UM11442

NXP Wi-Fi and Bluetooth Demo Applications User Guide for i.MX RT Platforms

Rev. 18 – 10 Jan 2025 User Manual

Document information

Information	Content
Keywords	i.MX RT crossover MCU, i.MX RT products, MX RT1060 EVK board, MCUXpresso SDK, 88W8801-based wireless module, IW416- based wireless module, 88W8987-based wireless module, IW611/612-based wireless module, 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 required tool set up

Revision history

Rev	Date	Description
v.1	20200717	Initial version
v.2	20210110	**Modifications**: • Extended the scope to IW416-based modules • Table 3: updated • Section 1 “About this Document”: updated • Section 2 “Tool Setup”: updated • Section 3 “Wi-Fi Sample Applications”: updated • Section 4 “Useful Wi-Fi APIs”: updated • Section 3.5 “wifi_test_mode Sample Application”: added • Section 3.6 “wifi_cert Sample Application”: added • Section 5 ”Bluetooth Classic/Low Energy Application”: added • Section 6 “Acronyms and abbreviations”: added
v.3	20210331	**Modifications**: • Section 1.3 “References”: updated • Section 3.1.4 “Run a Demo using ARM GCC”: updated • Section 3.5 “wifi_test_mode Sample Application”: updated • Section 3.6 “wifi_cert Sample Application”: updated • Section 3.6.1 “wifi_cert Application Execution”: updated • Section 5 ”Bluetooth Classic/Low Energy Application”: updated • Table 15: added
v.4	20210602	**Modifications**: • Document Format modifications • Section 1.3 “References”: updated • Section 1 “About this Document”: updated • Section 2 “Tool Setup”: updated • Table 3: updated • Section 3.1 “wifi_iperf Sample Application”: updated • Section 3.1.3.2 “Project Settings”: updated • Section 3.1.4.2 “Project Settings”: updated • Section 3.1.5.2 “Project Settings”: updated • Section 3.1.6.3 “Project Settings”: updated • Section 3.1.7.1 “Start-up logs”: updated • Section 3.2 “wifi_setup Sample Application”: added • Table 12: updated • Section 3.3.1.1 “Run the application”: updated • Section 3.3.1.3 “Wi-Fi Power Save”: added • Section 3.3.1.4 “Other useful CLI commands”: updated • Figure 35: updated • Section 3.4.2.1 “Start-up logs”: updated • Section 3.4.2.5 “Device reboot with configuration stored in mflash”: updated • Section 3.5.1.3 “Wi-Fi Packet count”: updated • Table 15: updated • Section 3.5.1.8 “Other useful CLI commands”: updated • Table 16: updated • Table 17: updated • Section 3.6.1.1 “Run the application”: updated • Section 3.6.1.6 “Set/Get Tx Rate Configuration”: updated

		• Table 21: updated • Section 2.2 "Wireshark Tool Setup": added • Section 5 "Bluetooth Classic/Low Energy Applications": updated • Section 5.14.2 "audio_profile Application Execution": updated • Section 5.15.2 "wifi_provisioning Application Execution": updated
v.5	20210823	**Modifications**: • Section 1.3 "References": updated • Table 2: updated • Section 3.1.3.2 "Project Settings": updated • Section 3.1.4.2 "Project Settings": updated • Section 3.1.5.2 "Project Settings": updated • Section 3.1.6.3 "Project Settings": updated • Section 5.14 "Wireless UART Sample Application": added • Section 5.15 "Shell Sample Application": added • Table 23: updated
v.6	20220114	**Modifications**: • Table 2: updated • Section 3.1.3.2 "Project Settings": updated • Section 3.1.4.2 "Project Settings": updated • Section 3.1.5.2 "Project Settings": updated • Section 3.1.6.3 "Project Settings": updated • Section 3.1.7.1 "Start-up logs": updated • Section 3.2.1.1 "Run the application": updated • Section 3.3.1.4 "Other useful CLI commands": updated • Section 3.4.2.1 "Start-up logs": updated • Section 3.4.2.5 "Device reboot with the configurations stored in mflash": updated • Section 3.5.1.8 "Other useful CLI commands": updated • Section 3.6.1.3 "Set/Get Tx Power Limit": updated • Table 23: updated • Section 5.1.1 "a2dp_sink Application Execution": updated • Section 5.15.1.1 "Shell Run the application": updated • Table 24: updated • Section 5.16.2.3 "Create IoT thing, private key, and certificate for device": updated • Section 5.16.2.5 "Configure the AWS IoT endpoint": updated • Section 5.16.2.4 "Configure the AWS IoT Certificate and Private Keys": updated • Section 5.17.2 "wifi_provisioning Application Execution": updated
v.7	20220314	**Modifications**: • Section 1.3 "References": updated • Section 3 "Wi-Fi Sample Application": updated • Section 3.1 "wifi_cli Sample Application": updated • Section 3.1.1 "Run a demo with MCUXPresso IDE": updated • Section 3.1.2 "Run a demo using ARM® GCC": updated • Section 3.1.3 "Run a demo using IAR IDE": updated

		• Section 3.1.4 "Run a demo using Keil MDK/µVision": updated • Section 3.1.5 "wifi_cli Application Execution": updated • Section 3.1.5.10 "IPerf Server/Client": updated • Section 3.6 "wifi_ipv4_ipv6_echo Sample Application": Added • Section 5.15 "Shell Sample Application": Updated • Section 5.15.1 "Shell Application Execution": updated • Section 5.15.1.1 "Shell Run the application": updated
v.8	20220627	**Modifications**: • Table 5 : added u-blox modules • Section 3.1.5.1 "Start-up logs: updated": updated logs • Section 3.1.5.3 "Reset Wi-Fi module": added new command • Section 3.1.5.9 "Start Soft AP": added bandwidth NOTE for 8977 and 8801 • Section 3.1.5.13 "Wi-Fi Host sleep/wowlan": added new command • Section 3.1.5.14 "Other useful CLI commands": added commands for heap stat and 8801 ext-coex • Section 3.4.1.1 "Run the application": updated startup logs • Section 3.4.1.2 "Prerequisite Commands": added 80MHz bandwidth option • Table 11: rename to 11bgn data rate parameters • Table 12: added 11ac data rates • Section 5.15 "Shell Sample Application": added new commands for RF Test and generic HCI command execution • Section 5.16 "peripheral_beacon Sample Application": added new application • Table 24: added new acronyms
v.9	20220812	**Modifications**: • Deprecated reference of 88W8977 from the document • Section 3 "Wi-Fi Sample Applications": Updated SDK version • Section 3.1.1.2 "Project Settings": Updated screenshot • Section 3.1.3.2 "Project Settings": Updated screenshot • Section 3.1.4.2 "Project Settings": Updated screenshot • Section 3.1.5.12 "Wi-Fi Power Save": Added NOTE for WNM • Section 3.1.5.14 "Other useful CLI commands": Updated FW version, added NOTE for heap-stat, and added new commands for encryption and decryption • Section 3.2.1.1 "Run the application": Updated FW version • Section 3.3.2.1 "Start-up logs": Updated FW version • Section 3.3.2.5 "Device reboot with the configurations stored in mflash": Updated FW version • Section 3.4.1.8 "Other useful CLI commands": Updated FW version • Section 5.1.1.1 "Run the Application": Updated logs • Section 5.3.1.1 "Run the application": Updated help details • Section 5.5.1.1 "Run the application": More commands added in the help • Section 5.5.1.2 "Serial Port Profile Server Configuration": Added connection and disconnection logs • Section 5.10.1.1 "Run the application": Updated connection logs • Section 5.12 "peripheral_ipsp Sample Application": Updated logs • Section 5.15.1.1 "Shell Run the application": Updated help console logs

v.10	20230103	**Modifications**: • Table 5 : Updated tested module information • Section 3 “Wi-Fi Sample Applications”: Updated SDK version • Section 3.1.1.2 “Project Settings”: Updated module macro and screenshot • Section 3.1.2.1 “Install ARM® GCC toolchain”: Updated armgcc and cmake version • Section 3.1.2.2 “Build the application”: Updated module macro details • Section 3.1.3.2 “Project Settings”: Updated module macro and screenshot • Section 3.1.4.2 “Project Settings”: Updated module macro and screenshot • Section 3.1.5.12 “Wi-Fi Power Save”: Updated logs and NOTEs • Section 3.1.5.14 “Other useful CLI commands”: Updated FW version, wlan_info output • Section 3.2.1.1 “Run the application”: Updated FW version • Section 3.3.2.1 “Start-up logs”: Updated FW version • Section 3.3.2.5 “Device reboot with the configurations stored in mflash”: Updated FW version • Section 3.4.1.8 “Other useful CLI commands”: Updated FW version • Section 3.5.1.4 “Set/Get Active/Passive Channel List”: Updated logs • Section 3.5.1.5 “Set Channel List and Tx Power Limit”: Updated logs • Section 5.16.1 “peripheral_beacon Application Execution”: iBeacon: Output changed • Section 5.17 “audio_profile Sample Application”: Updated screenshots • Section 5.17.2.4 “Configure the AWS IoT Certificate and Private Keys “: Added new method for converting PEM file to C string • Section 5.18 “wifi_provisioning Sample Application”: Removed
v.11	20230320	**Modifications**: • Section 3 “Wi-Fi Sample Applications”: Updated SDK version • Section 3.1.5.9 “Start Soft AP”: Added command for WPA3 SAE (R3) • Section 3.1.5.14 “Set/Get Antenna Diversity Configuration”: Added new command • Section 3.1.5.15 “Set/Get Region Code”: Added new command • Section 3.1.5.16 “Set RSSI low threshold”: Added new command • Section 3.1.5.17 “Roaming with 802.11k, 802.11r, and 802.11v”: Added new command • Section 3.1.5.18 “Other useful CLI commands”: Updated crypto commands • Section 3.2 “wifi_setup Sample Application”: Modification in sample app flow • Section 3.3.2 “wifi_webconfig Application Execution”: Updated logs and added NOTE for wpa3 • Section 3.5.1.9 “Set/Get ED MAC Feature”: Updated logs • Section 3.6 “wifi_ipv4_ipv6_echo Sample Application”: Updated logs
v.12	20230727	**Modifications**: • Section 2.3 “IPerf Remote Host Setup”: Updated iPerf version • Table 5: Added macro for IW612, updated SDK version, added foot NOTE for IW612 • Section 3.1.1 "Run a Demo with MCUXpresso IDE": Updated version • Section 3.1.2 "Run a demo using ARM® GCC": Updated version

		• Section 3.1.3 "Run a demo with IAR IDE": Updated version • Section 3.1.4 "Run a demo using Keil MDK/µVision": Updated version • Section 3.1.5.1 “Start-up logs”: updated logs • Section 3.1.5.2 "Help command": updated logs • Secion 3.1.5.4 "Scan command": updated logs and added new command wlan-scan-channel-gap • Section 3.1.5.6 "Station mode (connect to AP)": new command wlan-get-signal • Section 3.1.5.7 "Start Soft AP": Added new command wlan-set-uap-hidden-ssid • Section 3.1.5.10 "Wi-Fi Host sleep/wowlan": Updated logs • Section 3.1.5.11 "Wi-Fi Cloud Keep Alive": Added new • Section 3.1.5.15 "Roaming based on RSSI event": Added new • Section 3.7 "wifi_wpa_supplicant Sample Application": Added new • Section 4.2 "Enable Host based WPA supplicant Feature for Wi-Fi application": Added new • Table 20: Added macro for IW612 • Section 6 "802.15.4 Sample Application": Added new
v.13	20231018	**Modifications**: • Table 12: Added data rate table for 802.11ax • Section 3.7 “uart_wifi_bridge Sample Application”: Added new • Section 3.8.1.6 “Wi-Fi easy connect (DPP)”: Added new • Section 3.8.1.7 “wlan-cloud-keep-alive”: Added new • Section 5.14 to 5.21: Added new • Section 5.23 “Wi-Fi CLI over Wireless UART Sample Application”: Added new
v.14	20240110	**Modifications**: • Section 1.2 "Considerations": Added entry for IW612 • Table 4: Added entry for IW612 • Table 5: Added new • Section 3.1.2.3 "Flash the application program (no debugging)": Added NOTE for i.MX RT1060 EVKC and RT1170 EVKB • Section 3.1.5.1 "Start-up logs": Updated logs • Section 3.1.5.5 "Add network profile": Updated logs • Section 3.1.5.6 "Station mode (connect to AP)": Updated wlan-add command • Section 3.1.5.7 "Start Soft AP": Updated wlan-add command • Section 3.4.1.3 "Display and Clear Received Wi-Fi Packet Count": Updated logs • Section 3.8.1.3.3 WPA3- Enterprise”: Added information for Connection Establish time • Table 10: Updated • Table 11: Updated • Table 12: Updated • Section 3.4.1.7 "Transmit standard 802.11 packets": Updated command usage and logs • Section 3.7 "uart_wifi_bridge Sample Application": Removed NOTE for IW612 and added labtool link for 8987 and IW416 • Section 3.8.1.2 "Add network profile": Updated logs • Section 3.8.1.3.1 "Enable auto reconnect option": Added new • Section: 3.8.1.3.2 "Channel State Information (CSI)": Added New

