpr 1
```

Note: *For WPA3 R3, the command is the same as for WPA3.*

Start the AP using saved network profile:

```
# wlan-start-network xyz
[wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is on.
========================================
app_cb: WLAN: received event 14
========================================
app_cb: WLAN: UAP Started
========================================
Mobile AP "NXPAP" started successfully
========================================
DHCP Server started successfully
========================================
```

Connect the wireless client to the AP just created, NXPAP. The logs below can be observed once the Client is associated successfully.

```
Client => 38:E6:0A:C6:1A:EC Associated with mobile AP
```

Get the associated clients list:

```
# wlan-get-uap-sta-list
Number of STA = 1
STA 1 information:
====================
MAC Address: 38:E6:0A:C6:1A:EC
Power mfg status: power save
Rssi : -58 dBm
```

Get the IP and MAC information for the associated clients:

```
# dhcp-stat
DHCP Server Lease Duration : 86400 seconds
Client IP       Client MAC
192.168.10.2    38:E6:0A:C6:1A:EC
```

4.1.2.9 Stop the mobile AP

```
# wlan-stop-network
========================================
app_cb: WLAN: received event 19
========================================
app_cb: WLAN: UAP Stopped
========================================
mobile AP "NXPAP" stopped successfully
========================================
DHCP Server stopped successfully
========================================
```

4.1.2.10 STA filter for mobile AP

- Enable uAP STA filtering, and add a MAC address to the allow-list

```
# wlan-sta-filter 1 F2:A5:1E:D1:AD:59
```

- Enable uAP STA filtering, and add a MAC address to the deny-list

```
# wlan-sta-filter 2 E8:F4:08:F8:27:76
```

- Disable STA filter

```
# wlan-sta-filter 0
```

4.1.2.11 iPerf server/client

The sample application implements the protocol used by iPerf performance measurement tool. The performance is measured between RW61x EVK board and a computer running the iPerf tool. The instructions in this guide use an RW61x EVK board. The following figures show the setup overview to run the iPerf performance test.

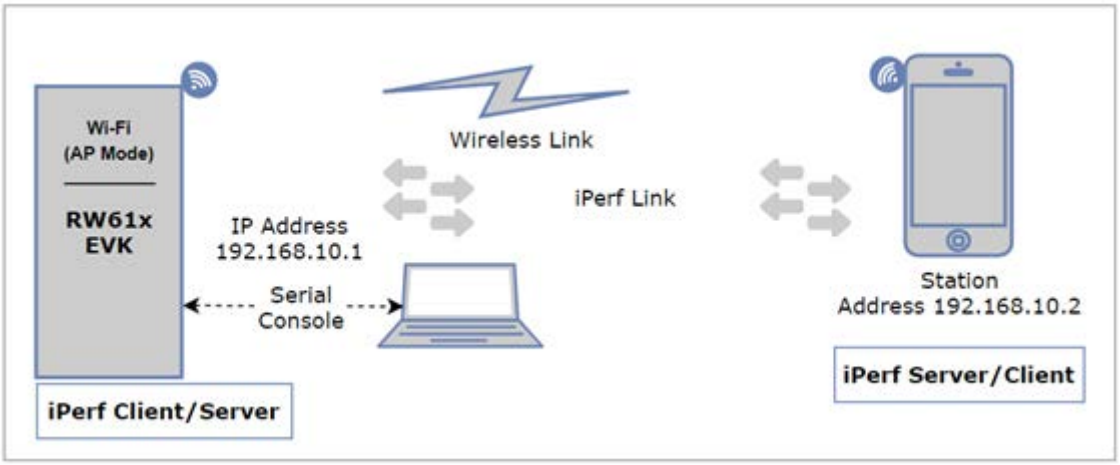


Figure 32. Hardware setup for iPerf performance test with mobile AP mode

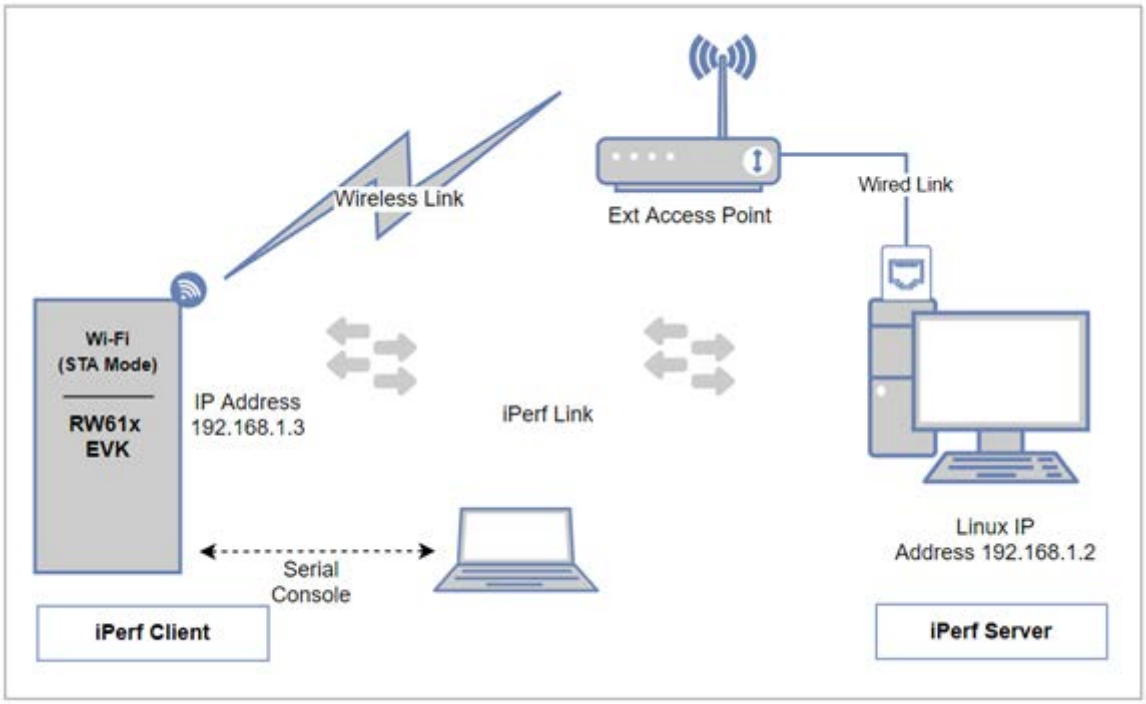


Figure 33. Hardware Setup for iPerf performance test with Station Mode

Note: *Refer to Section 2.3 for iperf remote host setup.*

The following commands are used for IPerf initialization:

IPerf usage:

```
# iperf
Incorrect usage
Usage:
        iperf [-s|-c <host>|-a] [options]
        iperf [-h]

        Client/Server:
           -u              use UDP rather than TCP
           -B    <host>    bind to <host> (including multicast address)
           -a              abort ongoing iperf session
        Server specific:
           -s              run in server mode
           -D              Do a bidirectional UDP test simultaneously and with -d from
 external iperf client
        Client specific:
           -c    <host>    run in client mode, connecting to <host>
           -d              Do a bidirectional test simultaneously
           -r              Do a bidirectional test individually
           -t    #         time in seconds to transmit for (default 10 secs)
           -b    #         for UDP, bandwidth to send at in Mbps, default 100Mbps without
 the parameter
           -S    #         QoS for udp traffic (default 0(Best Effort))
```

Note: *For iperf Windows, Linux and Mobile application commands refer to Table 1, Table 2, and Table 3 in Section 2.3.*

iPerf TCP

Start IPerf server:

```
# iperf -s
# IPERF initialization successful
New TCP client (settings flags 0x0)
------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 36874
 Bytes Transferred XXXX
 Duration (ms) 10130
 Bandwidth (Mbitpsec) XX
```

Start IPerf Client (TX Only):

```
# iperf -c 192.168.10.2
# IPERF initialization successful
------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 49153
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXX
 Duration (ms) 10001
 Bandwidth (Mbitpsec) XX
```

Start iPerf Client (TX and RX simultaneous):

```
# iperf -c 192.168.10.2 -d
IPERF initialization successful
New TCP client (settings flags 0x30313233)
------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 49154
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXX
 Duration (ms) 10001
 Bandwidth (Mbitpsec) XX
------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 36876
 Bytes Transferred XXXX
 Duration (ms) 10138
 Bandwidth (Mbitpsec) XX
```

Start iPerf Client (TX and RX individual):

```
# iperf -c 192.168.10.2 -r
# IPERF initialization successful
------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 49155
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXX
 Duration (ms) 10001
 Bandwidth (Mbitpsec) XX
New TCP client (settings flags 0x30313233)
------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 36878
 Bytes Transferred XXXX
 Duration (ms) 10095
 Bandwidth (Mbitpsec) XX
```

iPerf UDP

For UDP tests, specify the local interface IP address using -B option.

- Start iPerf server

```
# iperf -s -u -B 192.168.10.1
# IPERF initialization successful
New UDP client (settings flags 0x0)

Sending report back to client (0x80).

Jitter X.XXX,
Lost X/XXXX datagrams, OoO X
```

```
------------------------------------------------
 UDP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 54882
 Bytes Transferred XXXX
 Duration (ms) 10057
 Bandwidth (Mbitpsec) XX
```

- Start iPerf Client (TX only)

Note: *For UDP, indicate the bandwidth to send at in Mbps. The default value is 100 Mbps.*

```
# iperf -c 192.168.10.2 -u -B 192.168.10.1 -b 50
Ideal frame delay: 224 us
Send 4 frame(s) once per 1000 us
IPERF initialization successful

# Received report from server (0x80000000).

Jitter X.XXXX,
Lost XX/XXXX datagrams, OoO X
```

```
------------------------------------------------
 UDP_DONE_CLIENT (TX)
 Local address : 255.113.231.15  Port 49157
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXX
 Duration (ms) 10501
 Bandwidth (Mbitpsec) XX
```

4.1.2.12 Wi-Fi power save

The following commands are used to save Wi-Fi power in different power save modes.

IEEE power save

For IEEEPS mode Wi-Fi station should be connected with AP.

- IEEEPS usage

```
# wlan-ieee-ps
Usage: wlan-ieee-ps <0/1>
Error: Specify 0 to Disable or 1 to Enable
```

- Enable IEEEPS

```
# wlan-ieee-ps 1
Turned on IEEE Power Save mode
```

- Disable IEEEPS

```
# wlan-ieee-ps 0
Turned off IEEE Power Save mode
```

WMM power save

For WMM PS mode, the Wi-Fi station should be connected with the AP.

- WMM power save usage

```
# wlan-uapsd-enable
Usage: wlan-uapsd-enable <enable>
0 to Disable UAPSD
1 to Enable UAPSD
```

- Enable WMM power save

```
# wlan-uapsd-enable 1
```

- Disable WMM power save

```
# wlan-uapsd-enable 0
```

- Configure WMM power save sleeping period

```
# wlan-uapsd-sleep-period
Usage: wlan-uapsd-sleep-period <period(ms)>
# wlan-uapsd-sleep-period 30
```

WNM power save

- WNM power save usage

```
# wlan-wnm-ps
Usage: wlan-wnm-ps <0/1>
Error: Specify 0 to Disable or 1 to Enable
If enable, please specify sleep_interval
Example:
      wlan-wnm-ps 1 5
```

- Enable WNM power save

```
# wlan-wnm-ps 1 5
Turned on WNM Power Save mode
```

- Disable WNM power save

```
# wlan-wnm-ps 0
Turned off WNM Power Save mode
```

Deep sleep

For deep Sleep mode, Wi-Fi should be in disconnected state otherwise it does not enable the deep sleep.

- Check the Wi-Fi connection

```
# wlan-info
Station not connected
uAP not started
```

- Deep sleep usage

```
# wlan-deep-sleep-ps
Usage: wlan-deep-sleep-ps <0/1>
Error: Specify 0 to Disable or 1 to Enable
```

- Enable deep sleep

```
# wlan-deep-sleep-ps 1
Turned on Deep Sleep Power Save mode
```

- Disable deep sleep

```
# wlan-deep-sleep-ps 0
Turned off Deep Sleep Power Save mode
```

4.1.2.13 Host sleep

The following command is used to configure host sleep parameters and put host MCU into Sleep mode PM2.

- `wlan-auto-host-sleep` command usage

```
# wlan-auto-host-sleep
Usage:
wlan-auto-host-sleep <enable> <mode> <rtc_timeout> <periodic>
  enable      -- enable/disable host sleep
  0 - disable host sleep
  1 - enable host sleep
  mode        -- Mode of how host enter low power.
  manual - Manual mode. Need to use suspend command to enter low power.
  pm     - Power Manager.
  rtc_timeout -- RTC timer value. Unit is second.
  periodic    -- Host enter low power periodically or oneshot
  0 - Oneshot. Host will enter low power only once and keep full power after waking up.
  1 - Periodic. Host will enter low power periodically.
  Parameters <rtc_timeout> and <periodic> are for Power Manager ONLY!
    Examples:
    wlan-auto-host-sleep 1 pm 60 1
    wlan-auto-host-sleep 1 pm 5 0
    wlan-auto-host-sleep 1 manual
    wlan-auto-host-sleep 0
```

- Disable host sleep

```
# wlan-auto-host-sleep 0
Host Sleep disabled
```

- Host sleep using manual mode

```
# wlan-auto-host-sleep 1 manual
Manual mode is selected for host sleep
```

Note: *Use with the command* `suspend` *(Section 4.1.2.14).*

- Host sleep using power manager

RTC timer timeout value is 10 seconds, and the host enters low power mode only one time:

```
# wlan-auto-host-sleep 1 pm 10 0
Power Manager is selected for host sleep
Host will enter low power only once

# Enter low power mode PM2
```

RTC timer timeout value is 10 seconds, and the host enters low power periodically:

```
# wlan-auto-host-sleep 1 pm 10 1
Power Manager is selected for host sleep
Host will enter low power periodically

# Enter low power mode PM2
Exit low power mode
Woken up by RTC
Enter low power mode PM2
Exit low power mode
Woken up by RTC
Enter low power mode PM2
Exit low power mode
Woken up by RTC
Enter low power mode PM2
Exit low power mode
Woken up by RTC
```

Note: *For periodic host sleep, CPU3 keeps full power for 5 seconds after each wake-up. During this time, the user is allowed to issue other commands.*

If the command `wlan-wakeup-condition` *is never issued, the wake-up condition for Wi-Fi wake-up source is* `wlan-wakeup-condition wowlan 0x0`

4.1.2.14 Suspend

The `wlan-suspend` command is used to put manually the host MCU into a different power mode.

- Command usage:

```
# wlan-suspend
Usage:
     wlan-suspend <power mode>
     1:PM1 2:PM2 3:PM3 4:PM4
Example:
     wlan-suspend 3
```

Note: *If you use the command* `wlan-host-sleep` *to put the host to sleep manually, use* `wlan-suspend` *command to put the host to the targeted low power mode.*

4.1.2.15 Wake-up conditions

The `wlan-wakeup-condition` command is used to configure Wi-Fi wake-up conditions. Set up an STA connection or start the uAP accordingly before using the command.

- Command usage:

```
# wlan-wakeup-condition
Usage:
   wlan-wakeup-condition <wowlan [wake_up_conds]/mef>
   wowlan -- default host wakeup
   [wake_up_conds] -- value for wowlan host wakeup conditions only
        bit 0: WAKE_ON_ALL_BROADCAST
        bit 1: WAKE_ON_UNICAST
        bit 2: WAKE_ON_MAC_EVENT
        bit 3: WAKE_ON_MULTICAST
        bit 4: WAKE_ON_ARP_BROADCAST
        bit 6: WAKE_ON_MGMT_FRAME
        All bit 0 discard and not wakeup host
   mef     -- MEF host wakeup
Example:
    wlan-wakeup-condition mef
    wlan-wakeup-condition wowlan 0x1e
```

- Default host wake-up:

```
# wlan-wakeup-condition wowlan 0x1e
```

- MEF wake-up:

```
# wlan-wakeup-condition mef
No user configured MEF entries, use default ARP filters
```

Note:

- *Do not add wake-up conditions for MEF host wake-up. The method is ONLY for wowlan wakeup.*
- *Use the command `wlan-multi-mef` (Section 4.1.2.16) to configure MEF entries for MEF host wake-up. If the MEF entry is not configured, the driver uses the default MEF entry as MEF wake-up condition, that is broadcast or unicast ARP packet.*

4.1.2.16 Multi MEF configuration

The command is used to configure multiple MEF entries. Use `wlan-multi-mef` command with `wlan-host-sleep mef` command to set MEF wake-up conditions.

- Command usage:

```
# wlan-multi-mef
Usage:
    wlan-multi-mef <ping/arp/multicast/ns/del> [<action>]
      ping/arp/multicast/ns
                -- MEF entry type, will add one mef entry at a time
      del       -- Delete all previous MEF entries
      action    -- 0--discard and not wake host
                   1--discard and wake host
                   3--allow and wake host
Example:
    wlan-multi-mef ping 3
    wlan-multi-mef del
```

- Ping MEF entry:

```
# wlan-multi-mef ping 3
Add ping MEF entry successful
```

- Delete all MEF entries:

```
# wlan-multi-mef del
delete all MEF entries Successful
```

4.1.2.17 Wi-Fi reset

The following command is used to enable, disable, and reset Wi-Fi.

```
# wlan-reset
Usage: wlan-reset <options>
0 to Disable WiFi
1 to Enable WiFi
2 to Reset WiFi

# wlan-reset 0
--- Disable WiFi ---
--- Done ---

# wlan-reset 1
--- Enable WiFi ---
Initialize WLAN Driver
MAC Address: C0:95:DA:00:C0:45
--- Done ---

# wlan-reset 2
--- Disable WiFi ---
--- Enable WiFi ---
Initialize WLAN Driver
MAC Address: C0:95:DA:00:C0:45
--- Done ---
# ========================================
app_cb: WLAN: received event 11
========================================
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
ENHANCED WLAN CLIs are initialized
========================================
CLIs Available:
========================================

Help
……
```

4.1.2.18 802.11k commands

The following commands are used to enable 802.11k and send an 802.11k neighbor request.

- Enable 802.11k

```
# wlan-host-11k-enable
Usage: wlan-host-11k-enable <0/1> < 0--disable host 11k; 1---enable host 11k>
# wlan-11k-host-enable 1
```

- Send an 802.11k neighbor request after STA connection

```
# wlan-host-11k-neigbor-req
```

4.1.2.19 802.11d commands

The following command is used to enable 802.11d.

- Enable 802.11d

```
# wlan-11d-enable
Usage:
wlan-11d-enable <sta/uap> <0/1>, 0: disable, 1: enable
This command is only used to enable/disable 11D
Please use wlan-set-regioncode command to set region
```

4.1.2.20 Roaming commands

The following commands are used to enable Wi-Fi roaming.

- Enable roaming

Note: *The command `wlan-roaming` is used to configure roaming. One condition to trigger roaming is `rssi_low`.*

```
# wlan-roaming
Usage:
   wlan-roaming <0/1> rssi_low <rssi_threshold>
rssi_low is optional. Use default value 70 if not provided
Example:
   wlan-roaming 1 rssi_low 70

# wlan-roaming 1 rssi_low 70
```

If the current BSS RSSI is lower than the preconfigured threshold, RW61x STA switches to another BSS with better RSSI. When roaming occurs, the following message is printed.

```
=========================================
app_cb: WLAN: received event 0
=========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [test-2g], IP = [192.168.3.137]
```

- Disable roaming

```
# wlan-roaming 0
```

4.1.2.21 CSI commands

Refer to [1].

4.1.2.22 Net monitor commands

The following commands are used to configure net monitor.

- Configure net monitor parameters

```
# wlan-set-monitor-param
Usage             : wlan-set-monitor-param <action> <monitor_activity> <filter_flags> <radio_type>
 <chan_number>
action            : 0/1 to Action Get/Set
monitor_activity  : 1 to enable and other parameters to disable monitor activity
filter_flags      : network monitor fitler flag
chan_number       : channel to monitor

Usage example :
wlan-set-monitor-param 1 1 7 0 1

current parameters:
action            : 1
monitor_activity  : 1
filter_flags      : 7
radio_type        : 0
chan_number       : 1
filter_num        : 1
mac_addr      : 64:64:4A:D6:FA:7B
```

- Configure net monitor filter

```
# wlan-set-monitor-filter
Usage : wlan-set-monitor-filter <opt< <macaddr>
opt   : add/delete/clear/dump
add   : All options need to be filled in
delete: Delete recent mac addr
clear : Clear all mac addr
dump  : Dump monitor cfg information
Usage example
wlan-set-monitor-filter add 64:64:4A:D6:FA:7B
wlan-set-monitor-filter delete
wlan-set-monitor-filter clear
wlan-set-monitor-filter dump
```

- Apply net monitor configuration

```
# wlan-net-monitor-cfg
```

Net monitor data is not dumped to the console by default. Users must register a callback to receive these data in their application.

```
int net_monitor_data_recv_test(void *buffer, t_u16 data_len)
{
   for(int i =0 ; i < data_len; i++)
   {
      if(i % 16 == 0)
      {
        (void)PRINTF("\r\n");
      }
      (void)PRINTF("%02X ", *((t_u8 *)buffer + i));
   }
   return WM_SUCCESS;
}

wlan_register_monitor_user_callback(net_monitor_data_recv_test)
```

4.1.2.23 ECSA command

The following command is used to configure mobile AP ECSA.

```
# wlan-uap-set-ecsa-cfg
Usage        : wlan-uap-set-ecsa-cfg <block_tx> <oper_class> <new_channel> <switch_count>
 <bandwidth>
block_tx     : 0 -- no need to block traffic, 1 -- need block traffic
oper_class   : Operating class according to IEEE std802.11 spec
new_channel  : The channel will switch to
switch count : Channel switch time to send ECSA ie
bandwidth    : Channel width switch to(optional),RW610 only support 20M channels

Usage example :
TLV   : wlan-uap-set-ecsa-cfg 1 0 36 10 1
Action: wlan-uap-set-ecsa-cfg 1 115 36 0

# wlan-uap-set-ecsa-cfg 1 0 36 10 1
uap switch to channel 36 success!
```

4.1.2.24 EU crypto commands

The following commands are used to encrypt and decrypt preset sample data using AES-WRAP algorithm.

- Command usage

```
# wlan-eu-crypto
Usage:
Algorithm AES-WRAP encryption and decryption verification
wlan-eu-crypto <EncDec>
EncDec: 0-Decrypt, 1-Encrypt
```

- Encrypt sample data

```
# wlan-eu-crypto 1
Raw Data:
**** Dump @ 2002DE60 Len: 16 ****
12 34 56 78 90 12 34 56 78 90 12 34 56 78 90 12
******** End Dump *******
Encrypted Data:
**** Dump @ 2002DE98 Len: 24 ****
fa da 96 53 30 97 4b 61 77 c6 d4 3c d2 0e 1f 6d
43 8a 0a 1c 4f 6a 1a d7
******** End Dump *******
```

- Decrypt sample data

```
# wlan-eu-crypto 0
Raw Data:
**** Dump @ 2002DE80 Len: 24 ****
fa da 96 53 30 97 4b 61 77 c6 d4 3c d2 0e 1f 6d
43 8a 0a 1c 4f 6a 1a d7
******** End Dump *******
Decrypted Data:
**** Dump @ 2002DE98 Len: 16 ****
12 34 56 78 90 12 34 56 78 90 12 34 56 78 90 12
******** End Dump *******
```

Note: *Encryption and decryption sample data is in the function available at*

[SDK]\middleware\wifi_nxp\wlcmgr\wlan_test.c\test_wlan_eu_crypto

.

4.1.2.25 Set/get the antenna configuration

Commands to set or get the antenna configurations:

```
 # wlan-set-antcfg
Usage:
wlan-set-antcfg <ant_mode> <evaluate_time> <evaluate_mode>
        <ant_mode>:
                  Bit 0   -- Fixed to Tx/Rx antenna 1
                  Bit 1   -- Fixed to Tx/Rx antenna 2
                  0xFFFF  -- enable Tx/Rx antenna diversity
        [evaluate_time]:
                  If ant mode = 0xFFFF, use this to configure
                  SAD evaluate time interval in milli seconds unit.
                  If not specified, default value is 6000 milli seconds
        <evaluate_mode>:
                  0: PCB Ant. + Ext Ant0
                  1: Ext Ant0 + Ext Ant1
                  2: PCB Ant. + Ext Ant1
Examples:
wlan-set-antcfg 1
wlan-set-antcfg 0xffff
wlan-set-antcfg 0xffff 5000
wlan-set-antcfg 0xffff 6000 0
```

- Set the antenna configuration to use fixed antenna 1:

```
# wlan-set-antcfg 1
```

- Enable SAD with default evaluate time 6 s:

```
# wlan-set-antcfg 0xffff
```

- Enable SAD with evaluate mode configured:

```
# wlan-set-antcfg 0xffff 6000 0
```

- Get the antenna configuration:

```
# wlan-get-antcfg
Mode of Tx/Rx path is : 1
Current antenna is 1
```

4.1.2.26 Other useful CLI commands

Use the other commands to get the Wi-Fi information, driver version, firmware version, list of the networks and other information.

- Get the Wi-Fi information

```
# wlan-info
Station connected to:
"abc"
        SSID: nxp
        BSSID: 6E:C7:EC:33:A0:D0
        channel: 1
        role: Infra
        security: WPA2

        IPv4 Address
        address: DHCP
                IP:             192.168.43.113
                gateway:        192.168.43.1
                netmask:        255.255.255.0
                dns1:           192.168.43.1
                dns2:           0.0.0.0

        rssi threshold: 0
uAP started as:
"xyz"
        SSID: NXPAP
        BSSID: C0:95:DA:00:D5:0F
        channel: 1
        role: uAP
        security: WPA2
        wifi capability: 11ax
        user configure: 11ax

        IPv4 Address
        address: STATIC
                IP:             192.168.10.1
                gateway:        192.168.10.1
                netmask:        255.255.255.0
                dns1:           192.168.43.1
                dns2:           0.0.0.0

        rssi threshold: 0
```

- Get the Wi-Fi driver and firmware version

```
# wlan-version
WLAN Driver Version   : vX.X.rXX.pX
WLAN Firmware Version : rw610w-V0, RF878X, FP91, 18.91.1.p102, PVE_FIX 1, RF878X, FP91,
 18.91.1.p102
```

- Set the Wi-Fi MAC address

```
# wlan-set-mac C0:95:DA:00:D5:0F
STA MAC Address: C0:95:DA:00:D5:0F
uAP MAC Address: C0:95:DA:00:D6:0F
```

- Get the Wi-Fi MAC address

```
# wlan-mac
MAC address
STA MAC Address: C0:95:DA:00:D5:0F
UAP MAC Address: C0:95:DA:00:D5:0F
```

- Get the list of Wi-Fi networks

```
# wlan-list
2 networks:
"abc"
        SSID: nxp
        BSSID: 00:00:00:00:00:00
        channel: (Auto)
        role: Infra
        security: WPA2

        IPv4 Address
        address: DHCP
                IP:             0.0.0.0
                gateway:        0.0.0.0
                netmask:        0.0.0.0
                dns1:           0.0.0.0
                dns2:           0.0.0.0

        rssi threshold: 0
"xyz"
        SSID: NXPAP
        BSSID: 00:00:00:00:00:00
        channel: (Auto)
        role: uAP
        security: WPA2
        wifi capability: 11ax
        user configure: 11ax

        IPv4 Address
        address: STATIC
                IP:             192.168.10.1
                gateway:        192.168.10.1
                netmask:        255.255.255.0
                dns1:           192.168.43.1
                dns2:           0.0.0.0

        rssi threshold: 0
```

- Get the Wi-Fi state

```
# wlan-stat
Station connected (Active)
uAP started (Active)
```

- Get the Wi-Fi IP address

```
# wlan-address

      IPv4 Address
      address: DHCP
         IP:             192.168.3.20
         gateway:        192.168.3.1
         netmask:        255.255.255.0
         dns1:           192.168.3.1
         dns2:           0.0.0.0

      IPv6 Addresses
      Link-Local    :  FE80::C295:DAFF:FE00:D560 (Preferred)
```

- Get the mobile AP channel

```
# wlan-get-uap-channel
uAP channel: 6
```

- Set the maximum station count for the mobile AP

```
# wlan-set-max-clients-count
Usage: wlan-set-max-clients-count  max_clients_count
```

- Ping the IP address

```
# ping
Incorrect usage
Usage:
        ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ip_address>
Default values:
        packet_size: 56
        packet_count: 10
        timeout: 2 sec
# ping -s 56 -c 2 -W 2 192.168.43.1
PING 192.168.43.1 (192.168.43.1) 56(84) bytes of data
64 bytes from 192.168.43.1: icmp_req=1 ttl=64 time=196 ms
64 bytes from 192.168.43.1: icmp_req=2 ttl=64 time=95 ms
```

- Configure Wi-Fi RTS threshold

```
# wlan-rts
Usage: wlan-rts <sta/uap> <rts threshold>
```

- Configure Wi-Fi fragment threshold

```
# wlan-frag
Usage: wlan-frag <sta/uap> <fragment threshold>
```

- Configure hidden SSID

```
# wlan-set-uap-hidden-ssid
Usage: wlan-set-uap-hidden-ssid <0/1/2>
0: broadcast SSIS in beacons.
1: send empty SSID (length=0) in beacons.
2: clear SSID (ACSII 0), but keep the original length.
```

- Configure TX PER setting

```
# wlan-tx-pert
Usage:
    wlan-tx-pert <0/1> <STA/AP> <p:tx_pert_check_period> <r:tx_pert_check_ratio>
 <n:tx_pert_check_num>
Example:
    wlan-tx-pert 1 AP 5 3 5
```

- Get the Wi-Fi STA and mobile AP log

```
# wlan-get-log
Usage: wlan-get-log <sta/uap> <ext>

# wlan-get-log sta
dot11GroupTransmittedFrameCount    9
dot11FailedCount                   9
dot11RetryCount                    15
dot11MultipleRetryCount            13
dot11FrameDuplicateCount           0
dot11RTSSuccessCount               7
dot11RTSFailureCount               13
dot11ACKFailureCount               132
dot11ReceivedFragmentCount         24
dot11GroupReceivedFrameCount       7
dot11FCSErrorCount                 5
dot11TransmittedFrameCount         20
wepicverrcnt-1                     2517765259
wepicverrcnt-2                     0
wepicverrcnt-3                     0
wepicverrcnt-4                     0
beaconReceivedCount                0
beaconMissedCount                  2311
dot11TransmittedFragmentCount      0
dot11QosTransmittedFragmentCount   0 0 0 0 10 8 0 0
dot11QosFailedCount                0 0 0 0 1 8 0 0
dot11QosRetryCount                 0 0 0 0 7 7 0 0
dot11QosMultipleRetryCount         0 0 0 0 6 3 0 0
dot11QosFrameDuplicateCount        0 0 0 0 2 7 0 0
dot11QosRTSSuccessCount            0 0 0 0 0 25 0 0
dot11QosRTSFailureCount            0 0 0 0 0 89 0 0
dot11QosACKFailureCount            0 0 0 0 43 10 0 0
dot11QosReceivedFragmentCount      0 0 0 0 6 10 0 0
dot11QosTransmittedFrameCount      0 0 0 0 10 11 0 0
dot11QosDiscardedFrameCount        0 0 0 0 3 10 0 0
dot11QosMPDUsReceivedCount         0 0 0 0 6 5 0 0
dot11QosRetriesReceivedCount       0 0 0 0 3 0 0 0
dot11RSNAStatsCMACICVErrors          0
dot11RSNAStatsCMACReplays            0
dot11RSNAStatsRobustMgmtCCMPReplays  0
dot11RSNAStatsTKIPICVErrors          0
dot11RSNAStatsTKIPReplays            0
dot11RSNAStatsCCMPDecryptErrors      0
dot11RSNAstatsCCMPReplays            0
dot11TransmittedAMSDUCount           0
dot11FailedAMSDUCount                0
dot11RetryAMSDUCount                 0
dot11MultipleRetryAMSDUCount         0
dot11TransmittedOctetsInAMSDUCount   0
dot11AMSDUAckFailureCount            0
dot11ReceivedAMSDUCount              99
dot11ReceivedOctetsInAMSDUCount      99
dot11TransmittedAMPDUCount           0
dot11TransmittedMPDUsInAMPDUCount    2441
dot11TransmittedOctetsInAMPDUCount   529015416818064
dot11AMPDUReceivedCount              0
dot11MPDUInReceivedAMPDUCount        246
dot11ReceivedOctetsInAMPDUCount      0
dot11AMPDUDelimiterCRCErrorCount     0
```

- Get WMM TX statistics

```
# wlan-wmm-stat
1641493: [wifi] Warn: Dump priv[0] ac_queue[0]
1641497: [wifi] Warn: Dump priv[0] ac_queue[1]
1641501: [wifi] Warn: Dump priv[0] ac_queue[2]
1641505: [wifi] Warn: Dump priv[0] ac_queue[3]
1641510: [wifi] Warn: Dump priv[0] driver_error_cnt:
1641515: [wifi] Warn:     tx_no_media[0]
1641518: [wifi] Warn:     tx_err_mem[0]
1641522: [wifi] Warn:     tx_wmm_retried_drop[0]
1641526: [wifi] Warn:     tx_wmm_pause_drop[0]
1641531: [wifi] Warn:     tx_wmm_pause_replaced[0]
1641535: [wifi] Warn:     rx_reorder_drop[0]
1641539: [wifi] Warn: TX buffer pool: free_cnt[32] real_free_cnt[32]
```

4.1.3 Add commands to the wifi_cli sample application

User-definable commands can be called using CLI wrappers with the appropriate arguments. The new CLI command can be added in the existing demo application by using the existing structure that defines the list of commands. Command-line arguments can be passed based on the API requirement.

In the following example, a new command with arguments is added.

Command structure modification:

File: *wlan_tests.c* or *wlan_basic_cli.c*

Structure elements: {“command-name”, “help”, handler}

```
{"wlan-command-name", "<argument1> <argument2> <argument3>…", handler_wlan_command},
```

Command handler: `void handler_wlan_command (int argc, char *argv[])`

Store the input arg list and pass it to the relative APIs to be used by the driver/firmware.

The return value of API can be used to print the Error/Success message and command output.

