UG10171

NXP Wi-Fi and Bluetooth Demo Applications for FRDM-RW61x

Rev. 1.0 — 20 September 2024 **User guide**

Document information

Information	Content
Keywords	Wireless MCU RW610/RW612, FRDM-RW61x 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

This section outlines the purpose, scope, and key considerations for effectively utilizing this user guide.

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 FRDM-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, FRDM-RW61x information, hardware interconnection, board settings, bring-up, IDE setup, nor SDK download. These items are covered in [1]. The user must have FRDM-RW61x platform-related IDE and tools installed before going through the given demo process.

2 Tool setup

This section describes the setup procedures for different tools that are used for Wi-Fi and Bluetooth example applications.

2.1 Serial console tool setup

The serial console tool is used to read out the demo application logs on the computer connected to the FRDM-RW61x 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 FRDM-RW61x to the host computer running on Linux, Mac OS, or Windows.
- Open a terminal emulator program like Minicom or Tera term.
- For Minicom use the 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 the Wireshark tool for Windows and Mac OS from here.

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 from **here**.

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 a Debian package of iPerf 2.1.9 for Ubuntu 16.04 from **here**

```
$ 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 mobile devices

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 IPv6 tool is used to send or receive data via TCP or UDP connection to interact with the wifi_ipv4_ipv6_echo sample application (Section 3.5).

Remote host setup

- **ncat** - Recommended tool that supports both IPv4 and IPv6. ncat is part of nmap tools. See nmap website at https://nmap.org/download.html
- **nc (netcat)** - Similar to ncat. Antivirus applications tend to tag ncat as a virus, so it may be available for use on a PC or laptop.
- **echotool** - Supports only IPv4 and only for Windows. Can be downloaded from https://github.com/PavelBansky/EchoTool

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 the 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 the *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 FRDM-RW61x, additional patches are needed for the tools. Refer to the section FRDM-RW61x product image setup in the user manual reference UM11798.*

3 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.

3.1 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 the iPerf measurement tool

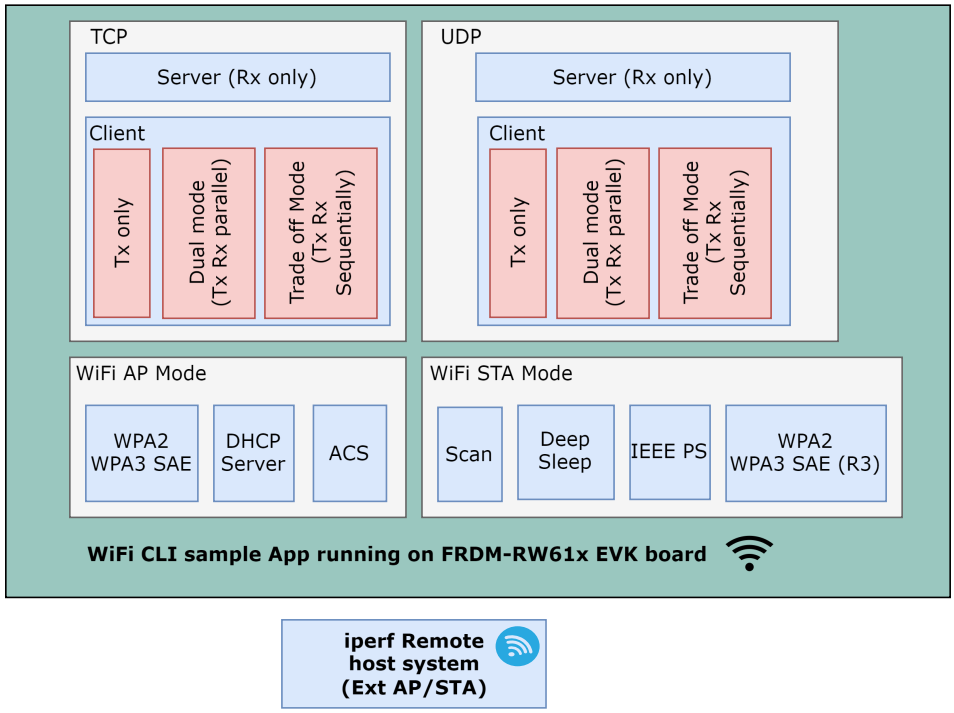

Figure 1. wifi_cli sample application components

Wi-Fi and iPerf features:

Table 4. Sample application features

Features	Details
Wi-Fi	Wi-Fi Soft 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)

3.1.1 Flash the Wi-Fi firmware

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

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

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

- Flash Wi-Fi firmware

The path to the Wi-Fi secure firmware binary is:

${SDK}\components\conn_fwloader\fw_bin\rw61x_sb_wifi_v2.bin for the A2 version of FRDM-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 2). Ensure that the Wi-Fi firmware is flashed before running any Wi-Fi demo application.*

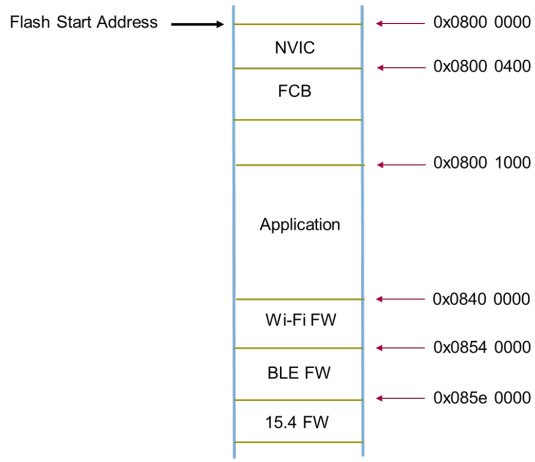

Figure 2. FRDM-RW61x FlexSPI flash layout

3.1.2 Run a demo using MCUXpresso IDE

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

3.1.2.1 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 3)

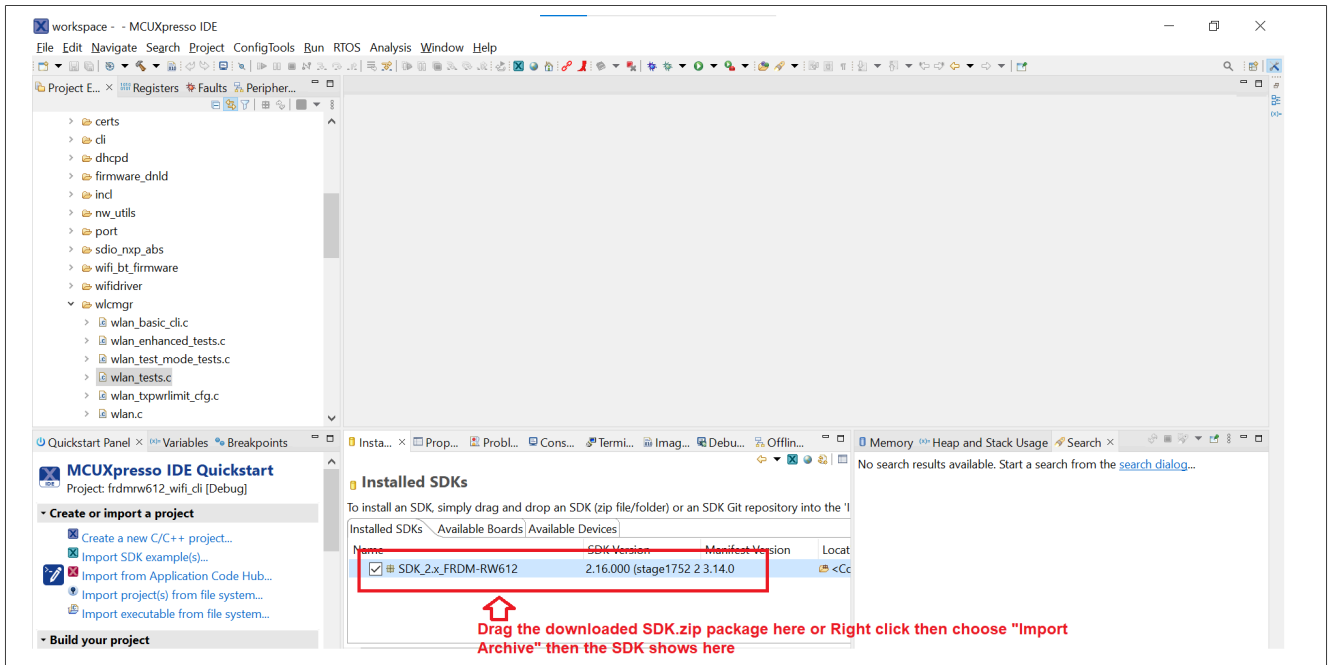

Figure 3. Drag and drop the downloaded SDK into Installed SDK tab

Step 2 - Import an example

- Go to the *Quickstart* panel and select the option *Import SDK examples* (Figure 4)

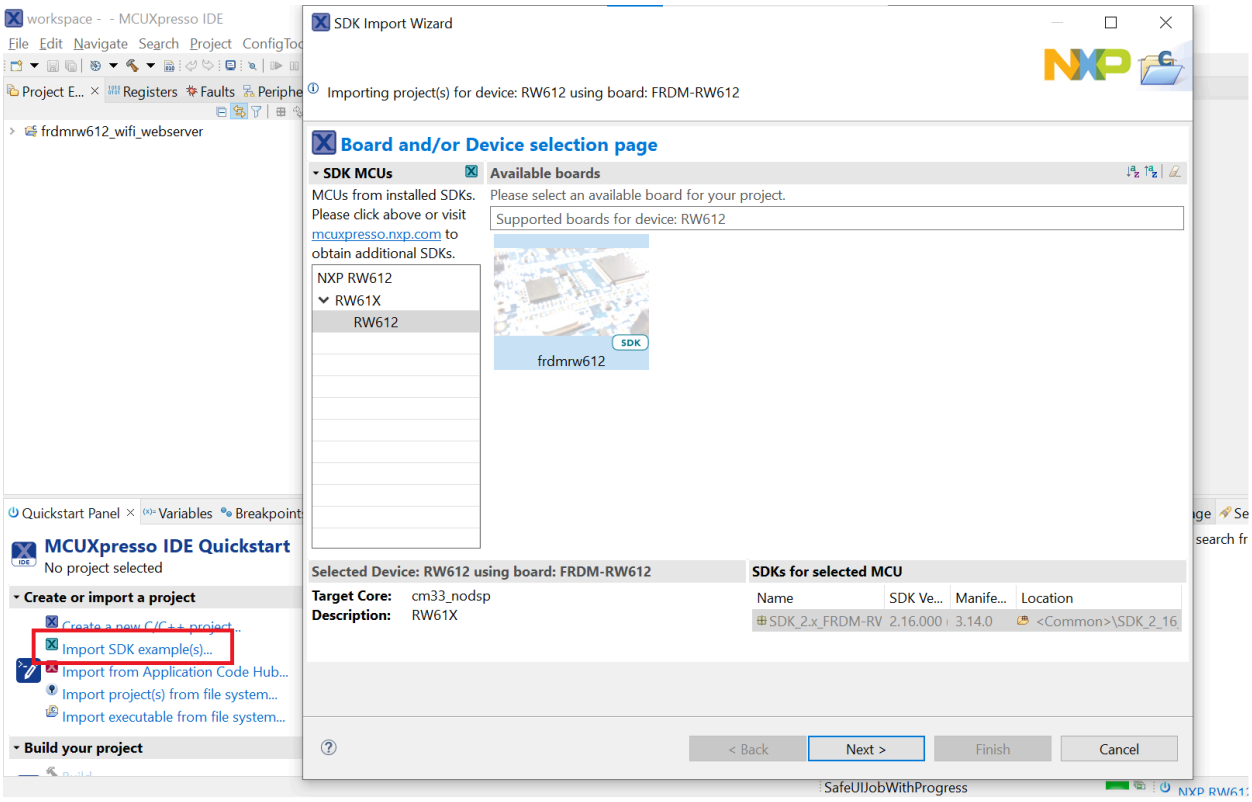

Figure 4. Import an example

Step 3 - Select the board

- Select the board (Figure 5)

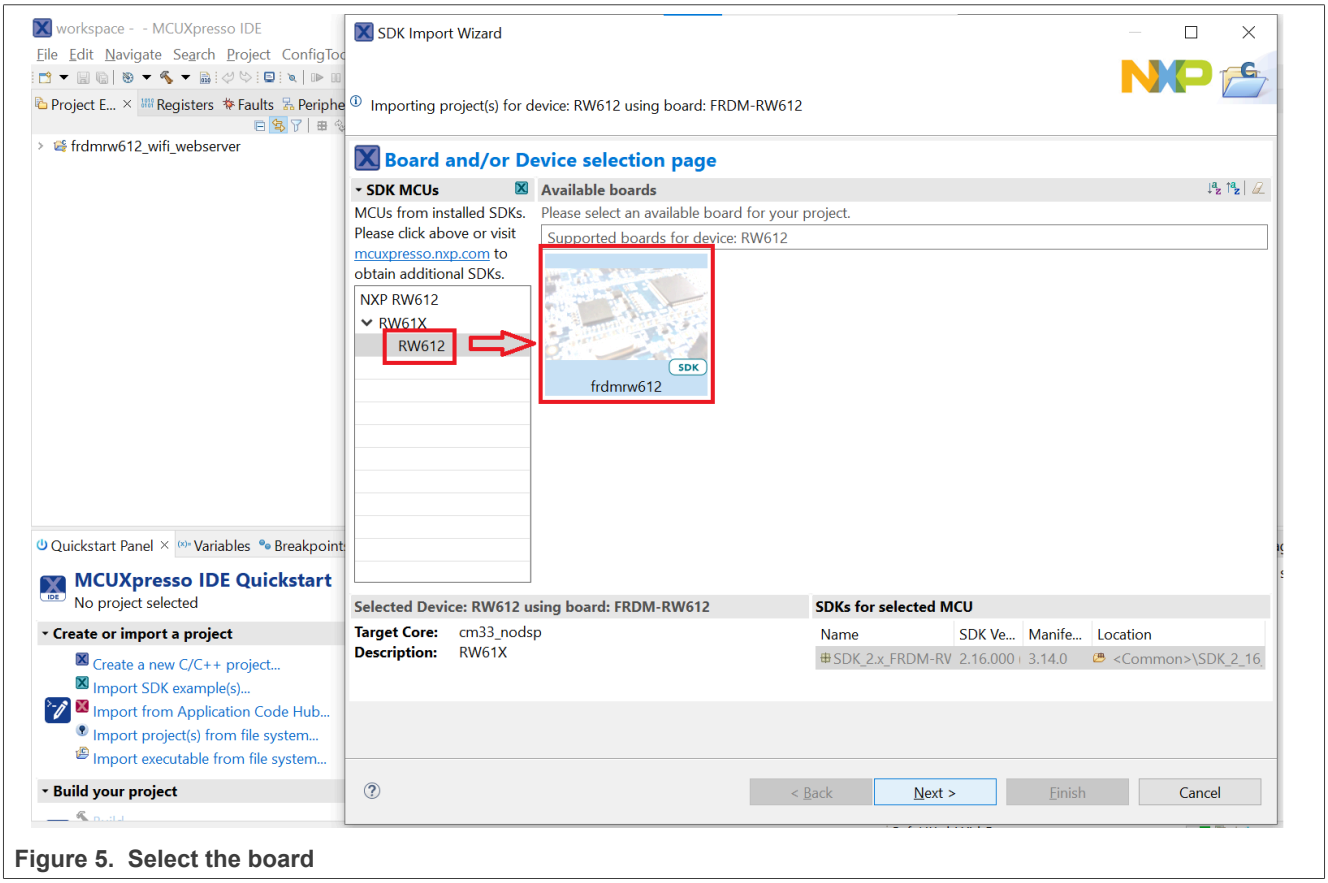

Figure 5. Select the board

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 6)

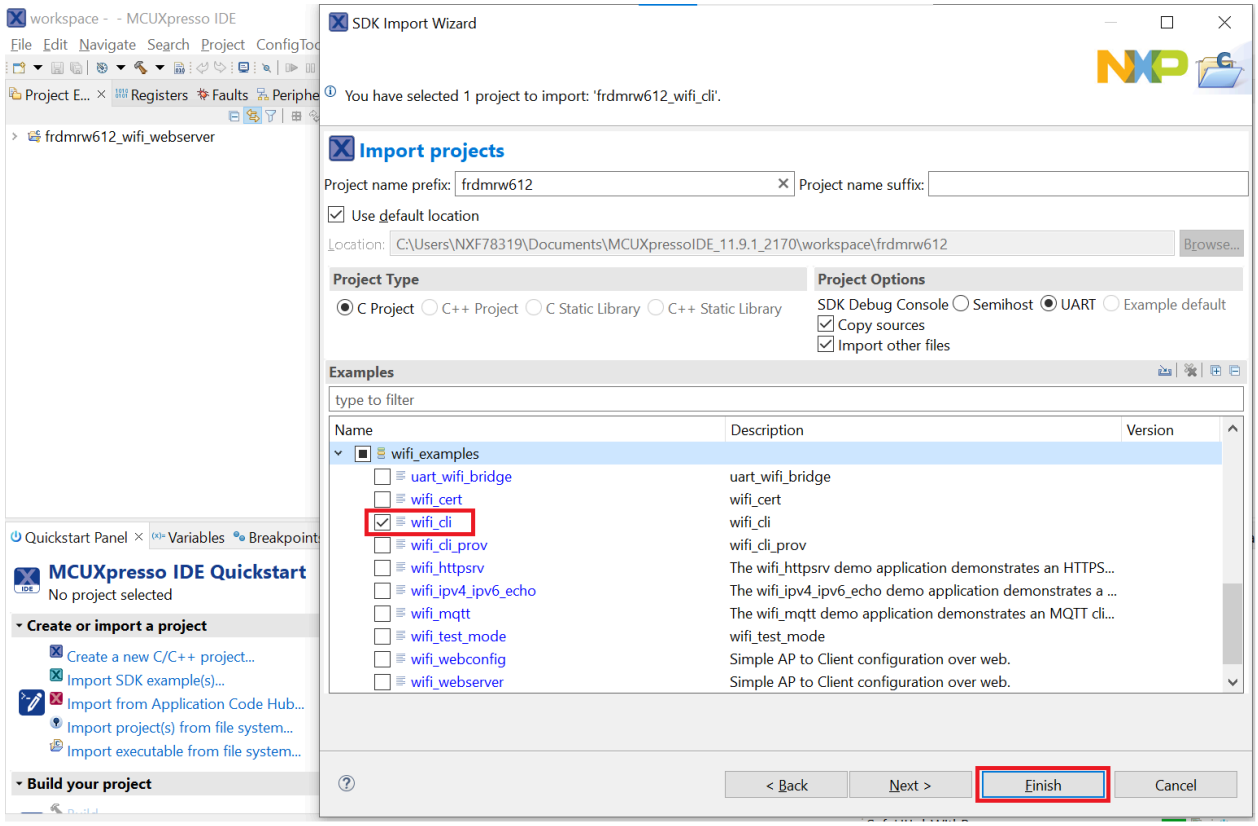

Figure 6. Import wifi_cli example

3.1.2.2 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) in the console window

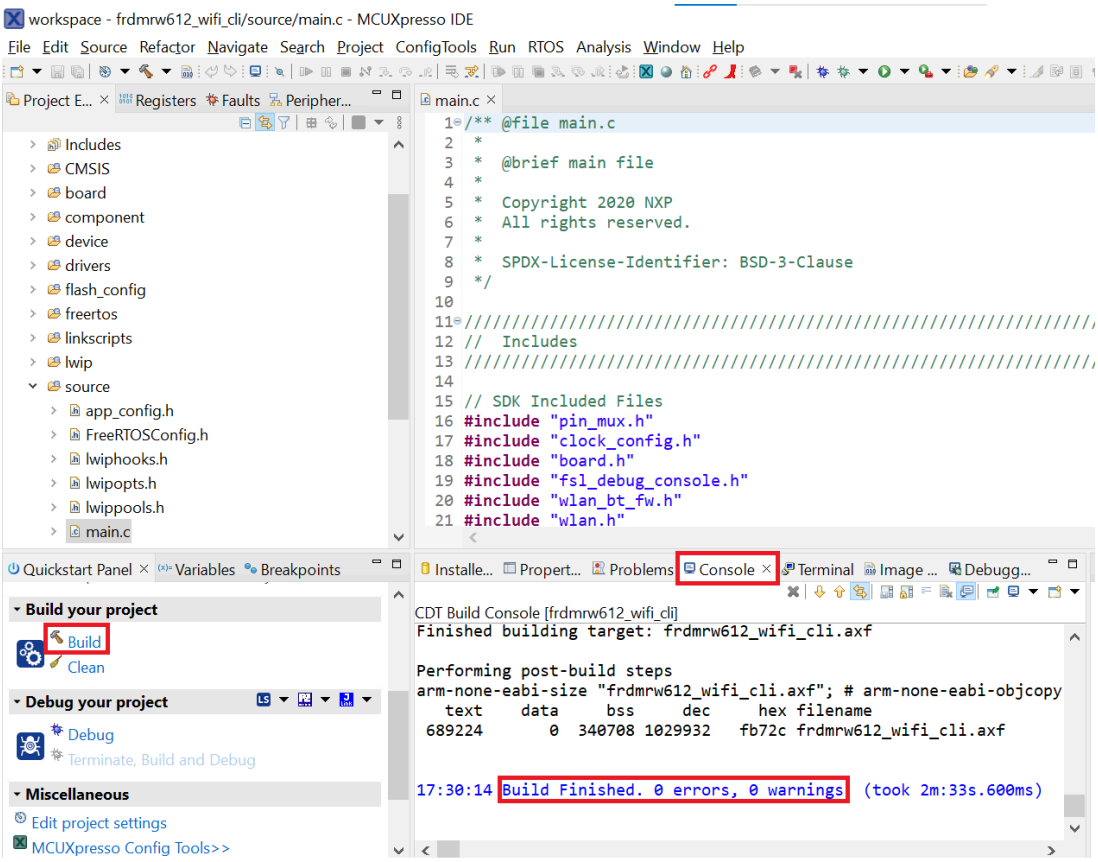

Figure 7. Select Build on Quickstart panel or in the toolbar

3.1.2.3 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 8)

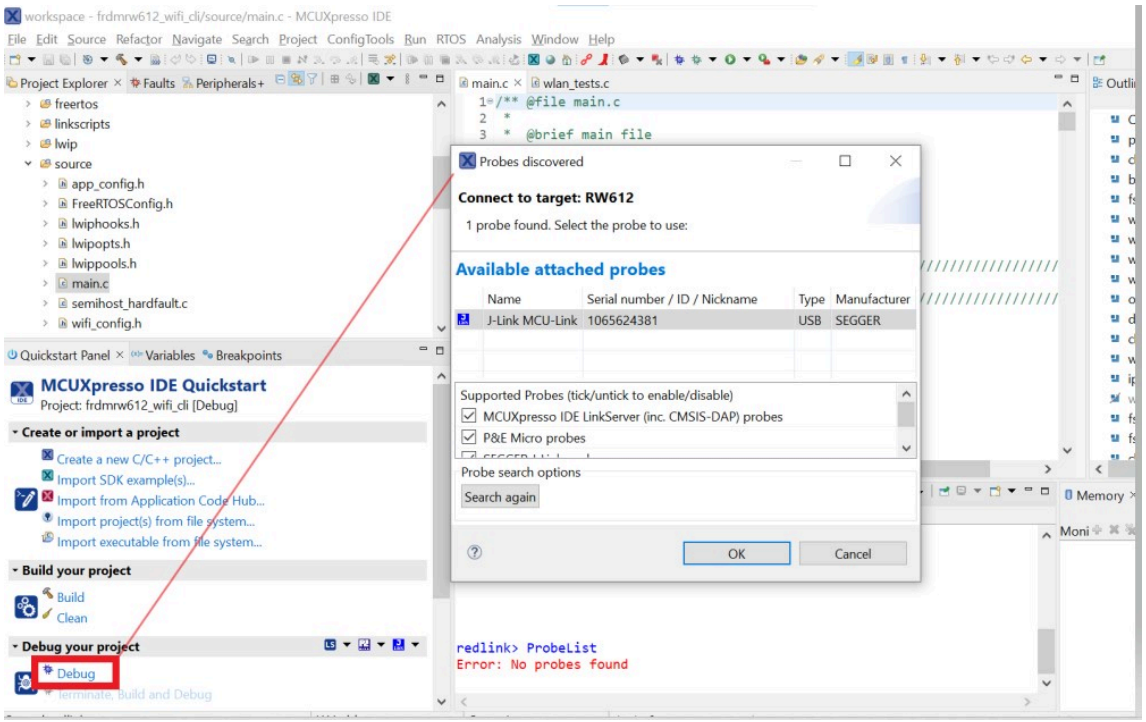

Figure 8. Select Debug mode and select the associated probe

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 9).

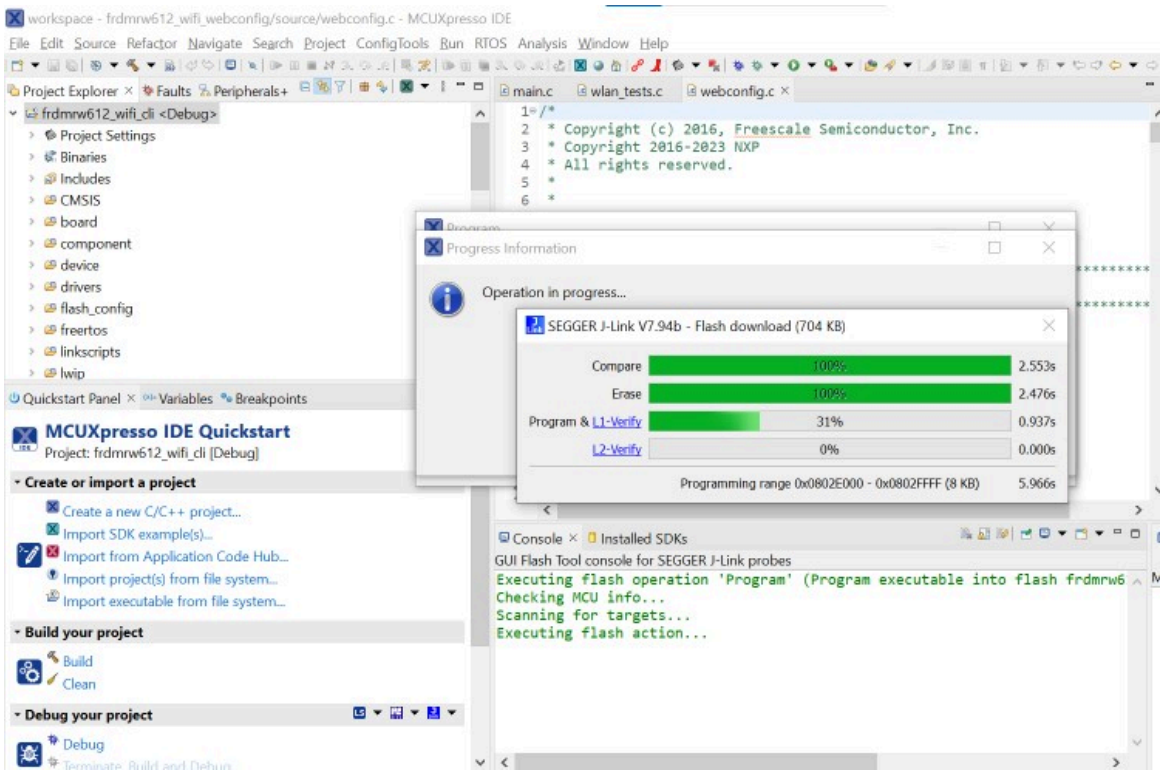

Figure 9. Application download start of program execution

- Click **Resume** to start the application
- To debug the application, use the **step into**, **step over** and **step return** buttons (Figure 10)
- To end the debugging session, use the **Terminate** button (Figure 10)

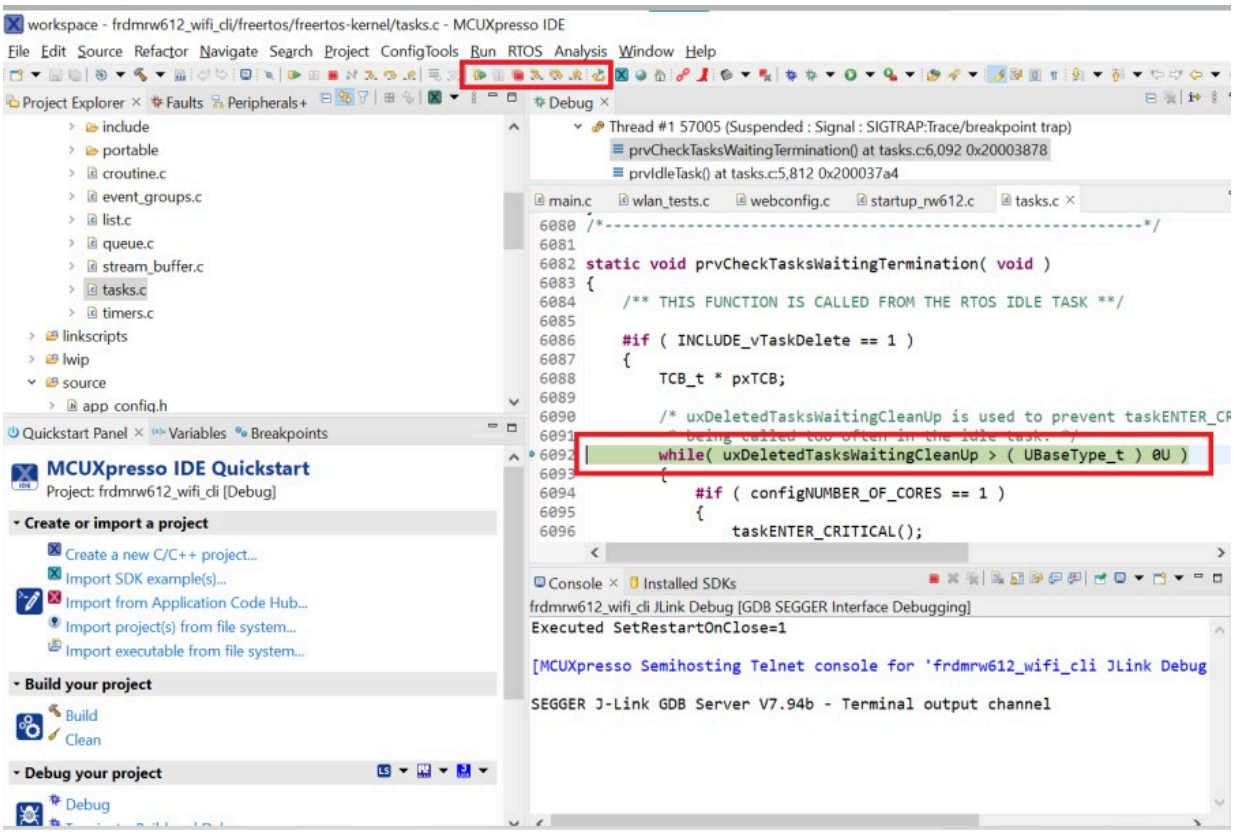

Figure 10. Run and debug the application

3.1.2.4 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 11)

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*.

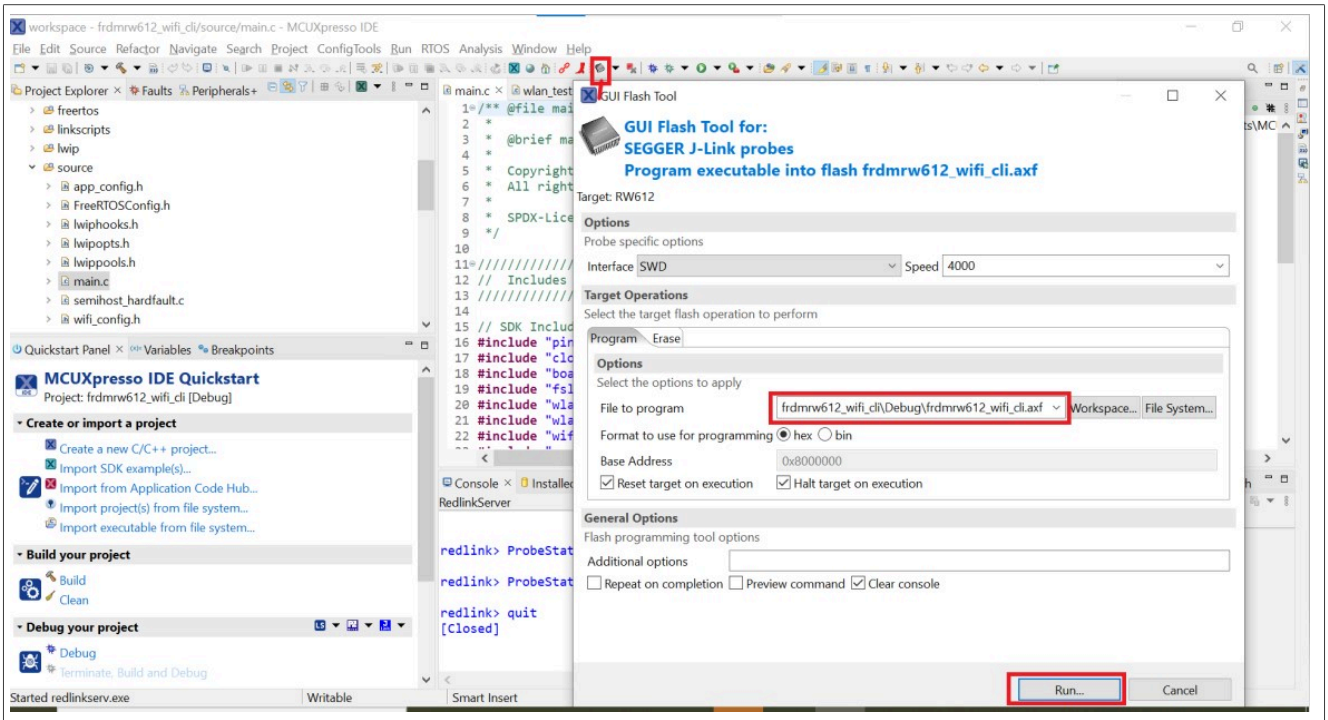

Figure 11. Using the GUI Flash tool for the pre-built binary or locally compiled binary

3.1.3 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.

3.1.3.1 Install Arm GCC toolchain

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

- Download the toolchain for the 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) using the Link for Linux system.
  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/
```