		• Section 3.8.1.3.3 "Other Security options": Added new EAP methods SIM, AKA, AKA-PRIME, FAST • Section 3.8.1.4.1 "Other Security options": Added new EAP methods SIM, AKA, AKA-PRIME, FAST • Section 3.8.1.6 "Independent Reset (IR)": Added new • Section 3.8.1.8 "wlan-cloud-keep-alive": Removed comment related to IW612
v.15	20240405	**Modifications**: • Table 4: Updated for 2.15.1 • Section 3.1.3: "Run a demo with IAR IDE": Updated IDE version • Section 3.1.5.6: "Station mode (connect to AP)": Update WPA3 Security • Section 3.1.5.7: “Start Soft AP”: Update WPA3 Security • Section 3.1.5.11: "Set/Get Antenna Diversity Configuration": Updated • Section 3.1.5.13: "Roaming based on RSSI event": Updated
v.16	20240628	**Modifications**: • Updated SDK version to 2.16.0 and foot note for AW611 • Features and Debug macros configurations restructured • Section 1.2 "Considerations": Added note for AW611 • Section 3 "Wi-Fi Sample Applications": Added note • Section 3.1.1 "Run a Demo with MCUXpresso IDE": Updated IDE version • Section 3.1.3 "Run a demo with IAR IDE": Updated IDE version • Section 3.1.4 "Run a demo using Keil MDK/µVision": Updated IDE version • Section 3.1.5.7 "Start Soft AP": Added note for PWE • Section 3.1.5.8 "IPerf Server/Client": Updated logs, Added -r option for server • Secion 3.1.5.10 "Wi-Fi Host sleep/wowlan": Added mef • Section 3.1.5.12 "Get Region Code": Removed set command and added note • Section 3.5.1.2 "Get Region Code": Removed set command and added note • Section 3.5.1.6 "Set/Get Tx Rate Configuration": Updated • Section 3.8.1.6 "Independent Reset (IR)": Added OOB • Secion 5.24.1.1 "Shell Run the application": Added command for HCI reset, independent reset, In-band reset, Out-of-band reset • Section 5.27: "Bluetooth Only firmware Download Test Procedure": Added new
v.17	20240925	**Modifications:** • Updated SDK version to 2.16.100 • Section 3.1.5.7 "Start Soft AP": Added WPA3 SAE examples • Section 3.1.5.9 "Wi-Fi Power Save": Updated idle time related info • Section: 3.8.1.3 "Station mode (connect to external AP)": Added info related to pwe along with WPA3 SAE examples • Section: 3.8.1.3.2 "Channel State Information (CSI)": Added UAP and STA mode examples • Section 3.8.1.4.1 "Other Security options": Added new WPA3 SAE examples • 3.1.5.6 "Station mode (connect to AP)": Added STA’s OWE examples • Section 3.1.5.7 "Start Soft AP": Added ACS mode info & examples • Section 3.1.5.7 "Start Soft AP": Added AP’s OWE mode info & examples • Section 3.1.5.7 "Start Soft AP": Added Hidden SSID cmd usage details

		• Section 3.1.5.8 "IPerf Server/Client": Updated usage details • Section 3.1.5.13 "Roaming based on RSSI event": updated • Section 3.1.5.17 "Roaming with 802.11k, 802.11r, and 802.11v": Added description and cmd usage examples • 3.7 "uart_wifi_bridge Sample Application": Added labtool option details. • 3.8.1.3.4 "Other Security options": Updated STA's OWE examples • 3.8.1.4 "Soft AP mode": Added ACS mode info & examples • 3.8.1.4 "Soft AP mode": Added Hidden SSID cmd usage details • Section 3.8.1.4.1 "Other Security options": Added AP's OWE examples • Section 3.8.1.9 "Wireless Location Service (WLS) using IEEE 802.11mc and IEEE 802.11az": Added • Section 5.6 to 5.9: Added PBAP and MAP profile examples
v.18	20241210	**Modifications:** • Updated SDK version to 24.12.00 • 3.8.1.10 "WLAN Offload Feature": Added ARP & NS Offload command examples • 3.8.1.3.1 "Enable auto reconnect option": Removed

1 About this Document

1.1 Purpose and Scope

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

1.2 Considerations

The i.MX RT is powered by FreeRTOS and the RTOS drivers are added to support the 88W8801, IW416, IW612 and 88W8987 NXP-based wireless modules. This document does not include NXP-based wireless modules information, i.MX RT product information, hardware interconnection, board settings, bring-up, IDE setup, SDK download, as these are covered in the UM11441. The user must have i.MX RT platform related IDE and tools installed before going through the given demo process.

Note:

The IW612/611 support is enabled in i.MX RT1170 EVKB and i.MX RT1060 EVKC.

AW611 module support is available only in i.MX RT1180 EVKA

1.3 References

Table 1: Reference Documents

Reference Type	Description
User manual	NXP – MCUXSDKGSUG - Getting Started with MCUXpresso SDK (link)
Web page	NXP - Getting Started with Wi-Fi on i.MX RT platforms (link)
User manual	NXP – UM11441 - Getting Started with NXP-based Wireless Modules and i.MX RT Platform Running on RTOS *SDK Documents available at SDK_<version>_EVK-<RT-Platform>\docs\wireless*
User manual	NXP - MCUXpresso_SDK_WLAN_Driver_Reference_Manual.pdf SDK Documents available at SDK_<version>_EVK-<RT-Platform>\docs\wireless\Wi-Fi
User manual	NXP - Hardware Rework Guide for MIMXRT1060-EVK and AW-AM457-uSD.pdf SDK Documents available at SDK_<version>_EVK-<RT-Platform>\docs\wireless\Wi-Fi.
User manual	SIG - Core Specification (link)
App NOTE	NXP - AN13296 Embedded Wi-Fi Subsystem API Specification v16 - Host driver firmware interface (link)
App NOTE	NXP - AN13612 Overview of 88W8801 Coexistence with External Radios (link)
Android Application	NXP – AwsMusicControl.apk SDK Source: SDK_<PATH>\boards\evkmimxrt1060\edgefast_bluetooth_examples\audio_profile\android_app.
Configuration file	NXP - aws_clientcredential.h SDK Source: SDK_<PATH>\ rtos\freertos\demos\include.
Configuration file	NXP - CertificateConfigurator.html SDK Source: SDK_<PATH>\ rtos\freertos\tools\certificate_configuration.
Mobile application	NXP - IoT Toolbox Android (IoT Toolbox on Google Play)IoT Toolbox on the APP Store)
Specifications	Specifications | Bluetooth® Technology Website

2 Tool Setup

2.1 Serial Console Tool Setup

The serial console tool is used to read out the demo application’s logs on the computer connected to i.MX RT EVK board.

- **Download and install the terminal emulator software such as minicom (Linux or Mac OS) or Tera Term (Windows)**
- **Use a micro-USB to USB cable to connect i.MX RT1060 EVK board to the host computer running on Linux, Mac OS or Windows.**
- **Open a terminal emulator program like minicom or Tera term.**
- **For minicom use following command and configure the below settings for serial console access:**

```
# minicom -s

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

Prior to 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:

Press **Ctrl + A** and then press **Z** key to open the Help menu

Press the **U** key to add a carriage return

2.2 Wireshark Tool Setup

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

Perform the following steps to complete the setup:

- Download package of IPerf 2.1.9 for Ubuntu 16.04 from here
- Extract the package

```
$ tar -xzf iperf-2.1.9.tar.gz
```

- Install the package using below commands