```
void handler_wlan_command (int argc, char *argv[])
{
    /* argv contains pointer to the arguments and argc is the number of arguments */
    return_value = wlan_command_driver_API(argument1, argument2, argument3,…);
    if (return_value == WM_SUCCESS) {
     /* Print success message and command output */
    } else {
     /* Print failure message and error number */
      }
}
```

4.2 wifi_webconfig sample application

This section describes *wifi_webconfig* sample application and its configuration along with the application execution. The *wifi_webconfig* sample application uses the uAP feature with an HTTP server to configure the Client mode and connect to an AP.

A simple LED control is implemented to check the operational mode. The LED is on if the device is in AP mode and it turns off after device is set to client mode.

The website in AP mode shows the available networks using scan. The desired network can be chosen by clicking the listed SSID. Once SSID and passphrase are entered and posted, the device attempts to connect to the chosen network with the given configuration.

The Wi-Fi credentials are stored in *mflash*, so the device can connect to the network after a reboot. Once the device comes up with the client mode, the AP mode goes down, and so the website closes down.

The website allows the user to reset the device to AP mode.

The following figure shows the logical flow diagram of the *wifi_webconfig* sample application.

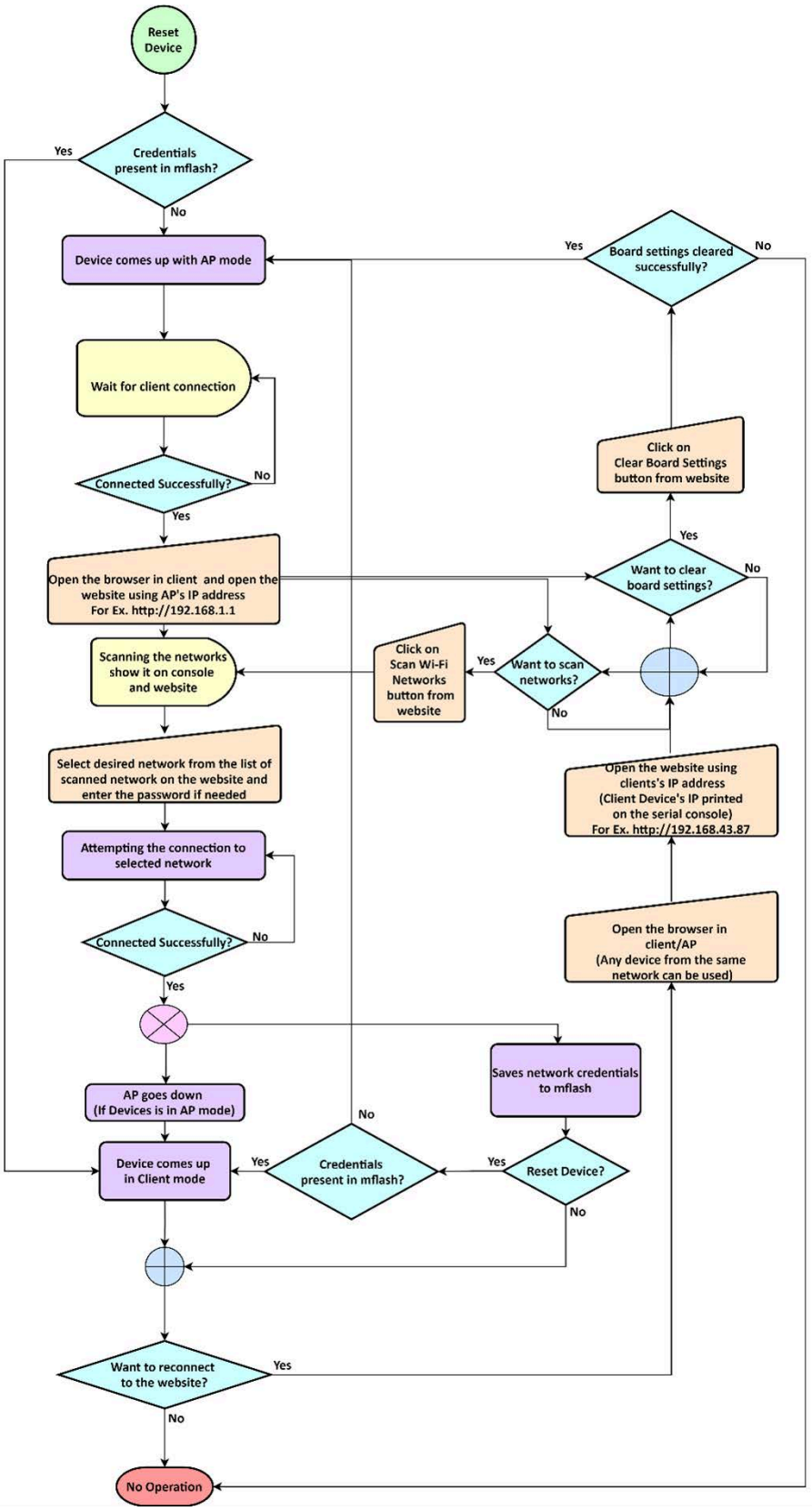


Figure 34. wifi_webconfig flow diagram

The *wifi_webconfig* application features are summarized in the table below.

Table 5. wifi_webconfig sample application features

Features	Details
Wi-Fi and HTTP	Wi-Fi Mobile AP mode Wi-Fi Station mode Wi-Fi Security (WPA2 by default for Mobile AP) Desired Channel Selection for AP HTTP server (Request GET/POST) DHCP Server/Client

4.2.1 User configurations

Table 6 lists the Wi-Fi features and feature-related macros that the user can configure.

Wi-Fi configurations

Table 6. wifi_webconfig application Wi-Fi configurations

Feature	Macro definition	Default value	File name	Details
Wi-Fi mobile AP	WIFI_SSID	“nxp_configuration_ access_point”	webconfig.h	Default SSID and passphrase to start mobile AP using the given sample application. It can be modified by changing the macro value. Default wpa2 security is used.
	WIFI_PASSWORD	“NXP0123456789”		
	WIFI_AP_CHANNEL	1		
	WIFI_AP_IP_ADDR	“192.168.1.1”		
	WIFI_AP_NET_MASK	“255.255.0.0”		

4.2.2 wifi_webconfig application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

Refer to Section 2.1 for information about the serial console setup.

4.2.2.1 Start-up logs

The following logs can be observed on the console once the RW61x EVK is up and running.

```
Starting webconfig DEMO
[i] Trying to load data from mflash.
[i] Nothing stored yet
[i] Initializing Wi-Fi connection...
MAC Address: C0:95:DA:00:D5:0F
821: [net] Initialized TCP/IP networking stack
[i] Successfully initialized Wi-Fi module
Starting Access Point: SSID: nxp_configuration_access_point, Chnl: 1
841: [wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is on.
 Now join that network on your device and connect to this IP: 192.168.1.1
```

4.2.2.2 Connect the client to the mobile AP

Connect the client to the mobile AP and observe the logs with the client MAC address.

```
Client => 0E:C4:21:F6:37:24 Associated with mobile AP
```

4.2.2.3 Open the website in the client web browser

Use the AP IP-192.168.1.1 open website in the client browser. Opening the website triggers the scan in the device and the available wireless networks are listed in the console and webpage. The current Wi-Fi mode AP is highlighted on the webpage. See Figure 35.

```
Initiating scan...
Galaxy M210997
     BSSID          : 8A:A3:03:B3:09:97
     RSSI           : -86dBm
     Channel        : 2
nxp
     BSSID          : 38:E6:0A:C6:1A:EC
     RSSI           : -90dBm
     Channel        : 165
```

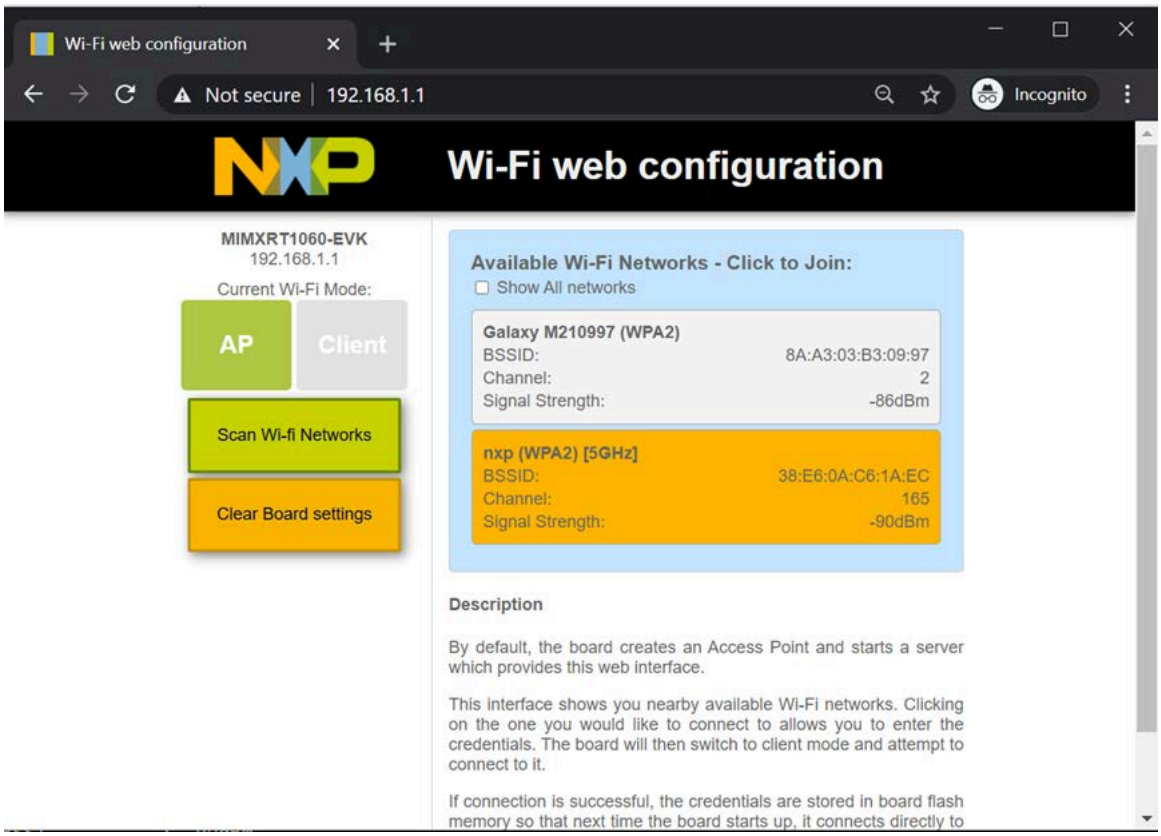


Figure 35. wifi_webconfig website in AP mode

4.2.2.4 Connect the device to the AP

Click the desired SSID on the webpage. If the AP uses Wi-Fi security, a dialog box opens and asks to enter a password. Once the credentials are posted, the device attempts the connection to the AP.

```
[i] Chosen ssid: nxp
[i] Chosen passphrase: "12345678"
[i] Joining: nxp
Switch to channel 165 success!
[i] Successfully joined: nxp
 Now join that network on your device and connect to this IP: 192.168.43.35
[i] mflash_save_file success
[i] Stopping AP!
```

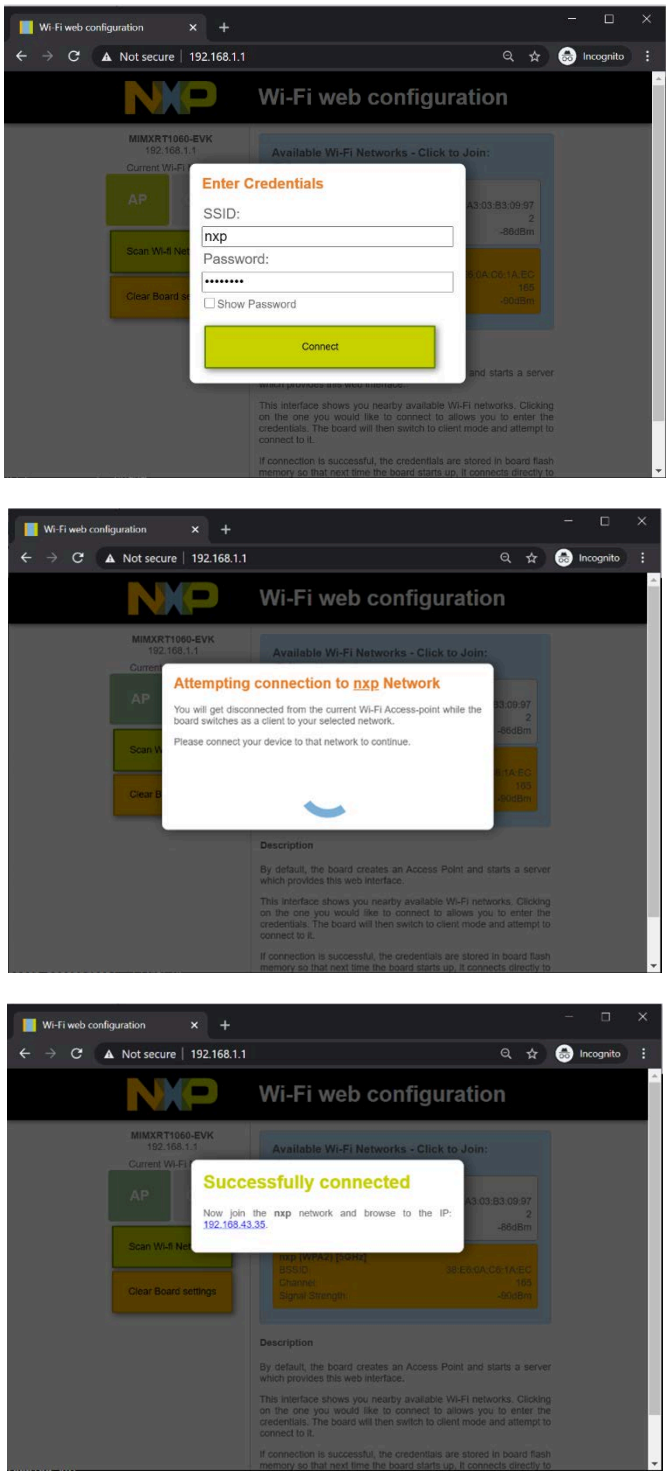


Figure 36. Connection attempt to AP using wifi_webconfig application

Note: *When the device has received by the configurations, the mobile AP goes down and the device switches to Client mode. To reconnect to the website, switch to the AP network and use the device (Client mode) IP (printed on the console) to open the website.*

For example, Figure 37 shows the attempt to reconnect to website.

The current Wi-Fi mode client is highlighted on the webpage shown in Figure 37.

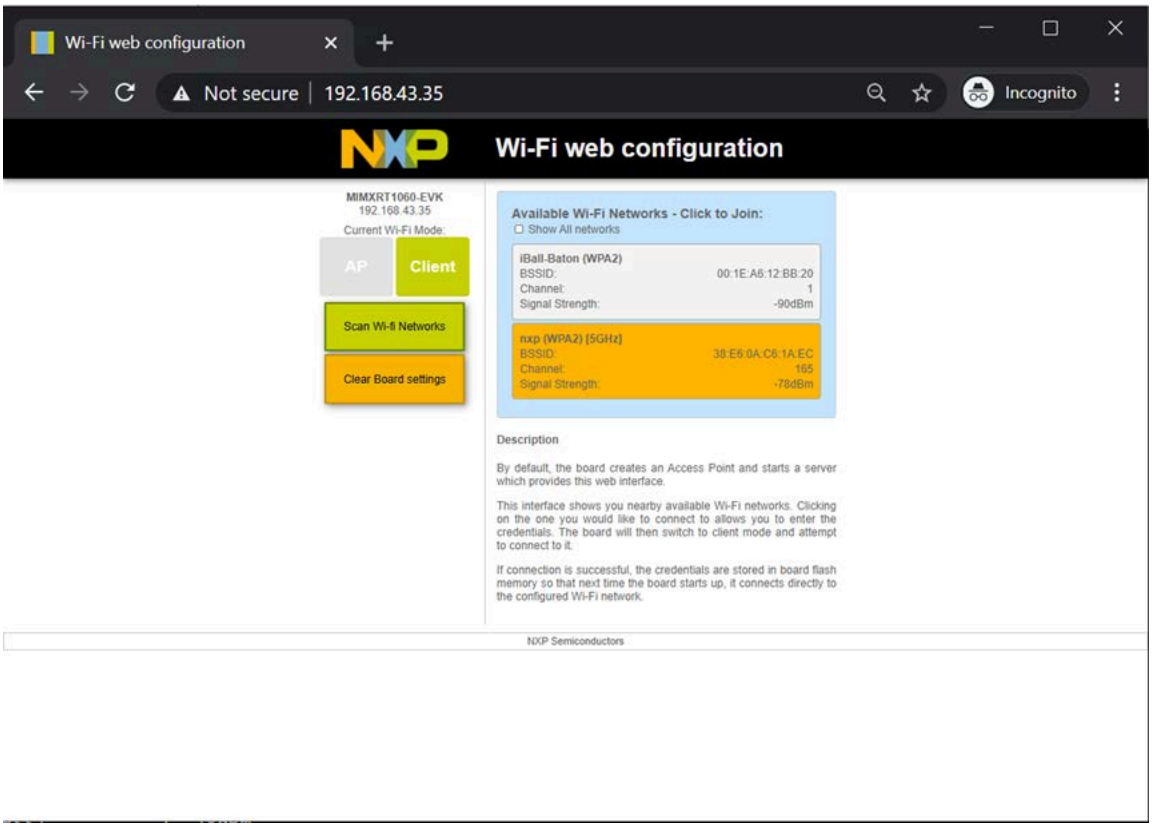


Figure 37. wifi_webconfig website in client mode

4.2.2.5 Device reboot with the configurations stored in mflash

The following logs can be observed when the device has the client configuration saved in *mflash*. It reads the stored information and uses it to configure client mode after a reboot.

```
MAC Address: C0:95:DA:00:D5:0F
818: [net] Initialized TCP/IP networking stack
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: nxp with password 12345678
[i] Connected to Wi-Fi
ssid: nxp
[!]passphrase: 12345678
 Now join that network on your device and connect to this IP: 192.168.43.35
```

4.2.2.6 Clear the settings on the website

To clear the configurations saved in *mflash*, click the **Clear Board settings** button available on the web page.

```
[i] mflash_save_file success
Starting Access Point: SSID: nxp_configuration_access_point, Chnl: 1
144614: [wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is
 on.
Now join that network on your device and connect to this IP: 192.168.1.1
```

Figure 38. Clear board settings

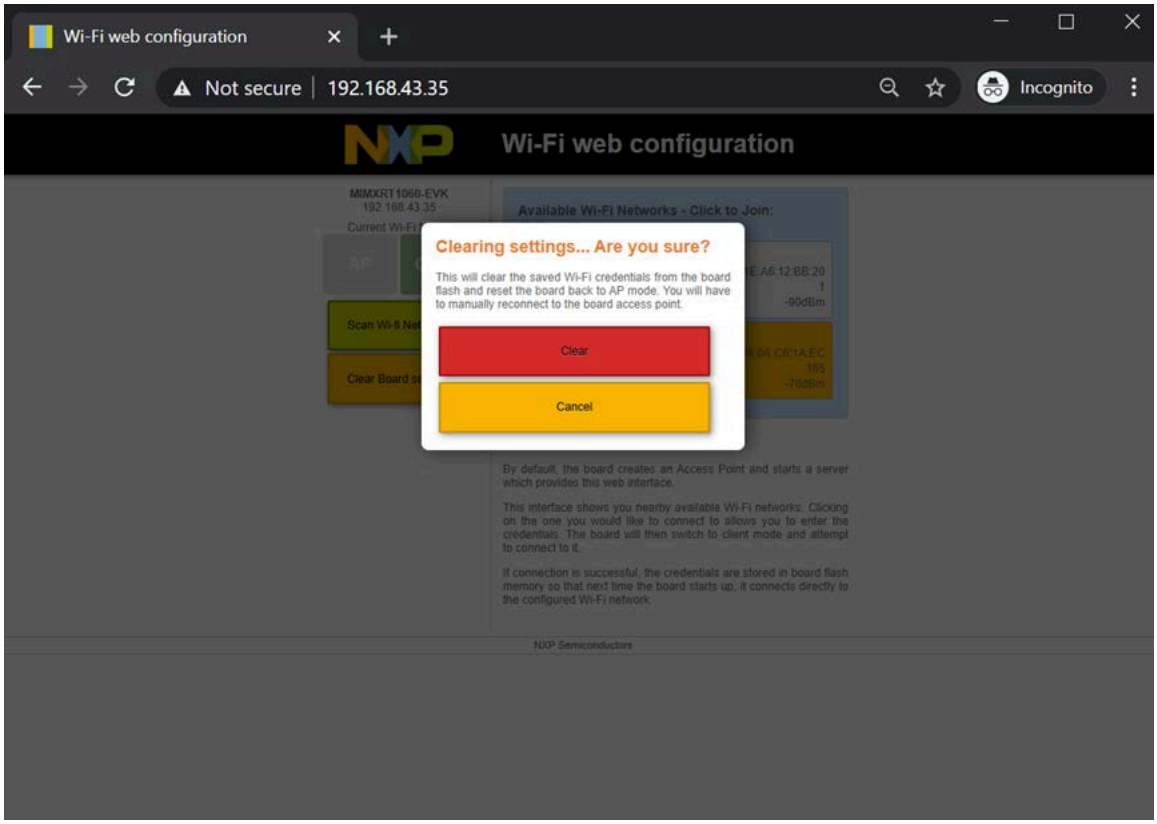


Figure 39. Clear configurations saved in mflash using the website

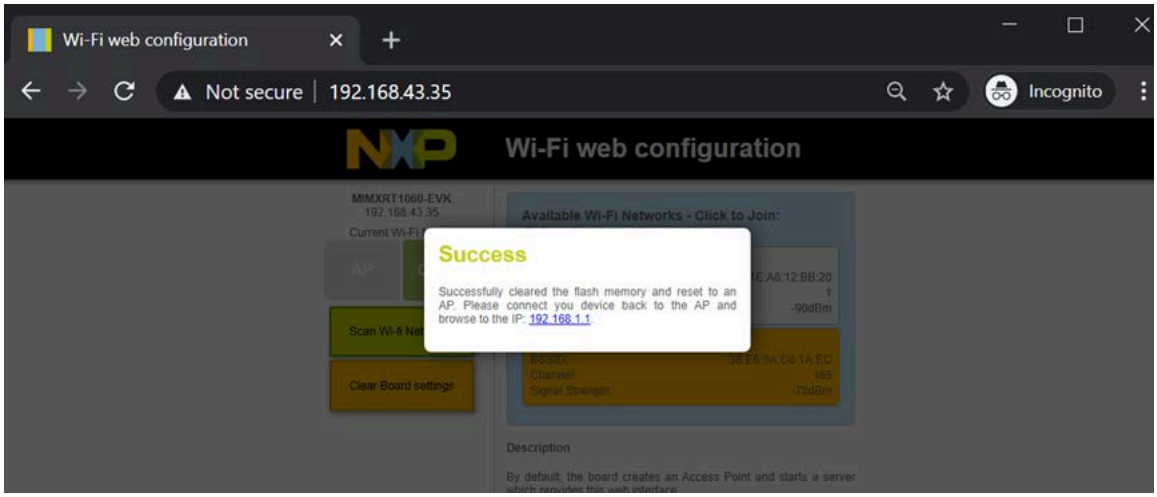


Figure 40. Clear configuration success message in wifi_webconfig application

4.3 wifi_cert sample application

This section describes the *wifi_cert* application to demonstrate the CLI support to handle and enable Wi-Fi configuration for different features. This sample application includes commands related to the Wi-Fi certification process. In this sample application Wi-Fi connection manager CLIs are available.

Table 7. wifi_cert application features

Features	Details
Wi-Fi	Wi-Fi Mobile AP mode Wi-Fi Station mode Wi-Fi Scan Wi-Fi TX Power Limit Wi-Fi Active/Passive Channel List Wi-Fi TX Data Rate Wi-Fi Management Frame Protection Wi-Fi Antenna Diversity Wi-Fi ED MAC
iPerf	TCP Client and Server TCP Client dual mode (TX and RX in simultaneous) TCP Client trade-off mode (TX and RX individual) UDP Client and Server UDP Client dual mode (TX and RX in simultaneous) UDP Client trade-off mode (TX and RX individual)

4.3.1 wifi_cert application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

Refer to Section 2.1 for information about the serial console setup.

4.3.1.1 Run the application

This section describes the available Wi-Fi commands. The application starts with the welcome message, press **Enter** for the command prompt.

```
=========================================
wifi cert demo
=========================================
Initialize CLI
=========================================
Initialize WLAN Driver
=========================================
MAC Address: C0:95:DA:00:D5:0F
821: [net] Initialized TCP/IP networking stack
=========================================
app_cb: WLAN: received event 10
=========================================
app_cb: WLAN initialized
=========================================
```

```
WLAN CLIs are initialized
========================================
ENHANCED WLAN CLIs are initialized
========================================
CLIs Available:
========================================

help
mem_rd <addr> [length]
mem_wr <addr> <value>
wlan-version
wlan-mac
wlan-set-mac MAC_Address
wlan-set-tx_buf_size buf_size <buf_size> bss_type <bss_type>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-deep-sleep-ps <0/1>
wlan-send-hostcmd
wlan-rts <sta/uap> <rts threshold>
wlan-frag <sta/uap> <fragment threshold>
wlan-sta-filter  <filter mode> [<mac address list>]
wlan-tx-pert <0/1> <STA/AP> <p> <r> <n>
wlan-roaming <0/1> rssi_low <rssi_threshold>
wlan-host-sleep <default/mef>
wlan-reset
wlan-11axcfg <11ax_cfg>
wlan-bcast-twt <bcast_twt_cfg>
wlan-twt-setup <twt_cfg>
wlan-twt-teardown <twt_cfg>
wlan-twt-report <twt_report_get>
wlan-ampdu-enable <sta/uap> <xx: rx/tx bit map. Tx(bit 0), Rx(bit 1> <xx: TID bit map>
wlan-mem-access <memory_address> [<value>]
wlan-delba <sta/uap> <direction> <tid> <mac addr>
wlan-set-regioncode <region-code>
wlan-get-regioncode
wlan-get-txpwrlimit <subband>
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
wlan-set-pmfcfg <mfpc> <mfpr>
wlan-get-pmfcfg
wlan-set-antcfg <ant mode> [evaluate_time]
wlan-get-antcfg
wlan-set-ed-mac-mode <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
wlan-get-ed-mac-mode
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ip_address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
========================================
```

Note: *Refer to section Section 4.1.2.1 for basic Wi-Fi features like Wi-Fi scan, Wi-Fi AP mode, Wi-Fi station mode, and iPerf.*

4.3.1.2 Set/get the region code

The following commands are used to set and get the region code.

Command usage:

```
# wlan-set-regioncode
Usage:
wlan-set-regioncode <region-code>
where, region code =
0x00 : World Wide Safe Mode
0x10 : US FCC, Singapore
0x20 : IC Canada
0x30 : ETSI, Australia, Republic of Korea
0x32 : France
0x50 : China
0xFF : Japan
```

Example of command to set the region code:

```
# wlan-set-regioncode 0x10
Region code: 0x10 set
```

Example of command to get the region code:

```
# wlan-get-regioncode
Region code: 0x10
```

Note: *If the region code is programmed in the device One Time Programmable (OTP) memory during the device production process, users cannot set the region code.*

4.3.1.3 Set/get the active/passive channel list

The following commands are used to set and get active and/or passive channel list.

Set the channel list:

```
#wlan-set-chanlist
------------------------------------------------
Number of channels configured: 15

ChanNum: 1      ChanFreq: 2412  Active
ChanNum: 2      ChanFreq: 2417  Active
ChanNum: 3      ChanFreq: 2422  Active
ChanNum: 4      ChanFreq: 2427  Active
ChanNum: 5      ChanFreq: 2432  Active
ChanNum: 6      ChanFreq: 2437  Active
ChanNum: 12     ChanFreq: 2467  Passive
ChanNum: 36     ChanFreq: 5180  Active
ChanNum: 40     ChanFreq: 5200  Active
ChanNum: 44     ChanFreq: 5220  Active
ChanNum: 48     ChanFreq: 5240  Active
ChanNum: 52     ChanFreq: 5260  Passive
ChanNum: 56     ChanFreq: 5280  Passive
ChanNum: 100    ChanFreq: 5500  Passive
ChanNum: 144    ChanFreq: 5720  Passive
```

Get the channel list:

```
# wlan-get-chanlist
------------------------------------------------
Number of channels configured: 15

ChanNum: 1      ChanFreq: 2412  Active
ChanNum: 2      ChanFreq: 2417  Active
ChanNum: 3      ChanFreq: 2422  Active
ChanNum: 4      ChanFreq: 2427  Active
ChanNum: 5      ChanFreq: 2432  Active
ChanNum: 6      ChanFreq: 2437  Active
ChanNum: 12     ChanFreq: 2467  Passive
ChanNum: 36     ChanFreq: 5180  Active
ChanNum: 40     ChanFreq: 5200  Active
ChanNum: 44     ChanFreq: 5220  Active
ChanNum: 48     ChanFreq: 5240  Active
ChanNum: 52     ChanFreq: 5260  Passive
ChanNum: 56     ChanFreq: 5280  Passive
ChanNum: 100    ChanFreq: 5500  Passive
ChanNum: 144    ChanFreq: 5720  Passive
```

4.3.1.4 Set/get TX rate configuration

The following commands are used to set and get TX rate.

Command to set TX rate configuration:

```
# wlan-set-txratecfg
Invalid arguments
```

Command usage:

```
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting>
Where:
<format> - This parameter specifies the data rate format used in this command
    0:    LG
    1:    HT
    2:    VHT
    3:    HE
    0xff: Auto
<index> - This parameter specifies the rate or MCS index
    If <format> is 0 (LG),
      0       1 Mbps
      1       2 Mbps
      2       5.5 Mbps
      3       11 Mbps
      4       6 Mbps
      5       9 Mbps
      6       12 Mbps
      7       18 Mbps
      8       24 Mbps
      9       36 Mbps
      10      48 Mbps
      11      54 Mbps
    If <format> is 1 (HT),
      0       MCS0
      1       MCS1
      2       MCS2
      3       MCS3
      4       MCS4
      5       MCS5
      6       MCS6
      7       MCS7
    If <format> is 2 (VHT),
      0       MCS0
      1       MCS1
      2       MCS2
      3       MCS3
      4       MCS4
      5       MCS5
      6       MCS6
      7       MCS7
      8       MCS8
      9       MCS9
<nss> - This parameter specifies the NSS. It is valid only for VHT
    If <format> is 2 (VHT),
        1       NSS1
        2       NSS2
    If <format> is 3 (HE),
        0       MCS0
        1       MCS1
        2       MCS2
        3       MCS3
        4       MCS4
        5       MCS5
        6       MCS6
```

```
        7       MCS7
        8       MCS8
        9       MCS9
        10      MCS10
        11      MCS11
<nss> - This parameter specifies the NSS. It is valid only for HE
    If <format> is 3 (HE),
        1       NSS1
        2       NSS2
<rate_setting> - This parameter can only specifies the GI types now.
    If <format> is 1 (HT),
        0x0000  Long GI
        0x0020  Short GI
    If <format> is 2 (VHT),
      0x0000  Long GI
      0x0020  Short GI
      0x0060  Short GI and Nsym mod 10=9
    If <format> is 3 (HE),
      0x0000  1xHELTF + GI0.8us
      0x0020  2xHELTF + GI0.8us
      0x0040  2xHELTF + GI1.6us
      0x0060  4xHELTF + GI0.8us if DCM = 1 and STBC = 1
      4xHELTF + GI3.2us, otherwise
```

Example of command to set TX rate configuration:

```
# wlan-set-txratecfg sta 0xff 0 0 1
Configured txratecfg as below:
Tx Rate Configuration:
Type:       0xFF (Auto)
```

Example of command to get TX rate configuration:

```
# wlan-get-txratecfg sta
Tx Rate Configuration:
Type:       0xFF (Auto)
```

Example of command to get the data rate:

```
# wlan-get-data-rate sta
Data Rate:
TX:
Type: LG
Rate: 2 Mbps
RX:
Type: HT
BW:   20 MHz
GI:   Long
MCS:  MCS 6
Rate: 58.50 Mbps
```

4.3.1.5 Get the management frame protection capability

Command to get the management frame protection (MFP) capability:

```
# wlan-get-pmfcfg
Management Frame Protection Capability: No
```

4.3.1.6 Set/get antenna diversity configuration

The following commands are used to get and set antenna diversity configuration.

Note: *Check that the second antenna is connected before setting the configuration.*

Command to set antenna diversity configuration:

```
# wlan-set-antcfg
```

Command usage:

```
wlan-set-antcfg <ant_mode> <evaluate_time> <evaluate_mode>

  <ant_mode>:
      Bit 0   -- Fixed to Tx/Rx antenna 1
      Bit 1   -- Fixed to Tx/Rx antenna 2
      0xFFFF  -- enable Tx/Rx antenna diversity
  <evaluate_time>:
      If ant mode = 0xFFFF, use this to configure
      SAD evaluate time interval in milli seconds unit.
      MAX evaluate time is 65535ms.
      If not specified, default value is 6000 milli seconds.
  <evaluate_mode>:
      0: Ant1 + Ant2
      1: Ant2 + Ant3
      2: Ant1 + Ant3
      255: invalid evaluate mode
      If not used, just keep this field empty.
Examples:
wlan-set-antcfg 1
wlan-set-antcfg 0xffff
wlan-set-antcfg 0xffff 5000
wlan-set-antcfg 0xffff 6000 0
Error: invalid number of arguments
```

Command to get antenna diversity configuration:

```
# wlan-get-antcfg
```

Note: *Read more about Antenna diversity in [2].*

4.3.1.7 Set/get energy detection (ED) MAC feature

This feature enables the European Union (EU) adaptivity test as per the compliance requirements in the ETSI standard.

Depending on the device and front-end loss, the ED threshold offset (*ed_ctrl_2g.offset* and *ed_ctrl_5g.offset*) must be adjusted. The ED threshold offset can be adjusted in steps of 1 dB.

Below are the get and set commands for ED-MAC adjustment.

```
#wlan-get-ed-mac-mode
```

```
#wlan-set-ed-mac-mode <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
```

Where:

Table 8. ED MAC parameters

Parameter	Description
ed_ctrl_2_g	0 = disable ED MAC threshold for 2.4 GHz band 1 = enable ED MAC threshold for 2.4 GHz band
ed_offset_2_g	ED MAC threshold for 2.4 GHz band. Hexadecimal value in units of dB Range: 0x80 to 0x7F, (-128 to 127), 0 = default offset value
ed_ctrl_5_g	0 = disable ED MAC threshold for 5 GHz band 1 = enable ED MAC threshold for 5 GHz band
ed_offset_5_g	ED MAC threshold for 5 GHz band. Hexadecimal value in units of dB Range: 0x80 to 0x7F, (-128 to 127), 0 = default offset value

For 2.4 GHz band:

In this example, the 2.4 GHz ED-MAC threshold is lowered by 1 dB.

Table 9. ED MAC 2.4 GHz command operations

Step	Operation	Command
1	Get ED-MAC status	`#wlan-get-ed-mac-mode` EU adaptivity for 2.4 GHz band: Enabled Energy Detect threshold offset: 0x9
2	Set ED-MAC threshold	`#wlan-set-ed-mac-mode 1 0x8` ED MAC MODE settings configuration successful

For 5 GHz band:

In this example, the 5 GHz ED-MAC threshold is lowered by 2 dB.

Table 10. ED MAC 5 GHz command operations

Step	Operation	Command
1	Get ED-MAC status	`#wlan-get-ed-mac-mode` EU adaptivity for 2.4 GHz band: Enabled Energy Detect threshold offset: 0x9 EU adaptivity for 5 GHz band: Enabled Energy Detect threshold offset: 0xC
2	Set ED-MAC threshold	`#wlan-set-ed-mac-mode 1 0x9 1 0x3` ED MAC MODE settings configuration successful

4.4 uart_wifi_bridge sample application

The *uart_wifi_bridge* application servers as a bridge between Windows NXP Labtool and RW61x Wi-Fi/Bluetooth LE/802.15.4 radios for wireless calibration and RF test. The application:

- Receives the command from Labtool running on a Windows system over UART port
- Passes the command to RW61x Wi-Fi/Bluetooth LE firmware to process
- Returns the command response back to Labtool

The exchanged commands and responses are transparent to *uart_wif_bridge* application.

uart_wif_bridge application must work with RW61x manufacturing firmware. To get Labtool release, reach out to your NXP support representative. The release includes labtool Windows application and the manufacturing firmware for Wi-Fi and Bluetooth LE separately.

4.4.1 Flash Wi-Fi MFG firmware

Refer to Section 4.1.1 to flash Wi-Fi firmware. Use the manufacturing firmware instead of the production firmware.

```
J-Link>loadbin [Wi-Fi MFG firmware],0x08400000
```

4.4.2 Flash Bluetooth MFG firmware

Refer to Section 6.1.1 to flash Bluetooth firmware. Use the manufacturing firmware instead of the production firmware.

```
J-Link>loadbin [Bluetooth LE MFG firmware],0x08540000
```

4.4.3 uart_wifi_bridge application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

4.4.3.1 Run the application

This application runs automatically in Bridge mode and does not require any user interaction.

Note: *The UART for serial console is used as communication port between NXP Labtool and RW61x MFG firmware. So there are no console logs for uart_wif_bridge application.*

Labtool is a user interactive command-line application running on Windows. Different options are defined to control RW61x internal Wi-Fi/Bluetooth LE radios to transmit and receive. Option `88` is used to read back RW61x manufacturing firmware version.