3.1.3.2 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.*

3.1.3.3 Flash the application program (no debugging)

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

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

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

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

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

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

- Reset the FRDM-RW61x 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 3.1.6 to view the output on the console once the application is executed.*

3.1.4 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.

3.1.4.1 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

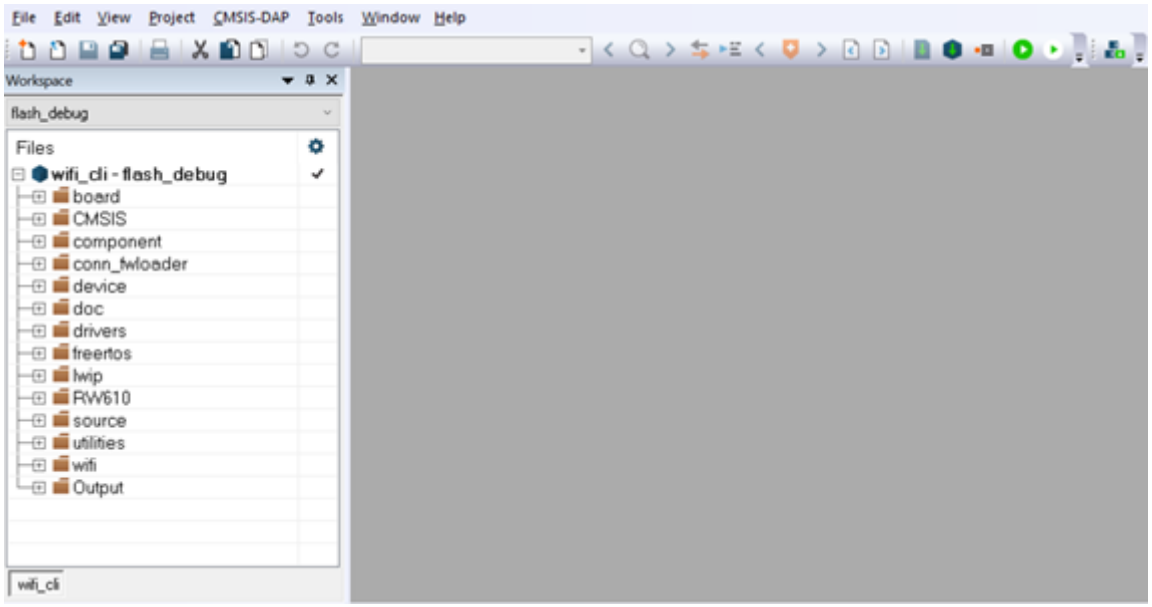

Figure 12. Open the project in IAR

3.1.4.2 Project settings

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

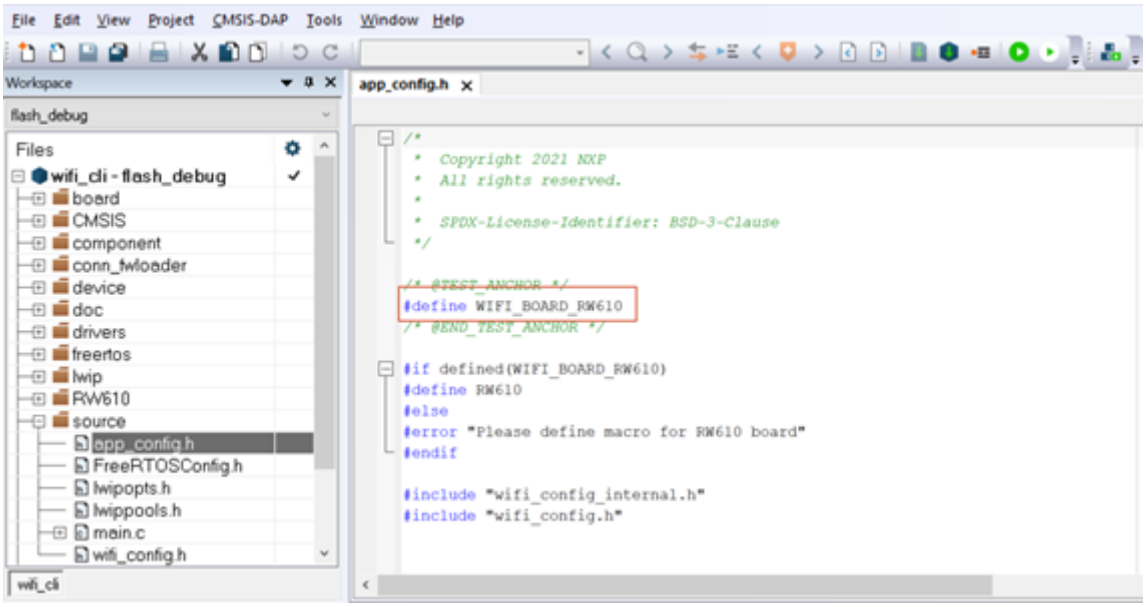

Figure 13. Wi-Fi board selection in IAR

3.1.4.3 Build the application

To build the *wifi_cli* application:

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

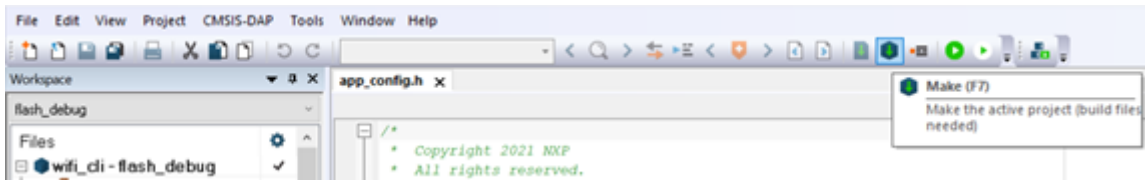

Figure 14. Application build in IAR

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

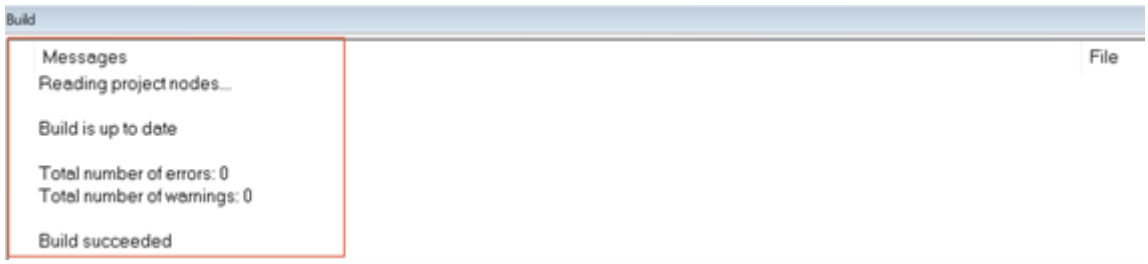

Figure 15. Build message in IAR

3.1.4.4 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.