```
$ cd iperf-2.1.9
$ ./configure
$ make
$ sudo make install
```

***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 phone:

Perform the following steps to run the iPerf:

- Download the iPerf application like Magic iPerf, HE.NET Network Tools etc.
- Open the application and select the iperf2. Run the suitable from the following table.

Table 3: iPerf Commands for Mobile 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 iPV4/6 Tool Setup

Remote host setup:

- ncat - Recommended tool. Supports both IPv4 and IPv6. It is part of nmap tools. It can be found at https://nmap.org/download.html.
- nc (netcat) - Basically, the same as ncat, but a lot of antiviruses consider this a virus.
- echotool - Supports only IPv4 and only for Windows. It can be obtained from https://github.com/PavelBansky/EchoTool

Zone Index:

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

***NOTE:** The demo has only a single interface, so do not append zone ID to any address typed to the demo terminal.*

3 Wi-Fi Sample Applications

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

These Wi-Fi examples can be configured based on the Wi-Fi modules used with the help of Wi-Fi module-specific macros.

Table 4 lists the Wi-Fi module specific macros that are common to all Wi-Fi examples. Macros are available in the file ***evk<RT-Platform>_wifi_cli\source\app_config.h***

Note: *The macro configSUPPORT_STATIC_ALLOCATION is not for user configuration.*

Table 4: Macros for Wi-Fi Modules

Module	Chipset	Macro
AzureWave AW-NM191NF	88W8801	WIFI_88W8801_BOARD_AW_NM191_USD WIFI_88W8801_BOARD_AW_NM191MA
AzureWave AW-AM457	IW416	WIFI_IW416_BOARD_AW_AM457_USD WIFI_IW416_BOARD_AW_AM457MA
AzureWave AW-AM510	IW416	WIFI_IW416_BOARD_AW_AM510_USD WIFI_IW416_BOARD_AW_AM510MA
AzureWave AW-CM358	88W8987	WIFI_88W8987_BOARD_AW_CM358_USD WIFI_88W8987_BOARD_AW_CM358MA
Murata Type 2DS	88W8801	WIFI_88W8801_BOARD_MURATA_2DS_USD WIFI_88W8801_BOARD_MURATA_2DS_M2[1]
Murata Type 1XK	IW416	WIFI_IW416_BOARD_MURATA_1XK_USD WIFI_IW416_BOARD_MURATA_1XK_M2[1]
Murata 1ZM	88W8987	WIFI_88W8987_BOARD_MURATA_1ZM_USD WIFI_88W8987_BOARD_MURATA_1ZM_M2[1]
Murata Type 2EL	IW611/612	WIFI_IW61x_BOARD_MURATA_2EL_USD[1][2] WIFI_IW612_BOARD_MURATA_2EL_M2[1][2]
EVK-LILY-W131	88W8801	WIFI_88W8801_BOARD_UBX_LILY_W1_USD
EVK-MAYA-W1	IW416	WIFI_IW416_BOARD_UBX_MAYA_W1_USD
EVK-JODY-W2	88W8987	WIFI_88W8987_BOARD_UBX_JODY_W2_USD
u-blox Jody W5	AW611	WIFI_AW611_BOARD_UBX_JODY_W5_M2[3]

[1] The module operation was tested during 24.12.00 release process.

[2] The module operation support is available in the i.MX RT1170 EVKB and i.MX RT1060 EVKC

[3] The module operation is available only with i.MX RT1180 EVKA

USD=microSD interface

M2=M.2 interface

Table 5: Memory used by Wi-Fi sample application on RT1060 EVKC

Memory Region	Total Size	wifi_cli	wifi_wpa_supplicant
BOARD FLASH	8 MB	1.57 MB	2.54 MB
SRAM_OC	768 KB	385.49 KB	518.77 KB
SRAM_DTC	128 KB	132 B	132 B
SRAM_ITC	128 KB	123.616 KB	130.82 KB
BOARD_SDRAM	30 MB	0	0
NCACHE_REGION	2 MB	0	0

3.1 wifi_cli Sample Application

This section describes the *wifi_cli* application to demonstrate the CLI support to handle and enable Wi-Fi configuration for the features including scan the visible access points, create and configure the access point, connection with the access point and Throughput performance check using iPerf measurement tool. The CLI module in the application allows users to add CLIs in the application. In this sample application Wi-Fi connection manager CLIs are available.

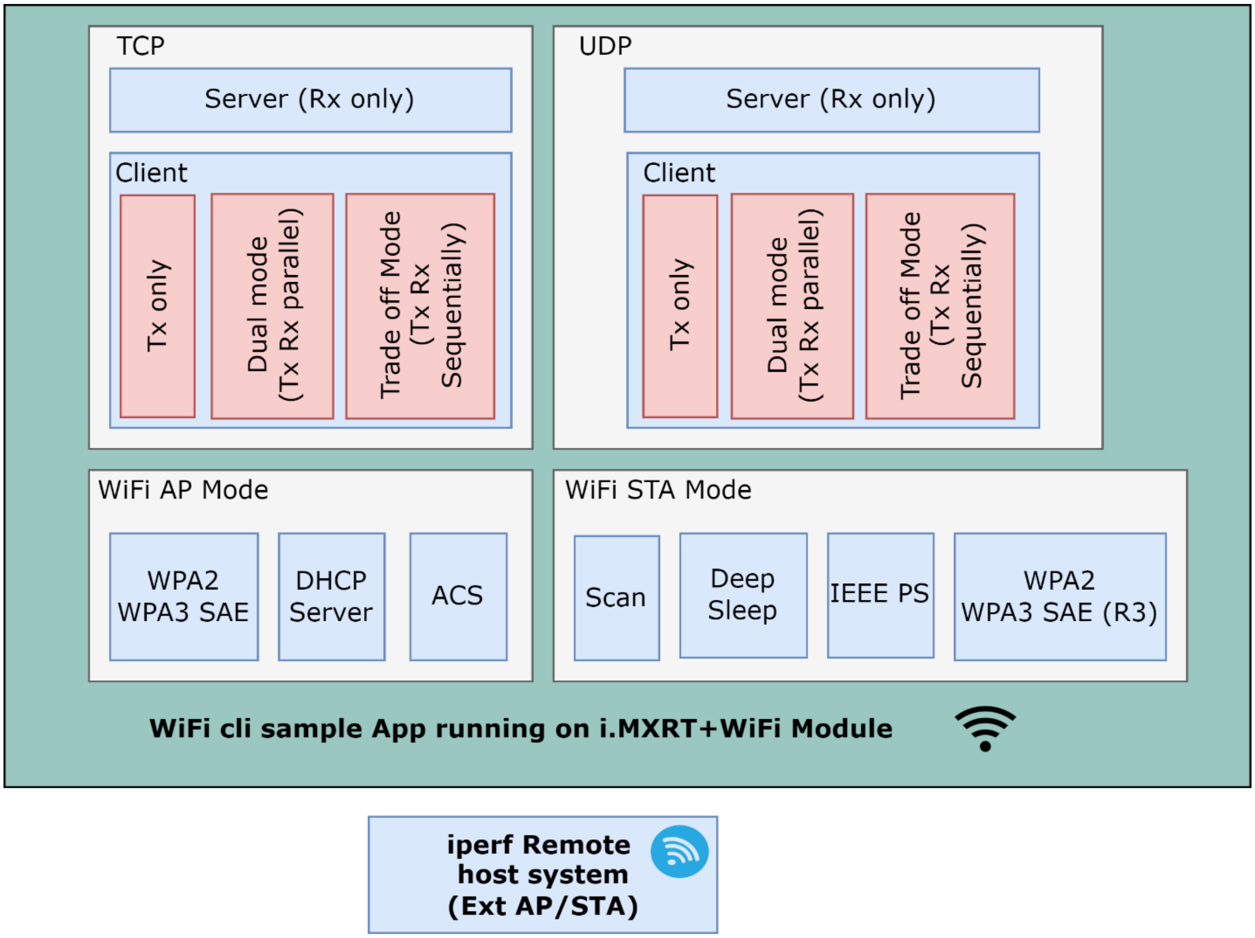

TCP
Server (Rx only)
Client
Tx only
Dual mode (Tx Rx parallel)
Trade off Mode (Tx Rx Sequentially)
UDP
Server (Rx only)
Client
Tx only
Dual mode (Tx Rx parallel)
Trade off Mode (Tx Rx Sequentially)
WiFi AP Mode
WPA2 WPA3 SAE
DHCP Server
ACS
WiFi STA Mode
Scan
Deep Sleep
IEEE PS
WPA2 WPA3 SAE (R3)
WiFi cli sample App running on i.MXRT+WiFi Module
iperf Remote host system (Ext AP/STA)

Figure 1: wifi_cli Sample Application Components

Wi-Fi and iPerf Features:

Table 6: Sample Application Features

Features	Details
Wi-Fi	Wi-Fi Soft AP mode Wi-Fi Station mode Wi-Fi Scan Wi-Fi IEEEPS power saving mode Wi-Fi deep-sleep power saving mode Wi-Fi host sleep/wowlan Wi-Fi RF Calibration Wi-Fi coexistence with external radios (for 88W8801) Wi-Fi 11r roaming 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)

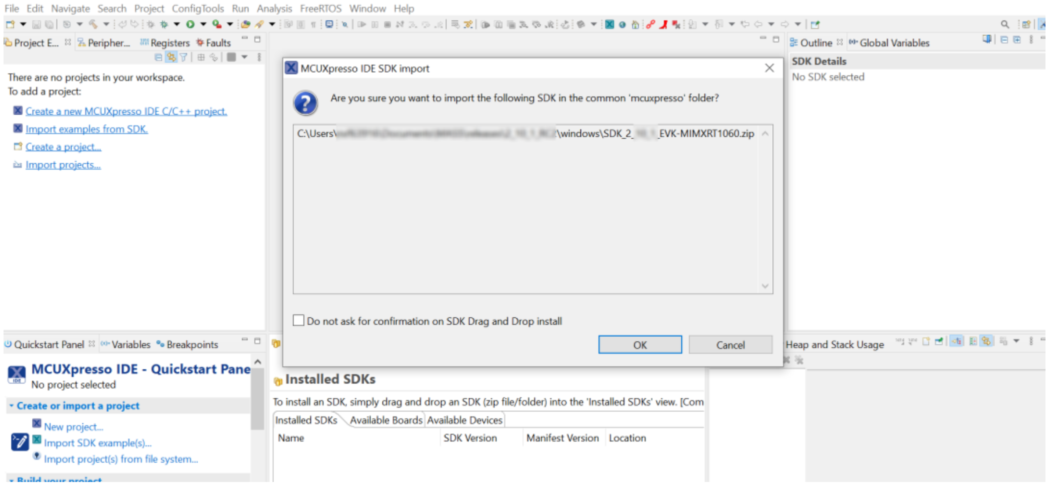
3.1.1 Run a Demo with MCUXpresso IDE

This section describes the steps to import, configure, build, debug and run the demo example through MCUXpresso IDE. MCUXpresso IDE version v11.10.0 is used for the following demo steps.

3.1.1.1 Project Import

Step 1: SDK Installation

- **Open MCUXpresso IDE.**
- **Locate the Installed SDKs tab at the bottom of the following image.**
- **Drag and drop the SDK into the Installed SDKs tab. Once done click "OK" on the pop-up window.**



File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Project E... Peripher... Registers Faults
Outline Global Variables
SDK Details
No SDK selected
Quickstart Panel Variables Breakpoints
Heap and Stack Usage
MCUXpresso IDE - Quickstart Pane
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Installed SDKs
Installed SDKs Available Boards Available Devices
OK
Cancel

There are no projects in your workspace.
To add a project:

- Create a new MCUXpresso IDE C/C++ project.
- Import examples from SDK.
- Create a project...
- Import projects...

MCUXpresso IDE SDK import

Are you sure you want to import the following SDK in the common 'mcuxpresso' folder?

C:\Users\[redacted]\windows\SDK_2_[redacted]_EVK-MIMXRT1060.zip

□ Do not ask for confirmation on SDK Drag and Drop install

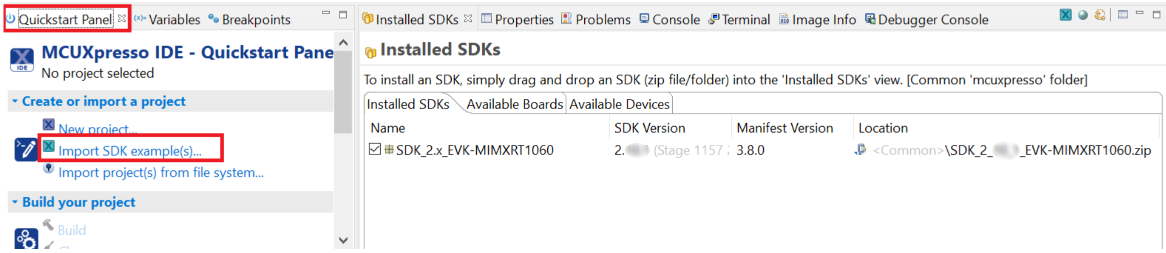
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Com

Name	SDK Version	Manifest Version	Location

Figure 2: SDK Drag and Drop in MCUXpresso

Step 2: Import an example

- **Go to the Quickstart panel and select the option Import SDK example(s).**



MCUXpresso IDE - Quickstart Pane

No project selected

Create or import a project

- New project...
- Import SDK example(s)...
- Import project(s) from file system...

Build your project

- Build

Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

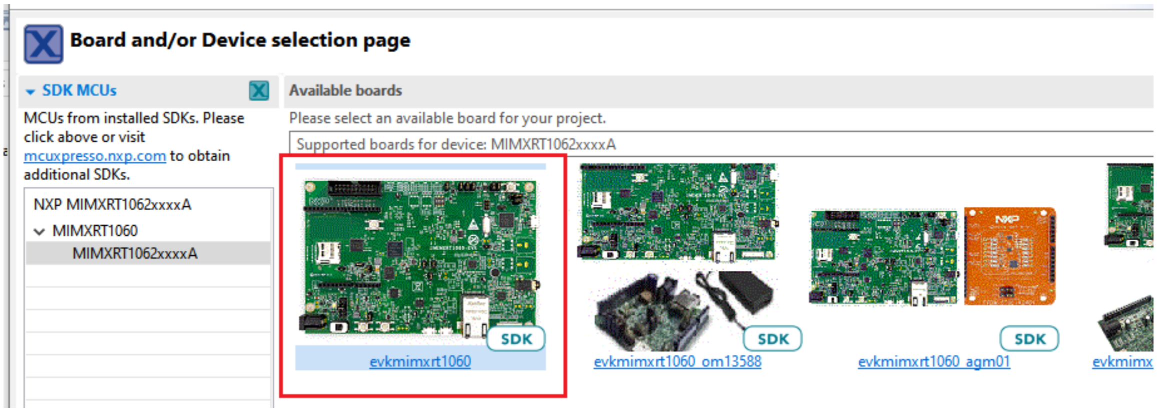
Installed SDKs | Available Boards | Available Devices

Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_EVK-MIMXRT1060	2.[redacted] (Stage 1157	3.8.0	<Common>\SDK_2_[redacted]_EVK-MIMXRT1060.zip

Figure 3: SDK Import Example in MCUXpresso

Step 3: Select EVK board.

- **Select the evaluation board.**



Board and/or Device selection page

SDK MCUs

MCUs from installed SDKs. Please click above or visit mcuxpresso.nxp.com to obtain additional SDKs.

- NXP MIMXRT1062xxxxA
  - MIMXRT1060
    - MIMXRT1062xxxxA

Available boards

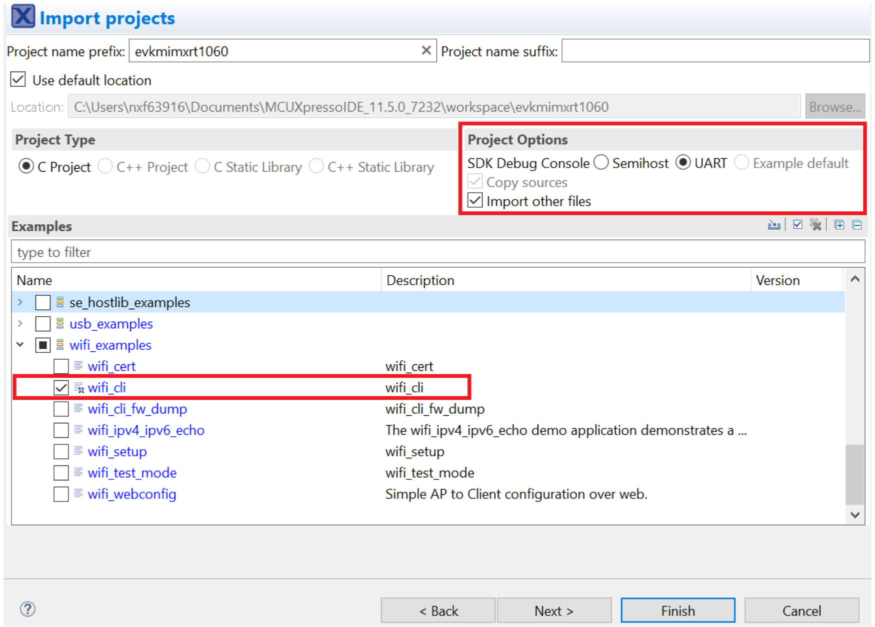
Please select an available board for your project.

Supported boards for device: MIMXRT1062xxxxA

Figure 3: Device/EVK Selection in MCUXpresso

Step 4: Select any Wi-Fi or Bluetooth example and verify default Project Options.

- **For example, select wifi_examples > wifi_cli and press Finish button to import the selected example into the workspace.**



Import projects

Project name prefix: evkmimxrt1060 Project name suffix:

☑ Use default location

Location: C:\Users\nxf63916\Documents\MCUXpressoIDE_11.5.0_7232\workspace\evkmimxrt1060 Browse...

Project Type

● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options

SDK Debug Console ○ Semihost ● UART ○ Example default

☑ Copy sources

☑ Import other files

Examples

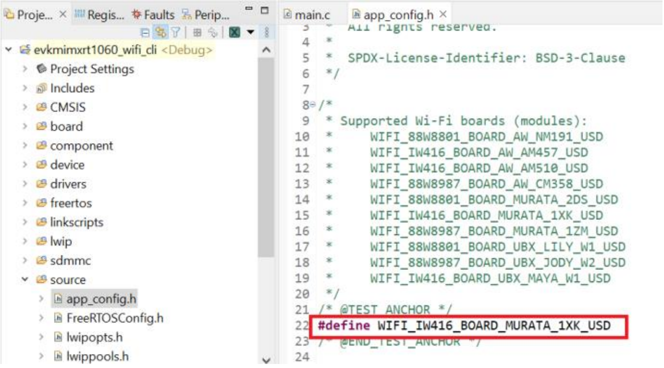
type to filter

Name	Description	Version
› □ se_hostlib_examples		
› □ usb_examples		
⌄ ■ wifi_examples		
□ wifi_cert	wifi_cert	
☑ wifi_cli	wifi_cli	
□ wifi_cli_fw_dump	wifi_cli_fw_dump	
□ wifi_ipv4_ipv6_echo	The wifi_ipv4_ipv6_echo demo application demonstrates a ...	
□ wifi_setup	wifi_setup	
□ wifi_test_mode	wifi_test_mode	
□ wifi_webconfig	Simple AP to Client configuration over web.	

Figure 4: Sample App Selection in MCUXpresso

3.1.1.2 Project Settings

- **By default, the project is configured to use the WIFI_IW416_BOARD_MURATA_1XK_USD Wi-Fi module based on IW416 chipset. Modify the value to match the module on your setup to include and compile the desired driver, components and application(s).**
- **The file "*app_config.h*" from the source folder is used for the macro definitions.**
- **Refer to Table 4 for the list of macros for Wi-Fi modules.**