```
Name: Dut labtool
Version: 1.0.0.0.2
Date: Mar 10 2022 (01:12:22)
Note:
1. =========WiFi tool=============
2. =========BT tool=============
3. =========15_4 tool=============
Enter CMD 99 to Exit
Enter option: 1
Name: DutApiClass
Interface: EtherNet
Version: 1.0.0.0.2
Date: Mar 10 2022 (01:12:08)
Note:
DutIf_InitConnection: 0
-------------------------------------------------------
RW610 (802.11a/g/b/n/ac/ax) TEST MENU
-------------------------------------------------------
Enter option: 88
DLL Version : 1.0.0.0.2
LabTool Version: 1.0.0.0.2
FW Version: 18.80.1.103 Mfg Version: 2.0.0.63
SFW Version: 0.0.0.00 SHAL Version: 0.0.0.0
SOC OR Version: 0.d Customer ID: 0
RF OR Version: 0.7 Customer ID: 0
Enter option:
```

4.5 wifi_ipv4_ipv6_echo sample application

The *wifi_ipv4_ipv6_echo* application demonstrates a TCP and UDP echo on the lwIP TCP/IP stack with FreeRTOS. The demo can use both TCP or UDP protocol over IPv4 or IPv6 and acts as an echo server. The application sends back the packets received from the PC, which can be used to test whether a TCP or UDP connection is available.

The demo generates a *IPv6* link-local address (the one from range FE80::/10) after the start. To send data to this address from the remote computer, you must specify the interface over which the demo is reachable. To specify the interface, append the command with % followed by zone index. Refer to Section 2.4 for details about zone index.

4.5.1 wifi_ipv4_ipv6_echo application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

Refer to section Section 2.1 for information about the serial console setup and section Section 2.4 for ipv4/6 tool setup.

4.5.1.1 Run the application

This section describes the available Wi-Fi commands. The application starts with the welcome message, press **Enter** for the command prompt.

```
=========================================
Initialize WLAN Driver
=========================================
MAC Address: C0:95:DA:00:D5:69
129: [net] Initialized TCP/IP networking stack
=========================================
app_cb: WLAN: received event 11
=========================================
app_cb: WLAN initialized
=========================================
Initialize CLI

=========================================

Copyright  2022  NXP

SHELL>>
```

4.5.1.2 Help command

```
SHELL>> help

"help": List all the registered commands

"exit": Exit program

"echo_tcp_client ip_addr port":
   Connects to specified server and sends back every received data.
 Usage:
   ip_addr:     IPv6 or IPv4 server address
   port:        TCP port number

"echo_tcp_server port":
   Listens for incoming connection and sends back every received data.
 Usage:
   port:        TCP port number

"echo_udp port":
   Waits for datagrams and sends them back.
 Usage:
   port:        UDP port number

"end": Ends echo_* command.

"print_ip_cfg": Prints IP configuration.

"wlan_scan": Scans networks.

"wlan_connect ssid":
   Connects to the specified network without password.
 Usage:
   ssid:        network SSID or BSSID

"wlan_connect_with_password ssid password":
   Connects to the specified network with password.
 Usage:
   ssid:        network SSID or BSSID
   password:    password
SHELL>>
```

4.5.1.3 Scan command

The scan command is used to scan the visible access points.

```
SHELL>> wlan_scan
Scanning
SHELL>>
2 networks were found:
 #1  c4:ad:34:a3:86:11"pine5"
 #2  00:72:63:fa:1b:96"netis_FA1B96"
```

4.5.1.4 Connect to found access point

Connect to the network using one of the following commands.

```
wlan_connect <(b)ssid>
wlan_connect_with_password  <(b)ssid> <password>
```

Note: *SSID is the name of the network and BSSID is the MAC address of the interface.*

```
wlan_connect pine5
Connecting in progress. Wait for further messages from callback
```

Once connected to the AP, the console output shows that the Client is successfully connected to AP with ssid “pine5”

```
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [pine5]
```

4.5.1.5 Print the IP configuration

The command prints IPv4 and IPv6 address of the board received from the external access point.

```
SHELL>> print_ip_cfg
************************************************
 IPv4 Address : 10.10.0.203
 IPv4 Subnet mask : 255.255.254.0
 IPv4 Gateway : 10.10.0.1
 IPv6 Address0 : FE80::2E9:3AFF:FEB9:E035
 IPv6 Address1 : -
 IPv6 Address2 : -
 ************************************************
```

Note: *It is necessary to have installed the tools capable of sending and receiving data over TCP or UDP to interact with the demo. Refer to Section 2.4 about tool setup.*

4.5.1.6 TCP client echo

Run ncat on the remote host computer.

```
C:\Users\nxp>ncat -v -l -p 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::10001
Ncat: Listening on 0.0.0.0:10001
```

IPv4

Run the command `echo_tcp_client <Remote host PC IPv4 addr> 10001` in the demo shell.

```
SHELL>> echo_tcp_client 10.10.0.155 10001

Creating new socket.
Connecting...
Connected.
```

Verify the connection from the remote host console. Type some text and hit enter, the demo sends the line back.

```
C:\Users\nxp>
Ncat: Connection from 10.10.0.203.
Ncat: Connection from 10.10.0.203:49153.

hello
hello
```

Check the console logs. The logs show the number of bytes sent back to the remote Host PC.

```
Echoing data. Use end command to return...
ECHO_TCP_CLIENT>>
6B sent back.
```

IPv6

Run the command `echo_tcp_client <Remote host PC IPv6 addr FE80::***%<zone ID>> 10001` in the demo shell.

```
SHELL>> echo_tcp_client fe80::5178:81e4:639:89ca%6 10001

Creating new socket.
Connecting...
Connected.

Echoing data. Use end command to return...
ECHO_TCP_CLIENT>>
```

Verify the connection from the remote host console. Type some text and hit enter, the demo sends the line back.

```
C:\Users\nxp>
Ncat: Connection from fe80::2e9:3aff:feb9:e035.
Ncat: Connection from fe80::2e9:3aff:feb9:e035:49153.

hello
hello
```

Check the console logs. The logs show the number of bytes sent back to the remote Host PC.

```
Echoing data. Use end command to return...
ECHO_TCP_CLIENT>>
6B sent back.
```

Terminate the remote host connection by pressing **ctrl+c** and for demo shell type **end**.

4.5.1.7 TCP server echo

Run the command `echo_tcp_server 10001` in the demo shell.

```
SHELL>> echo_tcp_server 10001

Creating new socket.
Waiting for incoming connection.  Use end command to return...
```

IPv4

To connect with the TCP server, run the command `ncat -v <Demo IPv4 addr> 10001` on the remote Host PC.

```
C:\Users\nxp>ncat -v 10.10.0.203 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 10.10.0.203:10001.
```

Verify the connection from the remote host console. Type some text and hit enter, the demo sends the line back.

```
C:\Users\nxp>
Ncat: Connection from 10.10.0.203.
Ncat: Connection from 10.10.0.203:49153.

hello
```

Check the console logs. The logs show the number of bytes sent back to the remote Host PC.

```
ECHO_TCP_SERVER>>
Accepted connection
Echoing data. Use end command to return...

ECHO_TCP_SERVER>>
6B sent back.
```

IPv6

To connect with the TCP server, run the command `ncat -v <Demo IPv6 addr FE80::***%<zone ID>> 10001` on the remote Host PC.

```
C:\Users\nxp>ncat -v FE80::2E9:3AFF:FEB9:E035%6 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to fe80::2e9:3aff:feb9:e035:10001.
```

Verify the connection from the remote host console. Type some text and hit enter, the demo sends the line back.

```
C:\Users\nxp>
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to fe80::2e9:3aff:feb9:e035:10001.

hello
```

Check the console logs. The logs show the number of bytes sent back to the remote Host PC.

```
ECHO_TCP_SERVER>>
Accepted connection
Echoing data. Use end command to return...
ECHO_TCP_SERVER>>
6B sent back.
```

Terminate the remote host connection by pressing **ctrl+c** and for demo shell type **end**.

4.5.1.8 UDP echo

Run the command `echo_udp 10001` in the demo shell.

```
SHELL>> echo_udp 10001

Creating new socket.
Waiting for datagrams
Use end command to return...
```

IPv4

To connect with UDP server, run the command `ncat -v -u <Demo IPv4 addr> 10001` on the remote host PC.

```
C:\Users\nxp>ncat -v -u 10.10.0.203 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
```

Verify the connection from the remote host console. Type some text and hit enter, the demo sends the line back.

```
C:\Users\nxp>ncat -v -u 10.10.0.203 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 10.10.0.203:10001.

hello
```

Check the console logs. The logs show the number of bytes sent back to the remote host PC.

```
ECHO_UDP>> Datagram carrying 6B sent back.
```

IPv6

To connect with UDP server, run the command `ncat -v -u <Demo IPv6 addr FE80::***%<zone ID>> 10001` on the remote host PC.

```
C:\Users\nxp>ncat -v -u FE80::2E9:3AFF:FEB9:E035%6 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
```

Verify the connection from the remote host console. Type some text and hit enter, the demo sends the line back.

```
C:\Users\nxp>ncat -v -u 10.10.0.203 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to fe80::2e9:3aff:feb9:e035:10001.
hello
```

Check the console logs. The logs show the number of bytes sent back to the remote host PC.

```
ECHO_UDP>> Datagram carrying 6B sent back.
```

Terminate the remote host connection by pressing **ctrl+c** and for demo shell type **end**.

4.6 wifi_cli_prov sample application

The *wifi_cli_prov* application is another sample application used to demonstrate the CLI support to handle and enable Wi-Fi configurations. The application includes most of the commands of *wifi_cli* application and additional provisioning related commands like WPS and DPP.

The *wifi_cli_prov* application features are summarized in Table 11.

Table 11. wifi_cli_prov sample application features

Features	Details
Wi-Fi	Wi-Fi Mobile AP mode Wi-Fi Station mode Wi-Fi Scan Wi-Fi Roaming Wi-Fi TX Power Limit Wi-Fi Regulatory Domain/Operating Class/Country Wi-Fi Power Save (IEEEPS, WMMPS, WNMPS, Deep Sleep) Wi-Fi Security (WPA2/WPA2-Enterprise/WPA3) Wi-Fi ED MAC Wi-Fi Net Monitor Host Sleep WPS
IPerf	TCP Client and Server TCP Client dual mode (Tx and Rx in simultaneous) TCP Client trade-off mode (Tx and Rx individual) UDP Client and Server UDP Client dual mode (Tx and Rx in simultaneous) UDP Client trade-off mode (Tx and Rx individual)

4.6.1 wifi_cli_prov application execution

Refer to Section 3.1 to Section 3.4 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs.

Refer to section Section 2.1 for information about the serial console setup.

4.6.1.1 Run the application

This section describes the available Wi-Fi commands. The application starts with the welcome message.

- Press **Enter** for the command prompt

```
=========================================
wifi cli demo
=========================================
host init done
Initialize CLI
=========================================
Initialize Power Manager
MCU wakeup source 0x0...
=========================================
Initialize WLAN Driver
=========================================
MAC Address: 00:50:43:02:FE:01
456: [net] Initialized TCP/IP networking stack
=========================================
app_cb: WLAN: received event 11
=========================================
app_cb: WLAN initialized
=========================================
WLAN CLIs are initialized
=========================================
ENHANCED WLAN CLIs are initialized
=========================================
CLIs Available:
=========================================
help
wlan-version
wlan-mac
wlan-set-mac MAC_Address
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-set-regioncode <region-code>
wlan-get-regioncode
wlan-wnm-ps <0/1> <sleep_interval>
wlan-11d-enable <sta/uap> <0/1>
wlan-set-max-clients-count <max clients count>
wlan-set-hidden-ssid <0/1>
wlan-deep-sleep-ps <0/1>
wlan-rts <sta/uap> <rts threshold>
wlan-frag <sta/uap> <fragment threshold>
wlan-sta-filter  <filter mode> [<mac address list>]
wlan-tx-pert <0/1> <STA/UAP> <p> <r> <n>
wlan-roaming <0/1> rssi_low <rssi_threshold>
wlan-host-sleep <default [default_val]>/mef> <manual>
wlan-multi-mef <ping/arp/multicast/del> [<action>]
suspend <power mode>
wlan-reset
```

```
wlan-net-monitor-cfg
wlan-set-monitor-filter <opt> <macaddr>
wlan-set-monitor-param <action> <monitor_activity> <filter_flags> <radio_type> <chan_number>
wlan-wmm-stat <bss_type>
wlan-generate-wps-pin
wlan-start-wps-pbc
wlan-start-wps-pin <8 digit pin>
wlan-scan-channel-gap <channel_gap_value>
wlan-start-dpp <channel>
wlan-stop-dpp
wlan-get-signal
wlan-set-tx-omi <value>
wlan-set-toltime <value>
wlan-get-txpwrlimit <subband>
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
wlan-set-pmfcfg <mfpc> <mfpr>
wlan-get-pmfcfg
wlan-uap-get-pmfcfg
wlan-set-antcfg <ant mode> [evaluate_time]
wlan-get-antcfg
wlan-set-ed-mac-mode <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
wlan-get-ed-mac-mode
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6 address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
wlan-set-rtc-time <year> <month> <day> <hour> <minute> <second>
wlan-get-rtc-time
wlan-read-usb-file <type:ca-cert/client-cert/client-key> <file name>
wlan-dump-usb-file <type:ca-cert/client-cert/client-key>
========================================
```

Note: *Refer to Section 4.1.2 "wifi_cli application execution" for Wi-Fi features like Wi-Fi Scan, Wi-Fi AP mode, Wi-Fi Station mode, and IPerf.*

4.6.1.2 WPS commands

The following commands are used to set up Wi-Fi connections with WPS PIN and PBC methods. The commands apply to both STA and micro AP modes.

WPS PBC

Start WPS with PBC mode:

```
# wlan-start-wps-pbc
```

WPS PIN

Generate WPS PIN:

```
# wlan-generate-wps-pin
WPS PIN is: 37612368
```

Start WPS with PIN mode:

```
# wlan-start-wps-pin 37612368
Start WPS PIN session with 37612368 pin
# 37612368
```

4.6.1.3 Start/stop DPP

The following commands are used to start and stop the DPP (Wi-Fi Easy Connect) feature.

Start DPP

```
# wlan-start-dpp
Error: invalid channel
Usage: wlan-start-dpp <channel>
Usage example :
wlan-start-dpp 6
# wlan-start-dpp 6
Bootstrapping QR Code URI:
dpp_qr_code DPP:C:81/6;M:00504302fe01;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACWb+74Ju49Efwp0/
lcSSUXUml5x4nd+H2ZuaLOQeoELk=;;
```

Note: *Convert the generated* `dpp_qr_code` *info into QR code using tools like "QR code generator". Use a DPP capable device with a Wi-Fi connection to scan the QR code so the board is added to the Wi-Fi network.*

Stop DPP

```
# wlan-stop-dpp
```

4.6.1.4 Set/get RTC time

The following commands are used to set and get the RTC time.

Command usage:

```
# wlan-set-rtc-time
Error: invalid number of arguments
Usage: wlan-set-rtc-time <year> <month> <day> <hour> <minute> <second>
Usage example :
wlan-set-rtc-time 2022 12 31 19 00
Current datetime: 1970-01-01 00:05:52
```

Set RTC time:

```
# wlan-set-rtc-time 2023 3 31 15 50
Current datetime: 2023-03-31 15:50:00
```

Get RTC time:

```
# wlan-get-rtc-time
Current datetime: 2023-03-31 15:50:06
```

4.6.1.5 Read/dump USB file

The following commands are used to read/dump CA/key files (for WPA2-Enterprise) from an external USB disk.

Command usage:

```
# wlan-read-usb-file
Error: invalid number of arguments
Usage: wlan-read-usb-file <type:ca-cert/client-cert/client-key> <file name>
Usage example :
wlan-read-usb-file ca-cert 1:/ca.der
```

The log shows once USB disk is plugged:

```
# mass storage device attached:pid=0x1000vid=0x90c address=1
```

Read the CA/key files from USB disk:

```
# wlan-read-usb-file ca-cert 1:/ca.der
.........................fatfs test.....................
fatfs mount as logical drive 1......success
# wlan-read-usb-file client-cert 1:/client.der
.........................fatfs test.....................
fatfs mount as logical drive 1......success
# wlan-read-usb-file client-key 1:/c_key.der
.........................fatfs test.....................
fatfs mount as logical drive 1......success
```

Dump the read files:

```
# wlan-dump-usb-file ca-cert
[USB File] ca-cert
**** Dump @ 20052DC0 Len: 1021 ****
30 82 03 f9 30 82 02 e1 a0 03 02 01 02 02 09 00
da aa 1c 26 b3 0e 49 0e 30 0d 06 09 2a 86 48 86
f7 0d 01 01 0b 05 00 30 81 92 31 0b 30 09 06 03
55 04 06 13 02 55 53 31 13 30 11 06 03 55 04 08
0c 0a 43 61 6c 69 66 6f 72 6e 69 61 31 14 30 12
06 03 55 04 07 0c 0b 53 61 6e 74 61 20 43 6c 61
```

```
 72 61 31 17 30 15 06 03 55 04 0a 0c 0e 57 69 2d
 46 69 20 41 6c 6c 69 61 6e 63 65 31 1d 30 1b 06
 03 55 04 03 0c 14 57 46 41 20 52 6f 6f 74 20 43
 65 72 74 69 66 69 63 61 74 65 31 20 30 1e 06 09
 2a 86 48 86 f7 0d 01 09 01 16 11 73 75 70 70 6f
 72 74 40 77 69 2d 66 69 2e 6f 72 67 30 1e 17 0d
 31 33 30 33 31 31 31 39 30 32 32 36 5a 17 0d 32
 33 30 33 30 39 31 39 30 32 32 36 5a 30 81 92 31
 0b 30 09 06 03 55 04 06 13 02 55 53 31 13 30 11
 06 03 55 04 08 0c 0a 43 61 6c 69 66 6f 72 6e 69
 61 31 14 30 12 06 03 55 04 07 0c 0b 53 61 6e 74
 61 20 43 6c 61 72 61 31 17 30 15 06 03 55 04 0a
 0c 0e 57 69 2d 46 69 20 41 6c 6c 69 61 6e 63 65
 31 1d 30 1b 06 03 55 04 03 0c 14 57 46 41 20 52
 6f 6f 74 20 43 65 72 74 69 66 69 63 61 74 65 31
 20 30 1e 06 09 2a 86 48 86 f7 0d 01 09 01 16 11
 73 75 70 70 6f 72 74 40 77 69 2d 66 69 2e 6f 72
 67 30 82 01 22 30 0d 06 09 2a 86 48 86 f7 0d 01
 01 01 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01
 01 00 e9 33 82 bb 6d 26 fb dc cb 64 84 d8 02 11
 a6 b7 1c 32 25 0f f5 c7 2b 57 49 02 58 5f c2 cd
 61 44 bf 37 71 c6 d9 00 10 e0 62 83 87 b0 cf 0c
 8d 77 ce 17 5c d3 23 94 13 50 88 39 76 b5 11 32
 9b 1c 85 3d c4 00 6b 2f b2 9f 5f 61 c2 43 59 c2
 3e a4 4f 89 6f 7f c2 04 1c 87 c0 30 ee 0f f4 ad
 79 a0 d7 37 b0 3c 9d fa cc a9 ea e7 fe 50 d6 ec
 9d f1 87 da 5c 1c e1 c6 6d 48 62 5c 13 28 c9 42
 8b 10 c5 70 07 95 a8 76 fb e2 60 1c 75 e9 75 97
 17 5a 80 7b da 2f 9e 9f a5 41 31 9b b1 b2 30 65
 d8 77 e4 36 bd 86 bf a3 55 30 08 c1 9f 30 e8 fa
 14 b1 f0 2a 9c ba d8 a0 a8 2b 7d 35 e8 79 56 52
 56 e9 83 c6 06 a4 b0 ed d8 10 78 2f 3c 66 3d fc
 87 db 1f 97 8f be 66 01 c9 65 3b 1c 7b 18 ad 2b
 70 63 e9 20 d4 6c af 8b e0 a2 5b 66 e0 18 75 78
 85 b9 ee 1f a0 5e 81 5f 1c 22 4b 30 6e ba ba 15
 3f d9 02 03 01 00 01 a3 50 30 4e 30 1d 06 03 55
 1d 0e 04 16 04 14 0b 03 c2 3e 54 a2 28 bd 3e 49
 de 72 f1 5f 8e ab 0e 97 67 82 30 1f 06 03 55 1d
 23 04 18 30 16 80 14 0b 03 c2 3e 54 a2 28 bd 3e
 49 de 72 f1 5f 8e ab 0e 97 67 82 30 0c 06 03 55
 1d 13 04 05 30 03 01 01 ff 30 0d 06 09 2a 86 48
 86 f7 0d 01 01 0b 05 00 03 82 01 01 00 b0 dc 4e
 cf d0 d7 6f b4 e4 34 52 ad 3c e6 3f ee 56 62 83
 86 6d 76 ee 8b 81 fe aa 11 43 af 24 65 bf 07 0d
 56 c6 53 cc 8e 93 e4 f6 82 3e 25 d4 7c 65 9e 01
 f5 08 44 f8 b6 61 83 a4 89 6e 32 be 54 9a 94 71
 3b e7 5d 67 60 d1 ef 02 81 e5 30 ef 5f 06 35 98
 ad 84 81 25 59 82 b7 2a 1d 5c 4a 8d ec 26 dd 0f
 cf d0 a2 25 f8 f0 b4 10 f0 b1 d0 2a 0b dc d5 f6
 d8 29 d3 d4 24 12 46 af f3 85 f6 d1 5c 5a 21 36
 b7 1f 54 2e 2b f2 d6 4a 7f ce 46 bc dc 5f 42 25
 d7 46 28 17 a1 f3 dd 95 9a eb 99 94 a0 cc dd f0
 36 2d 0b 21 ad 71 b6 2b 4a df 8d 30 2a 91 07 5f
 3a 3c 34 27 e5 70 42 ca d8 05 f1 5b a2 0b 2c a4
 67 4e 21 a8 72 f3 00 57 36 d4 ff 72 87 02 86 8b
 70 47 0f f4 9e a7 c7 ef af dc b8 56 93 08 a8 74
 26 da 36 85 3f 16 35 57 18 4f 49 5f 94 1e 81 35
 7c 23 e2 b4 82 53 a7 f8 97 8b bb 97 4d
******** End Dump *******
# wlan-dump-usb-file client-cert
[USB File] client-cert
**** Dump @ 200535E0 Len: 1033 ****
 30 82 04 05 30 82 02 ed a0 03 02 01 02 02 02 10
 46 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b 05 00
 30 81 92 31 0b 30 09 06 03 55 04 06 13 02 55 53
 31 13 30 11 06 03 55 04 08 0c 0a 43 61 6c 69 66
 6f 72 6e 69 61 31 14 30 12 06 03 55 04 07 0c 0b
 53 61 6e 74 61 20 43 6c 61 72 61 31 17 30 15 06
 03 55 04 0a 0c 0e 57 69 2d 46 69 20 41 6c 6c 69
 61 6e 63 65 31 1d 30 1b 06 03 55 04 03 0c 14 57
 46 41 20 52 6f 6f 74 20 43 65 72 74 69 66 69 63
 61 74 65 31 20 30 1e 06 09 2a 86 48 86 f7 0d 01
 09 01 16 11 73 75 70 70 6f 72 74 40 77 69 2d 66
 69 2e 6f 72 67 30 1e 17 0d 31 33 30 35 31 30 32
 33 34 34 35 31 5a 17 0d 32 33 30 35 30 38 32 33
 34 34 35 31 5a 30 7e 31 0b 30 09 06 03 55 04 06
 13 02 55 53 31 13 30 11 06 03 55 04 08 0c 0a 43
 61 6c 69 66 6f 72 6e 69 61 31 17 30 15 06 03 55
 04 0a 0c 0e 57 69 2d 46 69 20 41 6c 6c 69 61 6e
 63 65 31 1f 30 1d 06 03 55 04 03 0c 16 43 6c 69
 65 6e 74 20 43 65 72 74 69 66 69 63 61 74 65 20
 49 44 4c 31 20 30 1e 06 09 2a 86 48 86 f7 0d 01
 09 01 16 11 73 75 70 70 6f 72 74 40 77 69 2d 66
 69 2e 6f 72 67 30 82 01 22 30 0d 06 09 2a 86 48
 86 f7 0d 01 01 01 05 00 03 82 01 0f 00 30 82 01
 0a 02 82 01 01 00 dc a3 b9 10 e1 6d 9b fd f0 70
```

```
 d9 98 12 01 75 56 04 29 89 5b 4c 9a c7 c2 ba f3
 69 11 88 79 41 d5 a2 79 22 6f 2b 4c 49 2d 66 53
 c3 41 58 68 3b 75 2b f8 88 fd 63 cb a4 0f 71 ea
 08 c6 bd e5 3d 5a dc bc bd 96 10 c8 30 4f d5 46
 a4 60 65 11 9a fc 50 bf 8e 48 8a f6 ef 52 80 85
 cf 66 a2 64 d6 cf c9 54 6e da ed d7 82 f0 1b ca
 8a a2 05 85 7a 07 97 43 78 9c 9c 70 54 0f 24 6a
 0e 3c 75 76 eb 4f 1a ce 1f 6e 82 f6 ac 14 f3 98
 92 fd cd 68 cc 72 4c 9a ad 39 d9 63 d2 41 92 08
 4a af 3e 56 6c 65 09 b6 d7 c5 0e e4 53 6a 19 93
 10 ea 5b 78 4c 23 73 8f 4b cd 2c f2 ae 2d 33 3b
 a2 b6 fc c5 3b 42 6f 00 78 21 e0 81 72 13 bc a0
 9c 81 3b fa 53 3d 7f 71 30 cc 40 b0 f6 71 01 74
 be c0 bd f9 d8 2e 19 48 10 8f 12 f3 06 53 52 99
 dd f1 70 39 42 ce d0 6b 8a cb 36 ea 88 db bd d9
 21 62 4b 87 db ff 02 03 01 00 01 a3 78 30 76 30
 0f 06 03 55 1d 13 01 01 ff 04 05 30 03 02 01 00
 30 0b 06 03 55 1d 0f 04 04 03 02 05 e0 30 16 06
 03 55 1d 25 01 01 ff 04 0c 30 0a 06 08 2b 06 01
 05 05 07 03 02 30 1d 06 03 55 1d 0e 04 16 04 14
 2d 0b 69 b1 f0 f3 a1 65 17 c1 fa c8 f8 c1 7d 59
 3b 37 6b 0b 30 1f 06 03 55 1d 23 04 18 30 16 80
 14 0b 03 c2 3e 54 a2 28 bd 3e 49 de 72 f1 5f 8e
 ab 0e 97 67 82 30 0d 06 09 2a 86 48 86 f7 0d 01
 01 0b 05 00 03 82 01 01 00 b2 f1 c9 f8 9d 98 4c
 2b 04 03 af 6f 8d 29 ac 1a 82 76 88 45 c3 b8 72
 3e e7 b8 c8 a3 ef 2a 45 52 bf 9b 58 ac d5 e8 94
 01 e7 c4 c4 3b 71 c4 a4 c4 3c fb 68 ba b9 58 3b
 c0 38 8b ba 1e a7 d4 f8 dc 17 bc 9f 05 59 76 6c
 73 7e ed 78 71 d2 ca 42 b1 3e be 20 ae 1d 04 dc
 04 b4 a8 02 fe bf 49 96 b7 31 e2 11 92 40 d2 3e
 d0 8f 4c 9f cf a7 e5 58 3c af 20 3e 57 3d a7 01
 2c fc a2 e0 17 12 bf 4a ae 91 0c d8 d3 3e ce 89
 30 28 71 8b 58 92 75 41 12 f2 6c f7 24 38 50 d9
 14 16 56 e8 9b 3e 8a e1 81 5a 8d 4f 25 c1 8c 55
 1b 63 14 d5 97 78 41 0a d2 bf 05 de 08 c8 54 d7
 2e 2f 74 5d 71 ec 78 d6 e3 da c1 ee dc f1 41 53
 b3 c6 e6 68 4a 40 c5 0e 62 ab d0 cc 40 c8 5d 3e
 92 0c 0f 8f ad ea 6a af 05 e7 fc 12 fc eb 52 ba
 19 5e 0b 71 f2 81 83 41 c7 75 6f 60 24 1c 99 08
 dd 36 8a 01 53 c3 9e 8a ce
******** End Dump *******
# wlan-dump-usb-file client-key
[USB File] client-key
**** Dump @ 20052548 Len: 1193 ****
 30 82 04 a5 02 01 00 02 82 01 01 00 dc a3 b9 10
 e1 6d 9b fd f0 70 d9 98 12 01 75 56 04 29 89 5b
 4c 9a c7 c2 ba f3 69 11 88 79 41 d5 a2 79 22 6f
 2b 4c 49 2d 66 53 c3 41 58 68 3b 75 2b f8 88 fd
 63 cb a4 0f 71 ea 08 c6 bd e5 3d 5a dc bc bd 96
 10 c8 30 4f d5 46 a4 60 65 11 9a fc 50 bf 8e 48
 8a f6 ef 52 80 85 cf 66 a2 64 d6 cf c9 54 6e da
 ed d7 82 f0 1b ca 8a a2 05 85 7a 07 97 43 78 9c
 9c 70 54 0f 24 6a 0e 3c 75 76 eb 4f 1a ce 1f 6e
 82 f6 ac 14 f3 98 92 fd cd 68 cc 72 4c 9a ad 39
 d9 63 d2 41 92 08 4a af 3e 56 6c 65 09 b6 d7 c5
 0e e4 53 6a 19 93 10 ea 5b 78 4c 23 73 8f 4b cd
 2c f2 ae 2d 33 3b a2 b6 fc c5 3b 42 6f 00 78 21
 e0 81 72 13 bc a0 9c 81 3b fa 53 3d 7f 71 30 cc
 40 b0 f6 71 01 74 be c0 bd f9 d8 2e 19 48 10 8f
 12 f3 06 53 52 99 dd f1 70 39 42 ce d0 6b 8a cb
 36 ea 88 db bd d9 21 62 4b 87 db ff 02 03 01 00
 01 02 82 01 01 00 dc 9d b0 9c da 6b 79 00 cf 7c
 67 76 90 fa 78 52 cb d2 a4 8f 6f e7 8c 3a 80 28
 87 34 8a db 84 22 93 54 c0 43 9c 6d a8 f6 06 4d
 56 fd 6d e2 bb 21 0c 18 75 11 b8 c9 94 80 05 0d
 58 3d 30 ff 98 fb d3 9f bd 89 e1 b9 e9 e4 c9 82
 db 35 af 99 8c f9 21 dc 87 ee ad 55 00 33 e4 62
 e7 e0 de 1c 2a 56 96 1f 20 c1 33 f4 bb ee 4e 3b
 95 a7 30 12 28 d4 92 41 5a c6 6d fe 3b 6e f0 a0
 43 1f dc ec 4d 97 6e 0a 99 17 a7 31 80 61 a9 cb
 28 4f 80 58 07 0d c1 87 cc 5e f9 73 7d e6 b5 e3
 d8 17 48 6e be 7e 45 61 9c f5 9b 33 e2 8e 56 99
 de 26 2f f2 6f 6c a7 a8 86 bb 1e 72 27 a7 e1 36
 51 cd 61 64 bb dd d7 57 99 22 7b 57 35 68 9c b5
 9c 9f 39 6c 62 64 e1 fa 1c 98 ff f6 4e 8c 50 ca
 c4 db 3d 80 b1 1d 54 c8 1e e0 03 2b fa 9f f5 21
 87 27 d1 7b 10 11 c4 ef ed 22 b2 ec 01 6e a3 83
 3b 34 bc ac 77 51 02 81 81 00 f0 97 95 a1 ee 98
 49 0b 21 1b f7 7a 78 69 1c f0 58 c6 70 e7 74 d0
 ad 4f 54 fa f8 2b a2 b7 fb df a3 a2 ba a0 0e e9
 89 46 9e 8f f1 8f 66 c0 ca 5e 16 b2 79 38 84 6a
 ae d0 32 29 d4 b4 69 22 0a 01 a5 56 d5 e0 85 7b
 82 67 3e 8a ac 7d 7d 5b ac 33 0b a3 db a2 49 aa
 05 96 a4 e0 b4 1c bc 66 af b7 75 73 8e 58 80 28
 b7 1d 9f a9 d0 b7 ff 9b a7 f8 ca 0c 93 8a 55 bb
```

```
db df 4b 23 4b 4d f9 02 1e 33 02 81 81 00 ea c5
06 01 3d 51 31 27 7d 01 c8 87 ba 72 88 cd 43 02
89 81 6d da e5 46 8b 3c 99 a4 25 db a7 07 a9 41
cd df ec 32 d5 b1 87 2f 33 a3 60 51 68 01 ab 31
7a 2c f0 ec 6b 9d 81 b6 9e db c5 88 3f 5c 8b 1a
ba 6d d7 97 2a 54 0b e5 09 89 81 b5 9d c2 f7 c1
6e 4f 71 58 80 41 19 a5 63 66 13 c3 a6 fb 95 0c
b5 df a6 b4 c8 e8 9d 60 0f 90 26 02 f9 a9 6a 51
9c de 22 cf 82 66 ce 42 1c a6 02 0e a7 05 02 81
81 00 e7 e4 42 fd fc 19 5d b5 d3 68 c0 44 93 d0
44 6b 48 35 a4 57 02 99 ab a1 de 37 b3 81 63 69
cf e4 03 35 72 89 99 35 f0 f1 57 1d 48 67 be 53
2d fa 38 08 37 9c 88 cc c6 c7 b6 c1 e8 d9 26 c7
ff 3b 0c e0 c0 6e 92 59 b4 1b cd 05 1e 32 29 e4
74 fa 12 4b 12 03 be da 98 5e 55 76 9b 43 63 37
da 3f 8a 7e 21 82 1d ac a0 aa 75 dc d8 66 b0 80
98 0a cc bc 08 6c fa 2b 82 46 1b 86 de 2b 3e 49
93 4f 02 81 80 1e ce ab f3 0d b0 d1 da 74 b4 ff
33 90 6b e7 37 c8 4b 54 ef ff 12 72 73 c7 61 b4
67 ad f0 1d 03 0c 5a ee 41 2c 25 9f 95 24 40 35
6e 82 fd 2b c0 cc 4e 39 d2 1b eb 6a 53 c8 e9 c5
fe e0 f4 f8 1b 94 c5 75 21 64 e1 19 54 de 1a b8
1d ab 3f c1 ec 0b c6 fe 4a be 7c f6 97 94 5d f7
a5 35 82 bf 2e d4 68 4e 95 82 b2 c6 8a 7f dc 53
2e 7f 4e 74 a4 9e a7 07 06 bf 5a ab aa 01 f6 fa
fb 6d d9 ae 61 02 81 81 00 ea 5d 06 14 7c 0c 14
13 67 a4 fb b1 1e 06 20 4b 9f b8 83 7f bc 0e 16
59 20 37 3b 21 f9 41 08 15 85 fd 9c 1b 33 e9 c7
f8 94 b1 89 86 23 bc 3e f3 20 18 67 76 a1 11 71
9f 9f bd b6 53 43 ca 32 9a bd e3 e4 01 12 33 46
5f 00 76 df b7 a2 05 41 68 c3 50 2d c8 0d 41 7f
e0 5d 1c 52 27 45 0b cb 26 bb a5 9a da d8 ef 48
8c fb 0d fb e6 0e c9 f5 97 ee ec f6 a4 2f 0f 17
72 3d ab 34 22 42 ff bf cc
******** End Dump *******
```

4.7 wifi_httpsrv sample application

The wifi_httpsrv application demonstrates an HTTP server on the lwIP TCP/IP stack with FreeRTOS. The application acts as an HTTP server and sends a web page back to the PC which uses an Internet browser to send a request for HTTP connection.

The *wifi_httpsrv* application features are summarized in Table 12.

Table 12. wifi_httpsrv sample application features

Features	Details
Wi-Fi and HTTP	Wi-Fi Station mode Wi-Fi Security HTTP server (Request GET/POST) DHCP Client

4.7.1 User configurations

Table 13 lists the Wi-Fi features and feature-related macros that the user can configure.

Table 13. Wi-Fi configurations of wifi_httpsrv application

Feature	Macro definition	Default value	File name	Details
Wi-Fi STA	AP_SSID	“my_network”	wifi_httpsrv.c	Default SSID and passphrase to connect Ex-AP with the given sample application. Can be modified by changing the macro value. Default wpa2 security is used.
	AP_PASSWORD	“my_password”		

4.7.2 wifi_httpsrv application execution

Refer to Section 3.1 to Section 3.4 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs.

Refer to Section 2.1 for information about the serial console setup.

4.7.2.1 Start-up logs

The following logs can be observed on the console once RW61x EVK is up and running.