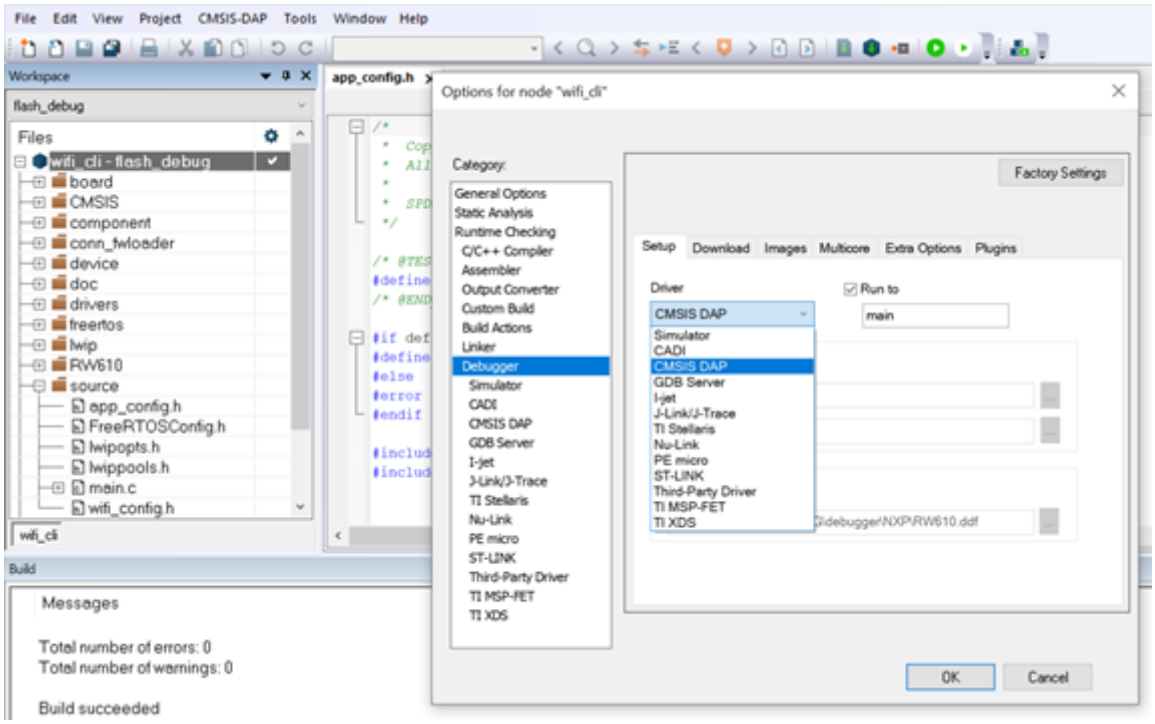

Figure 16. Debugger selection in IAR

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

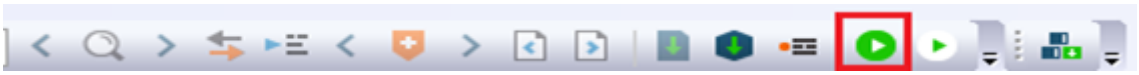

Figure 17. Initiate Debug in IAR

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.

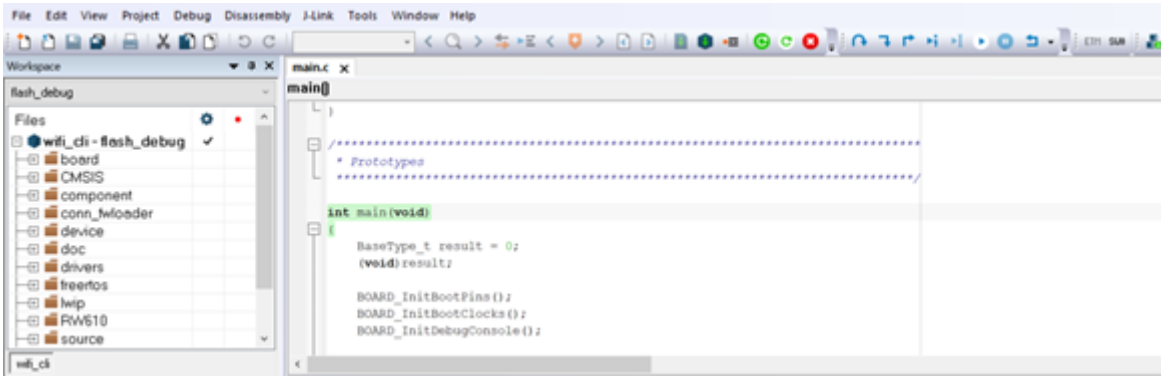

Figure 18. Application debugging in IAR

3.1.4.5 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.

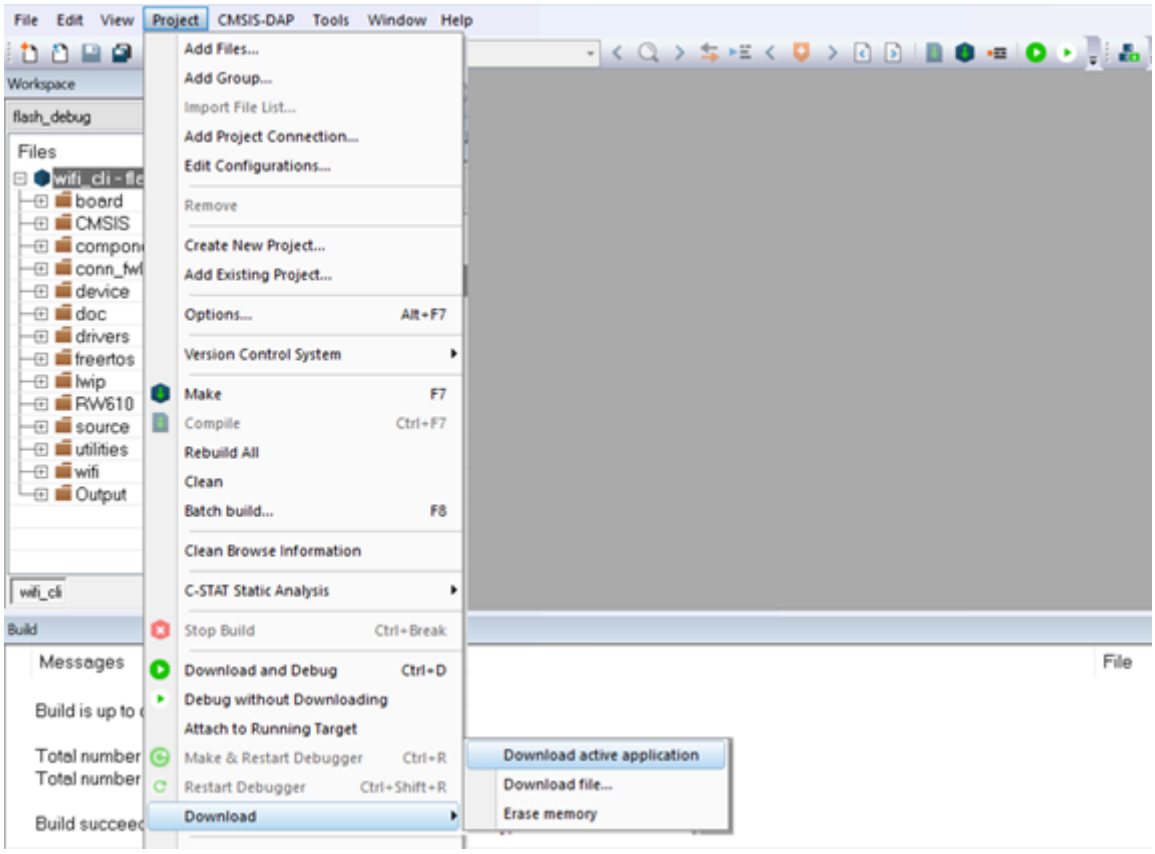

Figure 19. Binary flashing in IAR

Note: *Refer to Section 3.1.6 to view the output on the console once the application is executed.*

3.1.5 Run a demo using Keil MDK/µVision

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

3.1.5.1 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 the *RW612_DFP* file from NXP website
- Double click the downloaded file to install the RW612 software pack
- When the installation is complete, click 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.

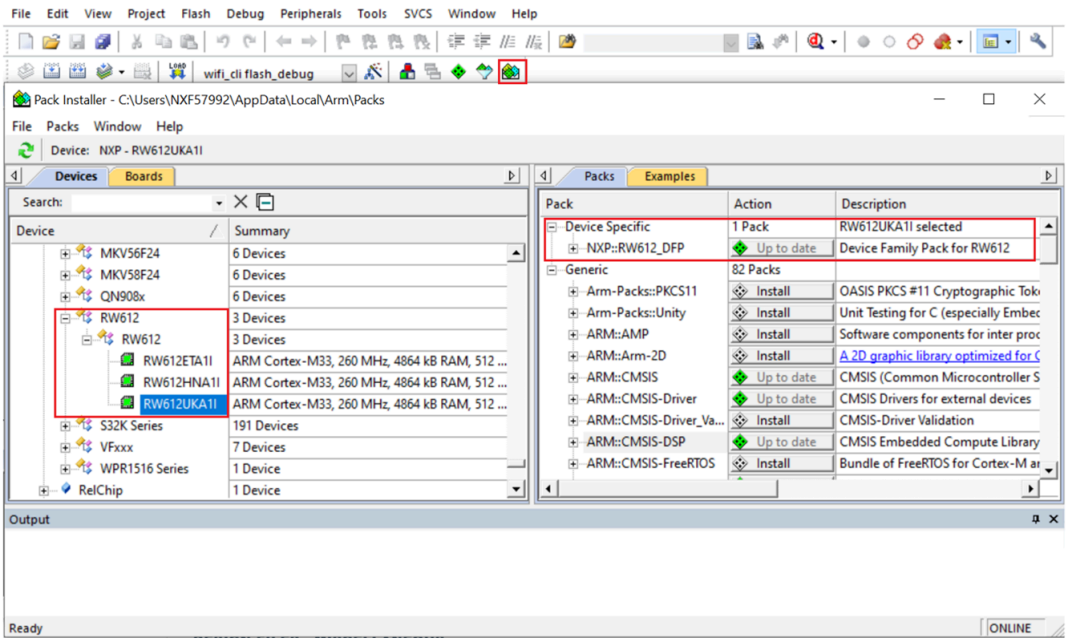

Figure 20. Pack Installer in Keil

3.1.5.2 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.*

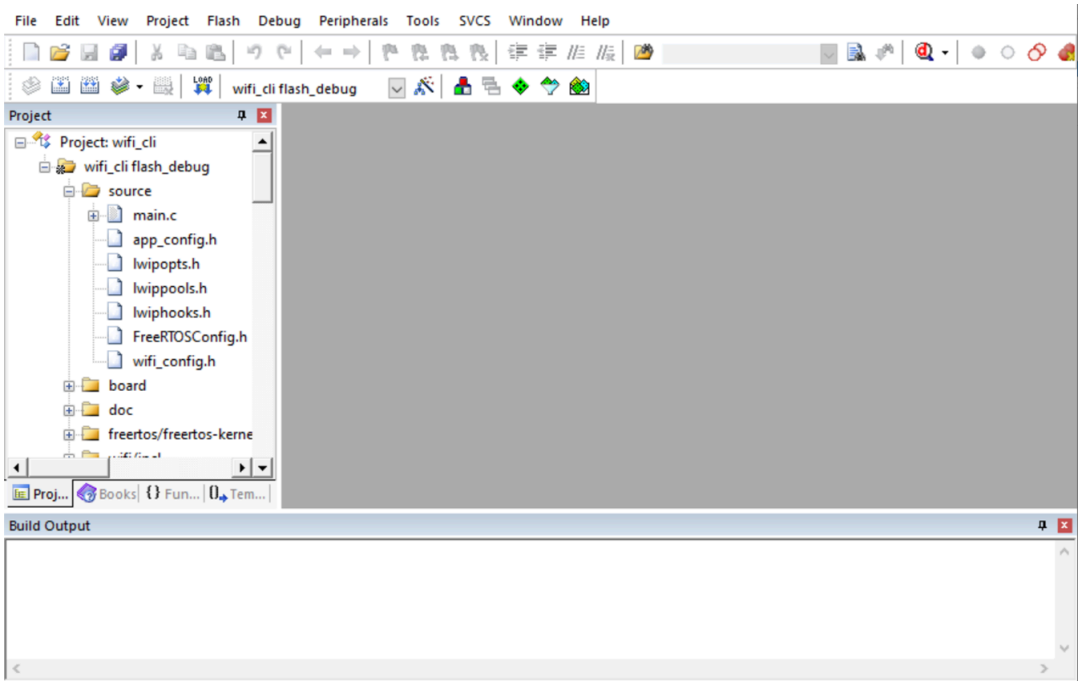

Figure 21. Open the project in Keil

3.1.5.3 Project settings

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

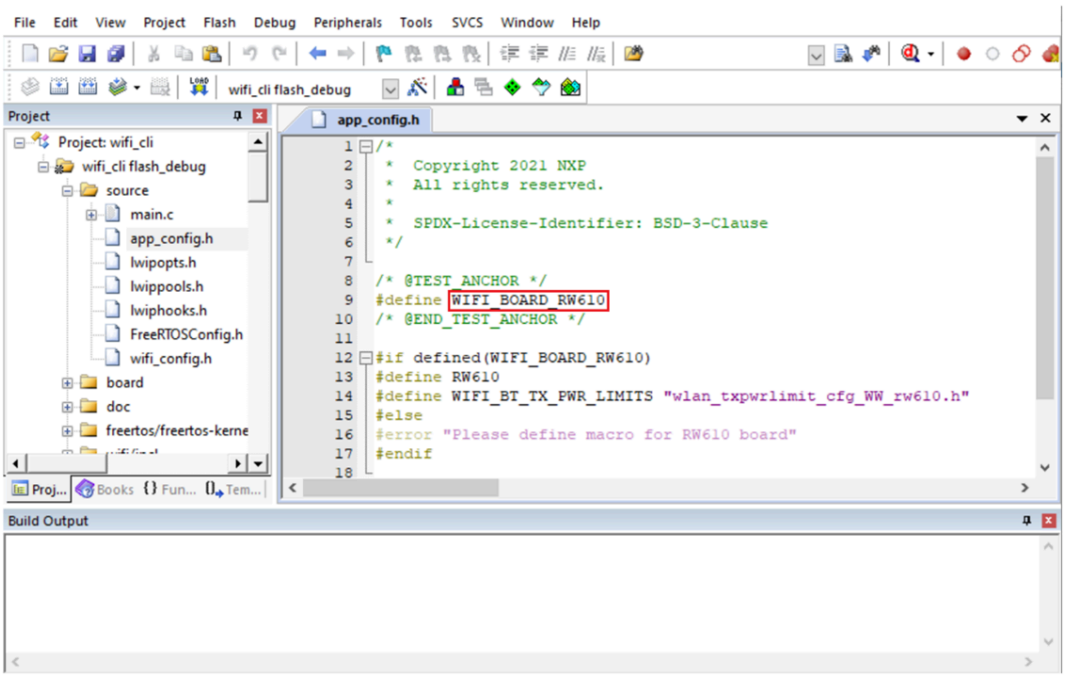

Figure 22. Wi-Fi board selection in Keil

3.1.5.4 Build the application

To build the application:

- Click the Build or Rebuild icons

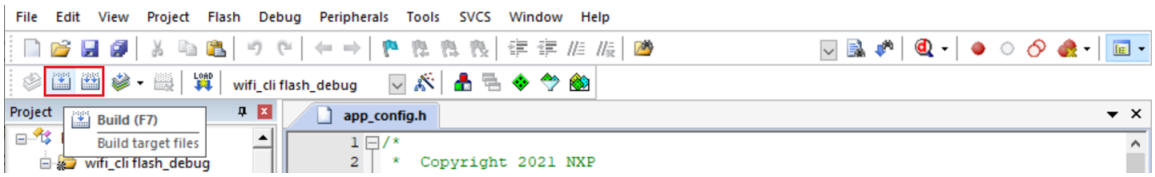

Figure 23. Build and Rebuild icons in Keil

- Verify the build progress in the Build Output window.

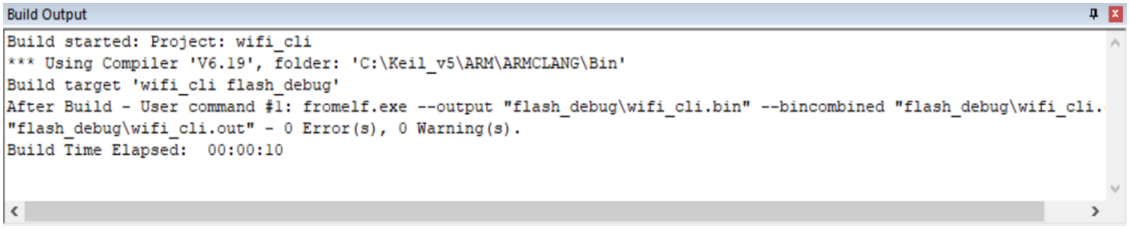

Figure 24. Build output window in Keil

3.1.5.5 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.

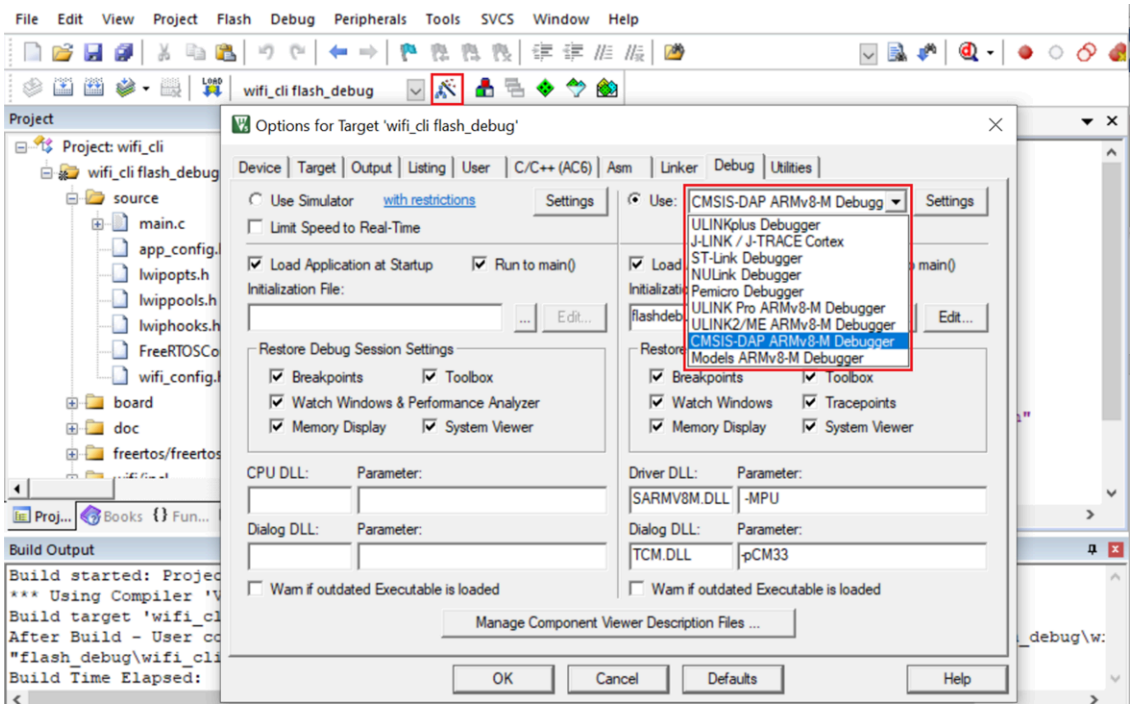

Figure 25. Select the debugger in Keil

To start the application debug:

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

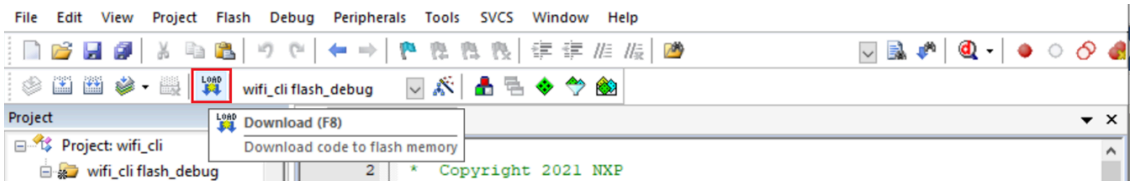

Figure 26. Load the application

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

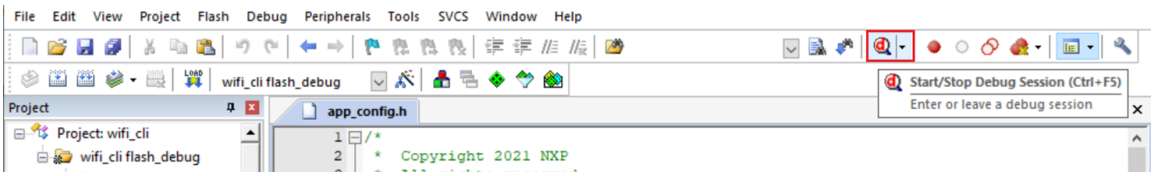

Figure 27. Start the debug session in Keil

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

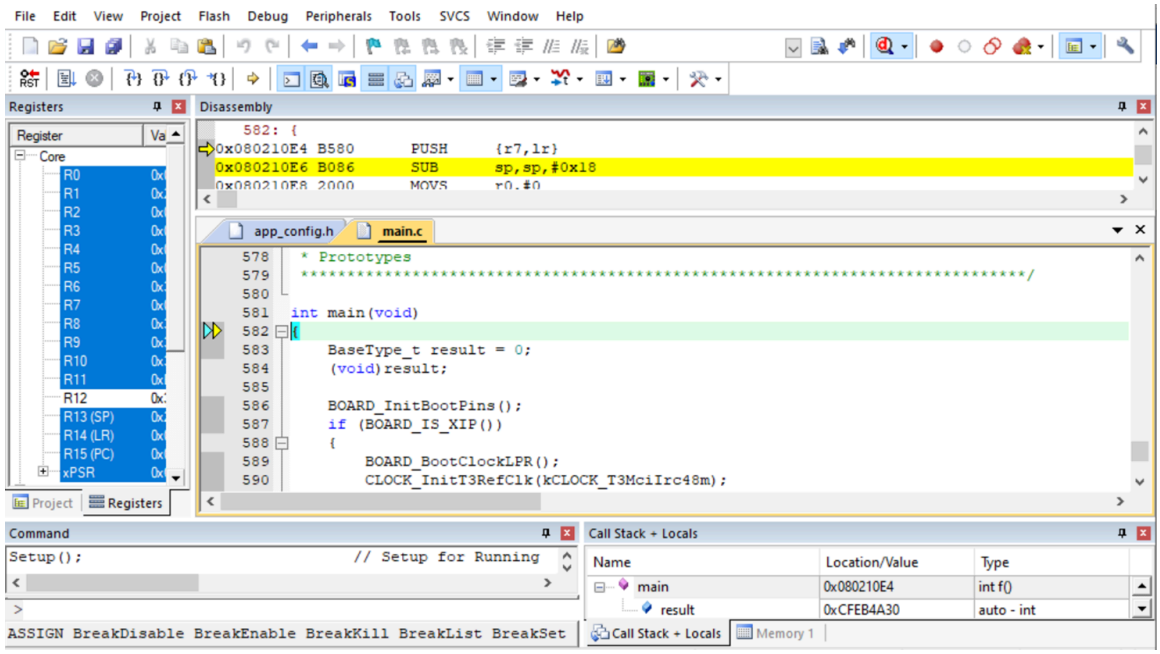

Figure 28. Set the program counter in Keil

- 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

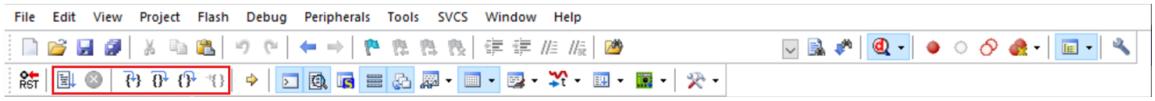

Figure 29. Application debugging features in Keil

3.1.5.6 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 3.1.6.1 to view the output on the console once the application is executed.