```
 *  All rights reserved.
 *
 *  SPDX-License-Identifier: BSD-3-Clause
 */

/*
 * Supported Wi-Fi boards (modules):
 *     WIFI_88W8801_BOARD_AW_NM191_USD
 *     WIFI_IW416_BOARD_AW_AM457_USD
 *     WIFI_IW416_BOARD_AW_AM510_USD
 *     WIFI_88W8987_BOARD_AW_CM358_USD
 *     WIFI_88W8801_BOARD_MURATA_2DS_USD
 *     WIFI_IW416_BOARD_MURATA_1XK_USD
 *     WIFI_88W8987_BOARD_MURATA_1ZM_USD
 *     WIFI_88W8801_BOARD_UBX_LILY_W1_USD
 *     WIFI_88W8987_BOARD_UBX_JODY_W2_USD
 *     WIFI_IW416_BOARD_UBX_MAYA_W1_USD
 */
/* @TEST_ANCHOR */
#define WIFI_IW416_BOARD_MURATA_1XK_USD
/* @END_TEST_ANCHOR */
```

Figure 5: Wi-Fi Module Selection in MCUXpresso

3.1.1.3 Build the Application

- **To build the application, go to the Quickstart panel and select Build, or select the Build icon in the main toolbar.**

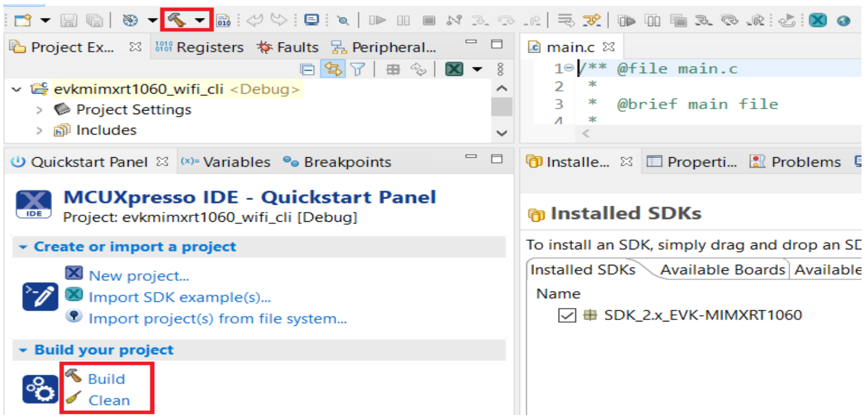
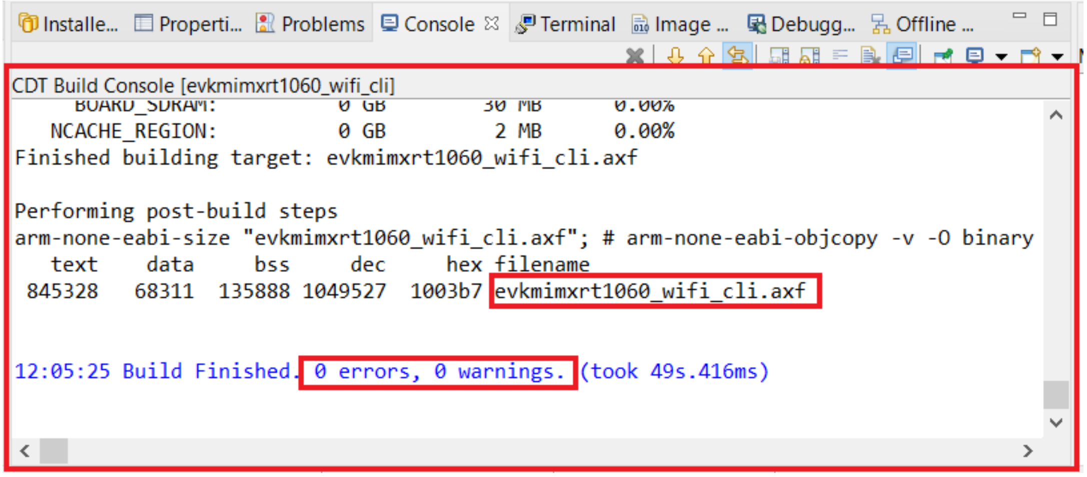


Figure 6: Application Build in MCUXpresso

- **Verify the build result (success or fail) on the console window.**



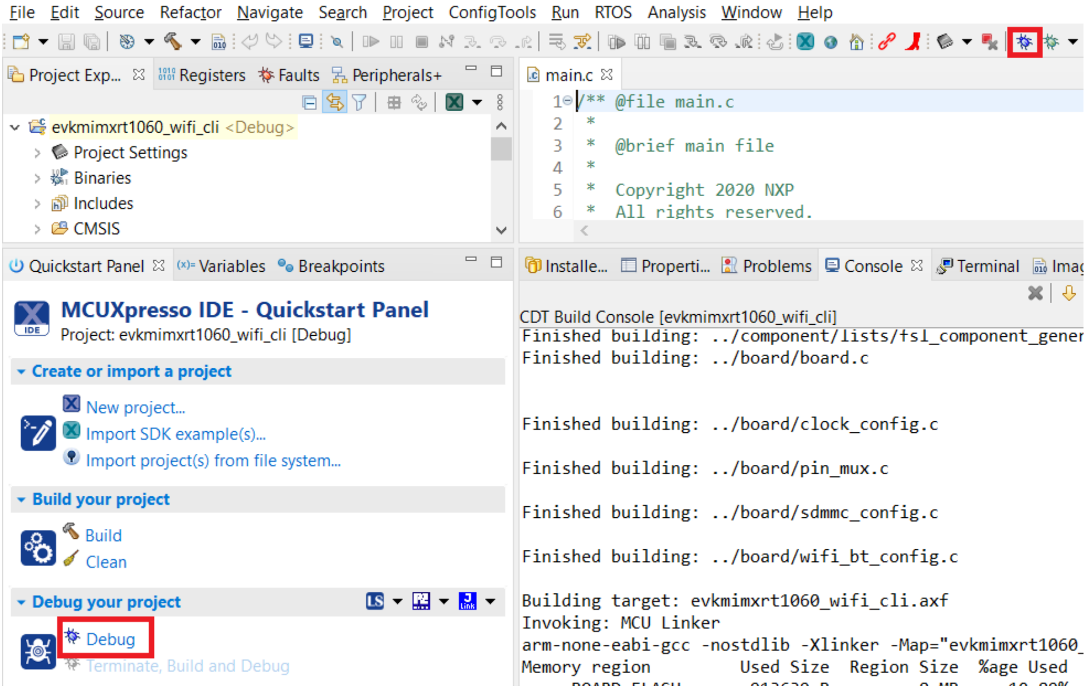
```
CDT Build Console [evkmimxrt1060_wifi_cli]
    BOARD_SDRAM:          0 GB        30 MB      0.00%
   NCACHE_REGION:          0 GB         2 MB      0.00%
Finished building target: evkmimxrt1060_wifi_cli.axf

Performing post-build steps
arm-none-eabi-size "evkmimxrt1060_wifi_cli.axf"; # arm-none-eabi-objcopy -v -O binary
   text    data     bss     dec     hex filename
 845328   68311  135888 1049527  1003b7 evkmimxrt1060_wifi_cli.axf