```
Starting httpsrv DEMO
[i] Initializing Wi-Fi connection...
MAC Address: 00:50:43:02:FE:01
451: [net] Initialized TCP/IP networking stack
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: my_network with password my_password
```

4.7.2.2 Connect Wi-Fi STA to Ex-AP

When the board is up and running, it tries to connect to Ex-AP and acquires the IP address through DHCP.

Make sure Ex-AP has a target SSID/password before running the application.

The network configuration is printed on the console when the board is connected to Ex-AP.

```
Starting httpsrv DEMO
[i] Initializing Wi-Fi connection...
MAC Address: 00:50:43:02:FE:01
451: [net] Initialized TCP/IP networking stack
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: my_network with password my_password
[i] Connected to Wi-Fi
ssid: my_network
[!]passphrase: my_password
 Now join that network on your device and connect to this IP: 192.168.0.10
*********************************************
 HTTP Server example
*********************************************
 IPv4 Address     : 192.168.0.10
 IPv4 Subnet mask : 255.255.255.0
 IPv4 Gateway     : 192.168.0.1
 mDNS hostname    : wifi-http
*********************************************
Ready
```

4.7.2.3 Open the website in the PC browser

Use the board IP-192.168.0.10 to open the website in the browser of the PC in the same AP network. The web page opens.

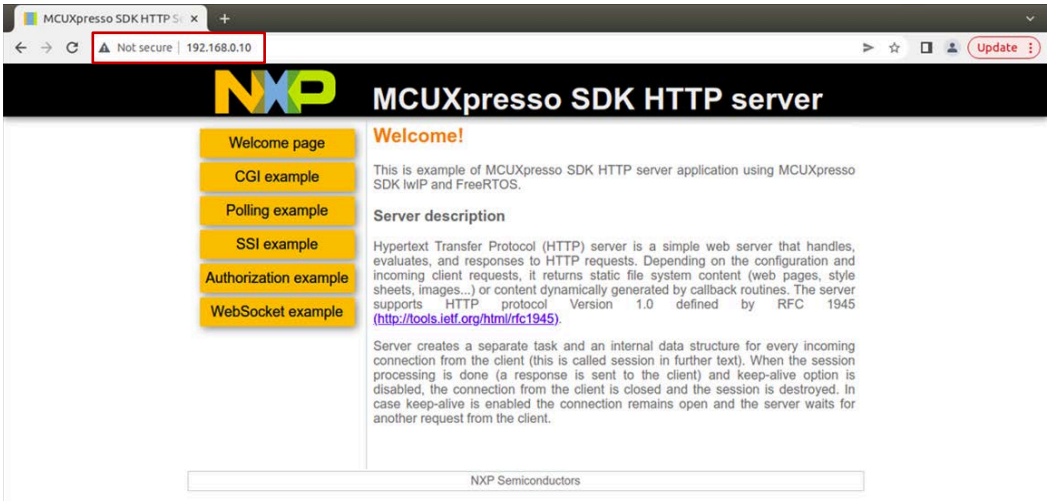


Figure 41. URL to open MCUXpresso SDK HTTP server

The board advertises itself using mDNS so it can be accessed using the URL *wifi-http.local*.

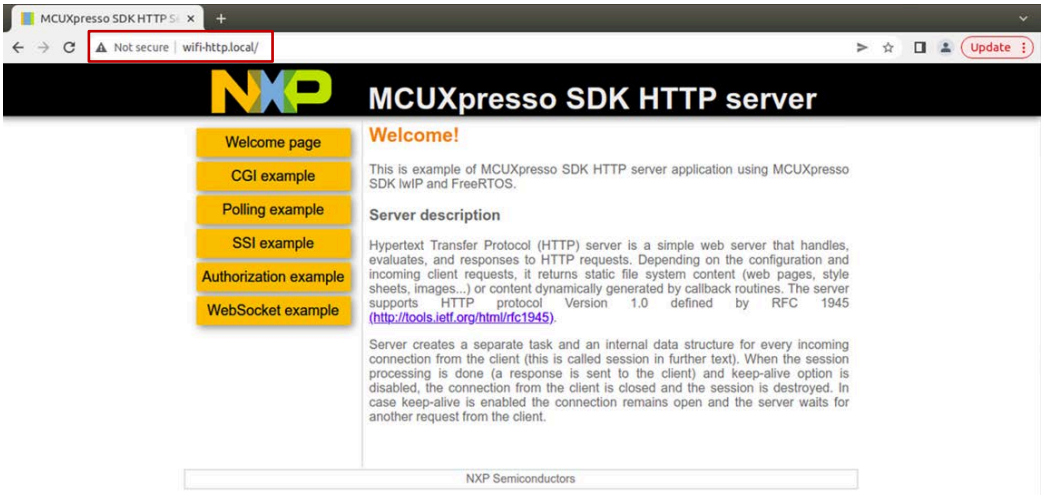


Figure 42. URL for the board on MCUXpresso SDK HTTP server

Note: *To support mDNS out-of-the-box, an mDNS resolver must be installed on the PC. Use Bonjour Print Services (Windows OS) or nss-mdns (Linux OS) to download mDNS.*

4.7.2.4 CGI example

The CGI example shows how to send an HTTP post/get request to the server through CGI functionality. The input text is sent to the HTTP server and stored in memory by sending an HTTP post request. An HTTP get request to the server retrieves the stored text.

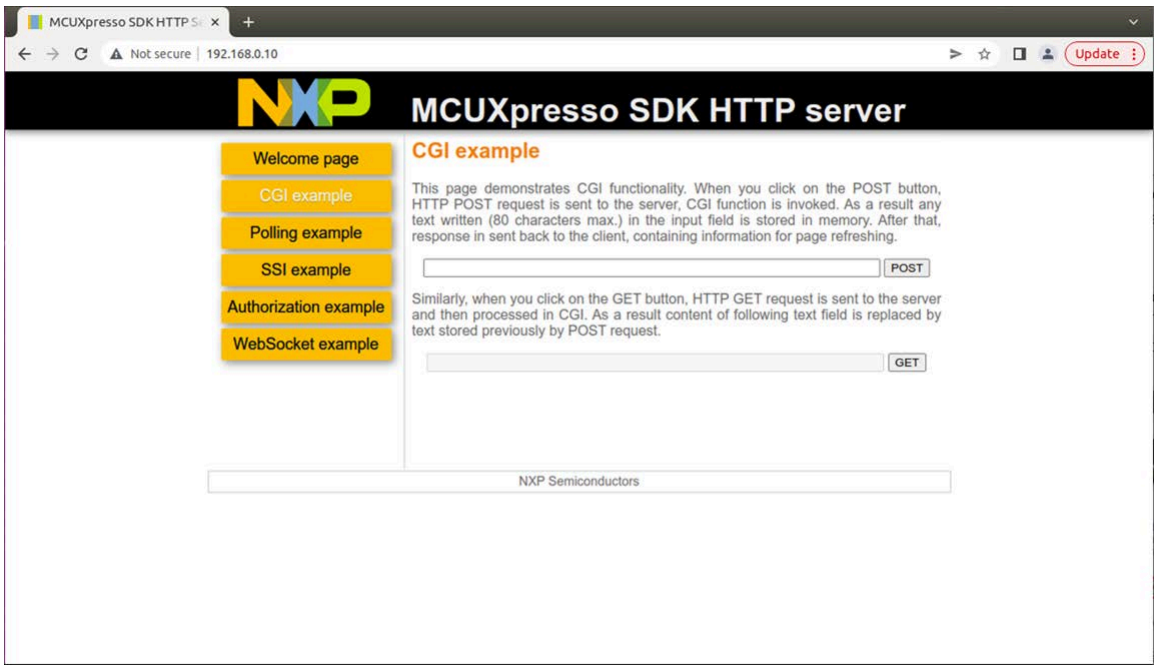


Figure 43. CGI example page

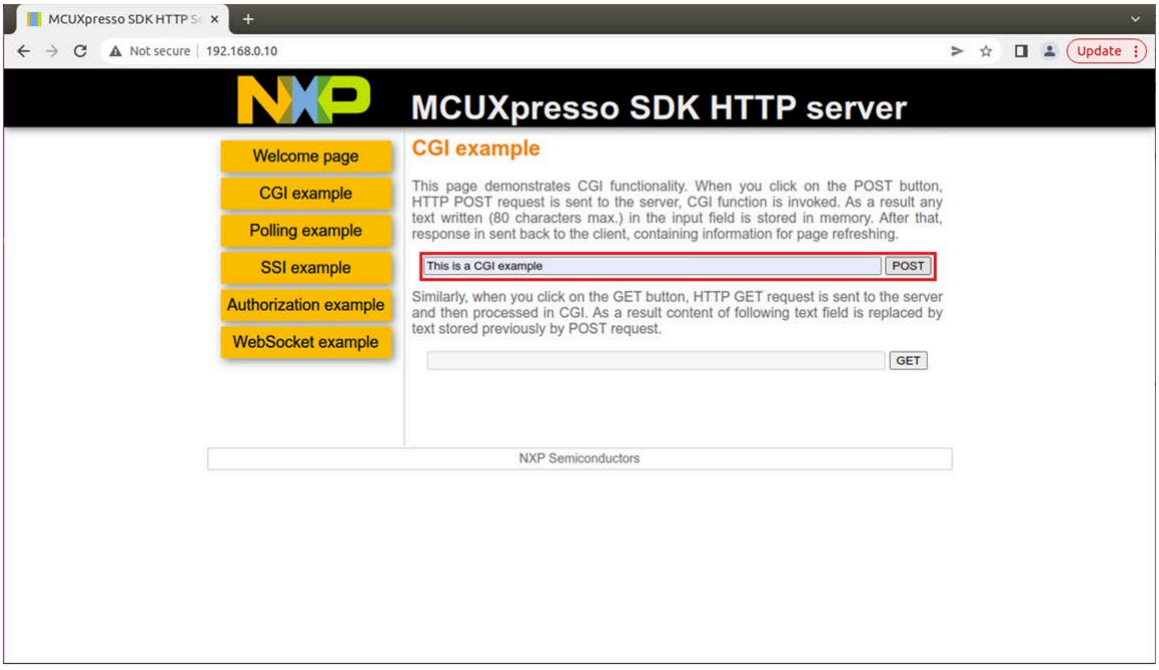


Figure 44. Using HTTP post

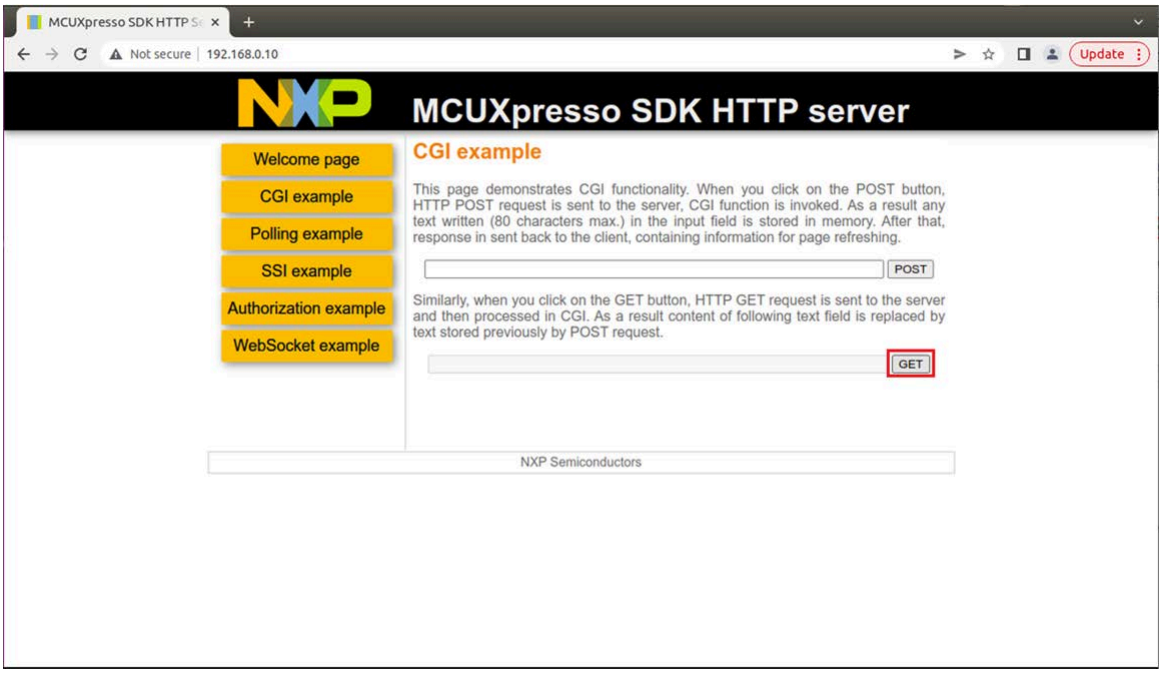


Figure 45. Using HTTP get

Figure 46. HTTP get response

4.7.2.5 Polling example

The polling example reads and displays the RTC time from the HTTP server every second.

Figure 47. Polling example page

4.7.2.6 Authorization example

The example of HTTP authorization uses the pre-set username and password “admin”.

```
static const HTTPSRV_AUTH_USER_STRUCT users[] = {
     {"admin", "admin"}, {NULL, NULL} /* Array terminator */
};
```

Figure 48. Sign in pop-up on the authorization example page

Figure 49. Capturing the username and password to sign in

Figure 50. Authorization example page once signed in

4.7.2.7 WebSocket example

The WebSocket example uses HTML5. The WebSocket connection is set up by sending a request from the browser on the PC. The WebSocket echo application runs on the board and sends back the messages sent to the server.

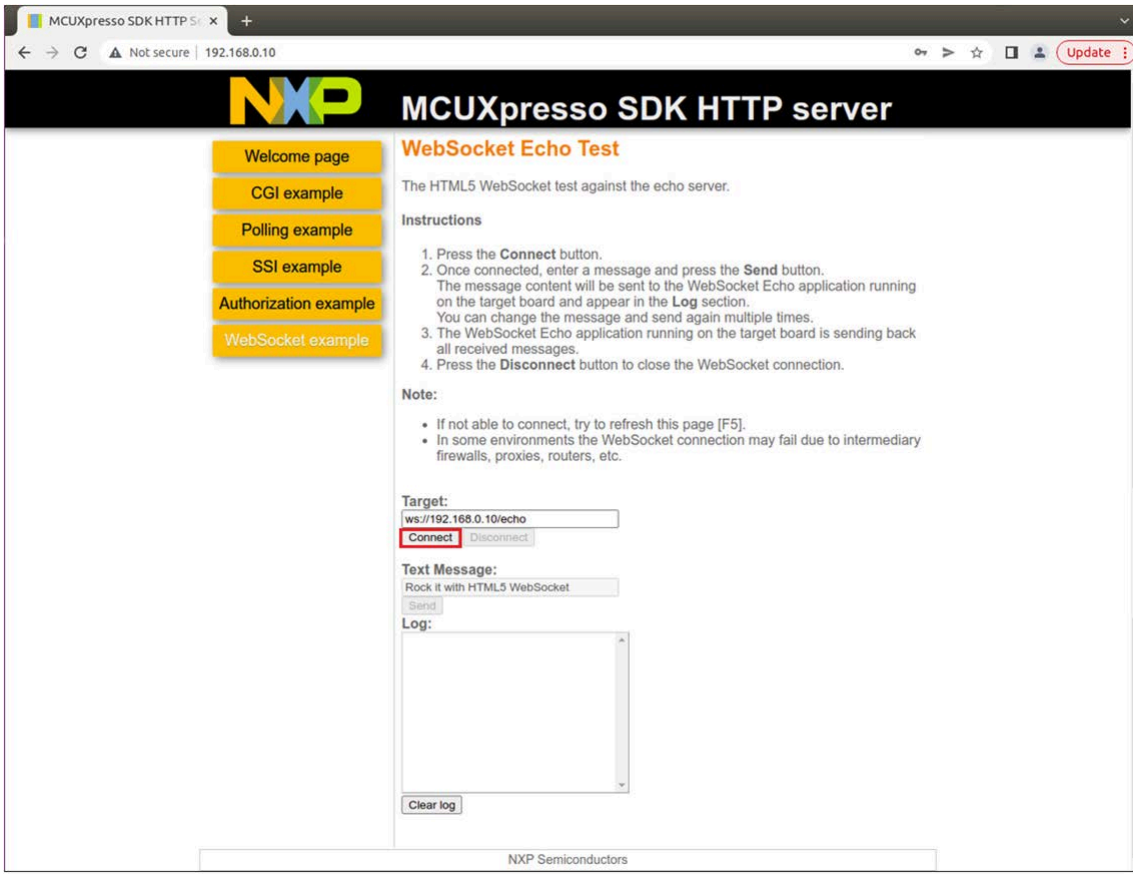


Figure 51. Connection request on Websocket example page

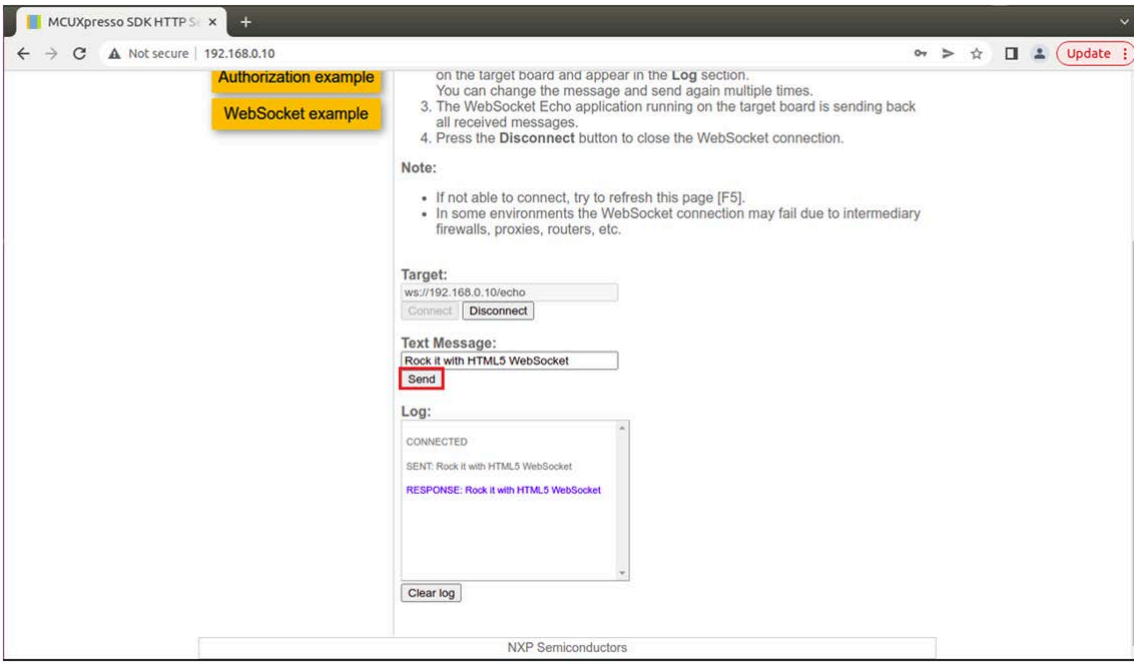


Figure 52. Sending a message on Websocket example page

4.7.2.8 Modify the static web page

To modify the contents of the static web page of wifi_httpsrv sample application:

- Modify, add or delete files in the directory *boards\<board_name>\wifi_examples\wifi_httpsrv\webpage*
- Run the script file *middleware\lwip\src\apps\httpsrv\mkfs\mkfs.pl <directory name>* to generate a new *httpsrv_fs_data.c*.
- Make sure to run the script in the directory with *httpsrv_fs_data.c* file.

For example:

```
C:\SDK\boards\rdrw610\wifi_examples\wifi_httpsrv>perl C:\SDK\middleware\lwip\src\apps
\httpsrv\mkfs\mkfs.pl webpage
Processing file webpage/auth.html
Processing file webpage/cgi.html
Processing file webpage/favicon.ico
Processing file webpage/httpsrv.css
Processing file webpage/index.html
Processing file webpage/NXP_logo.png
Processing file webpage/poll.html
Processing file webpage/request.js
Processing file webpage/ssi.shtml
Processing file webpage/websocket.html
Processing file webpage/welcome.html
Done.
```

Note: *To run the perl script, use a Windows tool like ActivePerl [10].*

- Make sure *httpsrv_fs_data.c* file has been overwritten with the newly generated content.
- Re-compile the wifi_httpsrv sample application and re-flash the board.

4.8 wifi_mqtt sample application

The wifi_mqtt application demonstrates an MQTT client on the lwIP TCP/IP stack with FreeRTOS.

The *wifi_mqtt* application features are summarized in Table 14.

Table 14. wifi_httpsrv sample application features

Features	Details
Wi-Fi and MQTT	Wi-Fi Station mode Wi-Fi Security MQTT Client DHCP Client

4.8.1 Wifi_mqtt application execution

Refer to Section 3.1 to Section 3.4 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs.

Refer to Section 2.1 for information about the serial console setup.

4.8.1.1 Start-up logs

The following logs show on the console once RW61x EVK is up and running.

```
*********************************************
 MQTT client example
*********************************************
[i] Initializing Wi-Fi connection...
MAC Address: 00:50:43:02:FE:01
450: [net] Initialized TCP/IP networking stack
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: my_network with password my_password
```

4.8.1.2 Connect Wi-Fi STA to Ex-AP

When the board is up and running, it tries to connect to Ex-AP and acquires the IP address through DHCP.

Make sure Ex-AP has a target SSID/password before running the application.

The network configuration is printed on the console when the board is connected to Ex-AP.

```
*********************************************
 MQTT client example
*********************************************
[i] Initializing Wi-Fi connection...
MAC Address: 00:50:43:02:FE:01
450: [net] Initialized TCP/IP networking stack
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: my_network with password my_password
[i] Connected to Wi-Fi
ssid: my_network
[!]passphrase: my_password
IPv4 Address     : 172.20.10.7
IPv4 Subnet mask : 255.255.255.240
IPv4 Gateway     : 172.20.10.1
```

4.8.1.3 Connect to MQTT broker and send messages

When the board is connected to Wi-Fi, it connects to HiveMQ MQTT broker to subscribe and publish messages, and to receive distributions from MQTT broker.

```
Resolving "broker.hivemq.com"...
Connecting to MQTT broker at 3.121.166.248...
MQTT client "nxp_fb2d9e35c2c415a6516ce025423dba68" connected.
Subscribing to the topic "lwip_topic/#" with QoS 0...
Subscribing to the topic "lwip_other/#" with QoS 1...
Going to publish to the topic "lwip_topic/100"...
Subscribed to the topic "lwip_topic/#".
Going to publish to the topic "lwip_topic/100"...
Subscribed to the topic "lwip_other/#".
Received 18 bytes from the topic "lwip_topic/100": "message from board"
Published to the topic "lwip_topic/100".
Received 18 bytes from the topic "lwip_topic/100": "message from board"
Published to the topic "lwip_topic/100".
Going to publish to the topic "lwip_topic/100"...
Received 18 bytes from the topic "lwip_topic/100": "message from board"
Published to the topic "lwip_topic/100".
Going to publish to the topic "lwip_topic/100"...
Published to the topic "lwip_topic/100".
Received 18 bytes from the topic "lwip_topic/100": "message from board"
Going to publish to the topic "lwip_topic/100"...
Received 18 bytes from the topic "lwip_topic/100": "message from board"
Published to the topic "lwip_topic/100".
```

Note: *The Wi-Fi network should have Internet access and no firewall limitation to connect the MQTT broker.*

4.9 wifi_test_mode sample application

The wifi_test_mode application demonstrates the CLI support for various RF and regulatory compliance tests. The application enables RF testing for the Wi-Fi module, and the measurement of RF parameters such as transmit power (2.4 GHz and 5GHz), RF packet counts, RF antenna configuration, and transmit standard 802.11 packets.

4.9.1 Wifi_test_mode application execution

Refer to Section 3.1 to Section 3.4 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs. Refer to section Section 2.1 for information about the serial console setup.

4.9.1.1 Run the application

The application starts with the welcome message. Press **Enter** for the command prompt.

```
=========================================
wifi test mode demo
=========================================
Initialize CLI
=========================================
Initialize WLAN Driver
=========================================
MAC Address: 00:50:43:02:FE:01
461: [net] Initialized TCP/IP networking stack
=========================================
app_cb: WLAN: received event 11
=========================================
app_cb: WLAN initialized
=========================================
WLAN Test Mode CLIs are initialized
=========================================
CLIs Available:
=========================================
help
wlan-version
wlan-mac
wlan-set-rf-test-mode
wlan-set-rf-tx-antenna <antenna>
wlan-get-rf-tx-antenna
wlan-set-rf-rx-antenna <antenna>
wlan-get-rf-rx-antenna
wlan-set-rf-band <band>
wlan-get-rf-band
wlan-set-rf-bandwidth <bandwidth>
wlan-get-rf-bandwidth
wlan-set-rf-channel <channel>
wlan-get-rf-channel
wlan-set-rf-radio-mode <radio_mode>
wlan-get-rf-radio-mode
wlan-set-rf-tx-power <tx_power> <modulation> <path_id>
wlan-set-rf-tx-cont-mode <enable_tx> <cw_mode> <payload_pattern> <cs_mode> <act_sub_ch> <tx_rate>
wlan-set-rf-tx-frame <start> <data_rate> <frame_pattern> <frame_len> <adjust_burst_sifs>
 <burst_sifs_in_us> <short_preamble> <act_sub_ch> <short_gi> <adv_coding> <tx_bf> <gf_mode> <stbc>
 <bssid>
wlan-get-and-reset-rf-per
wlan-set-rf-otp-mac-addr <mac_addr>
wlan-get-rf-otp-mac-addr
wlan-set-rf-otp-cal-data
wlan-get-rf-otp-cal-data
wlan-set-rf-he-tb-tx <enable> <qnum> <aid> <axq_mu_timer> <tx_power>
wlan-set-rf-trigger-frame
wlan-get-rf-trigger-frame
wlan-set-trigger-frame-parameters <index> <value>
wlan-get-trigger-frame-parameters <index>
=========================================
```

4.9.1.2 Prerequisite commands

This section includes the commands to start Wi-Fi RF test.

Enable Wi-Fi RF mode

Command to set Wi-Fi mode to RF test mode:

```
# wlan-set-rf-test-mode
RF Test Mode configuration successful
```

Set/get Wi-Fi RF band

Command usage:

```
# wlan-set-rf-band
Usage:
wlan-set-rf-band <band>
band: 0=2.4G, 1=5G
```

Command to set RF band:

```
# wlan-set-rf-band 0
RF Band configuration successful
```

Command to get RF band:

```
# wlan-get-rf-band
Configured RF Band is: 2.4G
```

Set/get Wi-Fi RF channel

Command usage:

```
# wlan-set-rf-channel
Usage:
wlan-set-rf-channel <channel>
```

Command to set RF channel:

```
# wlan-set-rf-channel 6
Channel configuration successful
```

Command to get RF channel:

```
# wlan-get-rf-channel
Configured channel is: 6
```

Set/get Wi-Fi RF bandwidth

Command usage:

```
# wlan-set-rf-bandwidth
Usage:
wlan-set-bandwidth <bandwidth>
        <bandwidth>:
                0: 20MHz
                1: 40MHz
```

Command to set RF bandwidth:

```
# wlan-set-rf-bandwidth 0
Bandwidth configuration successful
```

Command to get RF bandwidth:

```
# wlan-get-rf-bandwidth
Configured bandwidth is: 20MHz
```

Set/get Wi-Fi RF radio

Command usage:

```
# wlan-set-rf-radio-mode
Usage:
wlan-set-rf-radio-mode <radio_mode>
0: set the radio in power down mode
3: sets the radio in 5GHz band, 1X1 mode(path A)
11: sets the radio in 2.4GHz band, 1X1 mode(path A)
```

Command to set RF radio mode:

```
# wlan-set-rf-radio-mode 11
Set radio mode successful
```

Command to get RF radio mode:

```
# wlan-get-rf-radio-mode
Configured radio mode is: 11
```

4.9.1.3 Display and clear the received Wi-Fi packet count

Command to clear the received packet count and display the received multi-cast and error packet counts.

```
# wlan-get-and-reset-rf-per
PER is as below:
  Total Rx Packet Count                    : 6450
  Total Rx Multicast/Broadcast Packet Count: 4740
  Total Rx Packets with FCS error          : 2776
```

4.9.1.4 Wi-Fi antenna configuration

The following commands are used to set and get Wi-Fi TX/RX antenna configuration.

TX

Command usage:

```
# wlan-set-rf-tx-antenna
Usage:
wlan-set-rf-tx-antenna <antenna>
antenna: 1=Main, 2=Aux
```

Command to set TX antenna:

```
# wlan-set-rf-tx-antenna 1
Tx Antenna configuration successful
```

Command to get TX antenna configuration:

```
# wlan-get-rf-tx-antenna
Configured Tx Antenna is: Main
```

RX

Command usage:

```
# wlan-set-rf-rx-antenna
Usage:
wlan-set-rf-rx-antenna <antenna>
antenna: 1=Main, 2=Aux
```

Command to set RX antenna:

```
# wlan-set-rf-rx-antenna 2
Rx Antenna configuration successful
```

Command to get RX antenna configuration:

```
# wlan-get-rf-rx-antenna
Configured Rx Antenna is: Aux
```

4.9.1.5 Wi-Fi Tx power configuration

The following command is used to set the transmitter output power at the antenna using the stored calibration data. The power level is in dBm.

Command usage:

```
# wlan-set-rf-tx-power
Usage:
wlan-set-rf-tx-power <tx_power> <modulation> <path_id>
Power       (0 to 24 dBm)
Modulation  (0: CCK, 1:OFDM, 2:MCS)
Path ID     (0: PathA, 1:PathB, 2:PathA+B)
```

Command to set Tx power:

```
# wlan-set-rf-tx-power 8 1 1
Tx Power configuration successful
  Power          : 8 dBm
  Modulation     : OFDM
  Path ID        : PathB
```

4.9.1.6 Set Wi-Fi transmitter in continuous wave (CW) mode

The following command is used to set Wi-Fi transmitter in CW mode.

Command usage:

```
# wlan-set-rf-tx-cont-mode
Usage:
wlan-set-rf-tx-cont-mode <enable_tx> <cw_mode> <payload_pattern> <cs_mode> <act_sub_ch>
 <tx_rate>
Enable                (0:disable, 1:enable)
Continuous Wave Mode  (0:disable, 1:enable)
Payload Pattern       (0 to 0xFFFFFFFF) (Enter hexadecimal value)
CS Mode               (Applicable only when continuous wave is disabled) (0:disable,
 1:enable)
Active SubChannel     (0:low, 1:upper, 3:both)
Tx Data Rate          (Rate Index corresponding to legacy/HT/VHT rates)
To Disable:
  In Continuous Wave Mode:
    Step1: wlan-set-rf-tx-cont-mode 0 1 0 0 0 0
    Step2: wlan-set-rf-tx-cont-mode 0
  In none continuous Wave Mode:
    Step1: wlan-set-rf-tx-cont-mode 0
```

Note: *Refer to Table 15 and Table 16 for the data rate values.*

Command to enable CW mode:

```
# wlan-set-rf-tx-cont-mode 1 1 B496DEB6 0 0 7
Tx continuous configuration successful
  Enable                : enable
  Continuous Wave Mode  : enable
  Payload Pattern       : 0x7FFFFFFF
  CS Mode               : disable
  Active SubChannel     : low
  Tx Data Rate          : 7
```

Command to disable CW mode:

```
# wlan-set-rf-tx-cont-mode 0 1 0 0 0 0
Tx continuous configuration successful
  Enable                : disable
  Continuous Wave Mode  : enable
  Payload Pattern       : 0x00000000
  CS Mode               : disable
  Active SubChannel     : low
  Tx Data Rate          : 0
# wlan-set-rf-tx-cont-mode 0
Tx continuous configuration successful
  Enable                : disable
  Continuous Wave Mode  : disable
  Payload Pattern       : 0x00000000
  CS Mode               : disable
  Active SubChannel     : low
  Tx Data Rate          : 0
```

Note: *Disable CW mode when the test is completed. CW mode test and Tx frame test do not support parallel operation.*

Table 15. 802.11n/a/g/b data rate index

Data rate index	Data rate
1	1Mbits/sec
2	2Mbits/sec
3	5.5Mbits/sec
4	11Mbits/sec
5	Reserved
6	6Mbits/sec
7	9Mbits/sec
8	12Mbits/sec
9	18Mbits/sec
10	24Mbits/sec
11	36Mbits/sec
12	48Mbits/sec
13	54Mbits/sec
14	Reserved
15	HT_MCS 0
16	HT_MCS 1
17	HT_MCS 2
18	HT_MCS 3
19	HT_MCS 4
20	HT_MCS 5
21	HT_MCS 6
22	HT_MCS 7

Table 16. 802.11ac/802.11ax data rate index

Rate number format : (XYRR) X: 1 – 11ac VHT MCS rates, 2 – 11ax HE MCS rates Y: Number of streams. 1 – SS1 RR: MCS rate number		
Data rate Index	**Date rate XYRR**	**Data rate**
802.11ac VHT MCS rates		
4352	0x1100	VHT_SS1_MCS0
4353	0x1101	VHT_SS1_MCS1
4354	0x1102	VHT_SS1_MCS2
4355	0x1103	VHT_SS1_MCS3
4356	0x1104	VHT_SS1_MCS4
4357	0x1105	VHT_SS1_MCS5
4358	0x1106	VHT_SS1_MCS6
4359	0x1107	VHT_SS1_MCS7
4360	0x1108	VHT_SS1_MCS8
4361	0x1109	VHT_SS1_MCS9
802.11ax HE MCS rates		
8448	0x2100	HE_SS1_MCS0
8449	0x2101	HE_SS1_MCS1
8450	0x2102	HE_SS1_MCS2
8451	0x2103	HE_SS1_MCS3
8452	0x2104	HE_SS1_MCS4
8453	0x2105	HE_SS1_MCS5
8454	0x2106	HE_SS1_MCS6
8455	0x2107	HE_SS1_MCS7
8456	0x2108	HE_SS1_MCS8
8457	0x2109	HE_SS1_MCS9

4.9.1.7 Transmit standard 802.11 packets

The following command is used to continuously transmit packets, with an adjustable time gap of 0 to 255 microseconds between packets.

Command usage:

```
# wlan-set-rf-tx-frame
Usage:
wlan-set-rf-tx-frame <start> <data_rate> <frame_pattern> <frame_len> <adjust_burst_sifs>
 <burst_sifs_in_us> <short_preamble> <act_sub_ch> <short_gi> <adv_coding> <tx_bf>
 <gf_mode> <stbc> <bssid>
Enable                  (0:disable, 1:enable)
Tx Data Rate            (Rate Index corresponding to legacy/HT/VHT rates)(Enter
 hexadecimal value)
Payload Pattern         (0 to 0xFFFFFFFF) (Enter hexadecimal value)
Payload Length          (1 to 0x400) (Enter hexadecimal value)
Adjust Burst SIFS3 Gap  (0:disable, 1:enable)
Burst SIFS in us        (0 to 255us)
Short Preamble          (0:disable, 1:enable)
Active SubChannel       (0:low, 1:upper, 3:both)
Short GI                (0:disable, 1:enable)
Adv Coding              (0:disable, 1:enable)
Beamforming             (0:disable, 1:enable)
GreenField Mode         (0:disable, 1:enable)
STBC                    (0:disable, 1:enable)
BSSID                   (xx:xx:xx:xx:xx:xx)
To Disable:
wlan-set-rf-tx-frame 0
```

Note: *Refer to Table 15 and Table 16 for the data rate index values.*

Command to enable Tx frame:

```
# wlan-set-rf-tx-frame 1 13 2730 256 0 0 0 0 0 0 0 0 38:E6:0A:C6:1A:EC
Tx Frame configuration successful
  Enable                    : enable
  Tx Data Rate              : 19
  Payload Pattern           : 0x00002730
  Payload Length            : 0x00000256
  Adjust Burst SIFS3 Gap    : disable
  Burst SIFS in us          : 0 us
  Short Preamble            : disable
  Active SubChannel         : low
  Short GI                  : disable
  Adv Coding                : disable
  Beamforming               : disable
  GreenField Mode           : disable
  STBC                      : disable
  BSSID                     : 38:E6:0A:C6:1A:EC
```

Note: *<data_rate> parameter **13** in the command corresponds to the data rate index 19 (HT_MCS 4) in Table 15.*

Command to disable Tx frame:

```
# wlan-set-rf-tx-frame 0
Tx Frame configuration successful
  Enable                    : disable
  Tx Data Rate              : 0
  Payload Pattern           : 0x00000000
  Payload Length            : 0x00000001
  Adjust Burst SIFS3 Gap    : disable
  Burst SIFS in us          : 0 us
  Short Preamble            : disable
  Active SubChannel         : low
  Short GI                  : disable
  Adv Coding                : disable
  Beamforming               : disable
  GreenField Mode           : disable
  STBC                      : disable
  BSSID                     : 00:00:00:00:00:00
```

4.9.1.8 Transmit OFDMA packets

The section describes the commands to transmit 802.11ax OFDMA packets.

Enter/exit trigger frame response mode

The following command is used to enable/disable uplink OFDMA Tx for trigger frame response mode (respond to the received trigger frame by transmitting uplink OFDMA).