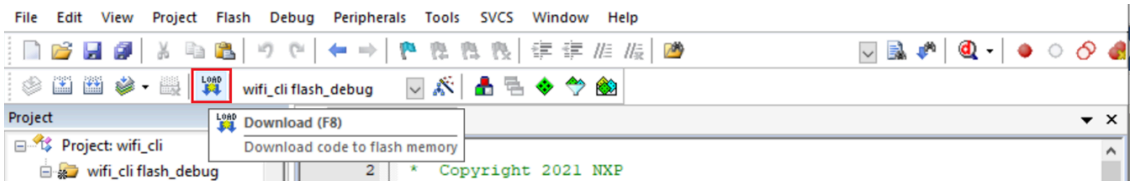

Figure 30. Flashing a binary file in Keil

3.1.6 wifi_cli application execution

3.1.6.1 Start-up logs

The following logs show on the console once the FRDM-RW61x 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
=========================================
CLI Build: Jun 18 2024 [19:28:12]
Copyright  2024  NXP
MCU Board: FRDM-RW61x
=========================================
MCU wakeup source 0x0...
Initialize WLAN Driver
=========================================
Wi-Fi cau temperature : 28
MAC Address: C0:95:DA:01:26:62
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-net-stats
wlan-set-mac <MAC_Address>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
…
…
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
=========================================
```

3.1.6.2 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-net-stats
wlan-set-mac <MAC_Address>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
…
…
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
```

3.1.6.3 Scan command

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

```
# wlan-scan
Scan scheduled...
# 1 network found:
5C:DF:89:0F:32:78  "NXP_V10" Infra
        mode: 802.11N
        channel: 1
        rssi: -43 dBm
        security: WPA2/WPA3 SAE Mixed
        WMM: YES
        802.11V: YES
        802.11W: Capable

# wlan-scan-opt ssid ASUS_2G
Scan for ssid "ASUS_2G" scheduled...

# 1 network found:
7C:10:C9:02:DA:48  "ASUS_2G" Infra
        mode: 802.11AX
        channel: 11
        rssi: -32 dBm
        security: WPA2
        WMM: YES
        802.11V: YES
        802.11W: NA
```

3.1.6.4 Add a network profile

Before adding a network profile for Soft AP mode or Station mode, 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: *Setting the channel value greater than or equal to 36 is mandatory, if UAP bandwidth is set to 80 MHz.*

```
[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
```

3.1.6.5 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:

Example of a command to add network profile in station mode with WPA2 security:

```
# wlan-add abc ssid Wifi 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: authenticated to network
app_cb: WLAN: connected to network
Connected to following BSS:

SSID = [Wifi]
IPv4 Address: [192.168.157.183]
```

Note: *Once connected to the AP, the console output shows that the Client is connected to the AP with* `ssid = [Wifi]` *and* `IP address = [192.168.157.183]` *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 abc ssid Wifi wpa3 sae 1234567890 mfpc 1 mfpr 1
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: authenticated to network
app_cb: WLAN: connected to network
Connected to following BSS:

SSID = [Wifi]
IPv4 Address: [192.168.157.183]
```

Note: *Once connected to the AP, the console output shows that the Client is connected to AP with* `ssid = [Wifi]` *and the* `IP address = [192.168.157.183]` *from the AP. For WPA3 R3, this configuration also works.*

3.1.6.6 WPA2/WPA3 station disconnection (from AP)

Disconnect from the AP network profile:

```
# wlan-disconnect
app_cb: disconnected
```

Remove the saved network profile:

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

3.1.6.7 Start soft 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 NXP__AP 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
Added "xyz"
```

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: UAP Started
========================================
Soft 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.

```
app_cb: WLAN: UAP a Client Associated
========================================
Client => 06:2A:94:36:5A:21 Associated with Soft AP
========================================
app_cb: WLAN: UAP a Client Connected
========================================
Client => 06:2A:94:36:5A:21 Connected with Soft AP
========================================
```

Get the associated clients list:

```
# wlan-get-uap-sta-list
Number of STA = 1
STA 1 information:
====================
MAC Address: F2:13:61:31:15:5A
Power mfg status: power save
Rssi : -25 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    F2:13:61:31:15:5A
```

3.1.6.8 Stop soft AP

```
# wlan-stop-network
# Pls set hidden_ssid before start uAP.
========================================
app_cb: WLAN: UAP Stopped
========================================
Soft AP stopped successfully
========================================
DHCP Server stopped successfully
========================================
```

3.1.6.9 STA filter for soft AP

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

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

- Enable soft-AP 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
```

3.1.6.10 iPerf server/client

The sample application implements the protocol used by iPerf performance measurement tool. The performance is measured between the FRDM-RW61x board and a computer running the iPerf tool. The instructions in this guide use the FRDM-RW61x board. Figure 31 and Figure 32 show the setup overview to run the iPerf performance test.

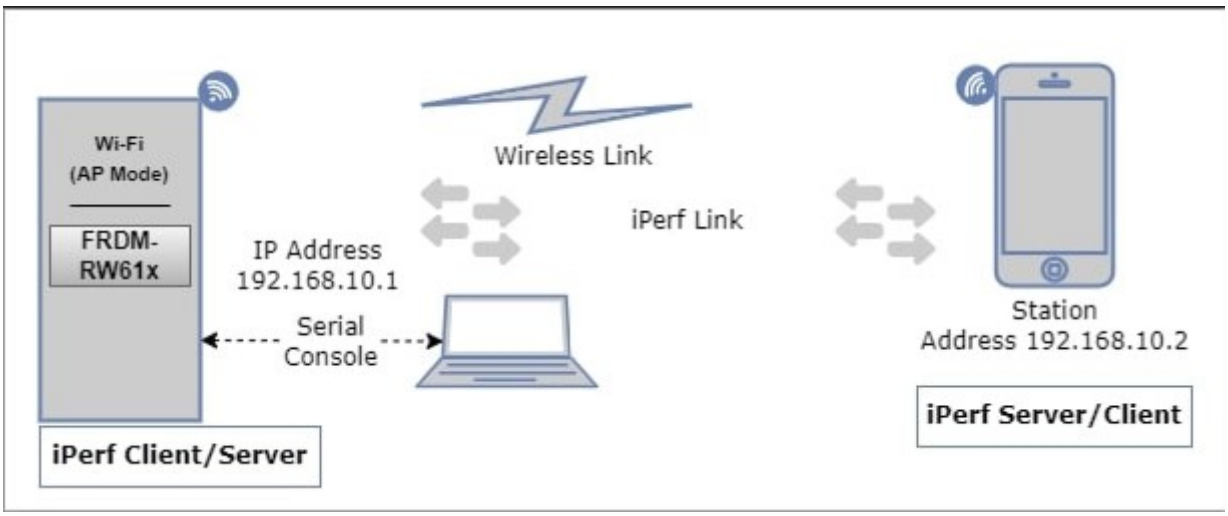

Figure 31. Hardware setup for iPerf performance test with soft AP mode

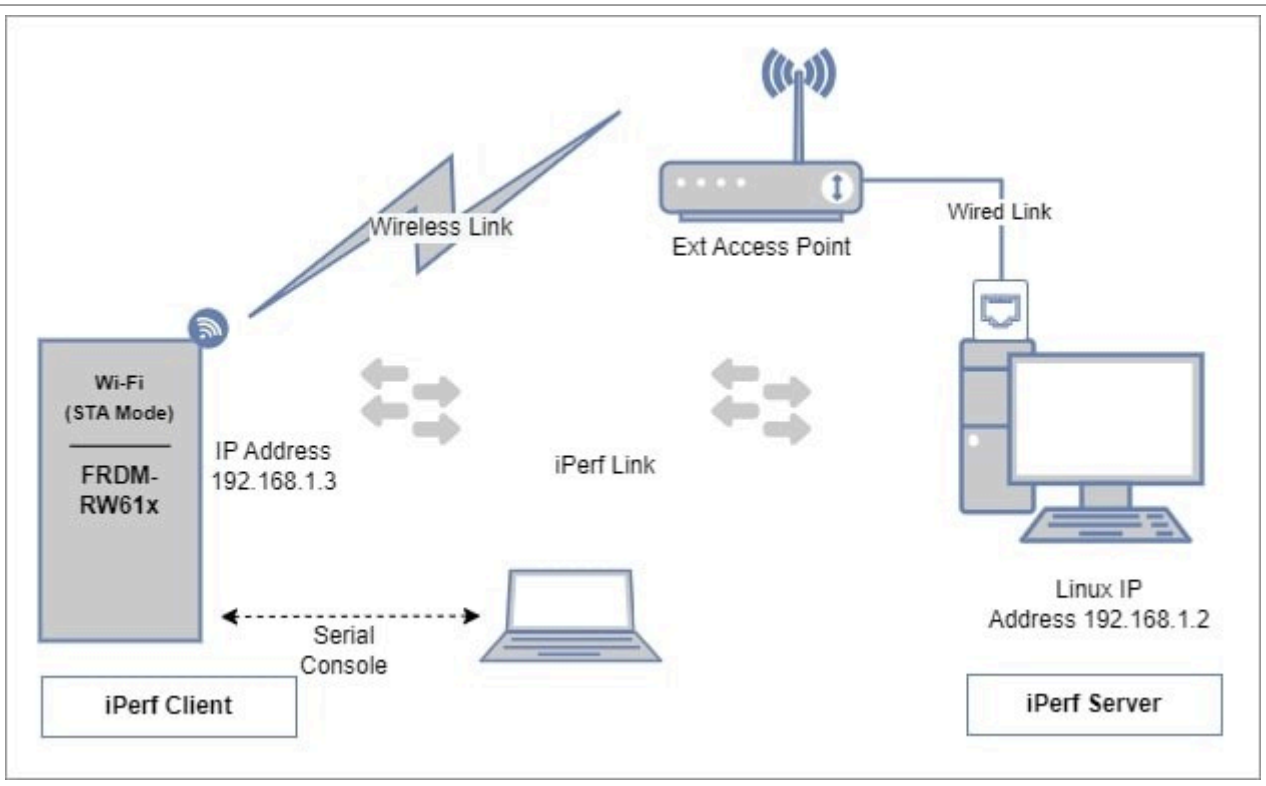

Figure 32. Hardware Setup for iPerf performance test with Station Mode

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)
           -V              Set the domain to IPv6 (send packets over IPv6)
           -a              abort ongoing iperf session
           -p              server port to listen on/connect to
           -r              Do a bidirectional UDP test individually

        Server specific:
           -s              run in server mode. Support 8 parallel traffic(-P) maximum from
 client side
           -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              reverse the test (client receives, server sends)
           -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))
           -l              length of buffer in bytes to write (Defaults: v4 TCP=1460, v6
 TCP=1440, v4 UDP=1470, v6 UDP=1450)
           Note: Limit length is smaller than default size.
```

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 the iPerf server:

```
# iperf -s
IPERF initialization successful
# New TCP client (settings flags 0x40010078)
------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 34656
 Bytes Transferred XXXXX
 Duration (ms) 10151
 Bandwidth (Mbitpsec) XX
```

Start the iPerf Client (TX Only):

```
# iperf -c 192.168.10.2
IPERF initialization successful
#
------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 54237
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXXX
 Duration (ms) 10001
 Bandwidth (Mbitpsec) XX
```

Start the iPerf Client (TX and RX simultaneous):

```
# iperf -c 192.168.10.2 -d
IPERF initialization successful
------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 54239
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXXX
 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 46370
 Bytes Transferred XXXXX
 Duration (ms) 10170
 Bandwidth (Mbitpsec) XX
```

Start the iPerf Client (TX and RX individual):

```
# iperf -r -c 192.168.10.2
IPERF initialization successful
------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 54240
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXXX
 Duration (ms) 10002
 Bandwidth (Mbitpsec) XX
------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 50006
 Bytes Transferred XXXXX
 Duration (ms) 10138
 Bandwidth (Mbitpsec) X
```

iPerf UDP

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

- Start iPerf server

```
nxp@thinkpad:.../iperf-2.1.9$ iperf -s -u -B 192.168.10.2
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 192.168.10.2 port 5001 connected with 192.168.10.1 port 49155
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.00-9.98 sec  XX.X MBytes  XX.X Mbits/sec   0.XXX ms X/XXXXX (0%)
```

- Start the iPerf Client (TX only)

```
# 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 (0x88000000).
Jitter X.XXX, Lost X/XXXX datagrams, OoO 0
------------------------------------------------
 UDP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 49155
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXXX
 Duration (ms) 10024
 Bandwidth (Mbitpsec) XX.XX
```

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

3.1.6.11 Wi-Fi power save

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

IEEE power save

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

- IEEE PS usage

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

- Enable IEEEPS

```
Enable IEEE PS
# wlan-ieee-ps 1
```

- 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
period = 20
# 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
```

3.1.6.12 Host sleep

The following command is used to configure host sleep parameters and put the 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_timer> 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
Auto 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 3.1.6.13).*

- Host sleep using power manager

The 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 PM3
```

The 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 PM3
Exit low power mode
Woken up by RTC
# Enter low power mode PM3
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 the Wi-Fi wake-up source is* `wlan-wakeup-condition wowlan 0x0`

3.1.6.13 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.*

3.1.6.14 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 <mef/wowlan wake_up_conds>
    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
```

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 3.1.6.15) to configure MEF entries for MEF host wake-up. If the MEF entry is not configured, the driver uses the default MEF entry as a MEF wake-up condition that is a broadcast or unicast ARP packet.*

3.1.6.15 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
```

3.1.6.16 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
Wi-Fi cau temperature : 28
MAC Address: C0:95:DA:01:26:62
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
…

# wlan-reset 2
--- Disable WiFi ---
--- Enable WiFi ---
Initialize WLAN Driver
Wi-Fi cau temperature : 27
MAC Address: C0:95:DA:01:26:62
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
```

```
...
```

3.1.6.17 802.11k 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 an STA connection

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

3.1.6.18 802.11d 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
```

3.1.6.19 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_threshold>
Example:
    wlan-roaming 1 40
# wlan-roaming 1 70
```

If the current BSS RSSI is lower than the preconfigured threshold, FRDM-RW61x STA switches to another BSS with better RSSI.

- Disable roaming

```
# wlan-roaming 0
```

3.1.6.20 CSI commands

The following commands are used to configure CSI.

- Configure CSI parameters

```
# wlan-set-csi-param-header
Usage: wlan-set-csi-param-header <sta/uap> <csi_enable> <head_id> <tail_id> <chip_id> <band_config>
 <channel> <csi_monitor_enable> <ra4us>

[csi_enable] :1/2 to Enable/Disable CSI
[head_id, head_id, chip_id] are used to seperate CSI event records received from FW
[Bandcfg] defined as below:
    Band Info - (00)=2.4GHz, (01)=5GHz
    t_u8  chanBand    : 2;
    Channel Width - (00)=20MHz, (10)=40MHz, (11)=80MHz
    t_u8  chanWidth   : 2;
    Secondary Channel Offset - (00)=None, (01)=Above, (11)=Below
    t_u8  chan2Offset : 2;
    Channel Selection Mode - (00)=manual, (01)=ACS, (02)=Adoption mode
    t_u8  scanMode    : 2;
[channel] : monitor channel number

[csi_monitor_enable] : 1-csi_monitor enable, 0-MAC filter enable

[ra4us] : 1/0 to Enable/DisEnable CSI data received in cfg channel with mac addr filter, not only RA
 is us or other

Usage example :
wlan-set-csi-param-header sta 1 66051 66051 170 0 11 1 1
The current csi_param is:
bss_type      : sta
csi_enable    : 0
head_id       : 0
tail_id       : 0
csi_filter_cnt: 0
chip_id       : 0
band_config   : 0
channel       : 0
csi_monitor_enable : 0
ra4us         : 0
```

- Configure CSI filter

```
# wlan-set-csi-filter
Error: invalid number of arguments
Usage : wlan-set-csi-filter <opt> <macaddr> <pkt_type> <type> <flag>
opt   : add/delete/clear/dump
add   : All options need to be filled in
delete: Delete recent filter information
clear : Clear all filter information
dump  : Dump csi cfg information

Usage example :
wlan-set-csi-filter add 00:18:E7:ED:2D:C1 255 255 0
wlan-set-csi-filter delete
wlan-set-csi-filter clear
wlan-set-csi-filter dump
```

- Apply CSI configuration

```
# wlan-csi-cfg
```

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

```
csi_data_recv_user (void* buffer, t_u16 data_len)
```

```
{
pcsi_record_ds data = pcsi_record_ds(buffer);
(void)PRINTF("Len :%d \r\n", data ->Len);
}
register_csi_user_callback(csi_data_recv_user)
```

3.1.6.21 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            : 0
monitor_activity  : 0
filter_flags      : 0
radio_type        : 0
chan_number       : 0
filter_num        : 0
```

- 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)
```

3.1.6.22 ECSA command

The following command is used to configure Soft 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. when 0 is used, only CSA
 IE will be used
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 : wlan-set-ecsa-cfg 1 0 36 10 1

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

3.1.6.23 EU crypto commands

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

- Command usage

```
# wlan-eu-crypto-rc4
Usage:
Algorithm RC4 encryption and decryption verification
wlan-eu-crypto-rc4 <EncDec>
EncDec: 0-Decrypt, 1-Encrypt
```

- Encrypt sample data

```
# wlan-eu-crypto-rc4 1
Raw Data:
**** Dump @ 2005776C Len: 16 ****
12 34 56 78 90 12 34 56 78 90 12 34 56 78 90 12
******** End Dump *******
Encrypted Data:
**** Dump @ 20057790 Len: 16 ****
d9 90 42 ad 51 ab 11 3f 24 46 69 e6 f1 ac 49 f5
******** End Dump *******
```

- Decrypt sample data

```
# wlan-eu-crypto-rc4 0
Raw Data:
**** Dump @ 2005775C Len: 16 ****
d9 90 42 ad 51 ab 11 3f 24 46 69 e6 f1 ac 49 f5
******** End Dump *******
Decrypted Data:
**** Dump @ 20057790 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 _rc4

.