12:05:25 Build Finished. 0 errors, 0 warnings. (took 49s.416ms)
```

Figure 7: Build Messages in MCUXpresso

3.1.1.4 Run the Application in Debug Mode

Please follow these steps to run the application in debug mode.

- **Initiate the application debug using the debug icon in the toolbar or go to the Quickstart panel and select Debug.**

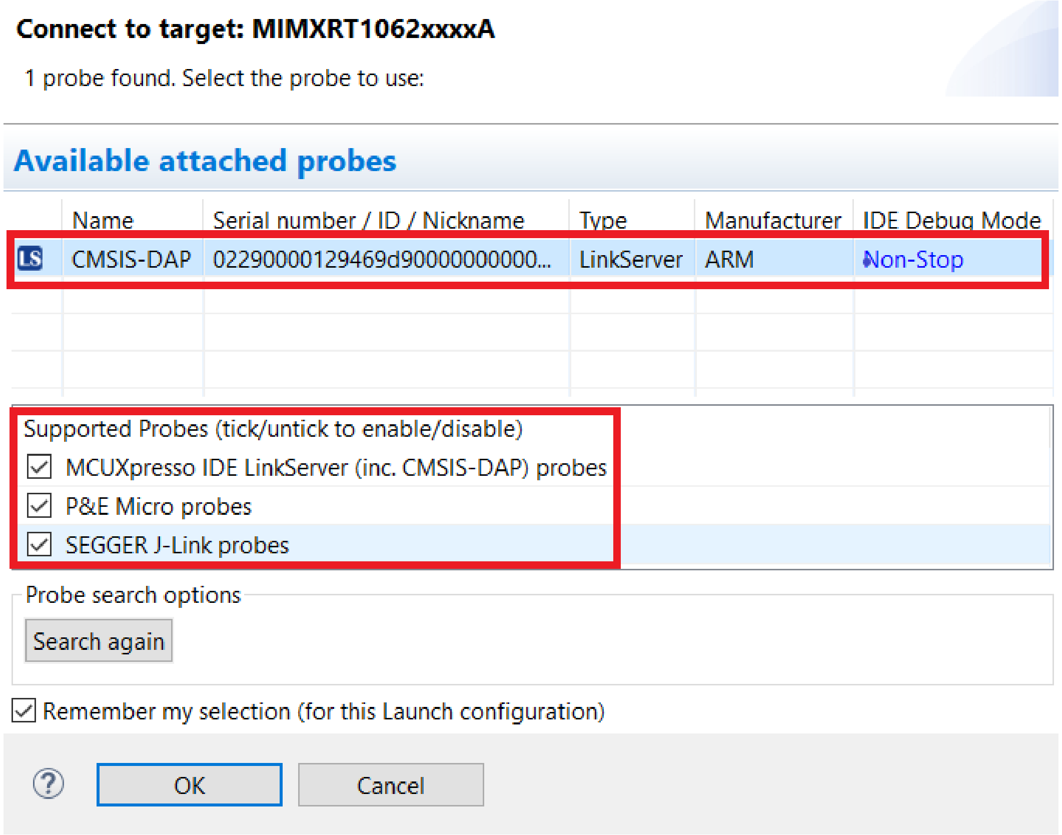


```
/** @file main.c
 *
 *  @brief main file
 *
 *  Copyright 2020 NXP
 *  All rights reserved.
```

```
CDT Build Console [evkmimxrt1060_wifi_cli]
Finished building: ../component/lists/fsl_component_gener
Finished building: ../board/board.c


Finished building: ../board/clock_config.c

Finished building: ../board/pin_mux.c

Finished building: ../board/sdmmc_config.c

Finished building: ../board/wifi_bt_config.c

Building target: evkmimxrt1060_wifi_cli.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -Xlinker -Map="evkmimxrt1060_
Memory region         Used Size  Region Size  %age Used
```

Figure 8: Initiate Debug in MCUXpresso

- **Select the associated emulator probe for the first time as illustrated below and press OK.**



Connect to target: MIMXRT1062xxxxA

1 probe found. Select the probe to use:

Available attached probes

	Name	Serial number / ID / Nickname	Type	Manufacturer	IDE Debug Mode
LS	CMSIS-DAP	02290000129469d90000000000...	LinkServer	ARM	Non-Stop

Supported Probes (tick/untick to enable/disable)

☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes

☑ P&E Micro probes

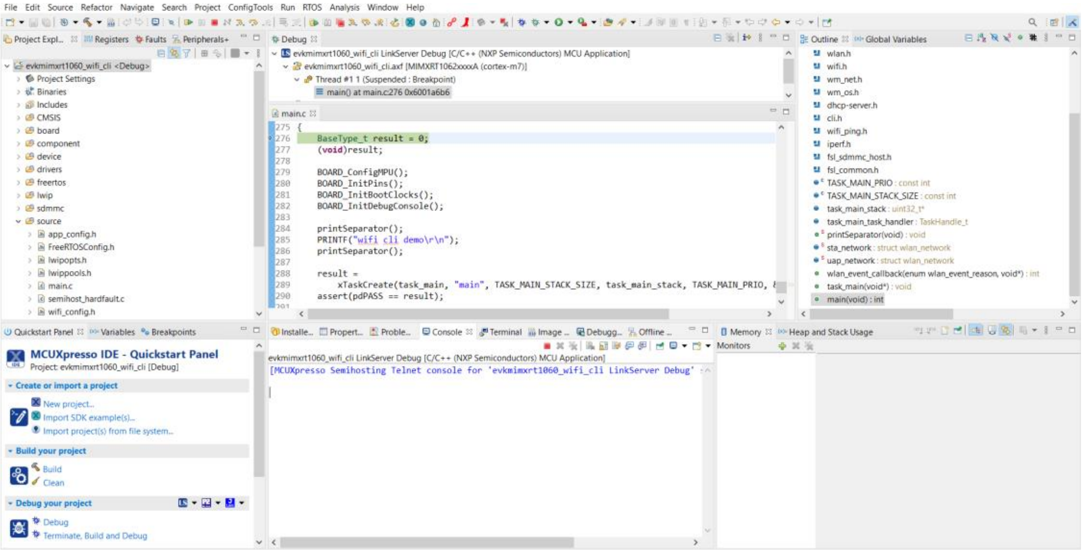
☑ SEGGER J-Link probes

Probe search options

☑ Remember my selection (for this Launch configuration)

Figure 9: Emulator Probe Selection in MCUXpresso

- **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. Press Resume to start the application. To debug the application, use the step into, step over and step return buttons. To end the debugging session, use the Terminate button.**



```
evkmimxrt1060_wifi_cli LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application]
  evkmimxrt1060_wifi_cli.axf [MIMXRT1062xxxxA (cortex-m7)]
    Thread #1 1 (Suspended : Breakpoint)
      main() at main.c:276 0x6001a6b6
```

```
{
    BaseType_t result = 0;
    (void)result;

    BOARD_ConfigMPU();
    BOARD_InitPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    printSeparator();
    PRINTF("wifi cli demo\r\n");
    printSeparator();

    result =
        xTaskCreate(task_main, "main", TASK_MAIN_STACK_SIZE, task_main_stack, TASK_MAIN_PRIO, &
    assert(pdPASS == result);
```

```
wlan.h
wifi.h
wm_net.h
wm_os.h
dhcp-server.h
cli.h
wifi_ping.h
iperf.h
fsl_sdmmc_host.h
fsl_common.h
TASK_MAIN_PRIO : const int
TASK_MAIN_STACK_SIZE : const int
task_main_stack : uint32_t*
task_main_task_handler : TaskHandle_t
printSeparator(void) : void
sta_network : struct wlan_network
uap_network : struct wlan_network
wlan_event_callback(enum wlan_event_reason, void*) : int
task_main(void*) : void
main(void) : int
```

```
evkmimxrt1060_wifi_cli LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application]
[MCUXpresso Semihosting Telnet console for 'evkmimxrt1060_wifi_cli LinkServer Debug'
```

MCUXpresso IDE - Quickstart Panel

Project: evkmimxrt1060_wifi_cli [Debug]

- Create or import a project
  - New project...
  - Import SDK example(s)...
  - Import project(s) from file system...
- Build your project
  - Build
  - Clean
- Debug your project
  - Debug
  - Terminate, Build and Debug

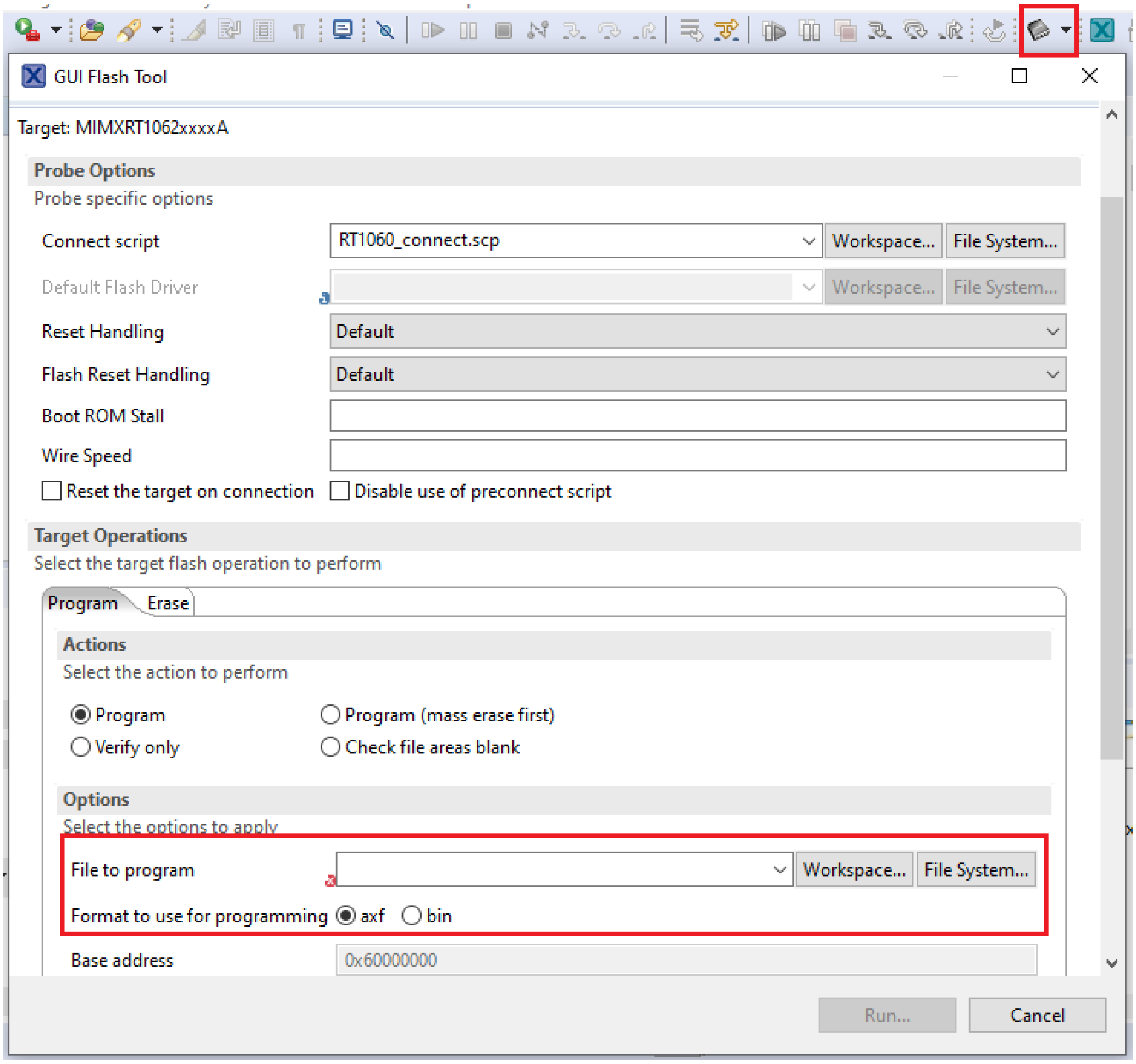
Figure 10: Application Debugging in MCUXpresso

3.1.1.5 Flash the Application Program (no debugging)

Please use the following steps to flash the application program.

To flash the required binaries, select the **GUI Flash Tool icon** in the toolbar as shown in the figure below. The GUI Flash Tool can be used to flash pre-build binary or locally compiled binary with **.axf* or **.bin* format. The path to the locally compiled binary is the following.

${workspace_loc}\evkmimxrt1060_wifi_cli\Debug\evkmimxrt1060_wifi_cli.axf



GUI Flash Tool

Target: MIMXRT1062xxxxA

Probe Options

Probe specific options

Connect script: RT1060_connect.scp

Default Flash Driver:

Reset Handling: Default

Flash Reset Handling: Default

Boot ROM Stall:

Wire Speed:

□ Reset the target on connection □ Disable use of preconnect script

Target Operations

Select the target flash operation to perform

Program | Erase

Actions

Select the action to perform

● Program ○ Program (mass erase first)

○ Verify only ○ Check file areas blank

Options

Select the options to apply

File to program:

Format to use for programming ● axf ○ bin

Base address: 0x60000000

Figure 11: Binary Flashing in MCUXpresso

***NOTE:** Please refer to section 3.1.5 to view the output on the console once the application is executed.*

3.1.2 Run a demo using ARM® GCC

This section describes the steps to configure the command line ARM® GCC tools to build and run demo applications. The wifi_cli application is used as an example, yet 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. Please refer to MCUXSDKGSUG for more details on ARM GCC toolchain setup.

3.1.2.1 Install ARM® GCC toolchain

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

- **Download the toolchain for Linux x86_64 system from the Link (package *Linux x86_64 tarball*).**
- **Create a directory at the location of your choice:**