```
# wlan-set-rf-he-tb-tx <enable> <qnum> <aid> <axq_mu_timer> <tx_power>
```

Command to enter trigger frame response mode:

```
# wlan-set-rf-he-tb-tx 1 1 5 400 9
Set he_tb_tx successful
```

Command to exit trigger frame response mode:

```
# wlan-set-rf-he-tb-tx 0 1 5 400 9
Set he_tb_tx successful
```

Set/get trigger frame

The following commands are used to transmit OFDMA packets and get the OFDMA configurations.

Command to set trigger frame:

```
# wlan-set-rf-trigger-frame
Set rf trigger frame successful
```

Command to get trigger frame configurations:

```
# wlan-get-rf-trigger-frame
trigger_frame_parameters_default
index   name                                value
0       enable_tx                           1
1       standalone_hetb                     0
2       frmCtl_type                         1
3       frmCtl_sub_type                     2
4       duration                            5484
5       trig_common_field_trigger_type      0
6       trig_common_field_ul_len            256
7       trig_common_field_more_tf           0
8       trig_common_field_cs_required       0
9       trig_common_field_ul_bw             0
10      trig_common_field_ltf_type          1
11      trig_common_field_ltf_mode          0
12      trig_common_field_ltf_symbol        0
13      trig_common_field_ul_stbc           0
14      trig_common_field_ldpc_ess          1
15      trig_common_field_ap_tx_pwr         60
16      trig_common_field_pre_fec_pad_fct   1
17      trig_common_field_pe_disambig       0
18      trig_common_field_spatial_reuse     21845
19      trig_common_field_doppler           0
20      trig_common_field_he_sig2           511
21      trig_user_info_field_aid12          5
22      trig_user_info_field_ru_alloc_reg   0
23      trig_user_info_field_ru_alloc       65
24      trig_user_info_field_ul_coding_type 0
25      trig_user_info_field_ul_mcs         1
26      trig_user_info_field_ul_dcm         0
27      trig_user_info_field_ss_alloc       0
28      trig_user_info_field_ul_target_rssi 80
29      basic_trig_user_info_mpdu_mu_sf     0
30      basic_trig_user_info_tid_al         0
31      basic_trig_user_info_ac_pl          0
32      basic_trig_user_info_pref_ac        0
```

Set/get trigger frame parameters

Command usage:

```
# wlan-set-trigger-frame-parameters
Usage:
wlan-set-trigger-frame-parameters <index> <value>
index        (0 to 32) (Num of the trigger_frame parameters)
value        (value set to parameters)
```

Note: *Refer to the output of* `wlan-get-rf-trigger-frame` *command for the parameters, and to Figure 53 for RU index.*

Bandwidth	20MHz								
RU Index	0	1	2	3	4	5	6	7	8
RU Tone	26	26	26	26	26	26	26	26	26
RU Index	37		38			39		40	
RU Tone	52		52			52		52	
RU Index	53					54			
RU Tone	106					106			
RU Index	61								
RU Tone	242								

Figure 53. RU index values for 20 MHz

Command to set trigger frame parameters:

```
# wlan-set-trigger-frame-parameters 23 61
Set_trigger_frame_parameters:
trig_user_info_field_ru_alloc = 61
```

Command to get trigger frame parameters:

```
# wlan-get-trigger-frame-parameters 23
Get_trigger_frame_parameters:
trig_user_info_field_ru_alloc = 61
```

4.9.1.9 Set/get OTP MAC address

Commands to write and read MAC address into/from OTP:

- Set OTP MAC address.

```
# wlan-set-rf-otp-mac-addr C0:95:DA:01:1B:6C
OTP MAC address configuration successful
```

- Get OTP MAC address.

```
# wlan-get-rf-otp-mac-addr
OTP MAC address: C0:95:DA:01:1B:6C
```

4.9.1.10 Set/get OTP calibration data

Commands to set/get OTP calibration data:

Set OTP calibration data

- Replace the OTP calibration data in *wifi/wlcmgr/wlan_test_mode_tests.c* file.

```
const uint8_t otp_cal_data[] = {
    0x01, 0x00, 0x0F, 0x00, 0x88, 0x00, 0x00, 0x20, 0x44, 0x0F, 0x00, 0x00, 0x00, 0x20, 0xFF, 0xFF,
    0x40, 0x00, 0x77, 0x00, 0x29, 0x12, 0x00, 0x00, 0x00, 0x10, 0x00, 0x04, 0x6A, 0xB1, 0x02, 0x00,
    0x00, 0x3F, 0x01, 0x00, 0x00, 0x0D, 0x00, 0x18, 0x97, 0x53, 0x00, 0x00, 0x00, 0x38, 0x39, 0x22,
    0x3C, 0x55, 0xBC, 0x68, 0x6A, 0x37, 0xBE, 0x82, 0x22, 0xB4, 0x41, 0x64, 0x8D, 0xCE, 0x00, 0x1C,
    0x9F, 0x37, 0x00, 0x00, 0x00, 0x54, 0x02, 0x04, 0x00, 0x01, 0x00, 0x00, 0x00, 0x08, 0x00, 0x2D,
    0xC6, 0xC0, 0x43, 0x00, 0x00, 0x66, 0x00, 0x00, 0x00, 0x50, 0x00, 0x1C, 0x49, 0x5F, 0x00, 0x00,
    0x00, 0x70, 0x02, 0x05, 0x00, 0x01, 0x00, 0x00, 0x00, 0x08, 0x00, 0x2D, 0xC6, 0xC0, 0x43, 0x00,
    0x00, 0x77, 0x00, 0x00, 0x00, 0x50, 0x00, 0x18, 0xB2, 0x68, 0xFF, 0xFF, 0xFF, 0xFF, 0xD3, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

Note: *The driver gets the OTP calibration data from the otp_cal_data array in wifi/wlcmgr/wlan_test_mode_tests.c, and sends the data to the FW through MFG commands.*

- Write calibration data into OTP:

```
# wlan-set-rf-otp-cal-data
OTP cal data configuration successful
```

Get OTP calibration data.

```
# wlan-get-rf-otp-cal-data
```

Note: *The command returns the status code 1 or 0 based on the status flag set if wlan-set-rf-otp-cal-data is successful. When OTP calibration data is set successfully, the expected output is* `OTP cal data read successfully: 1`.

4.9.1.11 Get the Wi-Fi driver and firmware versions

Command to get the Wi-Fi driver and firmware version:

```
# wlan-version
WLAN Driver Version   : v1.3.r34.p46
WLAN Firmware Version : rw610w-V1, RF878X, FP91, 18.91.2.p8, PVE_FIX 1
```

4.9.1.12 Get the Wi-Fi MAC address

Command to get the Wi-Fi MAC address:

```
# wlan-mac
MAC address
STA MAC Address: 00:50:43:02:FE:01
UAP MAC Address: 00:50:43:02:FE:01
```

4.9.1.13 Example of command sequence to adjust Tx power in 2.4 GHz

The radio is configured as:

- 2.4 GHz band
- Channel 6
- 20 MHz bandwidth
- 6 Mbps legacy data rate
- Test pattern transmitted: 0x00000AAA
- Output power: set to +10 dBm, then adjusted to +15 dBm

Table 17. Tx command sequences for 2.4 GHz

Step	Operation	Command
1	Set RF test mode	`# wlan-set-rf-test-mode` `RF Test Mode configuration successful`
2	Set radio mode	`# wlan-set-rf-radio-mode 11` `Set radio mode successful`
3	Set RF band	`# wlan-set-rf-band 0` `RF Band configuration successful`
4	Set RF channel	`# wlan-set-rf-channel 6` `Channel configuration successful`
5	Set RF bandwidth	`# wlan-set-rf-bandwidth 0` `Bandwidth configuration successful`
6	Set Tx antenna	`# wlan-set-rf-tx-antenna 1` `Tx Antenna configuration successful`

7	Set output power to +10 dBm	`# wlan-set-rf-tx-power 10 1 0`
		`Tx Power configuration successful`
		`Power : 10 dBm`
		`Modulation : OFDM`
		`Path ID : PathA`

8	Set continuous transmit mode	`# wlan-set-rf-tx-cont-mode 1 0 0xAAA 0 3 5`
		`Tx continuous configuration successful`
		`Enable : enable`
		`Continuous Wave Mode : disable`
		`Payload Pattern : 0x00000AAA`
		`CS Mode : disable`
		`Active SubChannel : both`
		`Tx Data Rate : 5`

9	Stop transmission	`# wlan-set-rf-tx-cont-mode 0`

Table 17. Tx command sequences for 2.4 GHz*...continued*

Step	Operation	Command

10	Set output power to +15 dBm	`# wlan-set-rf-tx-power 15 1 0`
		`Tx Power configuration successful`
		`Power : 15 dBm`
		`Modulation : OFDM`
		`Path ID : PathA`

11	Restart transmission	`# wlan-set-rf-tx-cont-mode 1 0 0xAAA 0 3 5`
		`Tx continuous configuration successful`
		`Enable : enable`
		`Continuous Wave Mode : disable`
		`Payload Pattern : 0x00000AAA`
		`CS Mode : disable`
		`Active SubChannel : both`
		`Tx Data Rate : 5`

12	Stop transmission	`# wlan-set-rf-tx-cont-mode 0`

4.9.1.14 Example of command sequence to adjust Tx power in 5 GHz

The radio is configured as:

- 5 GHz band
- Channel 36
- 20 MHz bandwidth
- MCS0 HT data rate
- Test pattern transmitted: 0x00BBBAAA
- Output power: set to +10 dBm, then adjusted to +8 dBm

Table 18. Tx command sequence for 5 GHz

Step	Operation	Command
1	Set RF test mode	`# wlan-set-rf-test-mode` `RF Test Mode configuration successful`
2	Set radio mode	`# wlan-set-rf-radio-mode 3` `Set radio mode successful`
3	Set RF band	`# wlan-set-rf-band 1` `RF Band configuration successful`
4	Set RF channel	`# wlan-set-rf-channel 36` `Channel configuration successful`
5	Set RF bandwidth	`# wlan-set-rf-bandwidth 0` `Bandwidth configuration successful`
6	Set Tx antenna	`# wlan-set-rf-tx-antenna 1` `Tx Antenna configuration successful`

7	Set output power to +10 dBm	`# wlan-set-rf-tx-power 10 1 0`
		`Tx Power configuration successful`
		`Power : 10 dBm`
		`Modulation : OFDM`
		`Path ID : PathA`

8	Set continuous transmit mode	`# wlan-set-rf-tx-cont-mode 1 0 0xBBBAAA 0 3 14`
		`Tx continuous configuration successful`
		`Enable : enable`
		`Continuous Wave Mode : disable`
		`Payload Pattern : 0x00BBBAAA`
		`CS Mode : disable`
		`Active SubChannel : both`
		`Tx Data Rate : 14`

9	Stop transmission	`# wlan-set-rf-tx-cont-mode 0`

10	Set output power to +8 dBm	`# wlan-set-rf-tx-power 8 1 0`
		`Tx Power configuration successful`
		`Power : 8 dBm`
		`Modulation : OFDM`
		`Path ID : PathA`

Table 18. Tx command sequence for 5 GHz*...continued*

Step	Operation	Command

11	Restart transmission	`# wlan-set-rf-tx-cont-mode 1 0 0xBBBAAA 0 3 14`
		`Tx continuous configuration successful`
		`Enable : enable`
		`Continuous Wave Mode : disable`
		`Payload Pattern : 0x00BBBAAA`
		`CS Mode : disable`
		`Active SubChannel : both`
		`Tx Data Rate : 14`

12	Stop transmission	`# wlan-set-rf-tx-cont-mode 0`

4.10 wifi_wpa_supplicant sample application

The *wifi_wpa_supplicant* application demonstrates the CLI support usage using wpa supplicant (host-based). The application includes similar commands as *wifi_cli* application, and new commands for host-based supplicant. That is WPA Enterprise and WPS.

Table 19. wifi_wpa_supplicant sample application features

Features	Details
Wi-Fi	Wi-Fi Host based supplicant Wi-Fi Mobile AP mode Wi-Fi Station mode Wi-Fi Scan Wi-Fi Roaming Wi-Fi IEEEPS power save mode Wi-Fi deep sleep power save mode Wi-Fi host sleep/wowlan WPA Enterprise WPS Wi-Fi Easy connect Wi-Fi Cloud keep alive Wi-Fi Turbo mode
IPerf	TCP Client and Server TCP Client dual mode (Tx and Rx in simultaneous) TCP Client trade-off mode (Tx and Rx individual) UDP Client and Server UDP Client dual mode (Tx and Rx in simultaneous) UDP Client trade-off mode (Tx and Rx individual)

4.10.1 wifi_wpa_supplicant application execution

Refer to Section 3.1 and Section 3.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Refer to Section 2.1 for information about the serial console setup.

4.10.1.1 Start-up logs

A welcome message pops up on the terminal when the demo application starts. This section describes the available Wi-Fi commands. Press **Enter** for the command prompt. Press **tab** or type help to list out all available CLI commands.

```
=========================================
wifi wpa supplicant demo
=========================================
host init done
Initialize CLI
=========================================
CLI Build: Mar 28 2024 [10:53:47]
Copyright  2024  NXP
MCU Board: RD-RW61X-BGA
=========================================
MCU wakeup source 0x0...
Initialize WLAN Driver
=========================================
MAC Address: C0:95:DA:01:1C:6C
supplicant_main_task: 298 Starting wpa_supplicant thread with debug level: 3
Successfully initialized wpa_supplicant
iface_cb: iface ml1 ifindex 2 c0:95:da:01:1b:6c
Using interface ml1
Initializing interface 0: ml1
PKG_TYPE: BGA
Set BGA tx power table data
app_cb: WLAN initialized
=========================================
WLAN CLIs are initialized
=========================================
ENHANCED WLAN CLIs are initialized
=========================================
HOST SLEEP CLIs are initialized
=========================================
CLIs Available:
=========================================
help
clear
wlan-version
wlan-mac
wlan-thread-info
wlan-net-stats
wlan-set-mac <MAC_Address>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-connect-opt <profile_name> ...
wlan-reassociate
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-set-ps-cfg <null_pkt_interval>
wlan-deep-sleep-ps <0/1>
wlan-get-beacon-interval
wlan-wnm-ps <0/1> <sleep_interval>
wlan-set-max-clients-count <max clients count>
wlan-rts <sta/uap> <rts threshold>
wlan-frag <sta/uap> <fragment threshold>
wlan-host-11k-enable <0/1>
wlan-host-11k-neighbor-req [ssid <ssid>]
wlan-host-11v-bss-trans-query <0..16>
wlan-mbo-nonprefer-ch "<oper_class>:<chan>:<preference>:<reason>
 <oper_class>:<chan>:<preference>:<reason>"
```

```
wlan-mbo-set-cell-capa <cell capa: 1/2/3(default)>
wlan-mbo-set-oce <oce: 1(default)/2>
wlan-set-okc <okc: 0(default)/1>
wlan-pmksa-list
wlan-pmksa-flush
wlan-set-scan-interval <scan_int: in seconds>
wlan-sta-filter  <filter mode> [<mac address list>]
wlan-get-log <sta/uap> <ext>
wlan-tx-pert <0/1> <STA/UAP> <p> <r> <n>
wlan-roaming <0/1> <rssi_threshold>
wlan-multi-mef <ping/arp/multicast/del> [<action>]
wlan-wakeup-condition <mef/wowlan wake_up_conds>
wlan-auto-host-sleep <enable> <mode> <rtc_timer> <periodic>
wlan-send-hostcmd
wlan-ext-coex-uwb
wlan-set-uap-hidden-ssid <0/1/2>
wlan-eu-crypto-rc4 <EncDec>
wlan-eu-crypto-aes-wrap <EncDec>
wlan-eu-crypto-aes-ecb <EncDec>
wlan-eu-crypto-ccmp-128 <EncDec>
wlan-eu-crypto-ccmp-256 <EncDec>
wlan-eu-crypto-gcmp-128 <EncDec>
wlan-eu-crypto-gcmp-256 <EncDec>
wlan-mem-access <memory_address> [<value>]
wlan-eu-validation <value>
wlan-ft-roam <bssid> <channel>
wlan-set-antcfg <ant_mode> <evaluate_time> <evaluate_mode>
wlan-get-antcfg
wlan-scan-channel-gap <channel_gap_value>
wlan-wmm-stat <bss_type>
wlan-reset
wlan-set-regioncode <region-code>
wlan-get-regioncode
wlan-11d-enable <sta/uap> <0/1>
wlan-uap-set-ecsa-cfg <block_tx> <oper_class> <new_channel> <switch_count> <bandwidth>
wlan-csi-cfg
wlan-set-csi-param-header  <csi_enable> <head_id> <tail_id> <chip_id> <band_config> <channel>
 <csi_monitor_enable> <ra4us>
wlan-set-csi-filter <opt> <macaddr> <pkt_type> <type> <flag>
wlan-txrx-histogram <action> <enable>
wlan-subscribe-event <action> <type> <value> <freq>
wlan-reg-access <type> <offset> [value]
wlan-uapsd-enable <uapsd_enable>
wlan-uapsd-qosinfo <qos_info>
wlan-uapsd-sleep-period <sleep_period>
wlan-tx-ampdu-prot-mode <mode>
wlan-rssi-low-threshold <threshold_value>
wlan-rx-abort-cfg
wlan-set-rx-abort-cfg-ext enable <enable> margin <margin> ceil <ceil_thresh> floor <floor_thresh>
wlan-get-rx-abort-cfg-ext
wlan-cck-desense-cfg
wlan-generate-wps-pin
wlan-start-wps-pbc
wlan-start-wps-pin <8 digit pin>
wlan-wps-cancel
wlan-start-ap-wps-pbc
wlan-start-ap-wps-pin <8 digit pin>
wlan-wps-ap-cancel
wlan-dpp-configurator-add
wlan-dpp-configurator-params  conf=<sta-dpp/ap-dpp> ssid=<ascii> configurator=<id>
wlan-dpp-mud-url https://...
wlan-dpp-bootstrap-gen type=<qrcode> chan=<op>/<ch> mac=<addr>
wlan-dpp-bootstrap-get-uri <bootstrap_gen id>
wlan-dpp-qr-code <DPP:...>
wlan-dpp-auth-init  peer=<id> role=<enrollee/configurator>
wlan-dpp-listen <frequency>...
wlan-dpp-stop-listen
wlan-dpp-pkex-add  own=<bootstrap_id> identifier=<string> code=<string>
wlan-dpp-chirp  own=<bootstrap id> listen=<freq>...
wlan-dpp-reconfig <network id> ...
wlan-dpp-configurator-sign  conf=<sta-dpp/ap-dpp> ssid=<ascii> configurator=<id>
wlan-net-monitor-cfg
wlan-set-monitor-filter <opt> <macaddr>
wlan-set-monitor-param <action> <monitor_activity> <filter_flags> <radio_type> <chan_number>
```

```
wlan-set-tsp-cfg <enable> <backoff> <highThreshold> <lowThreshold> <dutycycstep> <dutycycmin>
 <highthrtemp> <lowthrtemp>
wlan-get-tsp-cfg
wlan-get-signal
wlan-set-ips <option>
wlan-set-debug-htc <count> <vht> <he> <rxNss> <channelWidth> <ulMuDisable> <txNSTS> <erSuDisable>
 <erSuDisable> <erSuDisable>
wlan-enable-disable-htc <option>
wlan-set-su <0/1>
wlan-set-forceRTS <0/1>
wlan-set-mmsf <enable> <Density> <MMSF>
wlan-get-mmsf
wlan-get-turbo-mode <STA/UAP>
wlan-set-turbo-mode <STA/UAP> <mode>
wlan-set-multiple-dtim <value>
wlan-cloud-keep-alive <start/stop/reset>
wlan_tcp_client dst_ip <dst_ip> src_port <src_port> dst_port <dst_port>
wlan-set-country <country_code_str>
wlan-set-country-ie-ignore <0/1>
wlan-single-ant-duty-cycle <enable/disable> [<Ieee154Duration> <TotalDuration>]
wlan-dual-ant-duty-cycle <enable/disable> [<Ieee154Duration> <TotalDuration>
 <Ieee154FarRangeDuration>]
wlan-sta-inactivityto <n> <m> <l> [k] [j]
wlan-get-temperature
wlan-auto-null-tx <start/stop>
wlan-get-txpwrlimit <subband>
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting> <autoTx_set>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
wlan-get-pmfcfg
wlan-uap-get-pmfcfg
wlan-set-ed-mac-mode <interface> <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
wlan-get-ed-mac-mode <interface>
wlan-set-tx-omi <interface> <tx-omi> <tx-option> <num_data_pkts>
wlan-set-toltime <value>
wlan-set-rutxpwrlimit
wlan-11ax-cfg <11ax_cfg>
wlan-11ax-bcast-twt <bcast_twt_cfg>
wlan-11ax-twt-setup <twt_cfg>
wlan-11ax-twt-teardown <twt_cfg>
wlan-11ax-twt-report <twt_report_get>
wlan-get-tsfinfo <format-type>
wlan-set-clocksync <mode> <role> <gpio_pin> <gpio_level> <pulse width>
wlan-suspend <power mode>
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6 address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
wlan-hlr-cli <standard hlr cli options>
wlan-read-gsm-triplets <imsi> <kc> <sres> <rand>
wlan-read-milenage <imsi> <ki> <opc> <amf> <sqn>
wlan-set-rtc-time <year> <month> <day> <hour> <minute> <second>
wlan-get-rtc-time
wlan-read-usb-file <type:ca-cert/client-cert/client-key> <file name>
wlan-dump-usb-file <type:ca-cert/client-cert/client-key>
========================================
```

4.10.1.2 Add a network profile

Before adding a network profile for station (STA) and mobile AP (uAP) modes, check the command usage for different EAP methods.

```
# wlan-add
Usage:
For Station interface
  For DHCP IP Address assignment:
    wlan-add <profile_name> ssid <ssid> [wpa2 <psk/psk-sha256/ft-psk> <secret>] [mfpc <1> mfpr <0>]
      If using WPA2 security, set the PMF configuration as mentioned above.
If using proactive key caching set pkc as 1, to disable set to 0(default), if okc is set this is not
 used.
If using specific ciphers, set the group, pairwise and group mgmt using gc, pc and gmc options.
supported ciphers: ccmp=0x10, gcmp=0x40, gcmp_256=0x100, ccmp_256=0x200
supported group mgmt ciphers: aes_128_cmac=0x20, bip_gmac_128=0x800, bip_gmac_256=0x1000,
 bip_cmac_256=0x2000
    wlan-add <profile_name> ssid <ssid> [wpa3-sb/wpa3-sb-192] [eap-tls/eap-tls-sha256/eap-tls-ft/
eap-tls-ft-sha384 [tls_cipher <ECC_P384/RSA_3K>] id <identity> [key_passwd <client_key_passwd>][hash
 <hash>][domain_match <domain_match_string>][domain_suffix_match <domain_suffix_match_string>]]
 [mfpc <1> mfpr <0/1>] [mc 0x10 uc 0x10 gc 0x20]
    wlan-add <profile_name> ssid <ssid> [wpa3-sb/wpa3-sb-192] [eap-ttls aid <anonymous identity>
 [key2_passwd <client_key2_passwd>]] [mfpc <1> mfpr <0/1>]
    wlan-add <profile_name> ssid <ssid> [wpa3-sb/wpa3-sb-192] [eap-ttls-mschapv2 aid <anonymous
 identity> id <identity> pass <password> [key_passwd <client_key_passwd>]] [mfpc <1> mfpr <0/1>]
    wlan-add <profile_name> ssid <ssid> [wpa3-sb/wpa3-sb-192] [eap-peap-mschapv2/eap-peap-tls/eap-
peap-gtc [ver 0/1] id <identity> pass <password> [key_passwd <client_key_passwd>]] [mfpc <1> mfpr
 <0/1>]
    wlan-add <profile_name> ssid <ssid> [wpa3-sb/wpa3-sb-192] [eap-fast-mschapv2/eap-fast-gtc aid
 <anonymous identity> id <identity> pass <password> [key_passwd <client_key_passwd>]] [mfpc <1> mfpr
 <0/1>]
    wlan-add <profile_name> ssid <ssid> [eap-sim/eap-aka/eap-aka-prime id <identity> pass
 <password>]
      If using WPA2/WPA3 Enterprise security, set the PMF configuration as required.
    wlan-add <profile_name> ssid <ssid> <owe_only> [og <"19 20 21">] mfpc 1 mfpr 1
      If using OWE only security, always set the PMF configuration.
    wlan-add <profile_name> ssid <ssid> [wpa3 sae/ft-sae <secret> [sg <"19 20 21">] [pwe <0/1/2>]
 mfpc <1> mfpr <0/1>]
      If using WPA3 SAE security, always set the PMF configuration.
    wlan-add <profile_name> ssid <ssid> [wpa2 psk psk-sha256 <secret> wpa3 sae <secret>] [mfpc <1>
 mfpr <0>]
      If using WPA2/WPA3 Mixed security, set the PMF configuration as mentioned above.
  For static IP address assignment:
    wlan-add <profile_name> ssid <ssid>
    ip:<ip_addr>,<gateway_ip>,<netmask>
    [bssid <bssid>] [channel <channel number>]
    [wpa2 <psk/psk-sha256/ft-psk> <secret>] [wpa3-sb/wpa3-sb-192] [eap-tls/eap-tls-sha256/eap-tls-
ft/eap-tls-ft-sha384] [owe_only] [wpa3 sae/ft-sae <secret>] [mfpc <0/1> mfpr <0/1>]
For Micro-AP interface
    wlan-add <profile_name> ssid <ssid>
    ip:<ip_addr>,<gateway_ip>,<netmask>
    role uap [bssid <bssid>]
    [channel <channelnumber>]
    [wpa2 <psk/psk-sha256> <secret>] [wpa3 sae <secret> [sg <"19 20 21">] [pwe <0/1/2>] [tr
 <0/1/2/4/8>]]
    [ft-psk <secret>] [wpa3 ft-sae <secret>]
    [wpa3-sb/wpa3-sb-192] [eap-tls/eap-tls-sha256/eap-ttls/eap-ttls-mschapv2/eap-peap-mschapv2/eap-
peap-tls/eap-peap-gtc/eap-fast-mschapv2/eap-fast-gtc/eap-sim/eap-aka/eap-aka-prime]
    [eap-tls-ft/eap-tls-ft-sha384]
    [owe_only [og <"19 20 21">]]
    [mfpc <0/1>] [mfpr <0/1>]
    If using eap-sim/eap-aka/eap-aka-prime use read_gsm_triplets to add GSM authentication triplets
 and read_milenage to add Milenage keys and hlr_cli to start hlr_auc_gw
If setting dtim
The value of dtim is an integer. The default value is 10.
NoteSetting the channel value greater than or equal to 36 is mandatory,
      if UAP bandwidth is set to 80MHz.
    [capa <11ax/11ac/11n/legacy>]
If Set channel to 0, set acs_band to 0 1.
0: 2.4GHz channel   1: 5GHz channel  Not support to select dual band automatically.
Error: invalid number of arguments
```

4.10.1.3 Station mode (connect to AP)

This section demonstrates how to connect to External AP with Enterprise security.

Note: *A second RW61x is used as an External AP on which the radius certificates are configured. To configure your own certificates, refer to Section 4.10.1.5.*

WPA2 Enterprise Security

- Issue the command to add the network profile and configure the device in station mode using EAP-TLS method:

```
# wlan-add EapNet ssid EapNet_AP eap-tls id client1 key_passwd whatever
```

- Connect to the AP network using the save network profile:

```
# wlan-connect EapNet
Connecting to network...
Use 'wlan-stat' for current connection status.
# ml1: SME: Trying to authenticate with c0:95:da:01:20:c2 (SSID='EapNet_AP' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:20:c2 (SSID='EapNet_AP' freq=2437 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with c0:95:da:01:20:c2
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: CTRL-EVENT-EAP-STARTED EAP authentication started
ml1: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=13
ml1: CTRL-EVENT-EAP-METHOD EAP vendor 0 method 13 (TLS) selected
ml1: CTRL-EVENT-EAP-PEER-CERT depth=1 subject='C=IN, ST=MH, L=PUNE, O=NXP, CN=CA,
 emailAddress=ca@nxp.com'
hash=4f7f0a703ca723e3f0e5c7d11f7f5e0ec5d68975791370354f2a006f0100d4d2
ml1: CTRL-EVENT-EAP-PEER-CERT depth=0 subject='C=IN, ST=MH, L=PUNE, O=NXP, CN=SERVER,
 emailAddress=server@nxp.com'
hash=86f7f32f4450980966beac9df4695df908d532c0c1116e52d2ba07fef41cc764
ml1: CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
ml1: PMKSA-CACHE-ADDED c0:95:da:01:20:c2 0
app_cb: WLAN: authenticated to network
ml1: WPA: Key negotiation completed with c0:95:da:01:20:c2 [PTK=CCMP GTK=CCMP]
ml1: CTRL-EVENT-CONNECTED - Connection to c0:95:da:01:20:c2 completed [id=0 id_str=]
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [EapNet_AP]
IPv4 Address: [192.168.10.2]
```

Note: *Once connected to the AP, the console output shows the client successfully connected to AP with* `ssid = [EapNet_AP]` ***and*** `IP address = [192.168.10.2]`.

WPA3 enterprise security

To use WPA3 Suite B or Suite B 192 bit enterprise security:

- Add `wpa3-sb` or `wpa3-sb-192` before EAP security type (applies to all EAP securities).

```
# wlan-add EapNet ssid EapNet_AP <wpa3-sb/wpa3-sb-192> eap-tls id client1 key_passwd
 whatever mfpc 1 mfpr 1
```

- Connect to the AP network using the saved network profile:

```
# wlan-connect EapNet
Connecting to network...
Use 'wlan-stat' for current connection status.
# ml1: SME: Trying to authenticate with c0:95:da:01:20:c2 (SSID='EapNet_AP' freq=5745 MHz)
ml1: Trying to associate with c0:95:da:01:20:c2 (SSID='EapNet_AP' freq=5745 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with c0:95:da:01:20:c2
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: CTRL-EVENT-EAP-STARTED EAP authentication started
ml1: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=13
ml1: CTRL-EVENT-EAP-METHOD EAP vendor 0 method 13 (TLS) selected
ml1: CTRL-EVENT-EAP-PEER-CERT depth=1 subject='C=IN, ST=MH, L=PUNE, O=NXP, CN=CA,
 emailAddress=ca@nxp.com'
hash=4f7f0a703ca723e3f0e5c7d11f7f5e0ec5d68975791370354f2a006f0100d4d2
ml1: CTRL-EVENT-EAP-PEER-CERT depth=0 subject='C=IN, ST=MH, L=PUNE, O=NXP, CN=SERVER,
 emailAddress=server@nxp.com'
hash=86f7f32f4450980966beac9df4695df908d532c0c1116e52d2ba07fef41cc764
ml1: CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
app_cb: WLAN: authenticated to network
ml1: WPA: Key negotiation completed with c0:95:da:01:20:c2 [PTK=CCMP-256 GTK=CCMP-256]
ml1: CTRL-EVENT-CONNECTED - Connection to c0:95:da:01:20:c2 completed [id=0 id_str=]
ml1: PMKSA-CACHE-ADDED c0:95:da:01:20:c2 0
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [EapNet_AP]
IPv4 Address: [192.168.10.2]
```

Note: *Once connected to the AP, the console output shows that the Client is connected to the AP with* `ssid = [EapNet]` ***and*** `IP address = [192.168.10.2].`

Other security options

OWE

```
# wlan-add oweNet ssid oweNet_AP owe_only mfpc 1 mfpr 1
```

EAP_SIM_WPA2

```
# wlan-add abc ssid EAP eap-sim id 1232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123
```

EAP_SIM_WPA3

```
# wlan-add abc ssid EAP wpa3-sb-192 eap-sim id 1232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123 mfpc 1
 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

EAP_AKA_WPA2

```
# wlan-add abc ssid EAP eap-aka id 0232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123
```

EAP_AKA_WPA3

```
# wlan-add abc ssid EAP wpa3-sb-192 eap-aka id 0232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123
mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

AKA_PRIME_WPA2

```
# wlan-add abc ssid EAP eap-aka-prime id 6555444333222111 pass \
5122250214c33e723a5dd523fc145fc0:981d464c7c52eb6e5036234984ad0bcf:000000000123
```

AKA_PRIME_WPA3

```
# wlan-add abc ssid EAP wpa3-sb-192 eap-aka-prime id 6555444333222111 pass \
5122250214c33e723a5dd523fc145fc0:981d464c7c52eb6e5036234984ad0bcf:000000000123
mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

FT-SAE

```
# wlan-add abc ssid FTSAE wpa3 ft-sae 12345678 mfpc 1 mfpr 1
```

FT_Enterprise_WPA2

```
# wlan-add abc ssid FTEAP eap-tls-ft id client1 key_passwd whatever
```

FT_Enterprise_WPA3

```
# wlan-add abc ssid FTEAP wpa3-sb-192 eap-tls-ft-sha384 id client1 key_passwd whatever
 mfpc 1 mfpr 1
```

4.10.1.4 Mobile AP mode

Use the following commands to add the network profile to configure the device in Enterprise AP mode. Use the SSID, IP details, role, channel, security, user id and password of your AP in the argument.

Note: *To generate your own certificates, refer to Section 4.10.1.5.*

WPA2 EAP TLS

```
# wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0
role uap channel 6 eap-tls id client1 key_passwd whatever
```

WPA3 EAP TLS (suite B/suite B 192 bit)

```
# wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
 channel 149
<wpa3-sb/wpa3-sb-192> eap-tls id client1 key_passwd whatever mfpc 1 mfpr 1
```

- Start the AP using saved network profile:

```
# wlan-start-network EapNet
[wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is on.
ua2: interface state UNINITIALIZED->COUNTRY_UPDATE
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
# ua2: interface state COUNTRY_UPDATE->ENABLED
: AP-ENABLED
PKG_TYPE: BGA
Set BGA tx power table data
app_cb: WLAN: UAP Started
========================================
Mobile AP "EapNet_AP" started successfully
========================================
DHCP Server started successfully
========================================
```

- Connect the wireless client to the created AP.

Example of log showing that the Client is associated successfully:

```
app_cb: WLAN: UAP Started
ua2: STA c0:95:da:01:1b:6c IEEE 802.11: associated (aid 1)
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1b:6c
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=13
ml1: CTRL-EVENT-EAP-PEER-CERT depth=1 subject='C=IN, ST=MH, L=PUNE, O=NXP, CN=CA,
emailAddress=ca@nxp.com'
 hash=4f7f0a703ca723e3f0e5c7d11f7f5e0ec5d68975791370354f2a006f0100d4d2
ml1: CTRL-EVENT-EAP-PEER-CERT depth=0 subject='C=IN, ST=MH, L=PUNE, O=NXP, CN=Client,
emailAddress=client@nxp.com'
 hash=8bb701aedec525fbc4934c3a53a00adbcfb86f8c307504bcf600c004fb79148b
: CTRL-EVENT-EAP-SUCCESS c0:95:da:01:1b:6c
ua2: STA c0:95:da:01:1b:6c WPA: pairwise key handshake completed (RSN)
: EAPOL-4WAY-HS-COMPLETED c0:95:da:01:1b:6c
: AP-STA-CONNECTED c0:95:da:01:1b:6c
app_cb: WLAN: UAP a Client Connected
========================================
Client => C0:95:DA:01:1B:6C Connected with mobile AP
========================================
ua2: STA c0:95:da:01:1b:6c IEEE 802.1X: authenticated - EAP type: 0 (unknown)
```

- Get the associated clients list:

```
# wlan-get-uap-sta-list
Number of STA = 1
STA 1 information:
=====================
MAC Address: C0:95:DA:01:1B:6C
Power mfg status: power save
Rssi : -43 dBm
```

- Get the IP and MAC information for the associated clients:

```
# dhcp-stat
DHCP Server Lease Duration : 86400 seconds
Client IP       Client MAC
192.168.10.2    C0:95:DA:01:1B:6C
```

Other security options

OWE

```
# wlan-add oweNet ssid oweNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0
role uap owe_only mfpc 1 mfpr 1
```

EAP_SIM_WPA2

```
# wlan-add abc ssid EAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 36
 eap-sim id 1232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123
```

EAP_SIM_WPA3

```
# wlan-add abc ssid EAP wpa3-sb-192 ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
 channel 36 eap-sim id 1232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123 mfpc 1
 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

EAP_AKA_WPA2

```
# wlan-add abc ssid EAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 36
 eap-aka id 0232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123
```

EAP_AKA_WPA3

```
# wlan-add abc ssid EAP wpa3-sb-192 ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
 channel 36 eap-aka id 0232010000000000 pass \
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123 mfpc 1
 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

AKA_PRIME_WPA2

```
# wlan-add abc ssid EAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 36
 eap-aka-prime id 6555444333222111 pass \
5122250214c33e723a5dd523fc145fc0:981d464c7c52eb6e5036234984ad0bcf:000000000123
```

AKA_PRIME_WPA3

```
# wlan-add abc ssid EAP wpa3-sb-192 ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
 channel 36 eap-aka-prime id 6555444333222111 \
pass 5122250214c33e723a5dd523fc145fc0:981d464c7c52eb6e5036234984ad0bcf:000000000123 mfpc
 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

FT_SAE

```
# wlan-add abc ssid FTSAE ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 36
 wpa3 ft-sae 12345678 mfpc 1 mfpr 1
```

FT_Enterprise_WPA2

```
# wlan-add abc ssid FTEAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 36
 eap-tls-ft id client1 key_passwd whatever
```

FT_Enterprise_WPA3

```
# wlan-add abc ssid FTEAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap channel 36
 wpa3-sb-192 eap-tls-ft-sha384 id client1
key_passwd whatever mfpc 1 mfpr 1
```

4.10.1.5 Certificates and key configurations for enterprise security

For enterprise security, radius server (hostapd radius server) and server/client certificates are mandatory. This section describes how to configure CA certificate, client/server certificate, and client/server private key for WPA2/WPA3 enterprise.