3.1.6.24 Get the antenna configuration

Get the antenna configuration:

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

3.1.6.25 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: Wifi
        BSSID: FA:37:9B:63:18:D8
        mode: 802.11AC
        channel: 6
        role: Infra
        RSSI: -30dBm
        security: WPA2

        IPv4 Address
        address: DHCP
                IP:             192.168.157.183
                gateway:        192.168.157.231
                netmask:        255.255.255.0
                dns1:           192.168.157.231
                dns2:           0.0.0.0
        IPv6 Addresses
        Link-Local   :  FE80::C295:DAFF:FE01:2562 (Preferred)
        Global       :  2409:40C1:4036:BA6D:C295:DAFF:FE01:2562 (Preferred)
        rssi threshold: 0

uAP started as:
"xyz"
        SSID: NXPAP
        BSSID: C0:95:DA:01:26:62
        mode: 802.11AX
        channel: 6
        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:           192.168.10.1
                dns2:           0.0.0.0
        IPv6 Addresses
        Link-Local   :  FE80::C295:DAFF:FE01:2662 (Tentative)
        rssi threshold: 0
```

- Get the Wi-Fi driver and firmware version

```
# wlan-version
WLAN Driver Version   : v1.3.r48.p12
WLAN Firmware Version : rw610w-V2, IMU, FP99, 18.99.6.p10, PVE_FIX 1
```

- Set the Wi-Fi MAC address

```
# wlan-set-mac C0:95:DA:00:D5:0F
```

- 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:D6:0F
```

- Get the list of Wi-Fi networks

```
# wlan-list
2 networks:
"xyz"
        SSID: NXPAP
        BSSID: 00:00:00:00:00:00
        mode: 802.11AX
        channel: 6
        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:           192.168.10.1
                dns2:           0.0.0.0
        IPv6 Addresses
        Link-Local   :  FE80::C295:DAFF:FE00:D60F (Tentative)
        rssi threshold: 0
"abc"
        SSID: Wifi
        BSSID: 00:00:00:00:00:00
        mode: 802.11AC
        channel: (Auto)
        role: Infra
        RSSI: 0dBm
        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
        IPv6 Addresses
        Link-Local   :  FE80::C295:DAFF:FE00:D50F (Preferred)
        Global       :  2409:40C1:4036:BA6D:C295:DAFF:FE00:D50F (Tentative)

        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:             192.168.157.19
                gateway:        192.168.157.231
                netmask:        255.255.255.0
                dns1:           192.168.157.231
                dns2:           0.0.0.0
```

```
        IPv6 Addresses
        Link-Local    :  FE80::C295:DAFF:FE00:D50F (Preferred)
        Global        :  2409:40C1:4036:BA6D:C295:DAFF:FE00:D50F (Preferred)
```

- Get the Soft AP channel

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

- Set max station count for Soft 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>] <ipv4/ipv6 address>
Default values:
        packet_size: 56
        packet_count: 10
        timeout: 2 sec

# ping -s 56 -c 2 -W 2 192.168.157.231
PING 192.168.157.231 (192.168.157.231) 56(84) bytes of data
64 bytes from 192.168.157.231: icmp_req=1 ttl=64 time=6 ms
64 bytes from 192.168.157.231: icmp_req=2 ttl=64 time=12 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 SSID 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>
Options:
    <0/1>: Disable/enable Tx Pert tracking.
    <STA/UAP>: User needs to indicate which interface this tracking for.
    <p>: Tx Pert check period. Unit is second.
    <r>: Tx Pert ratio threshold (unit 10 percent). (Fail TX packet)/(Total TX packets).
 The default value is 5.
    <n>: A watermark of check number (default 5). Fw will start tracking Tx Pert after
 sending n packets.
```

```
Example:
    wlan-tx-pert 1 UAP 5 3 5
Note:
    Please verify by iperf or ping
    When the traffic quality is good enough, it will not be triggered
```

- Get Wi-Fi STA and Soft AP log

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

# wlan-get-log sta
dot11GroupTransmittedFrameCount    9
dot11FailedCount                   0
dot11RetryCount                    1
dot11MultipleRetryCount            1
dot11FrameDuplicateCount           0
dot11RTSSuccessCount               7
dot11RTSFailureCount               0
dot11ACKFailureCount               6
dot11ReceivedFragmentCount         45
dot11GroupReceivedFrameCount       7
dot11FCSErrorCount                 11294
dot11TransmittedFrameCount         56
wepicverrcnt-1                     65840
wepicverrcnt-2                     0
wepicverrcnt-3                     0
wepicverrcnt-4                     0
beaconReceivedCount                0
beaconMissedCount                  2016
dot11TransmittedFragmentCount      0
dot11QosTransmittedFragmentCount   0 0 0 0 46 0 0 0
dot11QosFailedCount                0 0 0 0 0 0 0 0
dot11QosRetryCount                 0 0 0 0 1 0 0 0
dot11QosMultipleRetryCount         0 0 0 0 1 0 0 0
dot11QosFrameDuplicateCount        0 0 0 0 0 0 0 0
dot11QosRTSSuccessCount            0 0 0 0 0 64 0 0
dot11QosRTSFailureCount            0 0 0 0 0 0 0 0
dot11QosACKFailureCount            0 0 0 0 6 3 0 0
dot11QosReceivedFragmentCount      0 0 0 35 0 10 0 0
dot11QosTransmittedFrameCount      0 0 0 0 46 0 0 0
dot11QosDiscardedFrameCount        0 0 0 0 0 3 0 0
dot11QosMPDUsReceivedCount         0 0 0 35 0 0 0 0
dot11QosRetriesReceivedCount       0 0 0 0 0 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              0
dot11ReceivedOctetsInAMSDUCount      6116033429504
dot11TransmittedAMPDUCount           0
dot11TransmittedMPDUsInAMPDUCount    420
dot11TransmittedOctetsInAMPDUCount   75784697938522
dot11AMPDUReceivedCount              0
dot11MPDUInReceivedAMPDUCount        1377
dot11ReceivedOctetsInAMPDUCount      0
dot11AMPDUDelimiterCRCErrorCount     0
```

- Get WMM TX statistics

```
# wlan-wmm-stat
[wifi] Warn: Dump priv[0] ac_queue[0]
[wifi] Warn:     [FA:XX:XX:XX:18:D8] drop_cnt[0] total_pkts[0]
[wifi] Warn:     [FF:XX:XX:XX:FF:FF] drop_cnt[0] total_pkts[0]
[wifi] Warn: Dump priv[0] ac_queue[1]
[wifi] Warn:     [FA:XX:XX:XX:18:D8] drop_cnt[0] total_pkts[0]
[wifi] Warn:     [FF:XX:XX:XX:FF:FF] drop_cnt[0] total_pkts[0]
[wifi] Warn: Dump priv[0] ac_queue[2]
[wifi] Warn:     [FA:XX:XX:XX:18:D8] drop_cnt[0] total_pkts[0]
[wifi] Warn:     [FF:XX:XX:XX:FF:FF] drop_cnt[0] total_pkts[0]
[wifi] Warn: Dump priv[0] ac_queue[3]
[wifi] Warn:     [FA:XX:XX:XX:18:D8] drop_cnt[0] total_pkts[0]
[wifi] Warn:     [FF:XX:XX:XX:FF:FF] drop_cnt[0] total_pkts[0]
[wifi] Warn: Dump priv[0] driver_error_cnt:
[wifi] Warn:     tx_no_media[0]
[wifi] Warn:     tx_err_mem[0]
[wifi] Warn:     tx_wmm_retried_drop[0]
[wifi] Warn:     tx_wmm_pause_drop[0]
[wifi] Warn:     tx_wmm_pause_replaced[0]
[wifi] Warn:     rx_reorder_drop[0]
[wifi] Warn: TX buffer pool: free_cnt[16] real_free_cnt[16]
```

3.1.7 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 */
}
}
```

3.2 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 the 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.

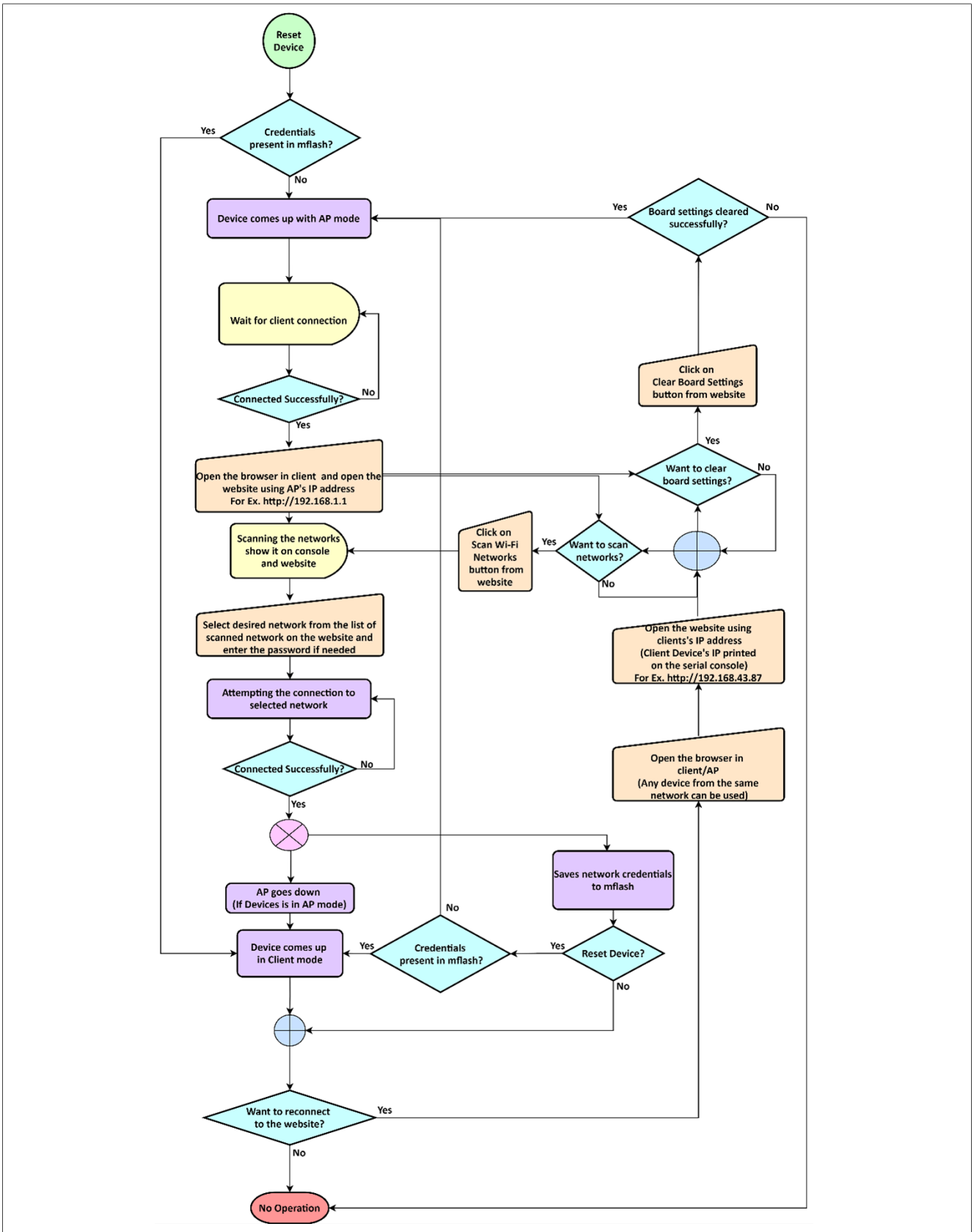

Figure 33. wifi_webconfig flow diagram

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

3.2.1 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 soft AP	WIFI_SSID	“nxp_configuration_ access_point”	webconfig.h	Default SSID and passphrase to start soft 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”		

3.2.2 wifi_webconfig application execution

Refer to Section 3.1.2 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.

3.2.2.1 Start-up logs

The following logs can be observed on the console once the FRDM-RW61x board 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
```

3.2.2.2 Connect the client to soft AP

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

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

3.2.2.3 Open the website in the client web browser

Use the AP IP-192.168.1.1 open website http://192.168.1.1 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 AP mode is highlighted on the webpage. See Figure 34.

```
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 34. wifi_webconfig website in AP mode

3.2.2.4 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!
```

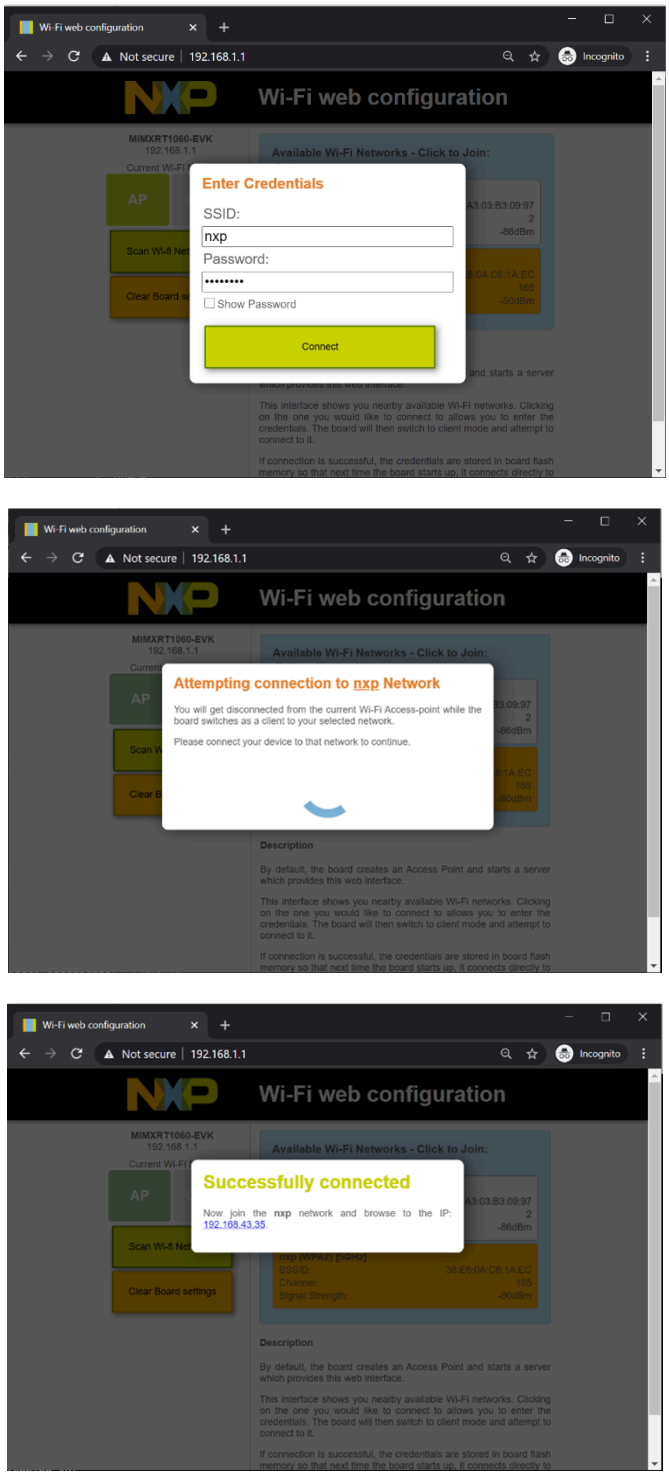

Figure 35. Connection attempt to AP using wifi_webconfig application

Note: *When the device has received by the configurations, soft 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 36 shows http://192.168.43.35 to reconnect to the website.

The current Wi-Fi mode client is highlighted on the webpage shown in Figure 36.

Figure 36. wifi_webconfig website in client mode

3.2.2.5 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
```

3.2.2.6 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 37. Clear board settings

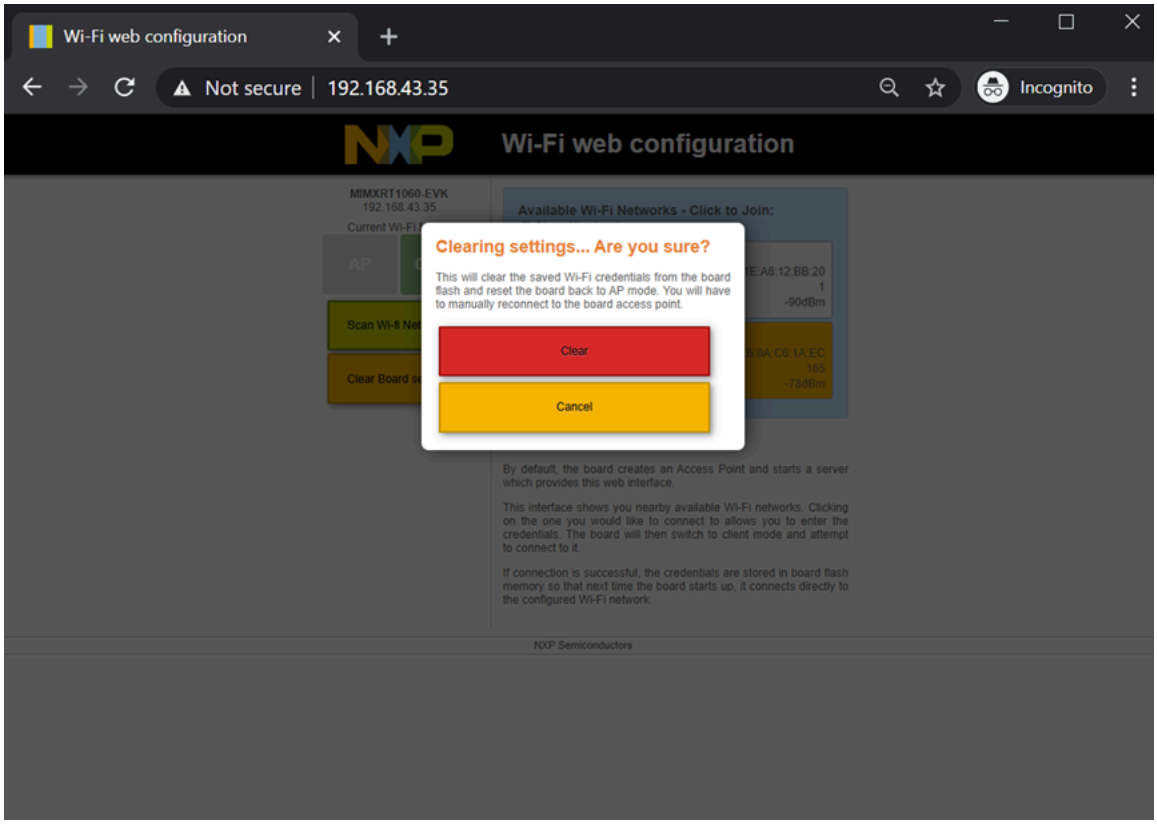

Figure 38. Clear configurations saved in mflash using the website

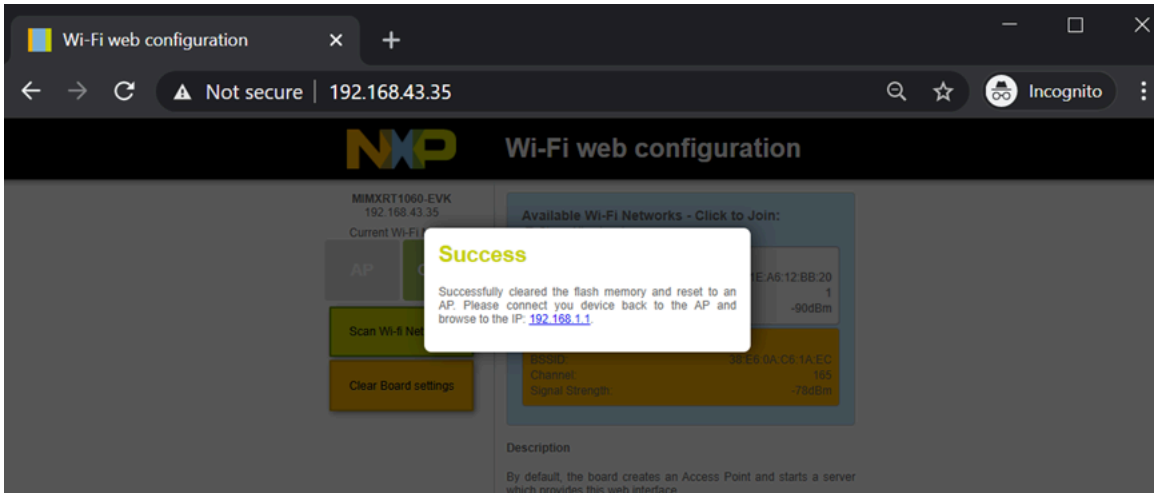

Figure 39. Clear configuration success message in wifi_webconfig application

3.3 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 Soft 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)

3.3.1 wifi_cert application execution

Refer to Section 3.1.2 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.

3.3.1.1 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
=========================================
CLI Build: Aug  7 2024 [16:58:23]
Copyright  2024  NXP
MCU Board: FRDM-RW612
=========================================
Initialize WLAN Driver
=========================================
Wi-Fi cau temperature : 32
```

```
MAC Address: C0:95:DA:01:26:62
PKG_TYPE: BGA
Set BGA tx power table data
========================================
app_cb: WLAN: received event 12
========================================
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
ENHANCED 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-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-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  <sta/uap> <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-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-net-monitor-cfg
wlan-set-monitor-filter <opt> <macaddr>
wlan-set-monitor-param <action> <monitor_activity> <filter_flags> <radio_type>
 <chan_number>
wlan-get-signal
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-get-turbo-mode <STA/UAP>
wlan-set-turbo-mode <STA/UAP> <mode>
wlan-set-multiple-dtim <value>
wlan-set-country <country_code_str>
wlan-set-country-ie-ignore <0/1>
wlan-get-temperature
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-11ax-twt-information <flow_identifier> <suspend_duration>
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6 address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
=======================================
=======================================
app_cb: WLAN: received event 14
=======================================
app_cb: WLAN: PS_ENTER
=======================================
app_cb: WLAN: received event 14
=======================================
app_cb: WLAN: PS_ENTER
```