```
$ mkdir toolchain-dir
```

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

```
$ cp <download_path>/arm-gnu-toolchain-12.2.rel1-x86_64-arm-none-eabi.tar.xz
toolchain-dir/
$ cd toolchain-dir/
$ tar -xvf arm-gnu-toolchain-12.2.rel1-x86_64-arm-none-eabi.tar.xz
```

- **Export the ARMGCC_DIR variable using the following command:**

```
$ export ARMGCC_DIR=<absolute-path>/toolchain-dir/ arm-gnu-toolchain-12.2.rel1-
x86_64-arm-none-eabi/
```

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

```
$ export PATH=$PATH:<absolute-path>/toolchain-dir/arm-gnu-toolchain-12.2.rel1-
x86_64-arm-none-eabi /bin/
```

- **Download and install *cmake* (source and binary distribution) using the Link for Linux system.**
- **Extract the source distribution and copy it to the */usr/share/* directory**

```
$ tar -zxf cmake-3.25.1.tar.gz
$ sudo cp -rf cmake-3.25.1 /usr/share/cmake-3.25
```

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

```
$ tar -zxf cmake-3.25.1-Linux-x86_64.tar.gz
$ sudo cp cmake-3.25.1-Linux-x86_64/bin/* /usr/bin/
```

3.1.2.2 Build the application

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

- **Go to the *armgcc* directory of the application**

```
$ cd <SDK-top-dir>/boards/evkmimxrt1060/wifi_examples/wifi_cli/armgcc/
```

Modify the configuration for a wireless module

- **By default, the project is configured to use the *WIFI_IW416_BOARD_MURATA_1XK_USD* Wi-Fi module based on IW416 chipset. Modify the value in the file "*<SDK_EXAMPLE_PATH>/source/app_config.h*" to match the module on your setup to include and compile the desired driver, components and application(s).**
- **Build the binary**

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

- **Generate *wifi_cli.bin* using following command**

```
arm-none-eabi-objcopy flexspi_nor_debug/wifi_cli.elf -O binary
flexspi_nor_debug/wifi_cli.bin
```

NOTE: *Please refer to* MCUXSDKGSUG *for more details to debug the application using GDB.*

3.1.2.3 Flash the application program (no debugging)

NOTE: *Step provided in this section will not be useful for i.MX RT1170 EVKB and i.MX RT1060 EVKC*

This section provides the steps to flash the binary on the i.MX RT board:

- **Connect the board to the Linux host system. The board shows as a Mass storage device in the Linux host system.**
- **Copy the application binary (*wifi_cli.bin*) to the Mass storage device and wait for the start of the binary download on the board.**

```
$ sudo cp flexspi_nor_debug/wifi_cli.bin /media/<user>/RT1060-EVK/
```

- **The board stops showing as Mass storage device and appears again once the flash process has completed. If any error occurs during the flashing, the *FAIL.txt* file is generated and stored in the Mass storage device.**
- **To access the device using the serial console please refer to 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: *Please refer to section 3.1.5 to view the actual output on the console once the application is executed.*

3.1.3 Run a demo with IAR IDE

This section provides the steps to open, configure, build, debug and run the demo example using IAR Embedded Workbench IDE. The instructions and illustrations refer to IAR version 9.60.1.

3.1.3.1 Open the project workspace

To open the wifi_cli project available in the SDK, double-click the project workspace file named *wifi_cli.eww* stored at the following location.
<install_dir>\boards\evkmimxrt1060\wifi_examples\wifi_cli\iar\wifi_cli.eww

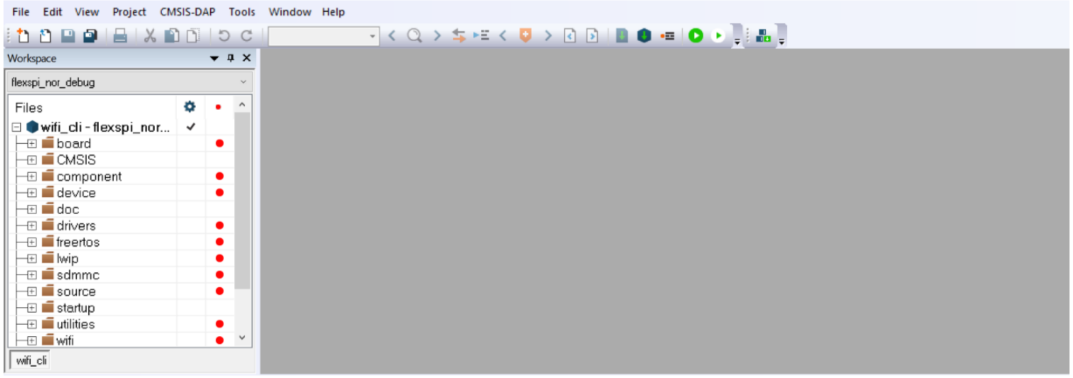
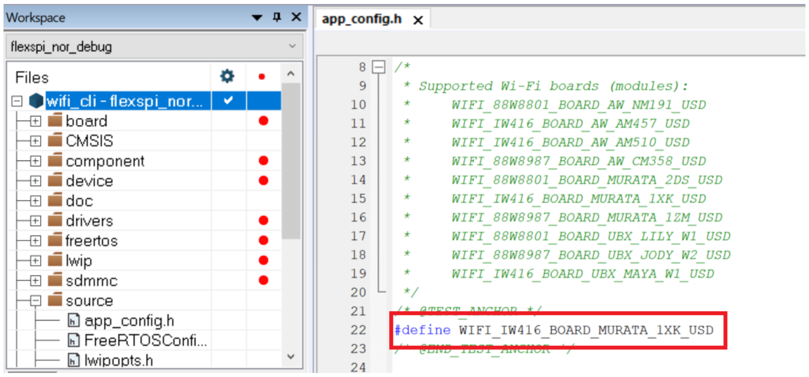


Figure 12: Open Project in IAR

3.1.3.2 Project Settings

- **By default, the project is configured to use the *WIFI_IW416_BOARD_MURATA_1XK_USD* Wi-Fi module based on IW416 chipset. Modify the value to match the module on your setup to include and compile the desired driver, components and application(s).**
- **The file "*app_config.h*" from the source folder is used for the macro definitions**
- **Refer to Table 4 for the list of macros for Wi-Fi modules.**

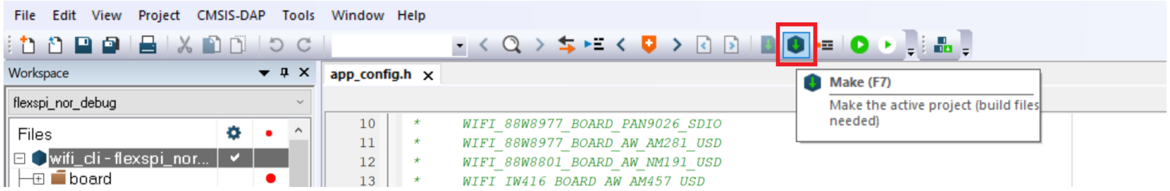


```
/*
 * Supported Wi-Fi boards (modules):
 *     WIFI_88W8801_BOARD_AW_NM191_USD
 *     WIFI_IW416_BOARD_AW_AM457_USD
 *     WIFI_IW416_BOARD_AW_AM510_USD
 *     WIFI_88W8987_BOARD_AW_CM358_USD
 *     WIFI_88W8801_BOARD_MURATA_2DS_USD
 *     WIFI_IW416_BOARD_MURATA_1XK_USD
 *     WIFI_88W8987_BOARD_MURATA_1ZM_USD
 *     WIFI_88W8801_BOARD_UBX_LILY_W1_USD
 *     WIFI_88W8987_BOARD_UBX_JODY_W2_USD
 *     WIFI_IW416_BOARD_UBX_MAYA_W1_USD
 */
/* @TEST_ANCHOR */
#define WIFI_IW416_BOARD_MURATA_1XK_USD
/* @END_TEST_ANCHOR */
```

Figure 13: Wi-Fi Module Selection in IAR

3.1.3.3 Build the application

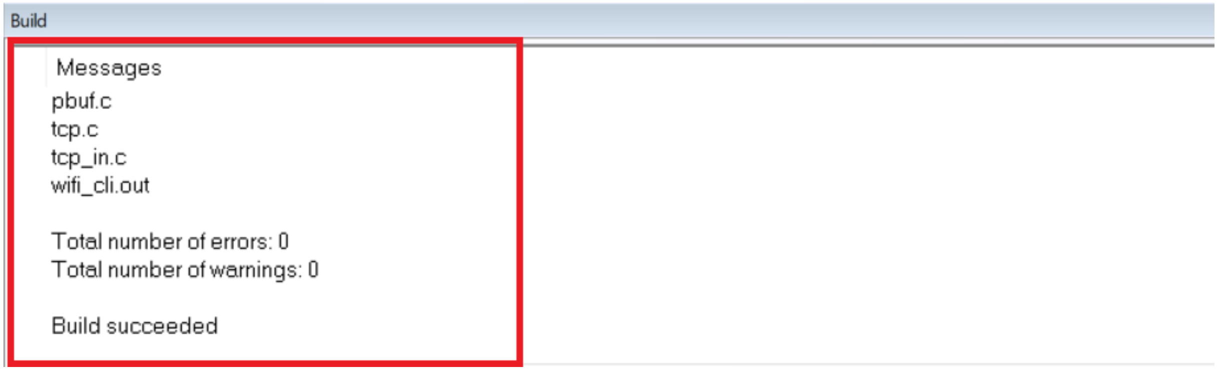
- **To build the *wifi_cli* application, press the Make icon as illustrated below.**



```
10  *    WIFI_88W8977_BOARD_PAN9026_SDIO
11  *    WIFI_88W8977_BOARD_AW_AM281_USD
12  *    WIFI_88W8801_BOARD_AW_NM191_USD
13  *    WIFI_IW416_BOARD_AW_AM457_USD
```

Figure 14: Application Build in IAR

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