The *wifi_wpa_supplicant* application supports two certificate configurations:

- Read certificates from USB disk.
- Read certificates from default .h files

Read from USB disk

Refer to Section 4.6.1.5 for the commands used to read certificate files (ca-cert/client-cert/client-key/server-cert/server-key/dh-params) from an external USB disk.

Read from default *.h* files

RW61x SDK supports certificates in *.h* format and server/client certificates are already available at location *<SDK_PATH>/middleware/wifi_nxp/certs/*. You can replace *ca-cert.h*, *client-cert.h*, *client-key.h*, *dh-param.h*, *server-cert.h*, and *server-key.h* files with your own certificate files.

To convert certificates on any Linux host where *openssl* and *xxd* are installed:

- Convert PEM certificate to DER certificate:

```
openssl x509 -inform pem -in ca.pem -outform der -out ca-cert.der
openssl x509 -inform pem -in client.pem -outform der -out client-cert.der openssl x509 -
inform pem -in server.pem
 -outform der -out server-cert.der
```

- Convert PEM private key to DER private key:

```
openssl rsa -inform pem -in client.key -outform der -out client-key.der
openssl rsa -inform pem -in server.key -outform der -out server-key.der
```

- Convert DER certificates and privet key to Header files:

ca-cert

```
xxd -i ca-cert.der ca-cert.h
```

- Change the array name and size in *ca-cert.h* file:

```
const unsigned char ca_der[]
unsigned int ca_der_len
```

client-cert

```
xxd -i client-cert.der client-cert.h
```

- Change the array name and size in *client-cert.h* file:

```
const unsigned char client_der[]
unsigned int client_der_len
```

client-key

```
xxd -i client-key.der client-key.h
```

- Change the array name and size in *client-key.h* file:

```
const unsigned char client_key_der[]
unsigned int client_key_der_len
```

Server-cert

```
xxd -i server-cert.der server-cert.h
```

- Change the array name and size in *server-cert.h* file:

```
const unsigned char server_der[]
unsigned int server_der_len
```

Server-key

```
xxd -i server-key.der server-key.h
```

- Change the array name and size inside *server-key.h* file:

```
const unsigned char server_key_der[]
unsigned int server_key_der_len
```

Note:

- *Defined certificates are read from USB disk. Undefined certificates are read from the .h files.*
- *The macro CONFIG_WIFI_USB_FILE_ACCESS in wifi_config.h is defined by default and can be modified.*

4.10.1.6 WPS

This section describes WPS related configurations. Two primary approaches are available for network setup within Wi-Fi Protected Setup: push-button and PIN entry.

WPS-PBC

- Start WPS PBC on the station:

```
# wlan-start-wps-pbc
ml1: WPS-PBC-ACTIVE
Started WPS PBC session
ml1: SME: Trying to authenticate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with c0:95:da:01:1c:6c
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: CTRL-EVENT-EAP-STARTED EAP authentication started
ml1: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=1
ml1: CTRL-EVENT-EAP-METHOD EAP vendor 14122 method 1 (WSC) selected
ml1: WPS-CRED-RECEIVED \
100e00321026000101104500074e585054455354100300020020100f000200081027000831323334353637381 0 \
200006c095da011fc2
ml1: WPS-SUCCESS
ml1: CTRL-EVENT-EAP-FAILURE EAP authentication failed
ml1: CTRL-EVENT-DISCONNECTED bssid=c0:95:da:01:1c:6c reason=3 locally_generated=1
app_cb: WLAN: network authentication failed
ml1: CTRL-EVENT-DSCP-POLICY clear_all
ml1: SME: Trying to authenticate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with c0:95:da:01:1c:6c
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: WPA: Key negotiation completed with c0:95:da:01:1c:6c [PTK=CCMP GTK=CCMP]
ml1: CTRL-EVENT-CONNECTED - Connection to c0:95:da:01:1c:6c completed [id=0 id_str=]
app_cb: WLAN: authenticated to network
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [NXPTEST]
IPv4 Address: [192.168.10.2]
```

- Start WPS PBC on the mobile AP:

```
# wlan-start-ap-wps-pbc
: WPS-PBC-ACTIVE
ua2: STA c0:95:da:01:1f:c2 IEEE 802.11: associated (aid 1)
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1f:c2
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1f:c2
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=254
: WPS-REG-SUCCESS c0:95:da:01:1f:c2 aed0f154-8c3e-5652-af0e-bf461a7e3807
: WPS-PBC-DISABLE
: WPS-SUCCESS
: CTRL-EVENT-EAP-FAILURE c0:95:da:01:1f:c2
ua2: STA c0:95:da:01:1f:c2 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
ua2: STA c0:95:da:01:1f:c2 IEEE 802.1X: Supplicant used different EAP type: 254 (expanded)
ua2: STA c0:95:da:01:1f:c2 IEEE 802.11: associated (aid 1)
: AP-STA-CONNECTED c0:95:da:01:1f:c2
app_cb: WLAN: UAP a Client Connected
========================================
Client => C0:95:DA:01:1F:C2 Connected with Soft AP
========================================
ua2: STA c0:95:da:01:1f:c2 WPA: pairwise key handshake completed (RSN)
: EAPOL-4WAY-HS-COMPLETED c0:95:da:01:1f:c2
```

WPS-PIN

- Generate the WPS PIN:

```
# wlan-generate-wps-pin
WPS PIN is: 27170991
```

- Start WPS PIN on the station:

```
# wlan-start-wps-pin 27170991
ml1: WPS-PIN-ACTIVE
Started WPS PIN session with pin as: 27170991
ml1: SME: Trying to authenticate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with c0:95:da:01:1c:6c
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: CTRL-EVENT-EAP-STARTED EAP authentication started
ml1: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=1
ml1: CTRL-EVENT-EAP-METHOD EAP vendor 14122 method 1 (WSC) selected
ml1: WPS-CRED-RECEIVED  \
100e00321026000101104500074e5850544553541003000200201000f000200081027000831323334353637 38 \
10200006c095da011fc2
ml1: WPS-SUCCESS
ml1: CTRL-EVENT-EAP-FAILURE EAP authentication failed
ml1: CTRL-EVENT-DISCONNECTED bssid=c0:95:da:01:1c:6c reason=3 locally_generated=1
app_cb: WLAN: network authentication failed
ml1: CTRL-EVENT-DSCP-POLICY clear_all
ml1: SME: Trying to authenticate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPTEST' freq=2437 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with c0:95:da:01:1c:6c
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: WPA: Key negotiation completed with c0:95:da:01:1c:6c [PTK=CCMP GTK=CCMP]
ml1: CTRL-EVENT-CONNECTED - Connection to c0:95:da:01:1c:6c completed [id=0 id_str=]
app_cb: WLAN: authenticated to network
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [NXPTEST]
IPv4 Address: [192.168.10.2]
```

- Start WPS PIN on the mobile AP:

```
# wlan-start-ap-wps-pin 27170991
Started AP WPS PIN session with pin as: 27170991
ua2: STA c0:95:da:01:1f:c2 IEEE 802.11: associated (aid 1)
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1f:c2
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1f:c2
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=254
: WPS-REG-SUCCESS c0:95:da:01:1f:c2 aed0f154-8c3e-5652-af0e-bf461a7e3807
: WPS-SUCCESS
: CTRL-EVENT-EAP-FAILURE c0:95:da:01:1f:c2
ua2: STA c0:95:da:01:1f:c2 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
ua2: STA c0:95:da:01:1f:c2 IEEE 802.1X: Supplicant used different EAP type: 254
 (expanded)
ua2: STA c0:95:da:01:1f:c2 IEEE 802.11: associated (aid 1)
: AP-STA-CONNECTED c0:95:da:01:1f:c2
app_cb: WLAN: UAP a Client Connected
========================================
Client => C0:95:DA:01:1F:C2 Connected with Soft AP
========================================
ua2: STA c0:95:da:01:1f:c2 WPA: pairwise key handshake completed (RSN)
: EAPOL-4WAY-HS-COMPLETED c0:95:da:01:1f:c2
```

4.10.1.7 Wi-Fi easy connect (DPP)

The Wi-Fi easy connect feature provides a simple and secure method to provision and connect Wi-Fi devices to a network without entering a password.

This section describes an example of test procedure of Wi-Fi easy connect with CLI commands supported in *wifi_wpa_supplicant* application, as well as configuration/connection of station and AP devices using DPP.

DPP test setup:

- The DUT (RW61x STA) operates as enrollee and authentication initiator.
- Device1 (RW61x STA) operates as configurator.
- Device2 (RW61x mobile AP) operates as enrollee and authentication responder.

Roles in DPP:

- Network role: STA and AP
- Provisioning roles: enrollee and configurator (role played in the entire DPP provisioning)
  - Configurator: Specifies the role of the device. Responsible for computing and passing the Network Access Key (NAK) and the Signing Key to the enrollee.
  - Enrollee: Receives the assigned role and configures the network according to the instructions of the configurator.
- Authentication roles: initiator and responder
  - Initiator: Sends the Authentication request.
  - Responder: Receives the authentication request and sends the authentication response.

Step 1 - Start mobile AP on Device2:

```
# wlan-add testAP ssid DPPNET01 ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
 channel 11 wpa2 psk 12345678
# wlan-start-network testAP
[wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is on.
ua2: interface state UNINITIALIZED->COUNTRY_UPDATE
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
# ua2: interface state COUNTRY_UPDATE->ENABLED
: AP-ENABLED
PKG_TYPE: BGA
Set BGA tx power table data
app_cb: WLAN: UAP Started
========================================
mobile AP "DPPNET01" started successfully
========================================
DHCP Server started successfully
========================================
```

Step 2 - Generate the QR code on Device2.

- Get the bootstrap ID:

```
# wlan-dpp-bootstrap-gen "type=qrcode chan=81/11 mac=C0:95:DA:01:20:C2"
 bootstrap generate id = 1
```

Note: *The MAC address of Device2 is the input in the command and the returned value “1” is the bootstrap info id required for the QR code string.*

- Get the QR code URI:

```
# wlan-dpp-bootstrap-get-uri 1
Bootstrapping QR Code URI:
DPP:C:81/11;M:c095da0120c2;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADhQ2oZmGPEq3pAv8zBZgbYFk1UK \
54C0OjyikiQoUapO4=;;
```

Note: *The generated QR code is used on Device1 with the command* `wlan-dpp-qr-code`*.*

Setp 3 - Configure Device1 as configurator

```
# wlan-dpp-configurator-add
conf_id = 1
```

Step 4 - Authenticate Device1 with Device2

- Add the QR code:

```
# wlan-dpp-qr-code
 DPP:C:81/11;M:c095da0120c2;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADhQ2 \
oZmGPEq3pAv8zBZgbYFk1UK
54C0OjyikiQoUapO4=;;
 DPP qr code id = 1
```

Note: *When the QR code addition is successful, a bootstrapping info id "1" is returned and used as input when sending an authentication request.*

- Send authentication request:

```
# wlan-dpp-auth-init " peer=1 conf=ap-dpp ssid=4450504e45543031 configurator=1"
ml1: DPP-TX dst=c0:95:da:01:20:c2 freq=2462 type=0
 DPP Auth Init OK!
# ml1: DPP-TX-STATUS dst=c0:95:da:01:20:c2 freq=2462 result=SUCCESS
ml1: DPP-RX src=c0:95:da:01:20:c2 freq=2462 type=1
ml1: DPP-AUTH-DIRECTION mutual=0
ml1: DPP-TX dst=c0:95:da:01:20:c2 freq=2462 type=2
ml1: DPP-TX-STATUS dst=c0:95:da:01:20:c2 freq=2462 result=SUCCESS
ml1: DPP-AUTH-SUCCESS init=1
ml1: DPP-CONF-REQ-RX src=c0:95:da:01:20:c2
ml1: DPP-RX src=c0:95:da:01:20:c2 freq=2462 type=11
ml1: DPP-CONF-SENT
```

Note: *The ssid parameter must be an hex string. In the example above,* `ssid=4450504e45543031` *is the hex string of* `DPPNET01`*.*

Output on Device2:

```
: DPP-RX src=c0:95:da:01:1b:6c freq=2462 type=0
: DPP-TX dst=c0:95:da:01:1b:6c freq=2462 type=1
: DPP-TX-STATUS dst=c0:95:da:01:1b:6c result=SUCCESS
: DPP-RX src=c0:95:da:01:1b:6c freq=2462 type=2
: DPP-AUTH-SUCCESS init=0
: GAS-QUERY-START addr=c0:95:da:01:1b:6c dialog_token=0 freq=2462
: GAS-QUERY-DONE addr=c0:95:da:01:1b:6c dialog_token=0 freq=2462 status_code=0 result=SUCCESS
: DPP-CONF-RECEIVED
: DPP-CONFOBJ-AKM dpp
: DPP-CONFOBJ-SSID DPPNET01
: DPP-CONNECTOR
 eyJ0eXAiOiJkcHBDb24iLCJraWQiOiItaDFmU25Yby1QR0YyZE94b25mQXFqV2pSdkh6c3dWSzNBRHc5Umc5e \
lB3IiwiYWxnIjoiRVMyNTYifQ.eyJncm91cHMiOlt7Imdyb3VwSWQiOiIqIiwibmV0Um9sZSI6ImFwIn1dLCJu \
ZXRBY2Nlc3NLZXkiOnsia3R5IjoiRUMiLCJjcnYiOiJQLTI1NiIsIngiOiJBNkNraGRS
: DPP-C-SIGN-KEY
 3039301306072a8648ce3d020106082a8648ce3d03010703220003b15b62da2a82f597d0b9158174523d3 \
ffe7c2a8a6045bb3c136e1ad65bd1bee7
: DPP-NET-ACCESS-KEY
 30770201010420cb3c255975d7dec6a25e89f8390cb515bf6a165b146b5257d2d5bff6e22b1143a00a \
06082a8648ce3d030107a1440342000403a0a485d44065b7ecb2925080496abf6234048c583e33a4e0ad038 \
0464efd8887eb1f131d7ea778a1e39f7eaeb00d0691bdf945ec0794b542ecbb0c
20/40 MHz: center segment 0 (=11) and center freq 1 (=0) not in sync
20/40 MHz: center segment 0 (=11) and center freq 1 (=0) not in sync
: DPP-TX dst=c0:95:da:01:1b:6c freq=2462 type=11
: DPP-TX-STATUS dst=c0:95:da:01:1b:6c result=SUCCESS
```

Step 5 - Generate the QR code on Device1 (configurator)

- Set the configurator parameter:

```
# wlan-dpp-configurator-params " conf=sta-dpp ssid=4450504e45543031 configurator=1"
```

Note: *There is a space character between " and* `conf`*.*

- Get the bootstrap ID:

```
# wlan-dpp-bootstrap-gen "type=qrcode chan=81/11 mac=C0:95:DA:01:1B:6C"
 bootstrap generate id = 2
```

- Get the QR code URI:

```
# wlan-dpp-bootstrap-get-uri 2
Bootstrapping QR Code URI:
DPP:C:81/11;M:c095da011b6c;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACZ2k
+yc2zLlHadBnnr5HvJcMzd6lNRJtHM+r7olKbzJY=;;
```

Note: *The QR code is used on the DUT with the command* `wlan-dpp-qr-code`*.*

Step 6 - Set Device1 in listening mode on a specific channel.

```
# wlan-dpp-listen "2462 role=configurator"
 DPP Listen OK!
```

Step 7 - Authenticate DUT (STA) and Device1 (STA)

- Add the QR code:

```
# wlan-dpp-qr-code
 DPP:C:81/11;M:c095da011b6c;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACZ2k+yc2zLlHad
Bnnr5HvJcMzd6lNRJtHM+r7olKbzJY=;;
 DPP qr code id = 1
```

Note: *Wthe QR code addition is successful, the bootstrapping info id is returned and used as input in for* `DPP_AUTH_INIT` *command.*

- Send the authentication request:

```
# wlan-dpp-auth-init " peer=1 role=enrollee"
ml1: DPP-TX dst=c0:95:da:01:1b:6c freq=2462 type=0
 DPP Auth Init OK!
# ml1: DPP-TX-STATUS dst=c0:95:da:01:1b:6c freq=2462 result=SUCCESS
ml1: DPP-RX src=c0:95:da:01:1b:6c freq=2462 type=1
ml1: DPP-AUTH-DIRECTION mutual=0
ml1: DPP-TX dst=c0:95:da:01:1b:6c freq=2462 type=2
ml1: DPP-RX src=c0:95:da:01:1b:6c freq=2462 type=1
ml1: DPP-AUTH-DIRECTION mutual=0
ml1: DPP-TX-STATUS dst=c0:95:da:01:1b:6c freq=2462 result=SUCCESS
ml1: DPP-AUTH-SUCCESS init=1
ml1: GAS-QUERY-START addr=c0:95:da:01:1b:6c dialog_token=46 freq=2462
ml1: GAS-QUERY-DONE addr=c0:95:da:01:1b:6c dialog_token=46 freq=2462 status_code=0 result=SUCCESS
ml1: DPP-CONF-RECEIVED
ml1: DPP-CONFOBJ-AKM dpp
ml1: DPP-CONFOBJ-SSID DPPNET01
ml1: DPP-CONNECTOR eyJ0eXAiOiJkcHBDb24iLCJraWQiOiItaDFmU25Yby1QR0YyZE94b25mQXFqV2pSdkh6c3dWSzNBRH \
c5Umc5elB3IiwiYWxnIjoiRVMyNTYifQ.eyJncm91cHMiOlt7Imdyb3VwSWQiOiIqIiwibmV0Um9sZSI6InN0YSJ9XSwibmV0Q
 \
WNjZXNzS2V5Ijp7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoiMlQ0a
ml1: DPP-C-SIGN-KEY 3039301306072a8648ce3d020106082a8648ce3d03010703220003b15b62da2a82f597d0b91581
 \
74523d3ffe7c2a8a6045bb3c136e1ad65bd1bee7
ml1: DPP-PP-KEY 3039301306072a8648ce3d020106082a8648ce3d030107032200021457d6b07b6ff77735928cdb4f8 \
631b6c1ffbf58551e4749b747244d0e0c49bb
ml1: DPP-NET-ACCESS-KEY 307702010104201c452ae0fb7b989a2e7b6af804b005b72e762943ede9b24a893e6d8d154 \
26113a00a06082a8648ce3d030107a14403420004d93e23c89841389150701ed345e4ca1f2416782d3be59b952482dd68 \
f8a32749eeff215b8566cf94ce3ec771861fcb98d0c15359f8be52de05d05
ml1: DPP-NETWORK-ID 1
ml1: DPP-TX dst=c0:95:da:01:1b:6c freq=2462 type=11
ml1: DPP-TX-STATUS dst=c0:95:da:01:1b:6c freq=2462 result=SUCCESS
ml1: DPP-TX dst=c0:95:da:01:20:c2 freq=2462 type=5
ml1: DPP-TX-STATUS dst=c0:95:da:01:20:c2 freq=2462 result=SUCCESS
ml1: DPP-RX src=c0:95:da:01:20:c2 freq=2462 type=6
ml1: PMKSA-CACHE-ADDED c0:95:da:01:20:c2 1
ml1: DPP-INTRO peer=c0:95:da:01:20:c2 status=0 version=3
ml1: SME: Trying to authenticate with c0:95:da:01:20:c2 (SSID='DPPNET01' freq=2462 MHz)
ml1: Trying to associate with c0:95:da:01:20:c2 (SSID='DPPNET01' freq=2462 MHz)
PKG_TYPE: CSP
Set CSP tx power table data
ml1: Associated with c0:95:da:01:20:c2
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml1: WPA: Key negotiation completed with c0:95:da:01:20:c2 [PTK=CCMP GTK=CCMP]
ml1: CTRL-EVENT-CONNECTED - Connection to c0:95:da:01:20:c2 completed [id=1 id_str=]
app_cb: WLAN: authenticated to network
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [DPPNET01]
IPv4 Address: [192.168.10.2]
```

Note: *The console output shows that the DUT is successfully connected to Device2 and* `IP address = [192.168.10.2]`*.*

Console output on Device1:

```
ml1: DPP-RX src=c0:95:da:01:2b:dc freq=2462 type=0
ml1: DPP-TX dst=c0:95:da:01:2b:dc freq=2462 type=1
ml1: DPP-TX-STATUS dst=c0:95:da:01:2b:dc freq=2462 result=SUCCESS
ml1: DPP-RX src=c0:95:da:01:2b:dc freq=2462 type=2
ml1: DPP-AUTH-SUCCESS init=0
ml1: DPP-CONF-REQ-RX src=c0:95:da:01:2b:dc
ml1: DPP-BAND-SUPPORT
 81,83,84,115,116,117,118,119,120,121,122,123,124,125,126,127,128,130
ml1: DPP-RX src=c0:95:da:01:2b:dc freq=2462 type=11
ml1: DPP-CONF-SENT
```

Console output on Device2:

```
: DPP-TX dst=c0:95:da:01:1b:6c freq=2462 type=11
: DPP-TX-STATUS dst=c0:95:da:01:1b:6c result=SUCCESS
: DPP-RX src=c0:95:da:01:2b:dc freq=2462 type=5
: DPP-TX dst=c0:95:da:01:2b:dc freq=2462 type=6 status=0
: DPP-TX-STATUS dst=c0:95:da:01:2b:dc result=SUCCESS
ua2: STA c0:95:da:01:2b:dc IEEE 802.11: associated (aid 1)
: AP-STA-CONNECTED c0:95:da:01:2b:dc
app_cb: WLAN: UAP a Client Connected
========================================
Client => C0:95:DA:01:2B:DC Connected with mobile AP
========================================
ua2: STA c0:95:da:01:2b:dc WPA: pairwise key handshake completed (RSN)
EAPOL-4WAY-HS-COMPLETED c0:95:da:01:2b:dc
```

Step 8 - Verify the connection between the DUT and Device2 using `ping`.

```
# ping 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data
64 bytes from 192.168.10.1: icmp_req=1 ttl=255 time=7 ms
64 bytes from 192.168.10.1: icmp_req=2 ttl=255 time=6 ms
64 bytes from 192.168.10.1: icmp_req=3 ttl=255 time=5 ms
64 bytes from 192.168.10.1: icmp_req=4 ttl=255 time=5 ms
64 bytes from 192.168.10.1: icmp_req=5 ttl=255 time=4 ms
64 bytes from 192.168.10.1: icmp_req=6 ttl=255 time=5 ms
64 bytes from 192.168.10.1: icmp_req=7 ttl=255 time=6 ms
64 bytes from 192.168.10.1: icmp_req=8 ttl=255 time=5 ms
64 bytes from 192.168.10.1: icmp_req=9 ttl=255 time=6 ms
64 bytes from 192.168.10.1: icmp_req=10 ttl=255 time=5 ms
--- 192.168.10.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss
```

4.10.1.8 Cloud keep alive

The cloud keep alive feature provides a method to send keep-alive packets from Wi-Fi to the cloud server periodically in host suspend state. The host can set keep-alive parameters like TCP/IP header info to the firmware when it goes to suspend. The Wi-Fi firmware sends keep-alive packets to the cloud server periodically with configured cycle time, and receives ACK from the cloud server for every keep-alive packet sent. If there is no ACK from server for three times continuously, the keep alive failure is indicated.

This section describes:

1. The test procedure of cloud keep-alive (TCP keep alive) using CLI commands on RW61x
2. The configuration of keep-alive parameters

Test setup:

- RW61x operates as STA.
- External AP with open security
- Cloud server is running on AP back-end.

Step 1 - Configure RW61x in station mode.

```
# wlan-add test ssid ax3600-2g
```

Step 2 - Connect to external AP.

```
# wlan-connect test
Connecting to network...
Use 'wlan-stat' for current connection status.
# ml1: SME: Trying to authenticate with 88:c3:97:c3:9f:24 (SSID='ax3600-2g' freq=2437
 MHz)
ml1: Trying to associate with 88:c3:97:c3:9f:24 (SSID='ax3600-2g' freq=2437 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml1: Associated with 88:c3:97:c3:9f:24
app_cb: WLAN: authenticated to network
ml1: CTRL-EVENT-CONNECTED - Connection to 88:c3:97:c3:9f:24 completed [id=0 id_str=]
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml1: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=CN
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [ax3600-2g]
IPv4 Address: [192.168.0.216]
```

Step 3 - Start TCP server in AP back-end Linux laptop.

Step 4 - Start cloud keep alive on RW61x.

```
# wlan-cloud-keep-alive start id <id> dst_mac <dst_mac> dst_ip <dst_ip> dst_port
 <dst_port>
```

Table 20. Cloud keep alive command parameters

Command parameters	Description
<id>	Cloud keep alive id (0~3)
<dst_mac>	MAC address of the server
<dst_ip>	IP address of the server
<dst_port>	Description port

Example:

```
# wlan-cloud-keep-alive start id 0 dst_mac 28:d2:44:07:53:cc dst_ip 192.168.0.157
 dst_port 9526
```

Step 5 - Set up the TCP connection with the server.

```
# wlan_tcp_client dst_ip 192.168.0.157 src_port 54236 dst_port 9526
```

Step 6 - Verify the TCP connection on the sniffer capture (Figure 54).

```
12657 2024-02-05 16:45:00.930343 192.168.0.216 192.168.0.157 TCP   12 [TCP Port numbers reused] 54236 → 9526 [SYN] Seq=0 W
12664 2024-02-05 16:45:00.934519 192.168.0.157 192.168.0.216 TCP    1 9526 → 54236 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
12672 2024-02-05 16:45:00.946703 192.168.0.157 192.168.0.216 TCP    2 [TCP ACKed unseen segment] 9526 → 54236 [ACK] Seq=1
12673 2024-02-05 16:45:00.946706 192.168.0.157 192.168.0.216 TCP    3 9526 → 54236 [ACK] Seq=1 Ack=2921 Win=62780 Len=0
12674 2024-02-05 16:45:00.946709 192.168.0.157 192.168.0.216 TCP    4 9526 → 54236 [ACK] Seq=1 Ack=4381 Win=61320 Len=0
12678 2024-02-05 16:45:00.948382 192.168.0.216 192.168.0.157 TCP   16 [TCP Previous segment not captured] 54236 → 9526 [PS
12679 2024-02-05 16:45:00.948386 192.168.0.216 192.168.0.157 TCP   17 54236 → 9526 [PSH, ACK] Seq=5841 Ack=1 Win=21900 Len
12684 2024-02-05 16:45:00.955656 192.168.0.157 192.168.0.216 TCP    2 [TCP ACKed unseen segment] 9526 → 54236 [ACK] Seq=1
12685 2024-02-05 16:45:00.955666 192.168.0.157 192.168.0.216 TCP    3 9526 → 54236 [ACK] Seq=1 Ack=2921 Win=62780 Len=0
12686 2024-02-05 16:45:00.955669 192.168.0.157 192.168.0.216 TCP    4 9526 → 54236 [ACK] Seq=1 Ack=4381 Win=61320 Len=0
12687 2024-02-05 16:45:00.955672 192.168.0.157 192.168.0.216 TCP    5 9526 → 54236 [ACK] Seq=1 Ack=5841 Win=62780 Len=0
12688 2024-02-05 16:45:00.955675 192.168.0.157 192.168.0.216 TCP    6 9526 → 54236 [ACK] Seq=1 Ack=7301 Win=62780 Len=0
```

Figure 54. Verify TCP connection on the sniffer capture

Step 7 - Configure MEF wake-up (use the default ARP filters for wake-up).

```
# wlan-wakeup-condition mef
No user configured MEF entries, use default ARP filters.
```

Step 8 - Set the host in suspend state.

```
# wlan-auto-host-sleep 1 manual
Manual mode is selected for host sleep
# wlan-suspend 2
Enter low power mode PM2
```

Once RW61x enters sleep state, packets show on the sniffer (Figure 55).

```
38699 2024-02-05 16:46:38.645701  192.168.0.216  192.168.0.157  TCP     0 [TCP Keep-Alive] 54236 → 9526 [PSH, ACK] Seq=7300 Ack=1
38704 2024-02-05 16:46:38.650845  192.168.0.157  192.168.0.216  TCP     7 [TCP Keep-Alive ACK] 9526 → 54236 [ACK] Seq=1 Ack=7301 W
```

Figure 55. Packets on sniffer

Step 9 -Stop or reset cloud keep alive after the host wakes up.

```
# wlan-cloud-keep-alive stop
```

Or

```
# wlan-cloud-keep-alive reset
```

Note: *The default period to send keep-alive packets is 55s in the application. The period to send retry packets is 20s with retry count of 3 by default. Modify the respective parameters in function* `test_wlan_cloud_keep_alive()` *in middleware/wifi_nxp/wlcmgr/wlan_tests.c. The period to send retry packets must be shorter than the period to send keep-alive packets.*

```
/* Period to send keep alive packet, set the default value to 55s(The unit is
 milliseconds) */
t_u32 send_interval_default = 55000;
/* Period to send retry packet, set the default value to 20s(The unit is milliseconds) */
t_u16 retry_interval_default = 20000;
/* Count to send retry packet, set the default value to 3 */
t_u16 retry_count_default = 3;
```

5 Useful Wi-Fi APIs

This section describes a few Wi-Fi driver APIs with their usage. These driver APIs can be called from the user application directly with the appropriate arguments to implement the required changes in the driver/firmware.

Note:

- *Refer to wifi_cert demo in Section 4.3, as it supports these APIs*
- *Refer to MCUXSDKGSUG for more details about the Wi-Fi driver APIs*

5.1 Set/get energy detection (ED) MAC feature

This feature enables the European Union (EU) adaptivity test as per the compliance requirements in the ETSI standard.

Depending on the device and front-end loss, the ED threshold offset (*ed_ctrl_2g.offset* and *ed_ctrl_5g.offset*) must be adjusted. The ED threshold offset can be adjusted in steps of 1 dB.

5.1.1 wlan_set_ed_mac_mode()

This API is used to configure ED MAC mode in the Wireless firmware.

Syntax: `int wlan_set_ed_mac_mode(wlan_ed_mac_ctrl_t wlan_ed_mac_ctrl)`

Where

Table 21. Set ED MAC API argument

Parameter	Description
[In] wlan_ed_mac_ctrl	A structure with parameters mentioned in section 4.1.3 to enable EU adaptivity.

Return value: `WM_SUCCESS` if the call is successful, -`WM_FAIL` if the call failed.

5.1.2 wlan_get_ed_mac_mode()

This API can be used to get current ED MAC mode configuration.

Syntax: `int wlan_get_ed_mac_mode(wlan_ed_mac_ctrl_t * wlan_ed_mac_ctrl)`

Where

Table 22. Get ED MAC API argument

Parameter	Description
[Out] wlan_ed_mac_ctrl	A pointer to a structure with parameters mentioned in section 4.1.3 to get ED MAC mode configuration.

Return value: `WM_SUCCESS` if the call is successful, -`WM_FAIL` if the call failed.

5.1.3 Usage and output

This section includes the output console logs and code snippets for reference. Use this section to add the feature-related commands in your user application.

To add new CLI command in the existing *wifi_cli* sample application, refer to Section 4.1.3.