Note: *Refer to section Section 3.1.6.1 for basic Wi-Fi features like Wi-Fi scan, Wi-Fi AP mode, Wi-Fi station mode, and iPerf.*

3.3.1.2 Set/get the region code

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

Syntax:

```
# wlan-set-regioncode <region-code>
```

Command usage:

```
# wlan-set-regioncode
where, regioncode =
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 output:

```
# wlan-set-regioncode 0x30
PKG_TYPE: BGA
Set BGA tx power table data
Region code: 0x30 set
```

Example of command to get the region code:

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

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.*

3.3.1.3 Get the active/passive channel list

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

Syntax:

```
# wlan-get-chanlist <channel-number>
```

Get the chanlist example output:

```
# wlan-get-chanlist
------------------------------------------------------------------------------
Number of channels configured: 37

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: 7      ChanFreq: 2442  Active
ChanNum: 8      ChanFreq: 2447  Active
ChanNum: 9      ChanFreq: 2452  Active
ChanNum: 10     ChanFreq: 2457  Active
ChanNum: 11     ChanFreq: 2462  Active
ChanNum: 12     ChanFreq: 2467  Active
ChanNum: 13     ChanFreq: 2472  Active
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: 60     ChanFreq: 5300  Passive
ChanNum: 64     ChanFreq: 5320  Passive
ChanNum: 100    ChanFreq: 5500  Passive
ChanNum: 104    ChanFreq: 5520  Passive
ChanNum: 108    ChanFreq: 5540  Passive
ChanNum: 112    ChanFreq: 5560  Passive
ChanNum: 116    ChanFreq: 5580  Passive
ChanNum: 120    ChanFreq: 5600  Passive
ChanNum: 124    ChanFreq: 5620  Passive
ChanNum: 128    ChanFreq: 5640  Passive
ChanNum: 132    ChanFreq: 5660  Passive
ChanNum: 136    ChanFreq: 5680  Passive
ChanNum: 140    ChanFreq: 5700  Passive
ChanNum: 149    ChanFreq: 5745  Active
ChanNum: 153    ChanFreq: 5765  Active
ChanNum: 157    ChanFreq: 5785  Active
ChanNum: 161    ChanFreq: 5805  Active
ChanNum: 165    ChanFreq: 5825  Active
```

3.3.1.4 Get the management frame protection capability

The following commands are used to get MFP capability.

Get MFP capability:

```
# wlan-get-pmfcfg
Management Frame Protection Capability: Yes
Management Frame Protection: Required
```

3.3.1.5 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.

Syntax:

```
# wlan-get-ed-mac-mode <interface>
```

Command usage:

```
# wlan-get-ed-mac-mode
        interface
            # 0       - for STA
            # 1       - for uAP
```

Example output:

```
# wlan-get-ed-mac-mode 0
EU adaptivity for 2.4GHz band : Disabled
EU adaptivity for 5GHz band : Disabled


# wlan-get-ed-mac-mode 1
EU adaptivity for 2.4GHz band : Disabled
EU adaptivity for 5GHz band : Disabled
```

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

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

3.4.1 Flash Wi-Fi MFG firmware

Refer to Section 3.1.1 to flash Wi-Fi firmware. Use the manufacturing firmware instead of the production firmware.

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

3.4.2 Flash Bluetooth MFG firmware

Refer to Section 5.1 to flash Bluetooth firmware. Use the manufacturing firmware instead of the production firmware.

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

3.4.3 uart_wifi_bridge application execution

Refer to Section 3.1.2 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.

3.4.3.1 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 a communication port between the NXP Labtool and the FRDM-RW61x MFG firmware. So, there are no console logs for the uart_wif_bridge application.*

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

Update the `<MFG-IW61X-MF-RTOS-BRG-WIN-X86>\labtool\SetUp.ini` to reflect the COM port settings.

```
[COMSET]
ComNo = 8
BaudRate = 115200
byParity = 0
byStopBits = 1
byByteSize = 8
```

Launch `<MFG-IW61X-MF-RTOS-BRG-WIN-X86>\labtool\DutApiSisoApApp_RW610.exe` and interact with DUT.

Demo execution

```
Name: Dut labtool
Version: 2.0.0.16
Date: May 30 2024 (05:48:05)
Note:
1. =========WiFi tool=============
2. =========BT tool=============
3. =========15_4 tool=============
Enter CMD 99 to Exit
Enter option: 1
Name: DutApiClass
Interface: EtherNet
Version: 2.0.0.16
Date: May 30 2024 (05:47:47)
Note:
DutIf_InitConnection: 0
-----------------------------------------------------
RW610 (802.11a/g/b/n/ac/ax) TEST MENU
-----------------------------------------------------
Enter option: 88
DLL Version : 2.0.0.16
LabTool Version: 2.0.0.16
FW Version: 18.80.6.11 Mfg Version: 2.0.0.63
SFW Version: 0.0.0.00 SHAL Version: 0.0.0.0
SOC OR Version: 1.2 Customer ID: 0
RF OR Version: 2.3 Customer ID: 0
Enter option:
```

3.5 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.

3.5.1 wifi_ipv4_ipv6_echo application execution

Refer to Section 3.1.2 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.

3.5.1.1 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
=========================================
Wi-Fi cau temperature : 32
MAC Address: C0:95:DA:01:26:62
PKG_TYPE: BGA
Set BGA tx power table data
Initialize CLI
=========================================

Copyright  2024  NXP

SHELL>>
```

3.5.1.2 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 one 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

"wlan_connect_with_password ssid password":
   Connects to the specified network with password.
Usage:
   ssid:        network SSID
   password:    password

"wlan_disconnect":
   Disconnect from connected network
SHELL>>
```

3.5.1.3 Scan command

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

```
SHELL>> wlan_scan

Initiating scan...
NXP_V10
     BSSID          : 5C:DF:89:0F:32:78
     RSSI           : -42dBm
     Channel        : 1
NXP_V10
     BSSID          : C8:08:73:39:E4:B8
     RSSI           : -74dBm
     Channel        : 1

SHELL>>
```

3.5.1.4 Connect to found access point

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

"wlan_connect_with_password ssid password":
   Connects to the specified network with password.
Usage:
   ssid:        network SSID
   password:    password
```

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

Once connected to the AP, the console output shows that the Client is successfully connected to AP with ssid “Dlink_2G”

```
SHELL>> wlan_connect Dlink_2G
Joining: Dlink_2G
Network joined
```

3.5.1.5 Print the IP configuration

The command prints the 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.*

3.5.1.6 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**.

3.5.1.7 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**.

3.5.1.8 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 the 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 the 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**.

3.6 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 11.

Table 11. 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

3.6.1 User configurations

Table 12 lists the Wi-Fi features and feature-related macros that the user can configure.

Table 12. 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”		

3.6.2 wifi_httpsrv application execution

Refer to Section 3.1.2 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.

3.6.2.1 Start-up logs

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

```
Starting httpsrv DEMO
[i] Initializing Wi-Fi connection...
Wi-Fi cau temperature : 28
MAC Address: C0:95:DA:01:1C:6C
PKG_TYPE: BGA
Set BGA tx power table data
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: my_network with password my_password
```

3.6.2.2 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...
Wi-Fi cau temperature : 33
MAC Address: C0:95:DA:01:1C:6C
PKG_TYPE: BGA
Set BGA tx power table data
[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: 192.168.0.126

************************************************************
 HTTP Server example
************************************************************
 IPv4 Address     : 192.168.0.126
 IPv4 Subnet mask : 255.255.255.0
 IPv4 Gateway     : 192.168.0.1
 IPv6 Address0    : FE80::C295:DAFF:FE01:1B6C
 IPv6 Address1    : -
 IPv6 Address2    : -
 mDNS hostname    : wifi-http
************************************************************

Ready
```

3.6.2.3 Open the website in the PC browser

Use the board IP-192.168.0.126 to open the website http://192.168.0.126/ in the browser of the PC on the same AP network. The webpage opens.

Figure 40. Webpage

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

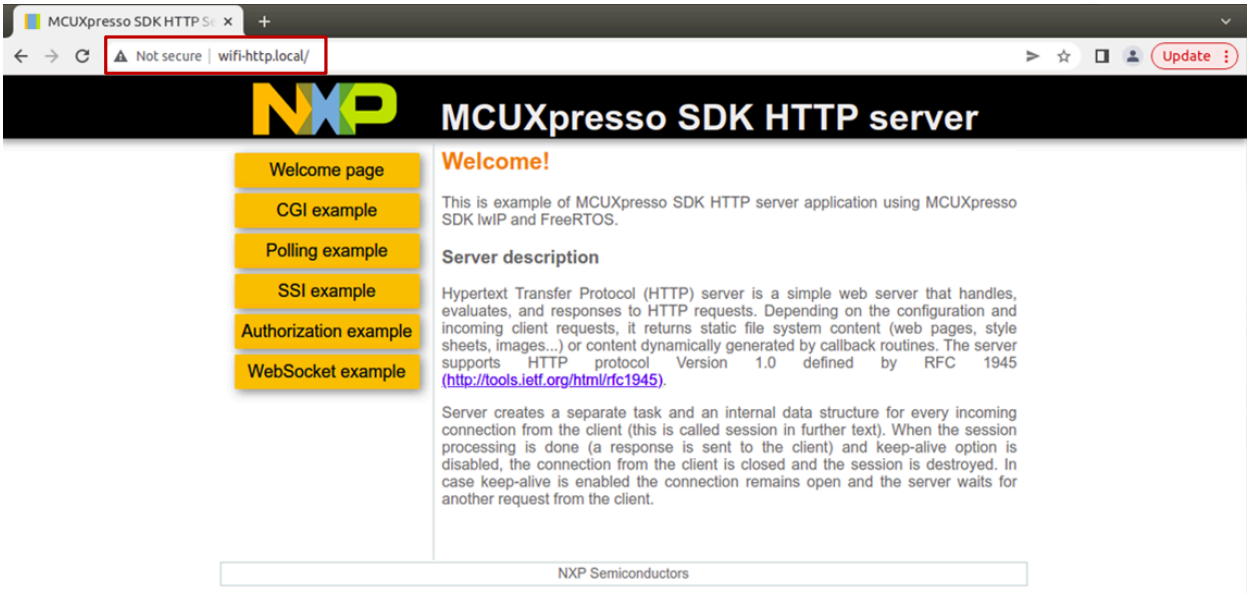

Figure 41. URL for the board on MCUXpresso SDK HTTP server

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.*

3.6.2.4 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.

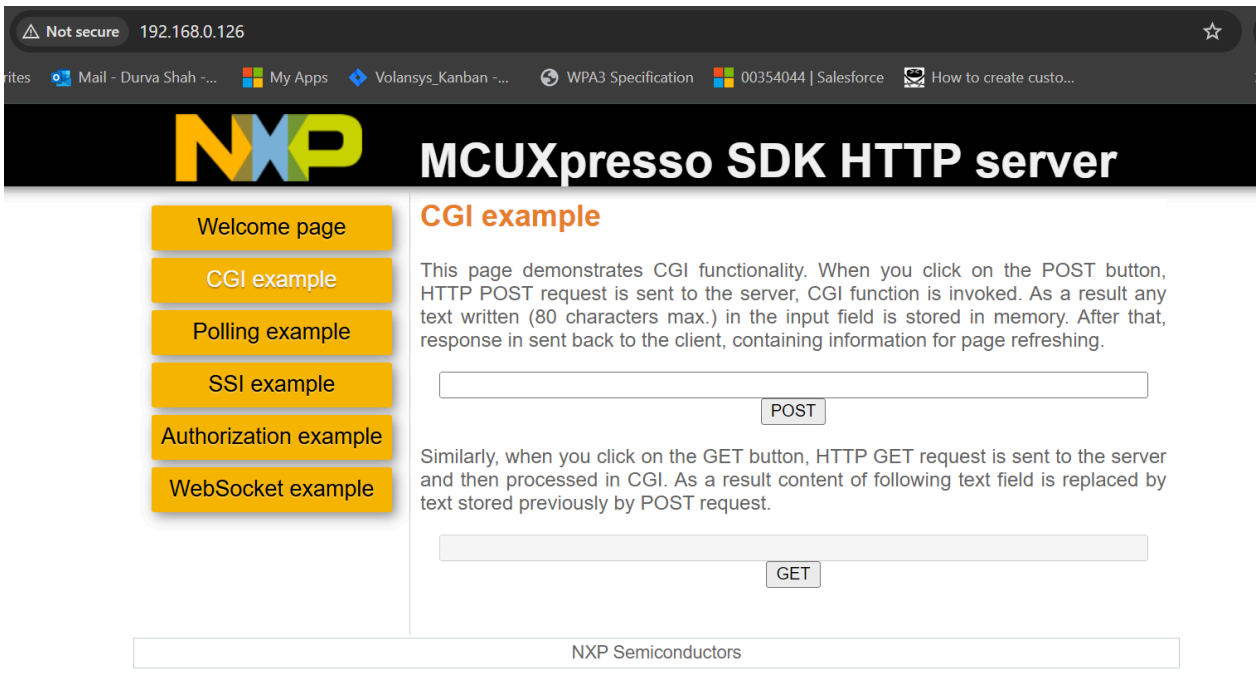

Figure 42. CGI Example Page

Figure 43. CGI Post Example

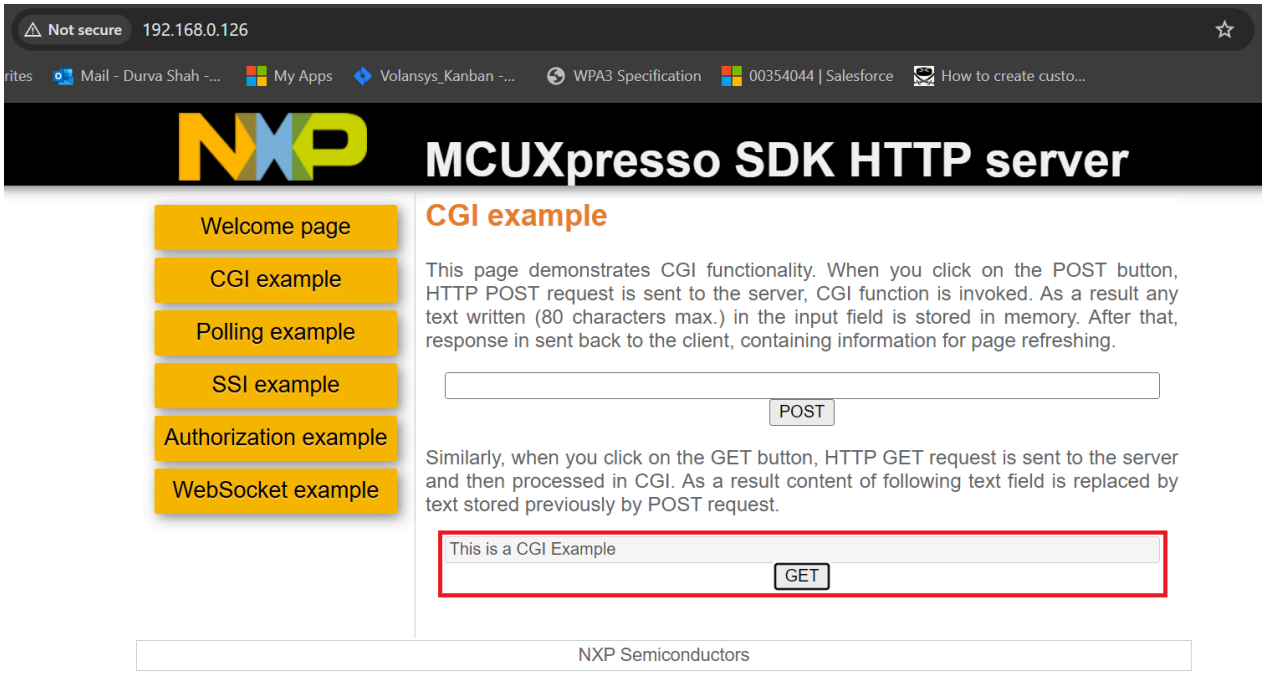

Figure 44. CGI Get Example

3.6.2.5 Polling example

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

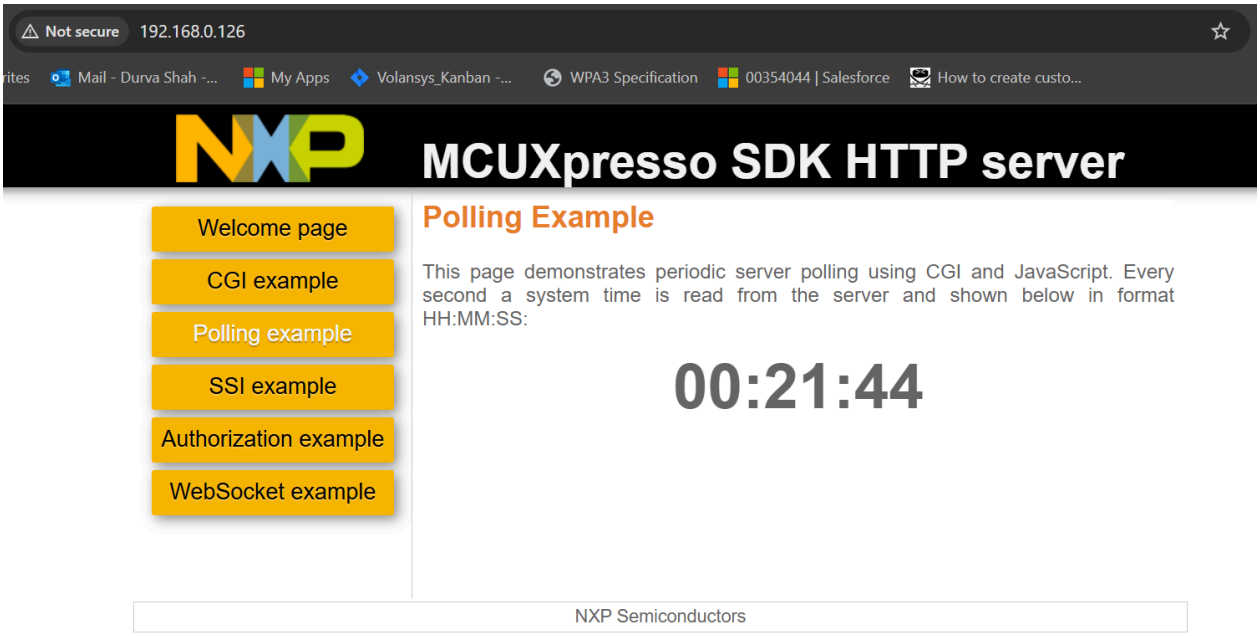

Figure 45. Polling example page

3.6.2.6 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 */
};
```