```
Messages
pbuf.c
tcp.c
tcp_in.c
wifi_cli.out

Total number of errors: 0
Total number of warnings: 0

Build succeeded
```

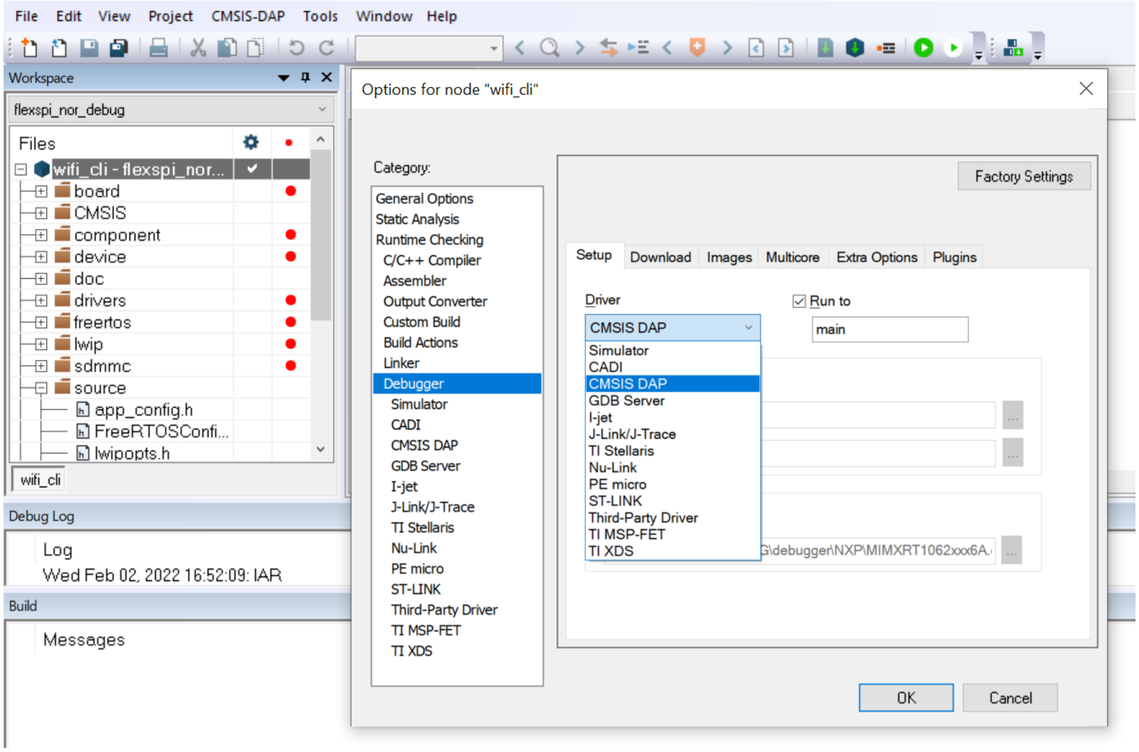
Figure 15: Build Message in IAR

3.1.3.4 Run the application in debug mode

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

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

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



Options for node "wifi_cli"

Category: Debugger

Setup tab — Driver: CMSIS DAP; ☑ Run to: main

Figure 16: Debugger Selection in IAR

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

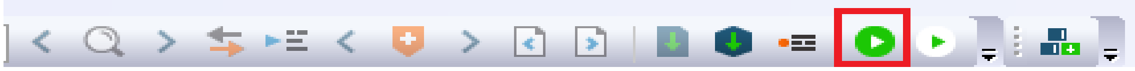
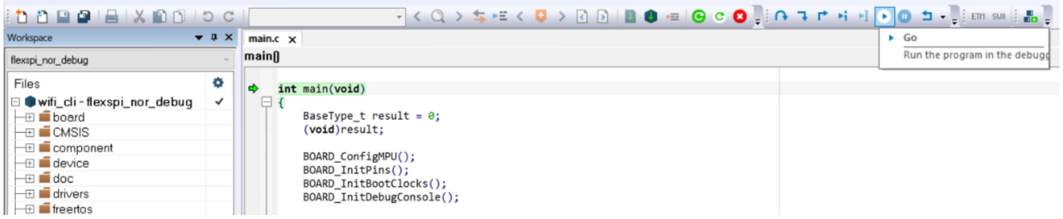


Figure 17: Initiate Debug in IAR

- **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.**



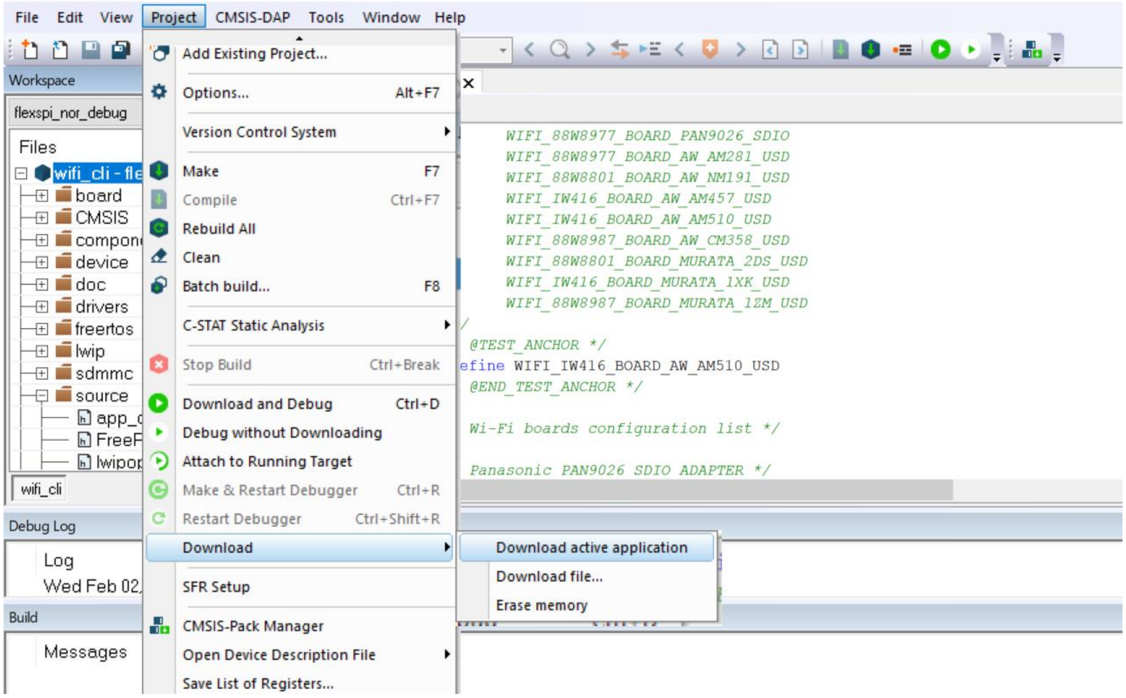
```
int main(void)
{
    BaseType_t result = 0;
    (void)result;

    BOARD_ConfigMPU();
    BOARD_InitPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();
```

Figure 18: Application Debugging in IAR

3.1.3.5 Flash the application program (no debugging)

Please use the following steps 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.**



```
    WIFI_88W8977_BOARD_PAN9026_SDIO
    WIFI_88W8977_BOARD_AW_AM281_USD
    WIFI_88W8801_BOARD_AW_NM191_USD
    WIFI_IW416_BOARD_AW_AM457_USD
    WIFI_IW416_BOARD_AW_AM510_USD
    WIFI_88W8987_BOARD_AW_CM358_USD
    WIFI_88W8801_BOARD_MURATA_2DS_USD
    WIFI_IW416_BOARD_MURATA_1XK_USD
    WIFI_88W8987_BOARD_MURATA_1ZM_USD
/
 @TEST_ANCHOR */
efine WIFI_IW416_BOARD_AW_AM510_USD
 @END_TEST_ANCHOR */

 Wi-Fi boards configuration list */

 Panasonic PAN9026 SDIO ADAPTER */
```

Figure 19: Binary Flashing in IAR

NOTE: *Refer to section 3.1.5 to view the output on the console once the application is executed.*

3.1.4 Run a demo using Keil MDK/µVision

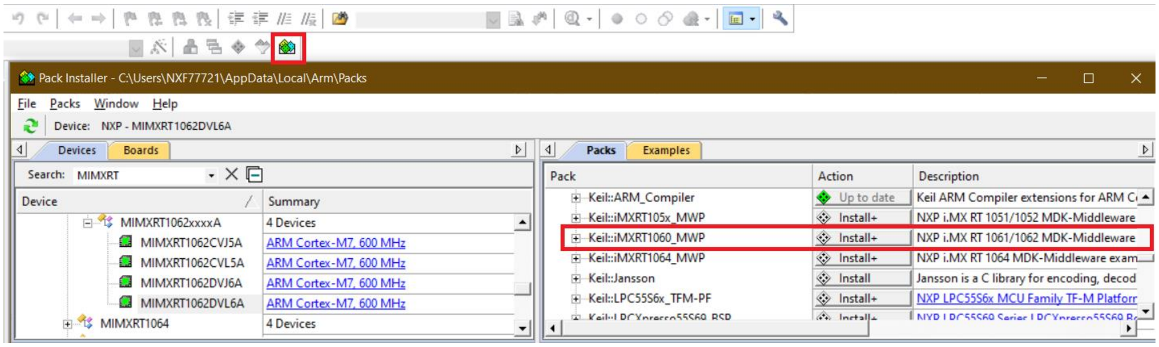
This section details the steps to open, configure, build, debug and run demo example through Keil IDE. The Keil version used in the following instructions is V5.39.0.0.

***NOTE:** For Bluetooth demo applications Keil MDK/ µVision IDE is not supported.*

3.1.4.1 Install CMSIS device pack

Following the installation of the MDK tools, install the CMSIS device packs so you can use the debug functionality on your device. The CMSIS device packs include the memory map information, register definitions and flash programming algorithms. The following steps install the MIMXRT106x CMSIS pack.

- **Click on the Pack Installer icon in the toolbar, look for iMXRT1060_MWP in the Packs tab. Press Install in the Action column.**



Pack Installer - C:\Users\NXF77721\AppData\Local\Arm\Packs

Device: NXP - MIMXRT1062DVL6A

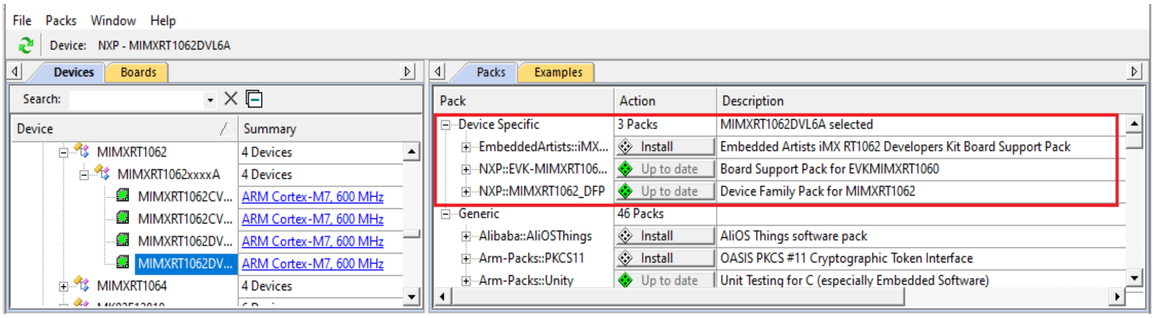
Search: MIMXRT

Device	Summary
MIMXRT1062xxxxA	4 Devices
MIMXRT1062CVJ5A	ARM Cortex-M7, 600 MHz
MIMXRT1062CVL5A	ARM Cortex-M7, 600 MHz
MIMXRT1062DVJ6A	ARM Cortex-M7, 600 MHz
MIMXRT1062DVL6A	ARM Cortex-M7, 600 MHz
MIMXRT1064	4 Devices

Pack	Action	Description
Keil::ARM_Compiler	Up to date	Keil ARM Compiler extensions for ARM C
Keil::iMXRT105x_MWP	Install+	NXP i.MX RT 1051/1052 MDK-Middleware
Keil::iMXRT1060_MWP	Install+	NXP i.MX RT 1061/1062 MDK-Middleware
Keil::iMXRT1064_MWP	Install+	NXP i.MX RT 1064 MDK-Middleware exam
Keil::Jansson	Install	Jansson is a C library for encoding, decod
Keil::LPC55S6x_TFM-PF	Install+	NXP LPC55S6x MCU Family TF-M Platforr

Figure 20: Install Packages using Pack Installer in Keil

- **When the installation is complete, Up to date is displayed in the Action column. Verify that the Board Support Pack (BSP) and Device Family Pack (DFP) are both listed in the Device > Packs tab.**



Device: NXP - MIMXRT1062DVL6A

Search:

Device	Summary
MIMXRT1062	4 Devices
MIMXRT1062xxxxA	4 Devices
MIMXRT1062CV...	ARM Cortex-M7, 600 MHz
MIMXRT1062CV...	ARM Cortex-M7, 600 MHz
MIMXRT1062DV...	ARM Cortex-M7, 600 MHz
MIMXRT1062DV...	ARM Cortex-M7, 600 MHz
MIMXRT1064	4 Devices

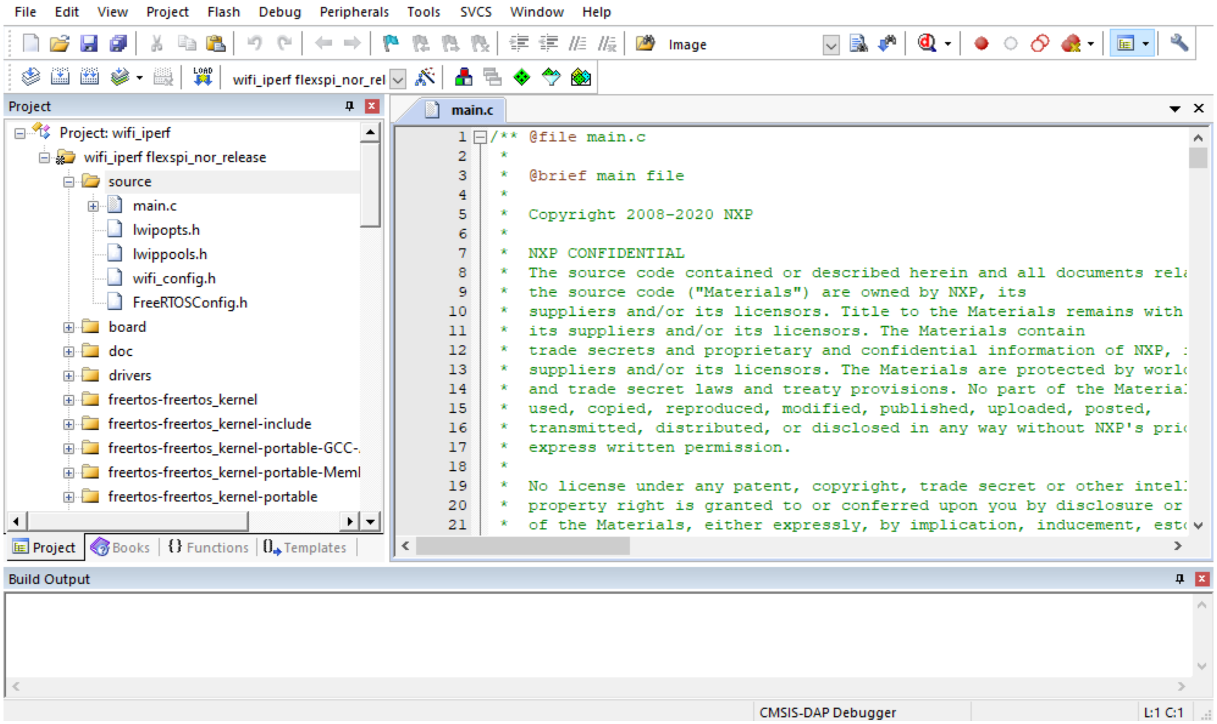
Pack	Action	Description
Device Specific	3 Packs	MIMXRT1062DVL6A selected
EmbeddedArtists::iMX...	Install	Embedded Artists iMX RT1062 Developers Kit Board Support Pack
NXP::EVK-MIMXRT106...	Up to date	Board Support Pack for EVKMIMXRT1060
NXP::MIMXRT1062_DFP	Up to date	Device Family Pack for MIMXRT1062
Generic	46 Packs	
Alibaba::AliOSThings	Install	AliOS Things software pack
Arm-Packs::PKCS11	Install	OASIS PKCS #11 Cryptographic Token Interface
Arm-Packs::Unity	Up to date	Unit Testing for C (especially Embedded Software)

Figure 21: DFP Verification in Pack Installer in Keil

3.1.4.2 Open the project workspace

To open the *wifi_cli* project: double-click the project workspace file *wifi_cli.uvprojx* located at the following path: *<install_dir>\boards\evkmimxrt1060\wifi_examples\wifi_cli\mdk\wifi_cli.uvprojx*

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

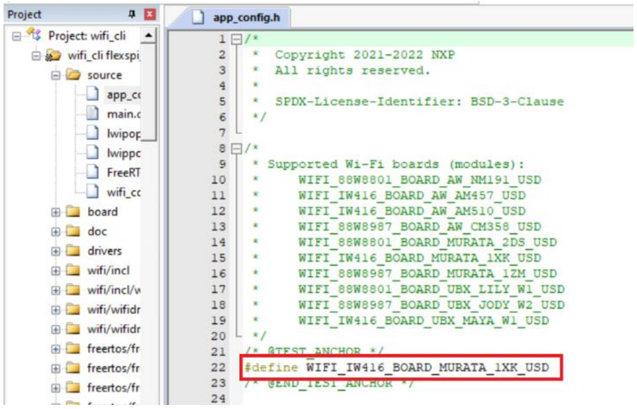