Usage:

Add a `set` command to the command list:

```
#ifdef CONFIG_5GHz_SUPPORT
 {"wlan-set-ed-mac-mode", "<ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>",
 wlan_ed_mac_mode_set},
#else
 {"wlan-set-ed-mac-mode", "<ed_ctrl_2g> <ed_offset_2g>",wlan_ed_mac_mode_set},
#endif
```

Print the usage of `set-ed-mac` command:

```
static void dump_wlan_set_ed_mac_mode_usage()
{
    PRINTF("Usage:\r\n");
#ifdef CONFIG_5GHz_SUPPORT
    PRINTF("wlan-set-ed-mac-mode <ed_ctrl_2g> <ed_offset_2g> <ed_ctrl_5g> <ed_offset_5g>
\r\n");
#else
    PRINTF("wlan-set-ed-mac-mode <ed_ctrl_2g> <ed_offset_2g>\r\n");
#endif
    PRINTF("\r\n");
    PRINTF("\ted_ctrl_2g \r\n");
    PRINTF("\t    # 0       - disable EU adaptivity for 2.4GHz band\r\n");
    PRINTF("\t    # 1       - enable EU adaptivity for 2.4GHz band\r\n");
    PRINTF("\ted_offset_2g \r\n");
    PRINTF("\t    # 0       - Default Energy Detect threshold\r\n");
    PRINTF("\t    #offset value range: 0x80 to 0x7F\r\n");
#ifdef CONFIG_5GHz_SUPPORT
    PRINTF("\ted_ctrl_5g \r\n");
    PRINTF("\t    # 0       - disable EU adaptivity for 5GHz band\r\n");
    PRINTF("\t    # 1       - enable EU adaptivity for 5GHz band\r\n");
    PRINTF("\ted_offset_2g \r\n");
    PRINTF("\t    # 0       - Default Energy Detect threshold\r\n");
    PRINTF("\t    #offset value range: 0x80 to 0x7F\r\n");
#endif
}
```

Set ED MAC mode using the structure parameter in driver (set) API:

```
static void wlan_ed_mac_mode_set(int argc, char *argv[])
{
    int ret;
    wlan_ed_mac_ctrl_t wlan_ed_mac_ctrl;
#ifdef CONFIG_5GHz_SUPPORT
    if (argc != 5)
#else
    if (argc != 3)
#endif
    {
        dump_wlan_set_ed_mac_mode_usage();
        return;
    }
    wlan_ed_mac_ctrl.ed_ctrl_2g   = strtol(argv[1], NULL, 16);
    wlan_ed_mac_ctrl.ed_offset_2g = strtol(argv[2], NULL, 16);
#ifdef CONFIG_5GHz_SUPPORT
    wlan_ed_mac_ctrl.ed_ctrl_5g   = strtol(argv[3], NULL, 16);
    wlan_ed_mac_ctrl.ed_offset_5g = strtol(argv[4], NULL, 16);
#endif
    if (wlan_ed_mac_ctrl.ed_ctrl_2g != 0 && wlan_ed_mac_ctrl.ed_ctrl_2g != 1)
    {
        dump_wlan_set_ed_mac_mode_usage();
        return;
    }
#ifdef CONFIG_5GHz_SUPPORT
    if (wlan_ed_mac_ctrl.ed_ctrl_5g != 0 && wlan_ed_mac_ctrl.ed_ctrl_5g != 1)
    {
        dump_wlan_set_ed_mac_mode_usage();
        return;
    }
#endif
    ret = wlan_set_ed_mac_mode(wlan_ed_mac_ctrl);
    if (ret == WM_SUCCESS)
    {
        PRINTF("ED MAC MODE settings configuration successful\r\n");
    }
    else
    {
        PRINTF("ED MAC MODE settings configuration failed\r\n");
        dump_wlan_set_ed_mac_mode_usage();
    }
}
```

Add a `get` command to the command list:

```
{"wlan-get-ed-mac-mode", NULL, wlan_ed_mac_mode_get},
```

Print the usage regarding `get-ed-mac`:

```
static void dump_wlan_get_ed_mac_mode_usage()
{
    PRINTF("Usage:\r\n");
    PRINTF("wlan-get-ed-mac-mode \r\n");
}
```

Get ED MAC mode values filled address of `wlan_ed_mac_ctrl` structure passed as a parameter to the driver (get) API:

```
static void wlan_ed_mac_mode_get(int argc, char *argv[])
{
    int ret;
    wlan_ed_mac_ctrl_t wlan_ed_mac_ctrl;
    if (argc != 1)
    {
        dump_wlan_get_ed_mac_mode_usage();
        return;
    }
    ret = wlan_get_ed_mac_mode(&wlan_ed_mac_ctrl);
    if (ret == WM_SUCCESS)
    {
        PRINTF("EU adaptivity for 2.4GHz band : %s\r\n", wlan_ed_mac_ctrl.ed_ctrl_2g ==
 1 ? "Enabled" : "Disabled");
        if (wlan_ed_mac_ctrl.ed_ctrl_2g)
            PRINTF("Energy Detect threshold offset : 0X%x\r\n",
 wlan_ed_mac_ctrl.ed_offset_2g);
#ifdef CONFIG_5GHz_SUPPORT
        PRINTF("EU adaptivity for 5GHz band : %s\r\n", wlan_ed_mac_ctrl.ed_ctrl_5g == 1 ?
 "Enabled" : "Disabled");
        if (wlan_ed_mac_ctrl.ed_ctrl_5g)
            PRINTF("Energy Detect threshold offset : 0X%x\r\n",
 wlan_ed_mac_ctrl.ed_offset_5g);
#endif
    }
    else
    {
        PRINTF("ED MAC MODE read failed\r\n");
        dump_wlan_get_ed_mac_mode_usage();
    }
}
```

Console output

```
# wlan-set-ed-mac-mode 1 0x9
ED MAC MODE settings configuration successful
# wlan-get-ed-mac-mode
EU adaptivity for 2.4GHz band : Enabled
Energy Detect threshold offset : 0X9
EU adaptivity for 5GHz band : Enabled
Energy Detect threshold offset : 0Xc
```

6 Bluetooth Low Energy applications

This section describes the Bluetooth Low Energy example applications that are available in the SDK. It also provides the instructions to configure, compile, debug, flash, and execute these examples.

The communication between the Host stack and the Link Layer (LL) is implemented via the standard host controller interface (HCI) specification ([5]).

The setup is done between RW61x EVK board and remote Bluetooth LE devices. The instructions in this guide use an RW61x EVK board.

6.1 peripheral_hps sample application

This application demonstrates the Bluetooth LE peripheral role. More specifically, the application exposes the HTTP Proxy GATT Service.

6.1.1 Flash Bluetooth LE firmware

RW61x application and Bluetooth firmware binary are stored in different partitions of FlexSPI NOR flash. The application reads the Bluetooth firmware during initialization and downloads it to RW61x internal Bluetooth MCU to run. This section describes the steps to flash Bluetooth firmware with SEGGER J-Link tool.

- Open J-Link commander in Windows and connect RW61x device

```
J-Link>con
Device>RW610
TIF>S
Speed><Enter>
```

- Flash Bluetooth LE firmware

The path to Bluetooth LE secure firmware binary is the following:

${SDK}\components\conn_fwloader\fw_bin\rw61x_sb_ble_v1.bin for A1 version of RW61x.

${SDK}\components\conn_fwloader\fw_bin\rw61x_sb_ble_v2.bin for A2 version of RW61x.

```
J-Link>loadbin rw61x_sb_ble_v<version number>.bin,0x08540000
```

Note: *Bluetooth firmware only must be flashed once unless it is erased. The firmware is stored at a given address. Ensure that Bluetooth firmware is flashed before running any Bluetooth LE demo application.*

6.1.2 peripheral_hps application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

Refer to section Section 2.1 for serial console tool setup.

6.1.2.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Advertising successfully started
```

The demo does not require user interaction.

The application automatically starts advertising the HTTP Proxy Service and it accepts the first connection request it receives. The application is then ready to process HTTP requests from the peer Bluetooth device.

The application simulates the processing of the HTTP request. It always returns HTTP Status Code 500 and preset values for HTTP Headers and HTTP Body.

```
Connected to peer: C0:95:DA:00:D5:0D (public)
Processing request..
Request processed.
Security changed: C0:95:DA:00:D5:0D (public) level 1 (error 8)
```

6.2 central_hpc sample application

This application demonstrates very basic Bluetooth LE central role functionality on RW61x EVK board. It scans for other Bluetooth LE devices and establishes a connection to the first Bluetooth LE device with a strong enough signal.

More specifically, the central_hpc application:

- Looks for HPS server
- Programs a set of characteristics to configure a Hyper Text Transfer Protocol (HTTP) request
- Initiates this request
- Read the response once connected

For this application, another setup of RW61x EVK board is used as *peripheral_hps*.

6.2.1 central_hpc application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

Refer to section Section 2.1 for serial console tool setup.

6.2.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Scanning started
[DEVICE]: C0:95:DA:00:D5:10 (random), AD evt type 3, AD data len 31, RSSI -94
```

The demo does not require user interaction.

The application automatically starts scanning and connects to the first advertiser who is advertising the HTTP Proxy Service.

If the connection is successful, the application performs service discovery to find the characteristics of the HTTP Proxy Service. If discovery is successful, the application performs a GET for the URI *http://nxp.com*. The GET command includes the URI and the Control Point characteristics of the HTTP Proxy Service.

The application displays the received response in the console after it gets notified through the HTTP Status Code characteristic.

```
Found device: Connected to peer: C0:95:DA:00:D5:10 (public)
Starting service discovery
GATT Write successful
Subscribed to HTTP Status Code
GATT Write successful
Received HTTP Status 500
Reading Headers..
HTTP Headers: HTTPHEADER
Reading Body...
Unsubscribed
HTTP Body: HTTPBODY
Security changed: C0:95:DA:00:D5:10 (public) level 1 (error 8)
```

6.3 peripheral_pxr sample application

This application demonstrates the Bluetooth LE Peripheral role on RW61x EVK board. More specifically, this application exposes the Proximity Reporter (including LLS, IAS, and TPS) GATT Service.

6.3.1 peripheral_pxr application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

The instructions are given for a few IDEs.

Refer to section Section 2.1 for serial console tool setup.

6.3.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Advertising successfully started
```

The demo does not require user interaction.

The application automatically starts advertising the Link Loss Service and it accepts the first connection request it receives. The application is then ready to process operations from the peer.

The application initially sets the default levels for the Link Loss Alert and the Immediate Alert.

```
Connected to peer: C0:95:DA:00:D5:0D (public)
Locally setting Link Loss Alert Level to OFF
Locally setting Immediate Alert...
ALERT: OFF
ALERT: OFF
```

The Proximity Monitor peer triggers or stops the Immediate Alert on the application depending on the connection RSSI.

```
Monitor is setting Immediate Alert...
ALERT: HIGH
Monitor is setting Immediate Alert...
ALERT: OFF
```

If the connection with the Proximity Monitor is timed out, the Link Loss Alert is triggered with the level previously set by the Monitor.

```
Security changed: C0:95:DA:00:D5:0D (public) level 4 (error 0)
Monitor is setting Link Loss Alert Level to HIGH
Monitor is setting Immediate Alert...
ALERT: HIGH
```

6.4 central_pxm sample application

This application demonstrates very basic Bluetooth LE Central role functionality on RW61x EVK board by scanning for other Bluetooth LE devices and establishing a connection to the first one with a strong enough signal.

More specifically, this application looks for Proximity Reporter.

For this application, another setup of RW61x EVK board is used as *peripheral_pxr*.

6.4.1 central_pxm application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for serial console tool setup.

6.4.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Scanning started
```

The application automatically starts scanning and connects to the first advertiser who is advertising the Link Loss Service.

If the connection is successful, the application performs service discovery to find:

- The characteristics of the Link Loss Service
- Additional services and characteristics specified by the Proximity Profile, for example Immediate Alert and TX Power services

```
Found device: Connected to peer: C0:95:DA:00:D5:10 (public)
Starting service discovery
GATT Write successful
Read successful - Tx Power Level: 0
Security changed: C0:95:DA:00:D5:10 (public) level 1 (error 8)
Connection RSSI: -11
```

If the TX Power service and its characteristics have been discovered, the application reads the TX power of the peer and displays it.

```
Read successful - Tx Power Level: 0
```

If the Immediate Alert service and its characteristics have been discovered, the application continuously monitors the connection RSSI, and triggers. Or the application stops the Immediate Alert on the peer when the value is crossing a preset threshold in either direction.

```
Connection RSSI: -11
Connection RSSI: -11
Connection RSSI: -11
Connection RSSI: -11
Connection RSSI: -11
```

After the mandatory Link Loss service is discovered, the application writes the Link Loss Alert Level on the peer as HIGH_ALERT.

To trigger the Link Loss Alert on the peer, the connection has to be timed out. To time out the connection, press the RST button on the board to reset the board.

6.5 peripheral_ht sample application

This application demonstrates the Bluetooth LE Peripheral role on RW61x EVK board. More specifically, this application exposes the HT (Health Thermometer) GATT Service.

When a Bluetooth device connects, it generates dummy temperature values.

6.5.1 peripheral_ht application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for serial console tool setup.

6.5.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board.

When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Advertising successfully started
```

The application does not require any user interaction.

The application automatically starts advertising the Health Thermometer Service, and accepts the first connection request it receives. If the peer subscribes to receive temperature indications, the indications are sent every second.

The temperature readings are simulated with values between 20°C and 25°C.

```
Connected to peer: C0:95:DA:00:D5:0D (public)
temperature is 20C
Indication success
temperature is 21C
Indication success
Passkey for C0:95:DA:00:D5:0D (public): 529639
temperature is 22C
Indication success
temperature is 23C
Indication success
```

6.6 central_ht sample application

This application demonstrates very basic Bluetooth LE Central role functionality on RW61x EVK board. It scans for other Bluetooth LE devices and establishes a connection to the first Bluetooth LE device with a strong enough signal.

More specifically, this application looks for health thermometer sensor and reports the temperature readings once connected.

For this application, another setup of RW61x EVK board is used as *peripheral_ht*.

6.6.1 central_ht application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for serial console tool setup.

6.6.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Scanning started
```

The demo does not require any user interaction.

The application automatically starts scanning and connects to the first advertiser who is advertising the Health Thermometer Service. If the connection is successful, the application performs service discovery to find the characteristics of the Health Thermometer Service.

If discovery is successful, the application subscribes to receive temperature indications from the peer.

The application displays the received indications in the console.

```
[DEVICE]: C0:95:DA:00:D5:10 (public), AD evt type 0, AD data len 9, RSSI -14
Found device: Connected to peer: C0:95:DA:00:D5:10 (public)
Starting service discovery
Subscribed to HTS
Temperature 20 degrees Celsius
Security changed: C0:95:DA:00:D5:10 (public) level 1 (error 8)
Temperature 21 degrees Celsius
Temperature 22 degrees Celsius
Temperature 23 degrees Celsius
```

6.7 peripheral_ipsp sample application

This application demonstrates the Bluetooth LE Peripheral role on RW61x EVK board. More specifically, this application exposes the Internet Protocol Support GATT Service.

6.7.1 peripheral_ipsp application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for serial console tool setup.

6.7.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board.

When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Advertising successfully started
IPSS Service ready
```

The demo does not require any user interaction.

The application automatically starts advertising the IPSP Service and it accepts the first connection request it receives.

The application performs the required setup for the L2CAP credit-based channel specified by the IPSP Profile. The application displays in the console any message that it receives from the peer through the L2CAP channel.

```
Connected to peer: C0:95:DA:00:D5:0D (public)
Security changed: C0:95:DA:00:D5:0D (public) level 1 (error 8)
Received message: hello
Received message: hello
```

6.8 central_ipsp sample application

This application demonstrates Bluetooth LE Central role functionality. It scans for other Bluetooth LE devices and establishes a connection to the first device with a strong enough signal.

More specifically, this application looks for IPSP Service and communicates between the devices that support IPSP. The application transfers IPv6 packets over the Bluetooth Low Energy transport once connected with a peer device.

For this application, another setup of RW61x EVK board is used as *peripheral_ipsp*.

6.8.1 central_ipsp application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for serial console tool setup.

6.8.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Bluetooth initialized
Scanning started
```

The demo does not require any user interaction.

The application automatically starts scanning and connects to the first advertiser who is advertising the IPSP Service.

After the L2CAP credit-based channel specified by the IPSP Profile is established, the application sends a predefined test message every 5 seconds through the channel.

```
[DEVICE]: C0:95:DA:00:D5:10 (public), AD evt type 0, AD data len 7, RSSI -13
Found device: Connected
Starting service discovery
Security changed: C0:95:DA:00:D5:10 (public) level 1 (error 8)
Sending message...
Sending message...
```

6.9 peripheral_beacon sample application

This application demonstrates the Bluetooth LE Peripheral role on RW61x EVK. More specifically, this application exposes three type of beacon types.

- General beacon: Describes Bluetooth LE Broadcaster role functionality by advertising
  - The company identifier
  - The beacon identifier
  - UUID, A, B, C, RSSI
- iBeacon: Describes the Bluetooth LE Broadcaster role functionality by advertising an Apple iBeacon
- Eddystone: Runs Eddystone Configuration Service as a GATT service in the beacon while it is connectable. The service is used to configure the advertised data, the broadcast power levels, and the advertising intervals.

6.9.1 peripheral_beacon application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for serial console tool setup.

Choose the beacon type by defining the corresponding macro to true in *app_config.h* while keeping the other two types as false.

```
#define BEACON_APP  1
#define IBEACON_APP 0
#define EDDYSTONE 0
```

6.9.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
Starting   Beacon Demo
Bluetooth initialized
Beacon started, advertising as C0:95:DA:00:D5:0D (public)
```

The demo does not require any user interaction. The application automatically broadcasts the packet in one of the following formats: SIG Beacon, Apple iBeacon, or Google Eddystone Beacon.

6.10 Wireless UART sample application

The application implements a custom GATT-based Wireless UART Profile that emulates UART over Bluetooth LE. The central and peripheral roles can be switched with the user button (SW4). To test the service/profile, you can use the "IoT Toolbox" application. IoT Toolbox is available on Apple App Store for iOS, and Google Play Store for Android.

6.10.1 wireless_uart application execution

Refer to Section 3.2 and Section 3.3 for instructions to:

- Import a project
- Build an application
- Run an application in Debug mode
- flash an application program

Refer to section Section 2.1 for information about the serial console tool setup.

6.10.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of RW61x EVK board. When the demo starts, the following message about the demo shows on the console.

```
BLE Wireless Uart demo start...
Bluetooth initialized
Advertising successfully started
```

The application works in peripheral role by default. It automatically starts advertising the Wireless UART Service after reset. And it only accepts one connection from the device with central role.

The demo requires user interaction. You can use "IoT Toolbox" ([6]) or another wireless_uart example with central role to test the Wireless UART device with peripheral role.

Peripheral role test

- Open "IoT Toolbox" application on an Android or iOS smartphone.
- Select the "Wireless UART" option.
- Look for the device named "NXP_WU".
- Connect to "NXP_WU" by selecting the device from the scan list.

The Android/iOS device should receive a prompt for a Bluetooth Pairing Request.

- Complete the pairing process by entering the passkey that is displayed on the debug terminal.

Once pairing is completed, you can transmit and receive data over the emulated UART interface.

```
BLE Wireless Uart demo start...
Bluetooth initialized
Advertising successfully started
Connected to C0:95:DA:00:C0:45 (public)
GATT MTU exchanged: 65
[ATTRIBUTE] handle 40
[ATTRIBUTE] handle 41
Security changed: C0:95:DA:00:C0:45 (public) level 2 (error 0)
```

Central role test

- Run wireless_uart sample application on another RW61x EVK board.
- After the application starts, apply a short press on the user button (SW4).

The example works in central role. It automatically starts scanning and connects to any discovered wireless_uart example. The application in central mode can connect up to eight devices. Each time you apply a short press on SW4 button, if a new device is found, the example scans and connects to the wireless UART service.

```
BLE Wireless Uart demo start...
Bluetooth initialized
Advertising successfully started
Scanning successfully started
[DEVICE]: 64:86:7F:5A:7C:7F (random), AD evt type 0, AD data len 23, RSSI -81
[DEVICE]: 64:86:7F:5A:7C:7F (random), AD evt type 4, AD data len 0, RSSI -80
[DEVICE]: 63:F2:B1:6A:FC:3D (random), AD evt type 0, AD data len 18, RSSI -80
[DEVICE]: 63:F2:B1:6A:FC:3D (random), AD evt type 4, AD data len 0, RSSI -80
[DEVICE]: 78:B3:AA:89:78:3B (random), AD evt type 0, AD data len 18, RSSI -80
[DEVICE]: 78:B3:AA:89:78:3B (random), AD evt type 4, AD data len 0, RSSI -79
[DEVICE]: C0:95:DA:00:C0:3C (public), AD evt type 0, AD data len 21, RSSI -43
Connected to C0:95:DA:00:C0:3C (public)
GATT MTU exchanged: 65
[ATTRIBUTE] handle 25
[ATTRIBUTE] handle 26
Security changed: C0:95:DA:00:C0:3C (public) level 2 (error 0)
```

Note: *The device address (AD), the event type `data len`, and RSSI are variables that depend on the Bluetooth device in the testing environment.*

- Send the data 12345 using the serial port terminal of the device with central role. The device with peripheral role prints the following log.

```
Data received from C0:95:DA:00:C0:45 (public)(length 1):1
Data received from C0:95:DA:00:C0:45 (public)(length 1):2
Data received from C0:95:DA:00:C0:45 (public)(length 1):3
Data received from C0:95:DA:00:C0:45 (public)(length 1):4
Data received from C0:95:DA:00:C0:45 (public)(length 1):5
```

- Send the data `123` using the serial port terminal of the device with peripheral role. The device with central role prints the following log.

```
Data received from C0:95:DA:00:C0:3C (public)(length 1):1
Data received from C0:95:DA:00:C0:3C (public)(length 1):2
Data received from C0:95:DA:00:C0:3C (public)(length 1):3
```

6.11 Shell sample application

The sample application demonstrates the interactive shell mode of Bluetooth commands and APIs. It provides full control over the Bluetooth interface and basic Bluetooth operations such as advertising/scanning, device discovery, connection and pairing. The application also provides direct access to HCI command interface.

6.11.1 Shell application execution

Refer to Section 3.1 to Section 3.4 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs. Refer to section Section 2.1 for information about the serial console setup.

6.11.1.1 Run the shell application

Press the power reset button on RW61x EVK board to run the demo application downloaded on the board. When the demo starts, the following message is displayed on the console.

```
Edgefast Bluetooth PAL shell demo start…
SHELL build: Jun 25 2023
Copyright 2020 NXP
@bt>
```

Note: *In the code sample above,* `SHELL build: Jun 25 2023` *is an example of compilation date.*

The shell command list can be accessed by typing `help` in the serial terminal. The demo can be configured to either `central` or `peripheral` by shell commands.

```
@bt> help
+---"help": List all the registered commands
+---"exit": Exit program
+---"echo": Set echo(0 - disable, 1 - enable)
+---"bt": bt command entry
    +---"init": init [no-settings-load], [sync]
    +---"settings-load": settings-load [none]
    +---"id-create": id-create [addr]
    +---"id-reset": id-reset <id> [addr]
    +---"id-delete": id-delete <id>
    +---"id-show": id-show [none]
    +---"id-select": id-select <id>
    +---"name": name [name]
    +---"appearance": appearance
    +---"scan": scan <value: on, passive, off> [filter: dups, nodups] [fal]
    +---"scan-filter-set": scan-filter-set Scan filter set commands
        +---"name": name <name>
        +---"addr": addr <addr>
        +---"rssi": rssi <rssi>
    +---"scan-filter-clear": scan-filter-clear Scan filter clear commands
        +---"all": all
        +---"name": name
        +---"addr": addr
    +---"advertise": advertise <type: off, on, scan, nconn> [mode: discov, non_discov] [filter-
accept-list: fal, fal-scan, fal-conn] [identity] [no-name] [one-time] [name-ad][disable-37]
 [disable-38] [disable-39]
    +---"directed-adv": directed-adv <address: XX:XX:XX:XX:XX:XX> <type: (public|random)> [mode:
 low] [identity] [dir-rpa]
    +---"connect": connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"auto-conn": auto-conn <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"connect-name": connect-name <name filter>
    +---"disconnect": disconnect [none]
    +---"select": select <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"info": info <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"conn-update": conn-update <min> <max> <latency> <timeout>
    +---"data-len-update": data-len-update <tx_max_len> [tx_max_time]
    +---"phy-update": phy-update <tx_phy> [rx_phy] [s2] [s8]
    +---"channel-map": channel-map <channel-map: XXXXXXXXXX> (36-0)
    +---"oob": oob [none]
```

```
    +---"clear": clear <remote: addr, all>
    +---"security": security <security level BR/EDR: 0 - 3, LE: 1 - 4> [force-pair]
    +---"bondable": bondable <bondable: on, off>
    +---"bonds": bonds [none]
    +---"connections": connections [none]
    +---"auth": auth <method: all, input, display, yesno, confirm, oob, status, none>
    +---"auth-cancel": auth-cancel [none]
    +---"auth-passkey": auth-passkey <passkey>
    +---"auth-passkey-confirm": auth-passkey-confirm [none]
    +---"auth-pairing-confirm": auth-pairing-confirm [none]
    +---"auth-oob-tk": auth-oob-tk <tk>
    +---"oob-remote": oob-remote <address: XX:XX:XX:XX:XX:XX> <type: (public|random)> <oob rand>
 <oob confirm>
    +---"oob-clear": oob-clear [none]
+---"gatt": gatt Bluetooth GATT shell commands
    +---"discover": discover [UUID] [start handle] [end handle]
    +---"discover-characteristic": discover-characteristic [UUID] [start handle] [end handle]
    +---"discover-descriptor": discover-descriptor [UUID] [start handle] [end handle]
    +---"discover-include": discover-include [UUID] [start handle] [end handle]
    +---"discover-primary": discover-primary [UUID] [start handle] [end handle]
    +---"discover-secondary": discover-secondary [UUID] [start handle] [end handle]
    +---"exchange-mtu": exchange-mtu [none]
    +---"read": read <handle> [offset]
    +---"read-uuid": read-uuid <UUID> [start handle] [end handle]
    +---"read-multiple": read-multiple <handle 1> <handle 2> ...
    +---"signed-write": signed-write <handle> <data> [length] [repeat]
    +---"subscribe": subscribe <CCC handle> <value handle> [ind]
    +---"resubscribe": resubscribe <address: XX:XX:XX:XX:XX:XX> <type: (public|random)> <CCC handle>
 <value handle> [ind]
    +---"write": write <handle> <offset> <data>
    +---"write-without-response": write-without-response <handle> <data> [length] [repeat]
    +---"write-without-response-cb": write-without-response-cb <handle> <data> [length] [repeat]
    +---"unsubscribe": unsubscribe [none]
    +---"get": get <start handle> [end handle]
    +---"set": set <handle> [data...]
    +---"show-db": show-db [uuid] [num_matches]
    +---"att_mtu": att_mtu Output ATT MTU size
    +---"metrics": metrics [value: on, off]
    +---"register": register register pre-predefined test service
    +---"unregister": unregister unregister pre-predefined test service
    +---"notify": notify [data]
    +---"notify-mult": notify-mult count [data]
+---"l2cap": l2cap Bluetooth L2CAP shell commands
    +---"connect": connect <psm> [sec_level]
    +---"disconnect": disconnect [none]
    +---"metrics": metrics <value on, off>
    +---"recv": recv [delay (in milliseconds)
    +---"register": register <psm> [sec_level] [policy: allowlist, 16byte_key]
    +---"send": send <number of packets>
    +---"allowlist": allowlist [none]
        +---"add": add [none]
        +---"remove": remove [none]
+---"le_test": le_test Bluetooth BLE test mode commands
    +---"set_tx_power": set_tx_power tx_power[1]
    +---"tx_test": tx_test tx_channel[1] data_length[1] payload[1] phy[1]
    +---"rx_test": rx_test rc_channel[1] phy[1] modulation[1]
    +---"end_test": end_test end the le test
+---"hci": hci Bluetooth HCI Command interface
    +---"generic_command": generic_command ogf[1] ocf[1] params....
```

Example of Bluetooth LE scanning devices

The Bluetooth LE host must be initialized before executing the scan command:

```
@bt> bt.init
@bt> Bluetooth initialized
Settings Loaded
@bt> bt.scan on
Bluetooth active scan enabled
@bt> [DEVICE]: 0B:F6:E9:7C:AA:AB (random), AD evt type 3, RSSI -47  C:0 S:0 D:0 SR:0 E:0
 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: C2:7E:06:31:17:0D (random), AD evt type 0, RSSI -84  C:1 S:1 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 1D:0C:29:D7:BB:73 (random), AD evt type 3, RSSI -81  C:0 S:0 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 51:FD:82:19:A4:03 (random), AD evt type 0, RSSI -39  C:1 S:1 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 51:FD:82:19:A4:03 (random), AD evt type 4, RSSI -41  C:0 S:1 D:0 SR:1 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 3D:BA:EC:58:43:77 (random), AD evt type 3, RSSI -87  C:0 S:0 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 48:76:6C:70:E3:7B (random), AD evt type 0, RSSI -76  C:1 S:1 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
@bt> bt.scan off
Scan successfully stopped
@bt>
```

Example of advertising

The Bluetooth LE host must be initialized before:

```
@bt> bt.advertise on
Advertising started
@bt> bt.advertise off
Advertising stopped
```

Example of Bluetooth LE pairing and bonding

GATT peripheral role side

Initialize the host

```
@bt> bt.init
```

Start advertising

```
@bt> bt.advertise on
```

When the connection is established, perform the pairing sequence. The pairing can start from the peripheral side with `bt.security <level>`, such as

```
@bt> bt.security 2
```

If the central role does not support `bondable`, issue the command below and repeat the previous step:

```
@bt> bt.bondable off
```

GATT central role side

Initialize the host

```
@bt> bt.init
```

Scan for advertising packets

```
@bt> bt.scan on
```

Stop the scanning after a few seconds

```
@bt> bt.scan off
```

Select the target board and create a new connection. If the target is not listed, repeat `scan on` and `scan off` then enter *`bt.connect <remote address: XX:XX:XX:XX:XX:XX> <type: (public|random)>`*.

```
@bt> bt.connect 11:22:33:44:55:66 public
```

When the connection is established, perform the pairing sequence. The pairing can start from the peripheral side with *`bt.security <level>`*, such as:

```
@bt> bt.security 2
```

If the central role does not support `bondable`, issue the command below and repeat the previous step:

```
@bt> bt.bondable off
```

After all the operations, initiate a disconnection from the central device:

```
@bt> bt.disconnect
```

Running generic HCI commands

Use this functionality to execute commands to the wireless controller.

Command syntax: `hci.generic_command <ogf> <ocf> <n parameters>`**..**

Vendor specific command to check the firmware version:

```
@bt> hci.generic_command 3f 0f
```

Command response:

```
HCI Command Response :
@bt> 00 02 19 12 08 00 00 02 04 00
```

6.11.1.2 Bluetooth LE RF test mode operations

This section includes the commands for Bluetooth LE RF test.

Note: *`command complete event` can be found in HCI log. The U-DISK should be connected to USB port to get HCI log captured. `CONFIG_BT_SNOOP macro` in app_config.h file is used to enable the stack to capture the HCI log.*

Set Bluetooth LE TX power

Command to set Bluetooth LE transmit power level.

```
@bt> le_test.set_tx_power 4
tx_power= 4
@bt>  HCI Command Response : 00
```

Note: *The value of `tx_power` parameter must be hexadecimal.*

Test Bluetooth LE transmitter

To start a test where the DUT generates test reference packets at a fixed interval, use LE transmitter test command. For more details on the command, refer to section 7.8.29 in Bluetooth Core Specification v5.3 Vol 0 Part A.

```
@bt> le_test.tx_test 01 FF 00 01
tx_channel= 1
test_data_len= ff
pkt_payload= 0
phy= 1
@bt>  HCI Command Response : 00
```

Note: *The value of `tx_channel` parameter must be hexadecimal.*

Observe the transmitter test packets over the air logs.

Test Bluetooth LE receiver

To start a test where the DUT receives test reference packets at a fixed interval, use LE receiver test command. For more details on the command, refer to section 7.8.28 in Bluetooth Core Specification v5.3 Vol 0 Part A.

```
@bt> le_test.rx_test 01 01 00
rx_channel= 1
@bt> phy= 1
modulation_index= 0
 HCI Command Response : 00
```

End a test for Bluetooth LE

Command to end any test for Bluetooth LE:

```
@bt> le_test.end_test
API returned success...
```

Note: *Observe the packet count in command complete event in HCI log during LE receiver test.*

7 Load external calibration data

This section shows how to load external Wi-Fi or Bluetooth calibration data.

7.1 Wi-Fi calibration data

The Wi-Fi calibration data is used to attain tighter RF performance tolerance for the Wi-Fi radio. The calibration data can be stored in the on-chip OTP memory or in as external configuration file. This section explains how to modify a Wi-Fi example (in this case, wifi_cli) from the latest RW61x SDK (v2.16.0 onwards) to load external Wi-Fi calibration data.

Step 1 – Define OVERRIDE_CALIBRATION_DATA and CONFIG_CUSTOM_CALDATA macros in the /source/wifi_config.h file.

```
#define OVERRIDE_CALIBRATION_DATA 1
#define CONFIG_CUSTOM_CALDATA 1
```

Step 2 – Define OVERRIDE_CALIBRATION_DATA and calibration data header file, wifi_cal_data_ext.h, in /wifi/wlcmgr/wlan.c.

```
#ifdef OVERRIDE_CALIBRATION_DATA
#include <wifi_cal_data_ext.h>
```

Step 3 – In /wifi/incl/wifi_cal_data_ext.h, copy the external calibration data from WlanCalData_ext.conf into the ext_cal_data array. The external calibration data is in hexadecimal with leading 0x format. Refer to AN13639-RW61x Calibration Structure for more information on generating WlanCalData_ext.conf file.

Example of ext_cal_data:

```
const uint8_t ext_cal_data[] = {
      0x01, 0x00,0x0F, 0x00, 0xD0, 0x01, 0x00, 0x20, 0xCE, 0x0F, 0x00, 0x00, 0x00, 0x20, 0xFF, 0xFF, 0x40, 0x00,
 0x77, 0x00, 0x28, 0x00,0x00, 0x00, 0x00, 0x10, 0x00, 0x04, 0x26, 0x79, 0x02, 0x00, 0x00, 0x3F, 0x01, 0x00, 0x00,
 0x12, 0x00, 0x8C, 0x37, 0x61, 0x00, 0x00, 0x00, 0xAC, 0xFF, 0xF0, 0x08, 0x00, 0x00, 0x05, 0x01, 0x00, 0x3B, 0x9E,
 0x2C, 0x23, 0x04, 0xD0, 0x6F, 0xEC, 0x0C, 0x62, 0x98, 0x7C, 0x00, 0x0B, 0x01, 0x01, 0x3B, 0xAE, 0x30, 0x24, 0x04,
 0xF0, 0x77, 0xEC, 0x0C, 0x72, 0x9C, 0x7E, 0x44, 0x28, 0x01, 0xF7, 0x3B, 0xAE, 0x4C, 0x25, 0x04, 0x90, 0x6B, 0xEB,
 0x0C, 0x12, 0x68, 0x74, 0x44, 0x38, 0x01, 0xF8, 0x3B, 0xDE, 0x58, 0x26, 0x04, 0xB0, 0x77, 0xED, 0x0C, 0x22, 0x6C,
 0x76, 0x48, 0x6C, 0x01, 0x02, 0x3C, 0x5E, 0x7C, 0x27, 0x05, 0x30, 0x97, 0xF7, 0x0C, 0xA2, 0x8C, 0x7D, 0x48, 0x84,
 0x01, 0x02, 0x3B, 0xDE, 0x58, 0x28, 0x04, 0xC0, 0x77, 0xEF, 0x0C, 0x42, 0x74, 0x78, 0x4C, 0x99, 0x01, 0x00, 0x3C,
 0x9E, 0x8C, 0x28, 0x05, 0x50, 0xA3, 0xFA, 0x0C, 0x42, 0x84, 0x7E, 0x4C, 0xA5, 0x01, 0xFE, 0x3B, 0xFE, 0x60, 0x29,
 0x04, 0xA0, 0x77, 0xEF, 0x0B, 0xC2, 0x60, 0x73, 0x00, 0x18, 0x2B, 0x53, 0x00, 0x00, 0x00, 0xC4, 0x39, 0x03, 0x93,
 0xD8, 0xBC, 0x6B, 0x70, 0x1B, 0x3D, 0x5F, 0xC9, 0xB2, 0x41, 0x65, 0xB2, 0xDF, 0x00, 0x70, 0xA8, 0x5A, 0x00, 0x00,
 0x01, 0x34, 0x00, 0x07, 0x02, 0x04, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,
 0xFF, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x00,
 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00,
 0x01, 0x00, 0x01, 0x00, 0x01, 0xFF, 0xFF, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x31, 0x63, 0x00, 0x00, 0x01, 0x44,
 0xF1, 0x11, 0x00, 0x01, 0x00, 0x00, 0x0D, 0x08, 0x00, 0x2C, 0x80, 0x4B, 0x00, 0x00, 0x01, 0x70, 0xFF, 0xFF, 0x04,
 0x00, 0x00, 0xD6, 0xDA, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0xDB, 0xDD, 0x47, 0x00, 0x00, 0x00, 0x00, 0x00, 0xD6,
 0xD9, 0x4B, 0x00, 0x00, 0x00, 0x00, 0x00, 0xCF, 0xCD, 0x4F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x5B, 0x62, 0x00,
 0x00, 0x01, 0x8C, 0xFF, 0xFF, 0x04, 0x00, 0x00, 0x02, 0x10, 0x84, 0x44, 0x00, 0x84, 0x21, 0x48, 0x01, 0x8C, 0x63,
 0x4C, 0x02, 0x10, 0x84, 0x00, 0x1C, 0xBD, 0x37, 0x00, 0x00, 0x01, 0xA8, 0x06, 0x04, 0x77, 0x01, 0x00, 0x00, 0x00,
 0x28, 0x00, 0x2D, 0xC6, 0xC0, 0xDA, 0x21, 0x12, 0x89, 0x10, 0xF0, 0xC0, 0x95, 0x00, 0x18, 0x7F, 0x68, 0x00, 0x00,
 0x01, 0xC0, 0x00, 0x00, 0x00, 0x01, 0x50, 0x60, 0x70, 0x80, 0x10, 0x20, 0x30, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x10, 0x8F, 0x64, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01
     }
```

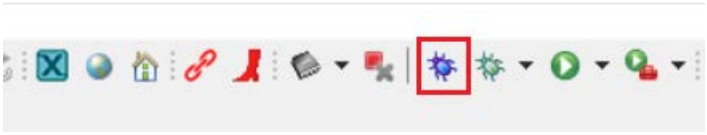
Step 4 – Define `test_get_cal_data` in */wifi/wlcmgr/wlan_enhanced_tests.c*.