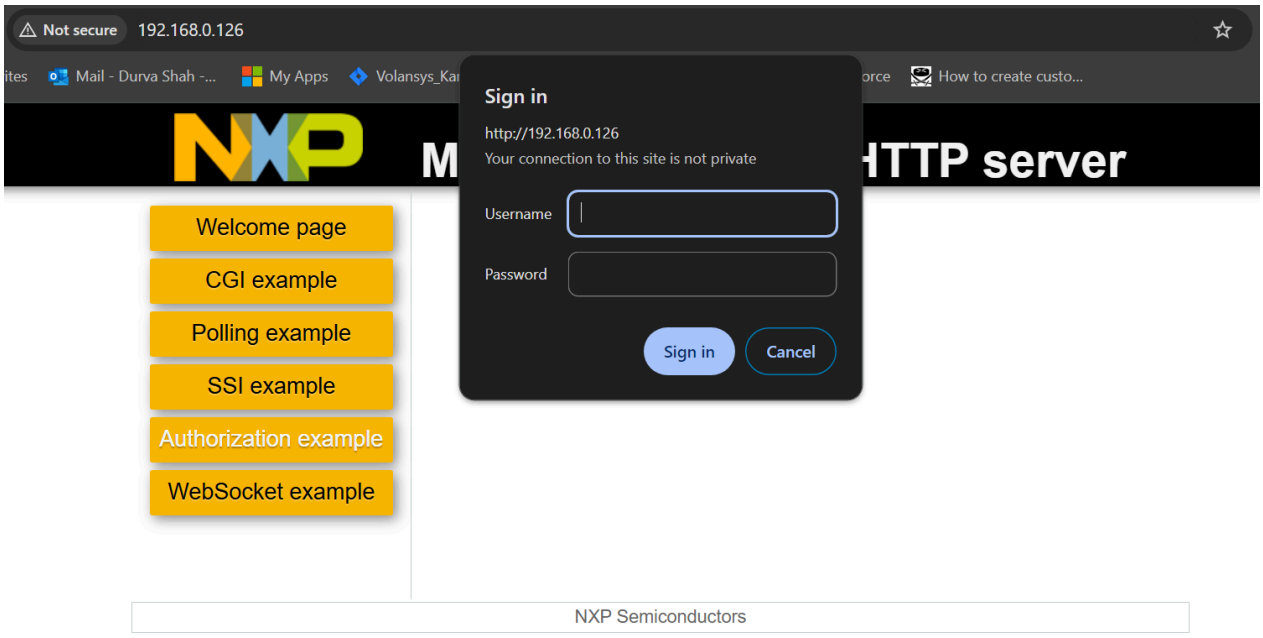

Figure 46. Authorization example page

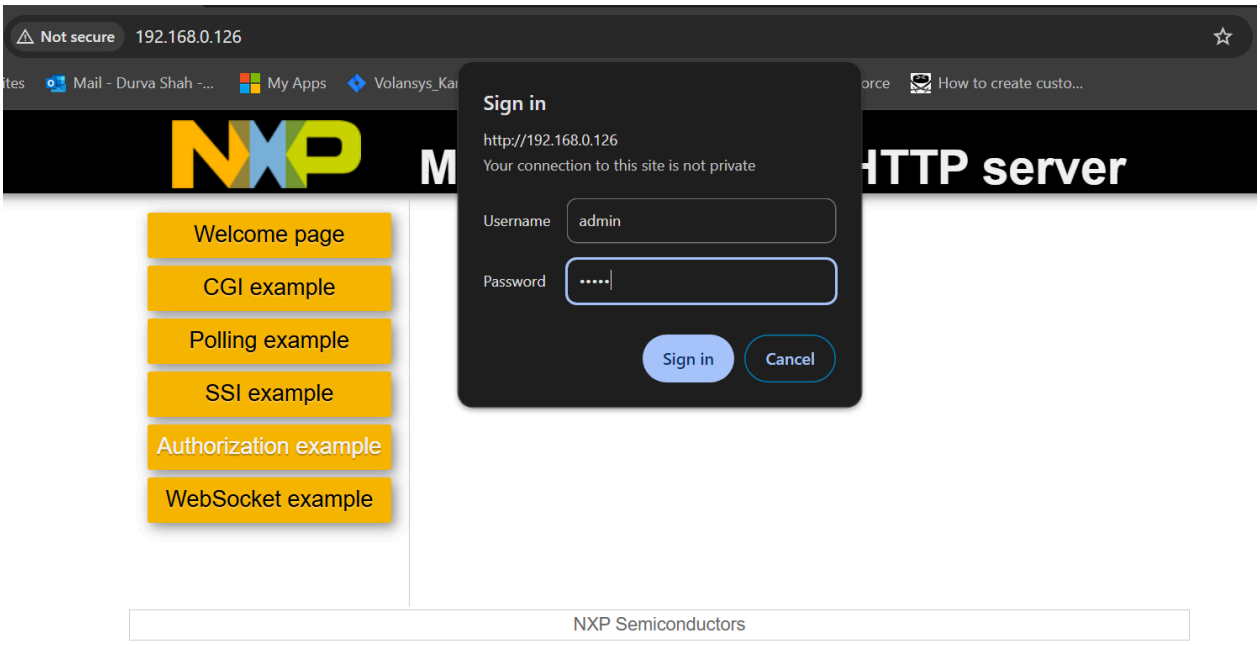

Figure 47. Capturing the username and password to sign in

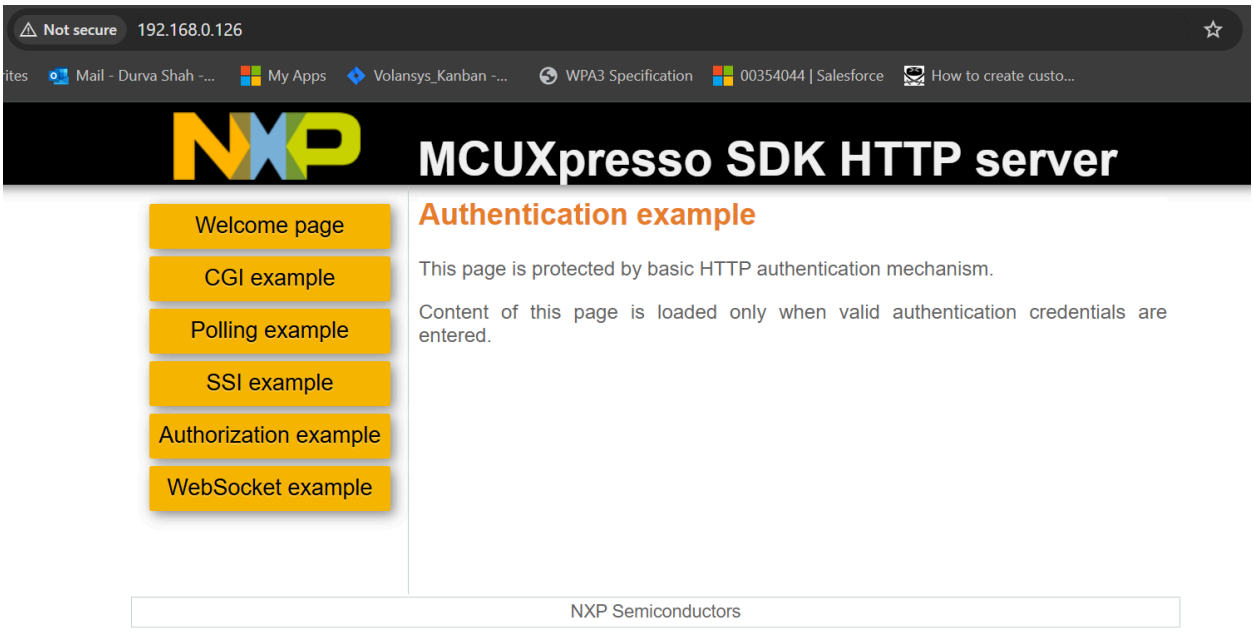

Figure 48. Authorization example page once signed in

3.6.2.7 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 49. Connection request on Websocket example page

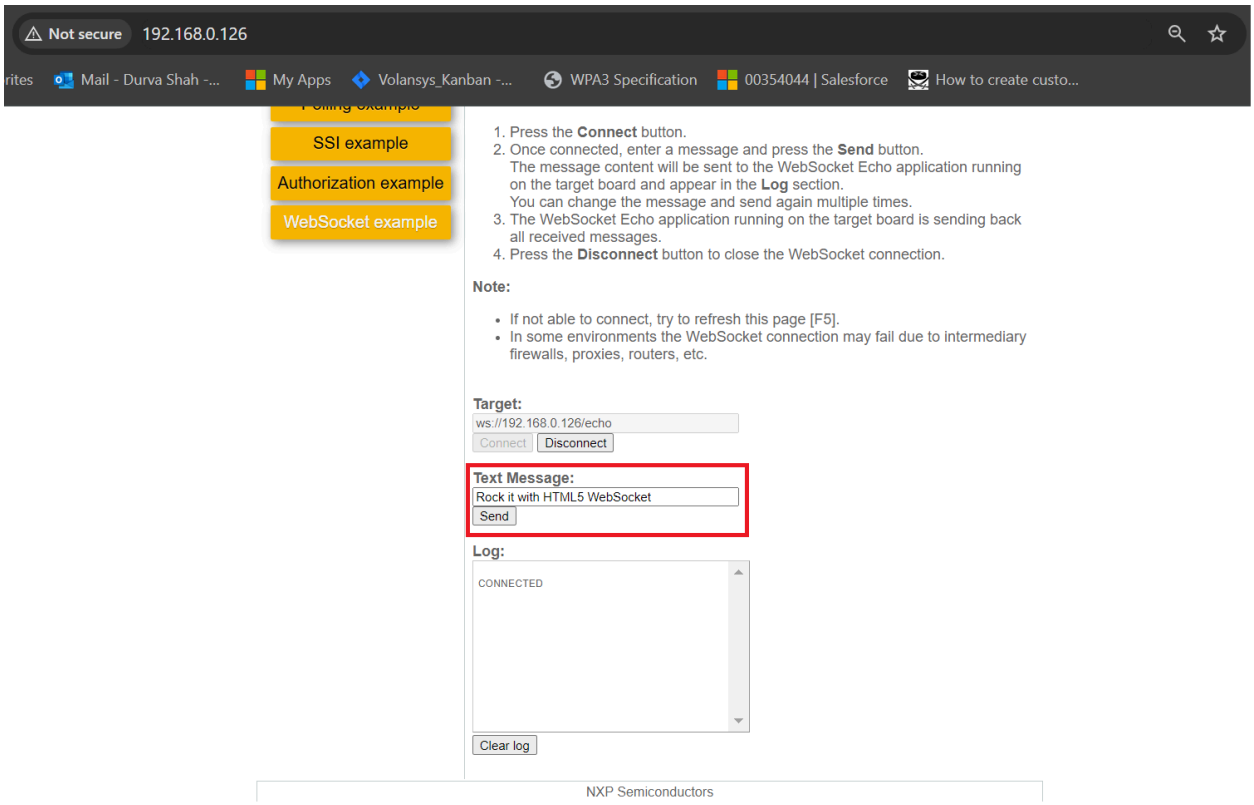

Figure 50. Sending a message on Websocket example page

3.6.2.8 Modify the static webpage

To modify the contents of the static webpage of the 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 the *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, a Windows tool, such as ActivePerl is used.*

- Make sure the *httpsrv_fs_data.c* file has been overwritten with the newly generated content.
- Recompile the wifi_httpsrv sample application and reflash the board.

3.7 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 13.

Table 13. wifi_httpsrv sample application features

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

3.7.1 Wifi_mqtt application execution

Refer to Section 3.1.2 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.

3.7.1.1 Start-up logs

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

```
*********************************************
 MQTT client example
*********************************************
[i] Initializing Wi-Fi connection...
Wi-Fi cau temperature : 32
MAC Address: C0:95:DA:01:1C:6C
PKG_TYPE: BGA
Set BGA tx power table data
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: my_network with password my_password
```

3.7.1.2 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...
Wi-Fi cau temperature : 32
MAC Address: C0:95:DA:01:1C:6C
PKG_TYPE: BGA
Set BGA tx power table data
[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     : 192.168.60.183
IPv4 Subnet mask : 255.255.255.0
```

```
IPv4 Gateway     : 192.168.60.75
```

3.7.1.3 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 3.69.4.220...
MQTT client "nxp_65509f9233382a84475a336551e26f66" connection refused: 5.
Connecting to MQTT broker at 3.69.4.220...
MQTT client "nxp_65509f9233382a84475a336551e26f66" connection refused: 5.
Connecting to MQTT broker at 3.69.4.220...
MQTT client "nxp_65509f9233382a84475a336551e26f66" connection refused: 3.
Connecting to MQTT broker at 3.69.4.220...
MQTT client "nxp_65509f9233382a84475a336551e26f66" 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/#".
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".
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".
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.*

3.8 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 5 GHz), RF packet counts, RF antenna configuration, and transmit standard 802.11 packets.

3.8.1 Wifi_test_mode application execution

Refer to Section 3.1.2 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.

3.8.1.1 Run the application

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

```
=========================================
wifi test mode demo
=========================================
Initialize CLI
=========================================
CLI Build: Aug 12 2024 [11:53:45]
Copyright  2024  NXP
MCU Board: FRDM-RW612
=========================================
Initialize WLAN Driver
=========================================
Wi-Fi cau temperature : 34
MAC Address: C0:95:DA:01:26:62
PKG_TYPE: BGA
Set BGA tx power table data
=========================================
app_cb: WLAN: received event 12
=========================================
app_cb: WLAN initialized
=========================================
WLAN Test Mode CLIs are initialized
=========================================
CLIs Available:
=========================================
help
clear
wlan-version
wlan-mac
wlan-set-rf-test-mode
wlan-unset-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-set-rf-trigger-frame-cfg <Enable_tx> <Standalone_hetb> <FRAME_CTRL_TYPE> <FRAME_CTRL_SUBTYPE>
 <FRAME_DURATION><TriggerType> <UlLen> <MoreTF> <CSRequired> <UlBw> <LTFType> <LTFMode><LTFSymbol>
 <UlSTBC> <LdpcESS> <ApTxPwr> <PreFecPadFct> <PeDisambig> <SpatialReuse><Doppler> <HeSig2> <AID12>
```

```
 <RUAllocReg> <RUAlloc> <UlCodingType> <UlMCS> <UlDCM><SSAlloc> <UlTargetRSSI> <MPDU_MU_SF> <TID_AL>
 <AC_PL> <Pref_AC>
wlan-set-rf-he-tb-tx <enable> <qnum> <aid> <axq_mu_timer> <tx_power>
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
========================================
========================================
app_cb: WLAN: received event 14
========================================
app_cb: WLAN: PS_ENTER
========================================
```

3.8.1.2 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
========================================
app_cb: WLAN: received event 15
========================================
app_cb: WLAN: PS EXIT
========================================
app_cb: WLAN: received event 15
========================================
app_cb: WLAN: PS EXIT
RF Test Mode Set configuration successful
```

Command to get RF band:

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

Command to set RF band:

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

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
                4: 80MHz
```

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
```

3.8.1.3 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                    : 68
  Total Rx Multicast/Broadcast Packet Count: 68
  Total Rx Packets with FCS error          : 216
```

3.8.1.4 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
```

3.8.1.5 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
```

3.8.1.6 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 14 and Table 15 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 14. 802.11n/a/g/b data rate index

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

Table 15. 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		
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		
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

3.8.1.7 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 14 and Table 15 for the data rate index values.*

Command to enable Tx frame:

```
# wlan-set-rf-tx-frame 1 7 2730 256 0 0 0 0 0 0 0 0 0 38:E6:0A:C6:1A:EC
Tx Frame configuration successful
Enable       : enable
Tx Data Rate : 7
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
```

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
```

3.8.1.8 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
Usage:
wlan-set-rf-he-tb-tx <enable> <qnum> <uint16_t aid> <axq_mu_timer> <tx_power>
Enable           (Enable/Disable trigger response mode)
qnum             (AXQ to be used for the trigger response frame)
aid              (AID of the peer to which response is to be generated)
axq_mu_timer     (MU timer for the AXQ on which response is sent)
tx_power         (TxPwr to be configured for the response)
```

Command to enter trigger frame response mode:

```
# wlan-set-rf-he-tb-tx 1 1 5 400 9
HE TB Tx configuration successful
Enable           : 1
qnum             : 1
aid              : 5
axq_mu_timer     : 400
tx_power         : 9
```

Command to exit trigger frame response mode:

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

HE TB Tx configuration successful
Enable           : 0
qnum             : 1
aid              : 5
axq_mu_timer     : 400
tx_power         : 9
```

3.8.1.9 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
```

3.8.1.10 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`.

3.8.1.11 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.r48.p12
WLAN Firmware Version : rw610w-V2, IMU, FP99, 18.99.6.p10, PVE_FIX 1
```

3.8.1.12 Get the Wi-Fi MAC address

Command to get the Wi-Fi MAC address:

```
# wlan-mac
MAC address
STA MAC Address: C0:95:DA:01:25:62
uAP MAC Address: C0:95:DA:01:26:62
```

3.8.1.13 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 16. Tx command sequences for 2.4 GHz

Step	Operation	Command

1	Set RF test mode	`# wlan-set-rf-test-mode`
		`========================================`
		`app_cb: WLAN: received event 15`
		`========================================`
		`app_cb: WLAN: PS EXIT`
		`========================================`
		`app_cb: WLAN: received event 15`
		`========================================`
		`app_cb: WLAN: PS EXIT`
		`RF Test Mode Set 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`
		`Tx continuous configuration successful`
		`Enable : disable`
		`Continuous Wave Mode : disable`
		`Payload Pattern : 0x00000000`

Table 16. Tx command sequences for 2.4 GHz*...continued*

Step	Operation	Command
		CS Mode : disable Active SubChannel : low Tx Data Rate : 0

Table 16. 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`
		`Tx continuous configuration successful`
		`  Enable : disable`
		`  Continuous Wave Mode : disable`
		`  Payload Pattern : 0x00000000`
		`  CS Mode : disable`
		`  Active SubChannel : low`
		`  Tx Data Rate : 0`

3.8.1.14 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 17. Tx command sequence for 5 GHz

Step	Operation	Command

1	Set RF test mode	`# wlan-set-rf-test-mode`
		`========================================`
		`app_cb: WLAN: received event 15`
		`========================================`
		`app_cb: WLAN: PS EXIT`
		`========================================`
		`app_cb: WLAN: received event 15`
		`========================================`
		`app_cb: WLAN: PS EXIT`
		`RF Test Mode Set 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`
		`Tx continuous configuration successful`
		`  Enable : disable`
		`  Continuous Wave Mode : disable`
		`  Payload Pattern : 0x00000000`

Table 17. Tx command sequence for 5 GHz*...continued*

Step	Operation	Command
		CS Mode : disable Active SubChannel : low Tx Data Rate : 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 17. 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`
		`Tx continuous configuration successful`
		`Enable : disable`
		`Continuous Wave Mode : disable`
		`Payload Pattern : 0x00000000`
		`CS Mode : disable`
		`Active SubChannel : low`
		`Tx Data Rate : 0`

3.9 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 18. wifi_wpa_supplicant sample application features

Features	Details
Wi-Fi	Wi-Fi Host based supplicant Wi-Fi Soft 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)

3.9.1 wifi_wpa_supplicant application execution

Refer to Section 3.1.2 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.