```
30 static void test_get_cal_data(int argc,char **argv)
31 {
32     wlan_cal_data_t cal_data;
33     wlan_get_cal_data(&cal_data);
34     for(int i=0;i<cal_data.data_len;i++)
35     {
36         PRINTF("0x%x ",cal_data.data[i]);
37     }
38 }
```

Step 5 – Define the function `wlan-get-cal-data` in */wifi/wlcmgr/wlan_enhanced_tests.c*. The command is used to read back the calibration data.

```
2025     {"wlan-set-clocksync", "<mode> <role> <gpio_pin> <gpio_level> <puls
2026 #endif /* CONFIG_WIFI_CLOCKSYNC */
2027     {"wlan-get-cal-data","wlan-get-cal-data",test_get_cal_data},
2028 };
```

Step 6 – Build and flash the Wi-Fi example application onto RW61x.

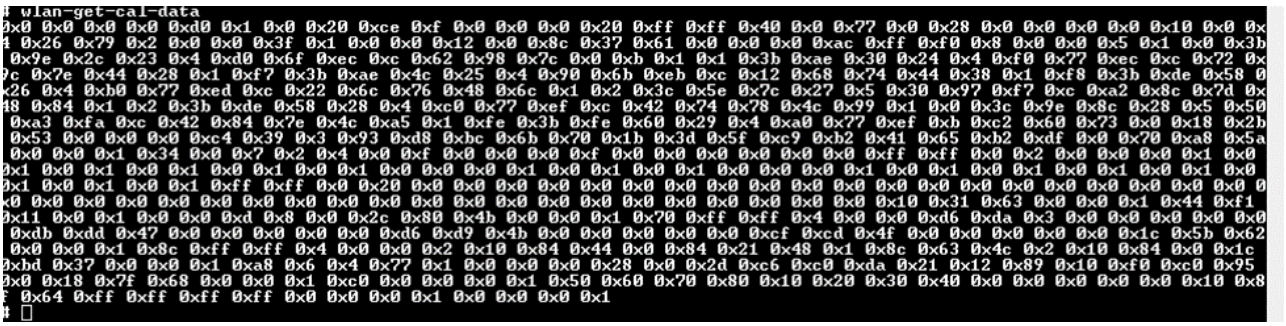


Step 7 – Open a serial terminal.

Step 8 – Read the external calibration data.

```
# wlan-get-cal-data
```

Example of output:



7.2 Bluetooth calibration data

Bluetooth calibration data is used to configure Bluetooth parameters such as the TX power class, crystal frequency, and front-end configuration. The calibration data can be stored in the on-chip OTP memory or externally in an application. This section explains how to modify a Bluetooth LE example (in this case, *edgefast_bluetooth_shell*) from the latest RW61x SDK (v2.16.0 onwards) to load Bluetooth-only external calibration data.

Step 1 – Set `gPlatformDisableSetBtCalData` and `gPlatformDisableSetBtCalDataAnnex100_d` to 0 in */framework/Platform/config/fwk_config.h.*

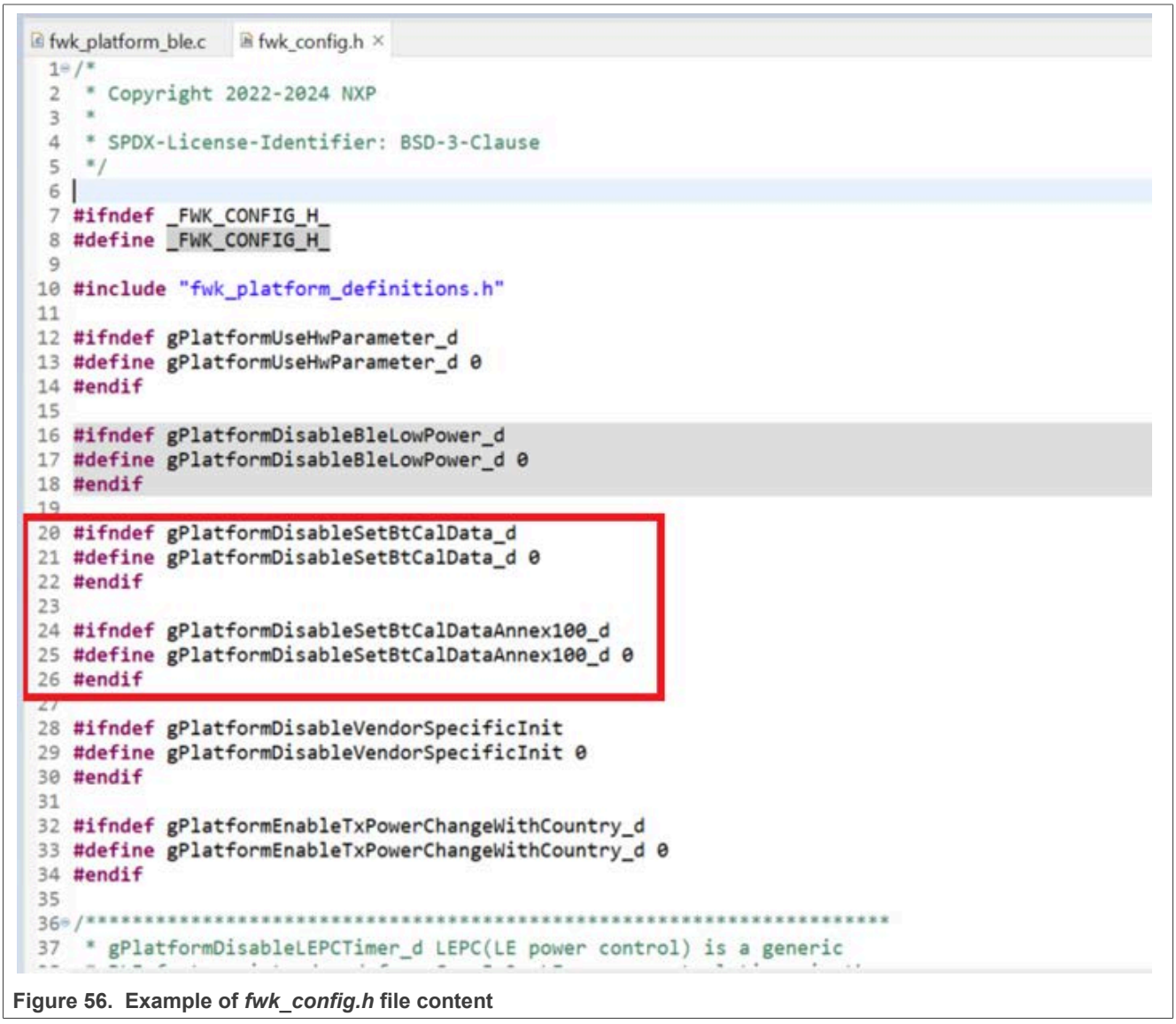


Figure 56. Example of *fwk_config.h* file content

Step 2 – Verify that `PLATFORM_VendorSpecificInit()` is called in */framework/Platform/fwk_platform_ble.c.*

Note: *By default, Bluetooth initialization (`bt.init`) calls this function to load the external Bluetooth calibration with the correct timings to load.*

fwk_config.h | fwk_platform_ble.c

```
418     } while (false);
419
420     status = OSA_MutexUnlock((osa_mutex_handle_t)bleMutexHandle);
421     assert(status == KOSA_StatusSuccess);
422     (void)status;
423
424     return ret;
425 }
426
427 void PLATFORM_VendorSpecificInit(void)
428 {
429 #if !defined(gPlatformDisableSetBtCalData_d) || (gPlatformDisableSetBtCalData_d == 0)
430     /* Send the BT Cal Data to Controller */
431     (void)PLATFORM_SetBtCalData();
432 #if !defined(gPlatformDisableSetBtCalDataAnnex100_d) || (gPlatformDisableSetBtCalDataAnnex100_d == 0)
433     /* After send annex55 to CPU2, CPU2 need reset,
434        a delay of at least 20ms is required to continue sending annex100*/
435     OSA_TimeDelay(BLE_RESET_DELAY_MS);
436
437     /* Send the BT Cal Data annex100 to Controller */
438     (void)PLATFORM_SetBtCalDataAnnex100();
439 #endif
440 #endif
```

Figure 57. `PLATFORM_VendorSpecificInit` **function**

Step 3 – Comment out or delete all but the first 4 bytes of the existing annex 55 parameters `hci_cal_data_params` in */framework/Platform/fwk_platform_ble.c.*. The first four parameters do not change.

```
290 static const uint8_t hci_cal_data_params[HCI_CMD_STORE_BT_CAL_DATA_PARAM_LENGTH] = {
291     0x00U,                              //  Sequence Number : 0x00
292     0x00U,                              //  Action : 0x00
293     0x01U,                              //  Type : Not use CheckSum
294     0x1CU,                              //  File Length : 0x1C
295 /*
296     0x37U,                              //  BT Annex Type : BT CFG
297     0x71U,                              //  Checksum : 0x71
298     0x1CU,                              //  Annex Length LSB: 0x001C
299     0x00U,                              //  Annex Length MSB: 0x001C
300     0xFFU,                              //  Pointer For Next Annex[0] : 0xFFFFFFFF
301     0xFFU,                              //  Pointer For Next Annex[1] : 0xFFFFFFFF
302     0xFFU,                              //  Pointer For Next Annex[2] : 0xFFFFFFFF
303     0xFFU,                              //  Pointer For Next Annex[3] : 0xFFFFFFFF
304     0x01U,                              //  Annex Version : 0x01
305     0x7CU,                              //  External Xtal Calibration Value : 0x7C
306     0x04U,                              //  Initial TX Power : 0x04
307     BT_CAL_DATA_ANNEX_FRONT_END_LOSS,   //  Front End Loss : 0x02 or 0x03
308     0x28U,                              //  BT Options :
309                                         //              BIT[0] Force Class 2 operation = 0
310                                         //              BIT[1] Disable Pwr Control for class 2= 0
311                                         //              BIT[2] MiscFlag(to indicagte external XTAL) = 0
312                                         //              BIT[3] Used Internal Sleep Clock = 1
313                                         //              BIT[4] BT AOA localtion support = 0
314                                         //              BIT[5] Force Class 1 mode = 1
315                                         //              BIT[7:6] Reserved
316     0x00U,                              //  AOANumberOfAntennas: 0x00
317     0x00U,                              //  RSSI Golden Low : 0
318     0x00U,                              //  RSSI Golden High : 0
319     0xC0U,                              //  UART Baud Rate[0] : 0x002DC6C0(3000000)
320     0xC6U,                              //  UART Baud Rate[1] : 0x002DC6C0(3000000)
321     0x2DU,                              //  UART Baud Rate[2] : 0x002DC6C0(3000000)
322     0x00U,                              //  UART Baud Rate[3] : 0x002DC6C0(3000000)
323     0x00U,                              //  BdAddress[0] : 0x000000000000
324     0x00U,                              //  BdAddress[1] : 0x000000000000
325     0x00U,                              //  BdAddress[2] : 0x000000000000
326     0x00U,                              //  BdAddress[3] : 0x000000000000
327     0x00U,                              //  BdAddress[4] : 0x000000000000
328     0x00U,                              //  BdAddress[5] : 0x000000000000
329     0xF0U,                              //  Encr_Key_Len[3:0]: MinEncrKeyLen = 0x0
330                                         //  Encr_Key_Len[7:4]: MaxEncrKeyLen = 0xF
331 #if defined(gPlatformEnableTxPowerChangeWithCountry_d) && (gPlatformEnableTxPowerChangeWithCountry_d == 0)
332     0x00U, //  RegionCode : 0x00
333 #else
334     0x00U, //  Reserved : 0x00
335 #endif  gPlatformEnableTxPowerChangeWithCountry_d */
336 };
```

Figure 58. `hci_cal_data_params` **with default values commented out**

Step 4 – Create *BtCalData_ext.conf* file using Labtool command `53` [4]. Open the file.

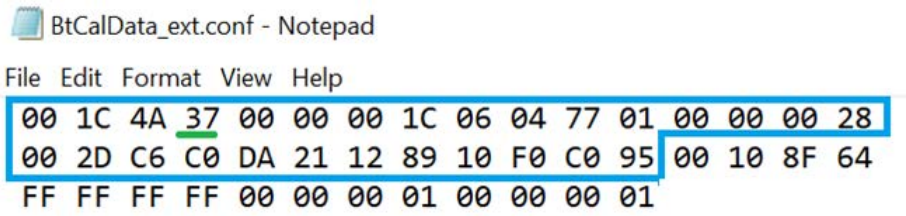


Figure 59. First 28 bytes of *BtCalData_ext.conf* file

Step 5 – Starting from “37”, convert 28 bytes of *BtCalData_ext.conf* to unsigned 4-byte aligned little-endian format.

Define `hci_cal_data_params` in */framework/Platform/fwk_platform_ble.c* file.

```
337 static const uint8_t hci_cal_data_params[HCI_CMD_STORE_BT_CAL_DATA_PARAM_LENGTH] = {
338     0x00U,                                  //  Sequence Number : 0x00
339     0x00U,                                  //  Action : 0x00
340     0x01U,                                  //  Type : Not use CheckSum
341     0x1CU,                                  //  File Length : 0x1C
342     0x37U, 0x4AU, 0x1CU, 0x00U,
343     0x1CU, 0x00U, 0x00U, 0x00U,
344     0x01U, 0x77U, 0x04U, 0x06U,
345     0x28U, 0x00U, 0x00U, 0x00U,
346     0xC0U, 0xC6U, 0x2DU, 0x00U,
347     0x89U, 0x12U, 0x21U, 0xDAU,
348     0x95U, 0xC0U, 0xF0U, 0x10U,
349 };
```

Figure 60. Updated `hci_cal_data_params` array with custom calibration data

Step 6 – Comment out or delete all the existing annex 100 parameters hci_cal_data_annex100_params / *framework/Platform/fwk_platform_ble.c.*

```
352 static const uint8_t hci_cal_data_annex100_params[HCI_CMD_STORE_BT_CAL_DATA_PARAM_ANNEX100_LENGTH] = {
353     /*                  BT_HW_INFO   START              */
354
355
356 /*
357     0x64U, //  Annex Type : 0x64
358     0x00U, //  CheckSum: Annex100 ignores checksum
359     0x10U, //  Length-In-Byte : 0x0010
360     0x00U, //  Length-In-Byte : 0x0010
361     0xFFU, // Pointer for next annex structure : 0xFFFFFFFF
362     0xFFU, // Pointer for next annex structure : 0xFFFFFFFF
363     0xFFU, // Pointer for next annex structure : 0xFFFFFFFF
364     0xFFU, // Pointer for next annex structure : 0xFFFFFFFF
365     0x01U, // Ext_PA Gain : Bit[7:1]   Ext_PA Present : Bit[0]
366 #if defined(gPlatformEnableTxPowerChangeWithCountry_d) && (gPlatformEnableTxPowerChangeWithCountry_d == 0)
367     0x00U, // Ext_Ant Gain : Bit[4:1]   Ext_Ant Present : Bit[0]
368 #else
369     0x00U, // Reserved
370 #endif                                          gPlatformEnableTxPowerChangeWithCountry_d
371     BT_CAL_DATA_ANNEX_100_EPA_FEM_MASK_LOW_BYTE, // BT_HW_INFO_EPA_FEM_Mask
372     0x00U,                                       // BT_HW_INFO_EPA_FEM_Mask
373     0x01U,                                       // Ext_LNA Present : Bit[0]   Ext_LNA Gain : Bit[7:1]
374     0x00U,                                       // multipurpose mask
375     BT_CAL_DATA_ANNEX_100_LNA_FEM_MASK_LOW_BYTE, // BT / LE ext LNA FEM BITMASK
376     0x00U,                                       // BT / LE ext LNA FEM BITMASK
377 */
378
379
380     /*                  BT_HW_INFO   END            */
381 };
382 #endif
```

Figure 61. `hci_cal_data_annex100_params` **with default values commented out**

Step 7 – Refer back to the *BtCalData_ext.conf* file.

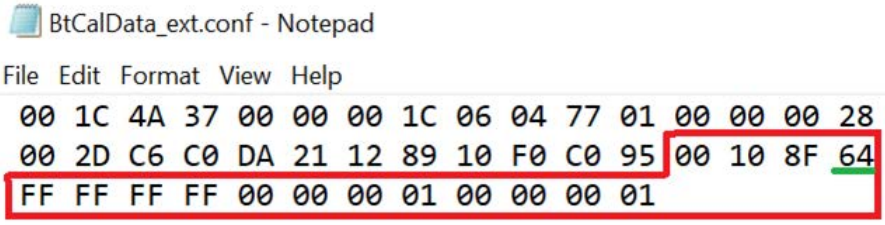


Figure 62. Last 16 bytes of *BtCalData_ext.conf* file

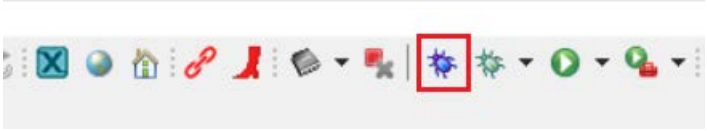
Step 8 – Starting from “64”, convert 16 bytes of *BtCalData_ext.conf* to unsigned 4 byte aligned little endian format and define hci_cal_data_params in /framework/Platform/fwk_platform_ble.c

```
#if !defined(gPlatformDisableSetBtCalDataAnnex100_d) || (gPlatformDisableSetBtCalDataAnnex100_d == 0)
/*
 * a.The following parameters are used in three cases,
 *    1.For share antenna case or ant2 with external FEM(ble only case).
 *    2.diversity case(enable ant3)
 *    3.diversity case(enable ant4)
 */
static const uint8_t hci_cal_data_annex100_params[HCI_CMD_STORE_BT_CAL_DATA_PARAM_ANNEX100_LENGTH] = {
    /*                  BT_HW_INFO   START              */
    0x64U, 0x8FU, 0x10U, 0x00U,
    0xFFU, 0xFFU, 0xFFU, 0xFFU,
    0x01U, 0x00U, 0x00U, 0x00U,
    0x01U, 0x00U, 0x00U, 0x00U,
    /*                  BT_HW_INFO   END                */

};
#endif
```

Figure 63. Updated hci_cal_data_annex100_params array with custom calibration data

Step 9 – Build and flash the Bluetooth example application onto RW61x.



Step 10 – Open a serial terminal.

Step 11 – Initialize Bluetooth.

Note: *The external Bluetooth calibration data is loaded after Bluetooth is initialized when* `PLATFORM_VendorSpecificInit()` *is called. Refer back to* ***Step 2*** *for more information.*

```
@bt > bt.init
```

Example of output:

```
@bt> bt.init
@bt> Bluetooth initialized
Settings Loaded

@bt>
```

8 Coexistence application

The coexistence application allows both Wi-Fi and Bluetooth LE to exist simultaneously. The standard SDK release (2.16.000 and above) includes the coexistence source and configuration files in the */middleware/wireless/coex* directory. This section explains how to compile and run the coexistence application.

Note: *For the coexistence application running Wi-Fi, Bluetooth LE, and Openthread, refer to middleware/wireless/coex/examples/coex_app/readme.md. The .md file explains the hardware and compilation requirements.*

Figure 64 shows the content of */middleware/wireless/coex* directory.

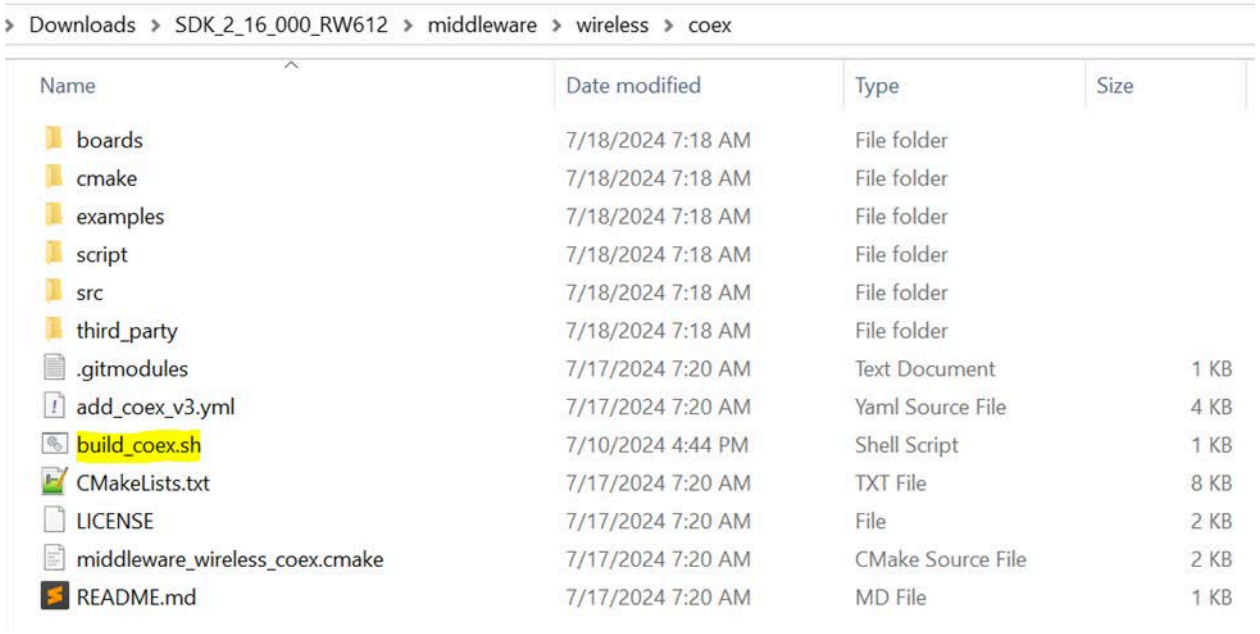


Figure 64. Content of */middleware/wireless/coex* directory

Step 1 – Configure and run the *build_coex.sh* script.

Modify the `NXP_RW612_SDK_ROOT` and `ARMGCC_DIR` parameters in the */wireless/coex/build_coex.sh* script with the SDK and toolchain path of your system.

Sample of *build_coex.sh* content:

```
export NXP_RW612_SDK_ROOT='../../../../SDK_2_16_000_RD-RW612-BGA'
export ARMGCC_DIR='/c/Program Files (x86)/GNU Arm Embedded Toolchain/10 2021.10'
rm -rf build_rw612
./script/build_rw612 coex_wpa_supplicant -DCOEX_ENABLE_WIFI=ON -DCOEX_ENABLE_BLE=ON -
DCOEX_ENABLE_OT=OFF -DCOEX_NXP_BASE=edgefast -DCOEX_EXAMPLE_BOARD=rdrw612bga
```

Step 2 – Open Segger J-Link tool and connect RW61x.

Example of output:

```
J-Link>connect
Device>RW612
TIF>S
Speed><Enter>
```

Step 3 – Flash the coexistence application by dragging and dropping the coexistence application binary into Segger J-Link tool.

Example of command:

```
loadbin ${SDK-top-dir}/middleware/wireless/coex/build_rw612/../coex_wpa_supplicant.bin,
 0x08000000
```

Example of command output:

```
ResetTarget() start
Reset via SYSRESETREQ and reset pin + halt after bootloader
ROM entered ISP command handling loop. Re-enable the debug access.
 MSPLIM cleared
ResetTarget() end - Took 124ms
Downloading file [C:\SDK_2_16_000_RD-RW612-BGA\middleware\wireless\coex
\build_rw612\rw612_coex_wpa_supplicant\bin\coex_wpa_supplicant.bin]...
J-Link: Flash download: Bank 0 @ 0x08000000: 1 range affected (3932160 bytes)
J-Link: Flash download: Total: 54.298s (Prepare: 0.208s, Compare: 13.034s, Erase:
 17.231s, Program: 17.254s, Verify: 6.459s, Restore: 0.109s)
J-Link: Flash download: Program speed: 222 KB/s
O.K.
```

Step 4 – Open MCUXpresso and locate the firmware binaries under */components/conn_fwloader/fw_bin/*.

Step 5 – Flash the Wi-Fi firmware by dragging and dropping the Wi-Fi firmware binary into Seggger J-Link tool.

Example of command:

```
J-Link>loadbin ${SDK-top-dir}/components/conn_fwloader/fw_bin/
rw61x_sb_wifi_a2.bin,0x08400000
```

Example of command output:

```
'loadbin': Performing implicit reset & halt of MCU.
ResetTarget() start
Reset via SYSRESETREQ and reset pin + halt after bootloader
ROM entered ISP command handling loop. Re-enable the debug access.
MSPLIM cleared
ResetTarget() end - Took 122ms
Downloading file [C:\SDK_2_16_000_RD-RW612-BGA\components\conn_fwloader\fw_bin
\rw61x_sb_wifi_a2.bin]...
J-Link: Flash download: Bank 0 @ 0x08000000: 1 range affected (589824 bytes)
J-Link: Flash download: Total: 8.101s (Prepare: 0.207s, Compare: 1.868s, Erase: 2.535s,
 Program: 2.416s, Verify: 0.965s, Restore: 0.109s)
J-Link: Flash download: Program speed: 238 KB/s
O.K.
```

Step 6 – Flash the Bluetooth LE firmware by dragging and dropping the Bluetooth LE firmware binary into Seggger J-Link tool.

Example of command:

```
loadbin ${SDK-top-dir}/components/conn_fwloader/fw_bin/rw61x_sb_ble_a2.bin,0x08540000
```

Example of command output:

```
'loadbin': Performing implicit reset & halt of MCU.
ResetTarget() start
Reset via SYSRESETREQ and reset pin + halt after bootloader
ROM entered ISP command handling loop. Re-enable the debug access.
MSPLIM cleared
ResetTarget() end - Took 123ms
Downloading file [C:\SDK_2_16_000_RD-RW612-BGA\components\conn_fwloader\fw_bin
\rw61x_sb_ble_a2.bin]...
J-Link: Flash download: Bank 0 @ 0x08000000: 1 range affected (262144 bytes)
J-Link: Flash download: Total: 3.704s (Prepare: 0.208s, Compare: 0.898s, Erase: 1.133s,
 Program: 0.925s, Verify: 0.428s, Restore: 0.109s)
J-Link: Flash download: Program speed: 276 KB/s
O.K.
```

Step 7 – Close MCUXpresso and Segger J-Link tool.

Step 8 – Open a serial terminal.

Step 9 – Turn OFF/ON the RW61x board.

Step 10 – Enter a Wi-Fi command.

Example of command:

```
wifi.wlan-version
```

Example of command output:

```
WLAN Driver Version   : v1.3.r48.p25
@Coex> WLAN Firmware Version : rw610w-V2, IMU, FP99, 18.99.6.p19, PVE_FIX 1
```

Step 11 – Enter a Bluetooth Low Energy command.

Example of command:

```
bt.init
```

Example of command output:

```
Bluetooth initialized
Settings Loaded
```

9 Abbreviations

Table 23. Abbreviations

Abbreviation	Definition
ACS	Auto channel selection
AES	Advanced encryption standard
AP	Access point
API	Application program interface
AWS	Amazon web services
Bluetooth LE	Bluetooth Low Energy
BSS	Basic service set
CGI	Common gateway interface
CLI	Command line interface
CMSIS	Cortex® Microcontroller Software Interface Standard
CSI	Channel state information
DDP	Device provisioning protocol
DFP	Device family pack
DHCP	Dynamic host configuration protocol
DHCPD	DHCP daemon
DPP	Device provisioning protocol
ECSA	Extended channel switch announcement
ED	Energy detection
ETSI	European Telecommunications Standards Institute
EU	European Union
EVK	Evaluation kit
Ext AP	External access point
Ext STA	External station
FW	Firmware
HCI	Host controller interface
HTS	Health thermometer service
HTTP	Hypertext transfer protocol
IDE	Integrated development environment
IP	Internet protocol
IPSP	Internet protocol support profile
lwIP	Lightweight IP
MEF	Memory efficiency filtering
MFP	Management frame protection
MQTT	Message queuing telemetry transport

Table 23. Abbreviations...*continued*

Abbreviation	Definition
NAT	Network address translation
OFDMA	Orthogonal frequency division multiple access
OFDMA	Orthogonal frequency division multiple access
OTP	One time programmable
PBC	Push button configuration
PIN	Personal identification number
PS	Power save
PXM	Proximity monitor
PXR	Proximity reporter
QR	Quick response (code)
RSSI	Received signal strength indicator
Rx	Receive
SAD	Single antenna diversity
SD	Secure digital
SDK	Software development kit
SPP	Serial port profile
SSI	Server side includes
SSID	Service set identifier
STA	Station/client
SW	Software
TCP	Transmission control protocol
TRPC	Transmit rate-based power control
TX	Transmit
uAP	Mobile AP
UAPSD	Unscheduled automatic power save delivery
UART	Universal asynchronous receiver transmitter
UDP	User datagram protocol
USB	Universal serial bus
WLAN	Wireless local area network
WMM	Wireless multimedia
WNM	Wireless network management
WPA	Wi-Fi protected access
WPS	Wi-Fi protected setup

10 References

[1] Application note – AN14281: Channel State Information on FreeRTOS (link)
[2] Application note – AN14463: Antenna Diversity
[3] User manual – UM11798: Getting Started with Wireless on RW61x Evaluation board Running RTOS
[4] User manual – UM11801: Manufacturing Software User Manual for RW61x (link)
[5] Specification – Bluetooth Core Specification (link)
[6] Mobile application – IoT Toolbox Android (IoT Toolbox on Google Play) and (IoT Toolbox on the APP Store)
[7] Gitlab – CMake project (link)
[8] GitHub – EchoTool (link)
[9] Webpage – iPerf (link)
[10] Webpage – Perl Download (link)
[11] Webpage – Nmap (link)
[12] Webpage – Wireshark (link)

11 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2022-2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

12 Revision history

Table 24. Revision history

Document ID	Release date	Description
UM11799 v.10.0	24 March 2025	• Changed the document access to public. • No changes in the content.
UM11799 v.9.0	8 January 2025	• Section 3 "Running a demo": created the section and moved Section 3.1, Section 3.2, Section 3.3, and Section 3.4. • Section 4.1.2.21 "CSI commands": updated. • Section 7 "Load external calibration data": added. • Section 8 "Coexistence application": added. • Section 10 "References": updated.
UM11799 v.8.0	5 September 2024	• Section 4.1.2.12 "Wi-Fi power save": updated the code example for IEEEPS usage. • Section 4.3.1.2 "Set/get the region code": updated the region codes in the command usage and command examples. • *Set the channel list and TX power limit*: removed the section. • Section 4.3.1.4 "Set/get TX rate configuration": added. • Section 4.3.1.5 "Get the management frame protection capability": removed the set command. • Section 4.3.1.6: added. • Section 4.9.1.6 "Set Wi-Fi transmitter in continuous wave (CW) mode": corrected the data rate index values in the table *802.11n/a/g/b data rate index*. • Section 4.9.1.7 "Transmit standard 802.11 packets": – Updated the command usage (TX data rate). – Updated the code sample showing how to enable TX frame. – Added a note about <data_rate> parameter. • Section 6.11.1.2 "Bluetooth LE RF test mode operations": added notes about `tx-power` and `tx_channel` parameter values to be hexadecimal.
UM11799 v.7.0	22 April 2024	• Section 4.1.2.1 "Start-up logs": updated. • Section 4.1.2.2 "Help command": updated. • Section 4.1.2.4 "Add a network profile": updated. • Section 4.1.2.5 "Station mode (connect to AP)": added `psk` to WPA2 code sample. • Section 4.1.2.8 "Start the mobile AP": added `psk` to WPA2 code sample. • Section 4.1.2.13 "Host sleep": replaced `wlan-host-sleep` with `wlan-auto-host-sleep` in the code samples. • Section 4.1.2.23 "ECSA command": replaced `wlan-set` with `wlan-uap-set` in the code samples. • Section 4.1.2.25 "Set/get the antenna configuration": added. • Section 4.1.2.26 "Other useful CLI commands": updated the code sample for "Configure hidden SSID". • Section 4.9.1.1 "Run the application": updated • Section 4.9.1.9 "Set/get OTP MAC address": added. • Section 4.9.1.10 "Set/get OTP calibration data": added. • Section 4.10 "wifi_wpa_supplicant sample application": added. • Section 9 "Abbreviations": added OFDMA, DDP, and SAD.

Table 24. Revision history...*continued*

Document ID	Release date	Description
UM11799 v.6.0	9 October 2023	• Section 4.1.1 "Flash the Wi-Fi firmware": updated the firmware name • Section 3.2.3 "Flash the application program (no debugging)": updated the NOR Flash base address in the command example. • Section 4.1.2.1 "Start-up logs": updated `wlan-host-sleep` • Section 4.1.2.2 "Help command": updated `wlan-host-sleep` • Section 4.1.2.13 "Host sleep": updated • Section 4.1.2.14 "Suspend": added • Section 4.1.2.15 "Wake-up conditions": added • Section 4.1.2.16 "Multi MEF configuration": added • Section 4.3.1.1 "Run the application": updated • Section *Set/get TX rate configuration*: removed • Section *Set/get TX power limit*: removed • Section *Set/get antenna diversity configuration*: removed • Section 4.6.1.1 "Run the application": updated • Section 6.1.1 "Flash Bluetooth LE firmware": updated the firmware name
UM11799 v.5.0	19 July 2023	• Section 2.3 "IPerf remote host setup":changed iPerf version to 2.1.9 • Section 4.9 "wifi_test_mode sample application": added the section • Section 6.11 "Shell sample application": added the section • Section 9 "Abbreviations": added OFDMA • Section 11 "Note about the source code in the document": added the section
UM11799 v.4.0	12 May 2023	• Section 2.3 "IPerf remote host setup": added a note about the default TCP/UDP port, and changed the version to 2.0.9 • Section 2.4 "IPv4 and IPv6 tool setup": added the first paragraph • Section 2.5 "J-Link commander setup": added the section • Table 4 "Sample application features": updated Wi-Fi features • Figure 31 "RW61x FlexSPI flash layout": added • Section 3.2.1 "Install ARM GCC toolchain": updated • Section 3.4 "Run a demo using Keil MDK/µVision": added the section • Section 4.1.2.1 "Start-up logs": updated • Section 4.1.2.5 "Station mode (connect to AP)": added the content for WPA3 security • Section 4.1.2.13 "Host sleep": updated • Section 4.1.2.18 "802.11k commands": updated • Section 4.6 "wifi_cli_prov sample application": added • Section 4.7 "wifi_httpsrv sample application": added • Section 4.8 "wifi_mqtt sample application": added • Section 9 "Abbreviations": updated

Table 24. Revision history...*continued*

Document ID	Release date	Description
UM11799 v.3.0	16 January 2023	• Section 3.2.3 "Flash the application program (no debugging)": corrected the start address of the application image
UM11799 v.2.0	10 August 2022	• Section 2.4 "IPv4 and IPv6 tool setup": added the section • Section 4.1.1 "Flash the Wi-Fi firmware": updated the instructions to flash Wi-Fi firmware • Section 3.1 "Run a demo using MCUXpresso IDE": added the section • Section 4.1.2.1 "Start-up logs": updated • Section 4.1.2.2 "Help command": updated • Section 4.1.2.12 "Wi-Fi power save": updated • Section 4.1.2.13 "Host sleep": added the section • Section 4.1.2.17 "Wi-Fi reset": added the section • Section 4.1.2.18 "802.11k commands": added the section • Section 4.1.2.19 "802.11d commands": added the section • Section 4.1.2.20 "Roaming commands": added the section • Section 4.1.2.21 "CSI commands": added the section • Section 4.1.2.22 "Net monitor commands": added the section • Section 4.1.2.23 "ECSA command": added the section • Section 4.1.2.24 "EU crypto commands": added the section • Section 4.1.2.26 "Other useful CLI commands": . Added Get the Wi-Fi IP address, Set max station count for the mobile AP, Configure TX PER setting, Get Wi-Fi STA and mobile AP log, and Get WMM TX State . Removed Send RF calibration host command and Enable 802.11d command • Section 4.5 "wifi_ipv4_ipv6_echo sample application": added the section • Section 6.1.1 "Flash Bluetooth LE firmware": added the path to Bluetooth LE secure firmware binary • Section 6.10 "Wireless UART sample application": added the section • Section 9 "Abbreviations": updated
UM11799 v.1.0	9 May 2022	Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Figures

Contents