3.9.1.1 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: Sep  2 2024 [12:28:13]
Copyright  2024  NXP
MCU Board: FRDM-RW612
=========================================

MCU wakeup source 0x0...
Initialize WLAN Driver
=========================================
Wi-Fi cau temperature : 27
MAC Address: C0:95:DA:01:1C:6C
supplicant_main_task: 603 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-uap-disconnect-sta <mac address>
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-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-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  <sta/uap> <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-external-coex-pta enable <PTA/WCI-2/WCI-2 GPIO> ExtWifiBtArb <enable/disable> PolGrantPin
 <high/low> PriPtaInt <enable/disable> StateFromPta <state pin/ priority >
wlan-sta-inactivityto <n> <m> <l> [k] [j]
wlan-get-temperature
wlan-auto-null-tx <sta/uap> <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-11ax-twt-information <flow_identifier> <suspend_duration>
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>
========================================
```

3.9.1.2 Add a network profile

Before adding a network profile for Station and Soft AP mode, 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_pa]
    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>]
    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_]
    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 <clien]
    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/]
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-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_aucw
If setting dtim
The value of dtim is an integer. The default value is 10.
Note: Setting 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
```

3.9.1.3 Station mode (connect to AP)

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

Note: *A second FRDM-RW61x is used as an External AP on which the radius certificates are configured. To configure your own certificates, refer to Section 3.9.1.5.*

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
```

3.9.1.4 Soft 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 3.9.1.5.*

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
========================================
Soft 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 Soft 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
```

3.9.1.5 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 Read/dump USB file 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

FRDM-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.*

3.9.1.6 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='NXPAP' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPAP' 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
 100e00301026000101104500054e585041501003000200201000f000200081027000831323334353637381020000 6c095da011fa9
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='NXPAP' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPAP' 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=1 id_str=]
app_cb: WLAN: authenticated to network
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [NXPAP]
IPv4 Address: [192.168.81.1]
```

- Start WPS PBC on soft AP:

```
# wlan-start-ap-wps-pbc
: WPS-PBC-ACTIVE
# ua2: STA c0:95:da:01:1f:a9 IEEE 802.11: associated (aid 1)
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1f:a9
: 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:a9 8f5d4c61-f31f-5a79-b8f0-8d0d92b861da
: WPS-PBC-DISABLE
: WPS-SUCCESS
: CTRL-EVENT-EAP-FAILURE c0:95:da:01:1f:a9
ua2: STA c0:95:da:01:1f:a9 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
ua2: STA c0:95:da:01:1f:a9 IEEE 802.1X: Supplicant used different EAP type: 254 (expanded)
ua2: STA c0:95:da:01:1f:a9 IEEE 802.11: associated (aid 1)
: AP-STA-CONNECTED c0:95:da:01:1f:a9
app_cb: WLAN: UAP a Client Connected
=======================================
Client => C0:95:DA:01:1F:A9 Connected with Soft AP
=======================================
ua2: STA c0:95:da:01:1f:a9 WPA: pairwise key handshake completed (RSN)
: EAPOL-4WAY-HS-COMPLETED c0:95:da:01:1f:a9
```

WPS-PIN

- Generate WPS PIN:

```
# wlan-generate-wps-pin
WPS PIN is: 26991825
```

- Start WPS PIN on the station:

```
# wlan-start-wps-pin 26991825
ml1: WPS-PIN-ACTIVE
Started WPS PIN session with pin as: 26991825

# ml1: SME: Trying to authenticate with c0:95:da:01:1c:6c (SSID='NXPAP' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPAP' 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
 100e00301026000101104500054e58504150100300020020100f000200081027000831323334353637381020000 6c095da011fa9
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='NXPAP' freq=2437 MHz)
ml1: Trying to associate with c0:95:da:01:1c:6c (SSID='NXPAP' 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 = [NXPAP]
IPv4 Address: [192.168.81.1]
IPv6 Address: Link-Local   :    FE80::C295:DAFF:FE01:1FA9 (Preferred)
```

- Start WPS PIN on soft AP:

```
# wlan-start-ap-wps-pin 26991825
Started AP WPS PIN session with pin as: 26991825

# ua2: STA c0:95:da:01:1f:a9 IEEE 802.11: associated (aid 1)
: CTRL-EVENT-EAP-STARTED c0:95:da:01:1f:a9
: 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:a9 8f5d4c61-f31f-5a79-b8f0-8d0d92b861da
: WPS-SUCCESS
: CTRL-EVENT-EAP-FAILURE c0:95:da:01:1f:a9
ua2: STA c0:95:da:01:1f:a9 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
ua2: STA c0:95:da:01:1f:a9 IEEE 802.1X: Supplicant used different EAP type: 254
 (expanded)
ua2: STA c0:95:da:01:1f:a9 IEEE 802.11: associated (aid 1)
: AP-STA-CONNECTED c0:95:da:01:1f:a9
app_cb: WLAN: UAP a Client Connected
========================================
Client => C0:95:DA:01:1F:A9 Connected with Soft AP
========================================
ua2: STA c0:95:da:01:1f:a9 WPA: pairwise key handshake completed (RSN)
: EAPOL-4WAY-HS-COMPLETED c0:95:da:01:1f:a9
```

3.9.1.7 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 (FRDM-RW61x STA) operates as enrollee and authentication initiator.
- Device1 (FRDM-RW61x STA) operates as configurator.
- Device2 (FRDM-RW61x soft 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 soft 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 ThisIsDppPassphrase
Added "testAP"

# 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
========================================
Soft 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:A9"

 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:c095da0120a9;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACjNY0eTUZxeTOqeFpK9zUKp8jtKRQ7mkjq1Gl4xiNLQ0=
```

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

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:c095da0120a9;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACjNY0eTUZxeTOqeFpK9zUKp8jtKRQ7m
 DPP qr code id = 1
```

- 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:a9 freq=2462 type=0

 DPP Auth Init OK!

# ml1: DPP-TX-STATUS dst=c0:95:da:01:20:a9 freq=2462 result=SUCCESS
ml1: DPP-RX src=c0:95:da:01:20:a9 freq=2462 type=1
ml1: DPP-AUTH-DIRECTION mutual=0
ml1: DPP-TX dst=c0:95:da:01:20:a9 freq=2462 type=2
ml1: DPP-TX-STATUS dst=c0:95:da:01:20:a9 freq=2462 result=SUCCESS
ml1: DPP-AUTH-SUCCESS init=1
ml1: DPP-CONF-REQ-RX src=c0:95:da:01:20:a9
ml1: DPP-RX src=c0:95:da:01:20:a9 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:01:1c:7a freq=2462 type=0
: DPP-TX dst=c0:95:da:01:1c:7a freq=2462 type=1
: DPP-TX-STATUS dst=c0:95:da:01:1c:7a result=SUCCESS
: DPP-RX src=c0:95:da:01:1c:7a freq=2462 type=2
: DPP-AUTH-SUCCESS init=0
: GAS-QUERY-START addr=c0:95:da:01:1c:7a dialog_token=0 freq=2462
: GAS-QUERY-DONE addr=c0:95:da:01:1c:7a dialog_token=0 freq=2462 status_code=0 result=SUCCESS
: DPP-CONF-RECEIVED
: DPP-CONFOBJ-AKM dpp
: DPP-CONFOBJ-SSID DPPNET01
: DPP-CONNECTOR
 eyJ0eXAiOiJkcHBDb24iLCJraWQiOiJOVGJvSkhDY2h4WlFNRlVwWmVLVllZVmYwRExKcjNGT0pVWmNSNjFjb3M0IiwiYWxnIjoiRVMyNTYifQ.e
: DPP-C-SIGN-KEY
 3039301306072a8648ce3d020106082a8648ce3d03010703220003a45e5d764d40c5dbeb65d41877fadf2ec0eca2f87ad30e4acbe3873cf9
: DPP-NET-ACCESS-KEY
 307702010104203c35cfa39c20b3c384b3e9d7614d7745c0497f3ba975f8bb4af52c5fd418c27ba00a06082a8648ce3d030107a144034200
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:01:1c:7a freq=2462 type=11
: DPP-TX-STATUS dst=c0:95:da:01:1c:7a result=SUCCESS
```

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

- Set the configurator parameter:

```
# wlan-dpp-configurator-params " conf=sta-dpp ssid=4E58504150 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:01:1D:7A"
 bootstrap generate id = 2
```

- Get the QR code URI:

```
# wlan-dpp-bootstrap-get-uri 2

Bootstrapping QR Code URI:

DPP:C:81/11;M:c095da011d7a;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACo6TnZAKFOfwlXTeQplEkZpxVXzAVoQH7xr3t3TcRqS8=
```

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:c095da011d7a;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACo6TnZAKFOfwlXTeQplEkZpxVXzAVoQ

 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 3039301306072a8648ce3d020106082a8648ce3d03010703220002145 7d6b07b6ff77735928cdb4f8 \
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 Soft 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
```

3.9.1.8 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 FRDM-RW61x
2. The configuration of keep-alive parameters

Test setup:

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

Step 1 - Configure FRDM-RW61x in station mode.

```
# wlan-add abc ssid NXP_FRDM
```

Step 2 - Connect to external AP.

```
# wlan-connect abc
Connecting to network...
Use 'wlan-stat' for current connection status.
# ml1: SME: Trying to authenticate with 50:2b:73:5d:b2:11 (SSID='NXP_FRDM' freq=2412 MHz)
ml1: Trying to associate with 50:2b:73:5d:b2:11 (SSID='NXP_FRDM' freq=2412 MHz)
ml1: Associated with 50:2b:73:5d:b2:11
app_cb: WLAN: authenticated to network
ml1: CTRL-EVENT-CONNECTED - Connection to 50:2b:73:5d:b2:11 completed [id=0 id_str=]
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [NXP_FRDM]
IPv4 Address: [192.168.1.188]
IPv6 Address: Link-Local   :    FE80::C295:DAFF:FE01:1C7A (Preferred)
```

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

Step 4 - Start cloud keep alive on FRDM-RW61x.

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

Table 19. 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 a4:fc:77:49:81:e7 dst_ip 192.168.1.189
 dst_port 9526
```

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

```
# wlan_tcp_client dst_ip 192.168.1.189 src_port 54236 dst_port 9526
```

Step 6 - Verify the TCP connection on the sniffer capture (Figure 51).

Figure 51. 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 FRDM-RW61x enters sleep state, packets show on the sniffer (Figure 52).

```
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 52. 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;
```

4 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 3.3, as it supports these APIs*
- *Refer to MCUXSDKGSUG for more details about the Wi-Fi driver APIs*

4.1 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.

4.1.1 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 20. 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.

4.1.2 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 21. 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.

4.1.3 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 3.1.7.

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
```

5 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 ([2]).

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

5.1 Flash Bluetooth LE firmware

FRDM-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 FRDM-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 FRDM-RW61x device

```
J-Link>con
Device> RW612
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_a2.bin for A2 version of FRDM-RW61x.

```
J-Link>loadbin rw61x_sb_ble_a2_ble_<version number>.bin, 0x08540000J-Link>loadbin
 rw61x_sb_ble_a2.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.*

5.2 peripheral_hps sample application

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

5.2.1 peripheral_hps application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.2.1.1 Run the application

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

```
BLE Peripheral HPS demo start...
Bluetooth initialized
Advertising successfully started
```

The demo does not require user interaction.

The application automatically starts advertising as temperature_sensor.

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:01:1C:7B (public) Processing request..
Request processed.
Security changed: C0:95:DA:01:1C:7B (public) level 2 (error 0)
```

5.3 central_hpc sample application

This application demonstrates very basic Bluetooth LE central role functionality on the FRDM-RW61x 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 the FRDM-RW61x board is used as *peripheral_hps*.

5.3.1 central_hpc application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.3.1.1 Run the application

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

```
BLE Central HPC demo start...
Bluetooth initialized
Scanning started
[DEVICE]: C0:95:DA:01:25:63 (public), AD evt type 0, AD data len 7, RSSI -51
[DEVICE]: D4:53:83:C3:E5:CD (public), AD evt type 0, AD data len 19, RSSI -82
```

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: C0:95:DA:01:25:63 (public)Connected to peer: C0:95:DA:01:25:63 (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:01:25:63 (public) level 2 (error 0)
```

5.4 peripheral_pxr sample application

This application demonstrates the Bluetooth LE Peripheral role on the FRDM-RW61x board. More specifically, this application exposes the Proximity Reporter (including LLS, IAS, and TPS) GATT Service.

5.4.1 peripheral_pxr application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.4.1.1 Run the application

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

```
BLE Peripheral PXR demo start...
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:01:25:63 (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
```

5.5 central_pxm sample application

This application demonstrates very basic Bluetooth LE Central role functionality on the FRDM-RW61x 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 FRDM-RW61x board is used as *peripheral_pxr*.

5.5.1 central_pxm application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.5.1.1 Run the application

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

```
BLE Central PXM demo start...
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

```
DEVICE]: C0:95:DA:01:1C:7B (public), AD evt type 0, AD data len 11, RSSI -73
Found device: C0:95:DA:01:1C:7B (public)
Connected to peer: C0:95:DA:01:1C:7B (public)
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: -51
Connection RSSI: -52
Connection RSSI: -51
Connection RSSI: -52
Connection RSSI: -53
```

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.

5.6 peripheral_ht sample application

This application demonstrates the Bluetooth LE Peripheral role on the FRDM-RW61x board. More specifically, this application exposes the HT (Health Thermometer) GATT Service.

When a Bluetooth device connects, it generates dummy temperature values.

5.6.1 peripheral_ht application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.6.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of the FRDM-RW61x board.

When the demo starts, the following message about the demo shows on the console.

```
BLE Peripheral HT demo start...
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: 46:00:97:EB:40:9D (random)
Passkey for 46:00:97:EB:40:9D (random): 992804
Security changed: 48:01:C5:27:E6:80 (public) level 4 (error 0)
temperature is 20C
Indication success
temperature is 21C
Indication success
temperature is 22C
```

5.7 central_ht sample application

This application demonstrates very basic Bluetooth LE Central role functionality on the FRDM-RW61x 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 the FRDM-RW61x board is used as *peripheral_ht*.

5.7.1 central_ht application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.7.1.1 Run the application

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

```
BLE Central HT demo start...
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:01:1B:5E (public), AD evt type 0, AD data len 9, RSSI -58
Found device: C0:95:DA:01:1B:5E (public)Connected to peer: C0:95:DA:01:1B:5E (public)
Starting service discovery
Subscribed to HTS
[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
Temperature 21 degrees Celsius
```

5.8 peripheral_ipsp sample application

This application demonstrates the Bluetooth LE Peripheral role on the FRDM-RW61x board. More specifically, this application exposes the Internet Protocol Support GATT Service.

5.8.1 peripheral_ipsp application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.8.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of the FRDM-RW61x board.

When the demo starts, the following message about the demo shows on the console.

```
BLE Peripheral IPSP demo start...
Bluetooth initialized
Advertising successfully started
```

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.

```
IPSS Service ready
Connected to peer: 6B:A4:65:28:CC:23 (random)
Passkey for 6B:A4:65:28:CC:23 (random): 704971
Security changed: D0:28:BA:D3:C4:86 (public) level 4 (error 0)
Received message: hello
Received message: hello
Received message: hello
```

5.9 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 the FRDM-RW61x board is used as *peripheral_ipsp*.

5.9.1 central_ipsp application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.9.1.1 Run the application

To run the demo application downloaded on the board, reset the power supply of the FRDM-RW61x 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]: 64:16:FA:4A:30:A4 (random), AD evt type 2, AD data len 4, RSSI -88
[DEVICE]: 48:70:10:18:A7:E8 (random), AD evt type 0, AD data len 17, RSSI -85
Found device: 48:70:10:18:A7:E8 (random)Connected
Starting service discovery
Discover complete, No attribute found
Passkey for 48:70:10:18:A7:E8 (random): 592137
Sending message...
Sending message...
Sending message...
```

5.10 peripheral_beacon sample application

This application demonstrates the Bluetooth LE Peripheral role on FRDM-RW61x. 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.

5.10.1 peripheral_beacon application execution

Refer to Section 3.1.2 and Section 3.1.5 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
```

5.10.1.1 Run the application

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

```
BLE Beacon demo start...
Bluetooth initialized
Beacon started, advertising as C0:95:DA:01:25:63 (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.

5.11 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.

5.11.1 Shell application execution

Refer to Section 3.1.2 to Section 3.1.5 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.

5.11.1.1 Run the shell application

Press the power reset button on the FRDM-RW61x 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: Aug  8 2024
Copyright  2020  NXP
@bt>
```

Note: *In the code sample above,* `SHELL build: Aug 8 2024` *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.

```
+---"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 <address: XX:XX:XX:XX:XX:XX>
    +---"id-reset": id-reset <id> <address: XX:XX:XX:XX:XX:XX>
    +---"id-delete": id-delete <id>
    +---"id-show": id-show [none]
    +---"id-select": id-select <id>
    +---"name": name [name]
    +---"appearance": appearance [none]
    +---"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 <address: XX:XX:XX:XX:XX:XX>
        +---"rssi": rssi <rssi>
        +---"pa_interval": pa_interval <pa_interval>
    +---"scan-filter-clear": scan-filter-clear Scan filter clear commands
        +---"all": all
        +---"name": name
        +---"addr": addr
    +---"scan-verbose-output": scan-verbose-output <value: on, off>
    +---"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] [appearance]
 [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 <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"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 [all] [<address: XX:XX:XX:XX:XX:XX> <type: (public|random)>]
    +---"security": security <security level BR/EDR: 0 - 3, LE: 1 - 4> [force-pair]
    +---"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]: 2A:6C:C2:C9:D3:5E (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 us), SID: 0xff
[DEVICE]: 70:3B:0E:BC:84:B5 (random), AD evt type 3, RSSI -93  C:0 S:0 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 us), 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 D0:28:BA:D3:C4:86 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 06 19 12 08 00 00 02 04 00
```

5.11.1.2 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.

```
Set Bluetooth LE TX power
@bt> le_test.set_tx_power 2
tx_power= 2
feloss= 0
HCI Command Response : @bt> 00
 HCI Command Response : 00
```

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
@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
```

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
```

```
@bt> le_test.end_test API returned success...
@bt> le_test.end_test
Number of packets received: 0
@bt> API returned success...
```

End a test for Bluetooth LE

Command to end any test for Bluetooth LE:

```
@bt> le_test.end_test
Number of packets received: 0
@bt> API returned success...
```

Note: *Observe the packet count in command complete event in HCI log during LE receiver test.*

5.12 central_fmp sample application

This application demonstrates basic Bluetooth LE Central role functionality by scanning for other Bluetooth LE devices and establishing a connection to the first one with a strong enough signal.

This application specifically looks for a peer advertising the Intermediate Alert Service and implementing the 'Find Me Profile'.

5.12.1 central_fmp application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.12.1.1 Run the application

The Demo automatically starts scanning and connects to the first advertiser who is advertising the Intermediate Alert Service.

```
Bluetooth initialized
Scanning started
```

If the connection is successful, the application performs service discovery to find the Alert Level characteristic.

```
Found device: 65:39:DE:D6:D3:4F (random)
Connected to peer: 65:39:DE:D6:D3:4F (random)
Starting service discovery
Discover complete, No attribute found
```

```
Found device: 65:39:DE:D6:D3:4F (random)Connected to peer: 65:39:DE:D6:D3:4F (random)
Starting service discovery
Discover complete, No attribute found
GATT Write successful
GATT Write successful
```

After the disconnect from the target, the locator restarts scanning.

5.13 perhipheral_fmp sample application

This application demonstrates the Bluetooth LE Central role on FRDM-RW61x, except that this application specifically exposes the Find Me Profile in the Target role.

5.13.1 perhipheral_fmp application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.13.1.1 Run the application

The demo does not require user interaction. The application will automatically start advertising the Intermediate Alert Service and it accepts the first connection request it receives.

```
Bluetooth initialized
Fast Advertising successfully started
Slow Advertising successfully started
```

The application initially sets the default levels for the Link Loss Alert and the Immediate Alert.

```
Connected to peer: 5F:C0:44:21:4F:12 (random)
ALERT: OFF
Passkey for 5F:C0:44:21:4F:12 (random): 108015
Security changed: AC:C0:48:9F:82:5A (public) level 4 (error 0)
```

The Central triggers an Immediate Alert on the application and disconnects. The Target then resumes advertising.

```
Locator is setting Immediate Alert...

ALERT: OFF
```

5.14 central_tip sample application

This application demonstrates the Bluetooth LE Central role on the FRDM-RW61x board. The application demonstrates basic Bluetooth LE Central role functionality by scanning and connecting with the first scanned Time Server.

5.14.1 central_tip application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.14.1.1 Run the application

The demo does not require user interaction. The application automatically starts scanning and connects to the first advertiser who is advertising the Current Time Service.

```
Bluetooth initialized
Scanning started
[DEVICE]: F6:B4:B3:7B:57:B1 (random), AD evt type 0, AD data len 30, RSSI -93
[DEVICE]: C0:95:DA:01:1B:5E (public), AD evt type 0, AD data len 11, RSSI -67
```

If the connection is successful, the application performs service discovery to find the characteristics of the Current Time Service. If discovery is successful, the application subscribes to receive time notifications from the peer.

The application displays the received notifications in the console.

```
Monday, 12 of July 2022, 0:0:5
Monday, 12 of July 2022, 0:0:10
Monday, 12 of July 2022, 0:0:15
Monday, 12 of July 2022, 0:0:20
```

5.15 perhipheral_tip sample application

This application demonstrates the Bluetooth LE Central role on FRDM-RW61x. More specifically, the application demonstrates the BLE Peripheral role, except that this application specifically implements the Time Profile.

5.15.1 perhipheral_tip application execution

Refer to Section 3.1.2 and Section 3.1.5 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.

5.15.1.1 Run the application

The demo does not require user interaction. The application automatically starts advertising the Current Time Service and the Reference Time Update Service and it accepts the first connection request it receives.

```
Bluetooth initialized
Advertising successfully started
Connected to peer: C0:95:DA:01:25:63 (public)
Security changed: C0:95:DA:01:25:63 (public) level 2 (error 0)
```

Once the client configures the Current Time Characteristic CCCD for notifications, the server periodically notifies its local time value to the client.

```
Monday, 12 of July 2022, 0:0:5
Monday, 12 of July 2022, 0:0:10
Monday, 12 of July 2022, 0:0:15
Monday, 12 of July 2022, 0:0:20
```

6 Acronyms and abbreviations

Table 22. Acronyms and abbreviations

Terms	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 22. Acronyms and abbreviations...*continued*

Terms	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
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

7 References

[1] User manual - Getting Started with Wireless on FRDM-RW61x Evaluation board Running RTOS (UG10160)

[2] Specification - SIG - Bluetooth Core Specification (link)

8 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 2022-2024 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.

9 Revision history

Table 23 summarizes revisions to this document.

Table 23. Revision history

Document ID	Release date	Description
UG10171 v.1.0	20 September 2024	Initial public release

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.

Contents