```
/** @file main.c
 *
 *  @brief main file
 *
 *  Copyright 2008-2020 NXP
 *
 *  NXP CONFIDENTIAL
 *  The source code contained or described herein and all documents rela
 *  the source code ("Materials") are owned by NXP, its
 *  suppliers and/or its licensors. Title to the Materials remains with
 *  its suppliers and/or its licensors. The Materials contain
 *  trade secrets and proprietary and confidential information of NXP, :
 *  suppliers and/or its licensors. The Materials are protected by worlc
 *  and trade secret laws and treaty provisions. No part of the Materia:
 *  used, copied, reproduced, modified, published, uploaded, posted,
 *  transmitted, distributed, or disclosed in any way without NXP's pric
 *  express written permission.
 *
 *  No license under any patent, copyright, trade secret or other intel:
 *  property right is granted to or conferred upon you by disclosure or
 *  of the Materials, either expressly, by implication, inducement, estc
```

Figure 22: Open Project in Keil

3.1.4.3 Project Settings

- **By default, the project is configured to use the *WIFI_IW416_BOARD_MURATA_1XK_USD* Wi-Fi module based on IW416 chipset. Modify the value to match the module on your setup to include and compile the desired driver, components and application(s).**
- **The file "*app_config.h*" from the source folder is used for the macro definitions**
- **Refer to Table 4 for the list of macros for Wi-Fi modules.**



```
/*
 *  Copyright 2021-2022 NXP
 *  All rights reserved.
 *
 *  SPDX-License-Identifier: BSD-3-Clause
 */

/*
 * Supported Wi-Fi boards (modules):
 *     WIFI_88W8801_BOARD_AW_NM191_USD
 *     WIFI_IW416_BOARD_AW_AM457_USD
 *     WIFI_IW416_BOARD_AW_AM510_USD
 *     WIFI_88W8987_BOARD_AW_CM358_USD
 *     WIFI_88W8801_BOARD_MURATA_2DS_USD
 *     WIFI_IW416_BOARD_MURATA_1XK_USD
 *     WIFI_88W8987_BOARD_MURATA_1ZM_USD
 *     WIFI_88W8801_BOARD_UBX_LILY_W1_USD
 *     WIFI_88W8987_BOARD_UBX_JODY_W2_USD
 *     WIFI_IW416_BOARD_UBX_MAYA_W1_USD
 */
/* @TEST_ANCHOR */
#define WIFI_IW416_BOARD_MURATA_1XK_USD
/* @END_TEST_ANCHOR */
```

Figure 23: Wi-Fi Module Selection in Keil

3.1.4.4 Build the application

- **To build the *wifi_cli* application, press the Build or Rebuild icons.**

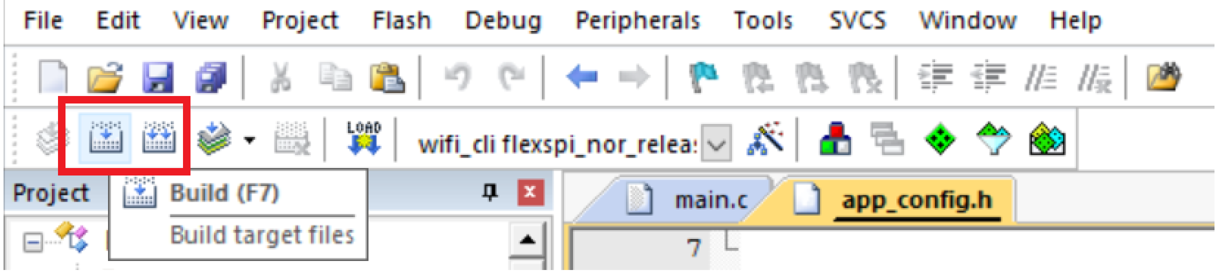
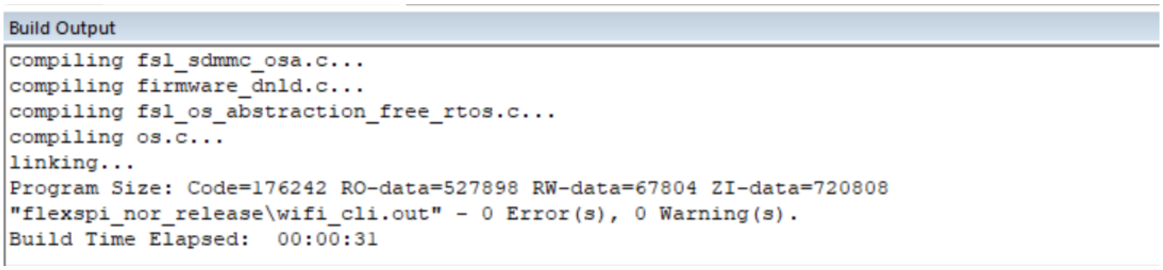


Figure 24: Application Build in Keil

- **Verify the build progress in the Build Output window.**


Build Output

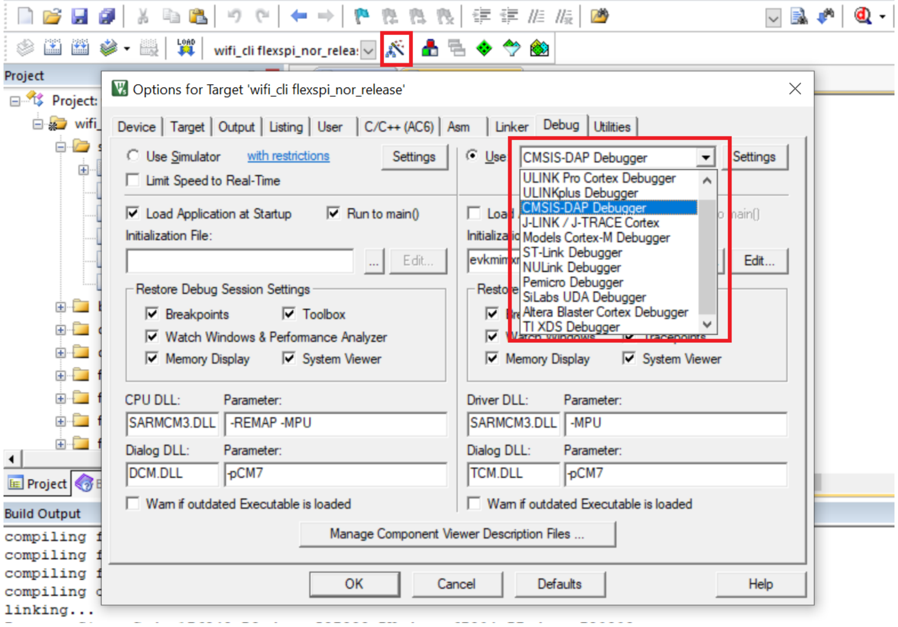
```
compiling fsl_sdmmc_osa.c...
compiling firmware_dnld.c...
compiling fsl_os_abstraction_free_rtos.c...
compiling os.c...
linking...
Program Size: Code=176242 RO-data=527898 RW-data=67804 ZI-data=720808
"flexspi_nor_release\wifi_cli.out" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:31
```

Figure 25: Build Message in Keil

3.1.4.5 Run the application in debug mode

Please refer to following steps to run the application in debug mode.

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



Options for Target 'wifi_cli flexspi_nor_release'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

○ Use Simulator with restrictions Settings

☐ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU

Dialog DLL: DCM.DLL Parameter: -pCM7

☐ Warn if outdated Executable is loaded

● Use: CMSIS-DAP Debugger Settings

- ULINK Pro Cortex Debugger
- ULINKplus Debugger
- CMSIS-DAP Debugger
- J-LINK / J-TRACE Cortex
- Models Cortex-M Debugger
- ST-Link Debugger
- NULink Debugger
- Pemicro Debugger
- SiLabs UDA Debugger
- Altera Blaster Cortex Debugger
- TI XDS Debugger

☐ Load

Initialization File: evkmim

☑ Memory Display ☑ System Viewer

Driver DLL: SARMCM3.DLL Parameter: -MPU

Dialog DLL: TCM.DLL Parameter: -pCM7

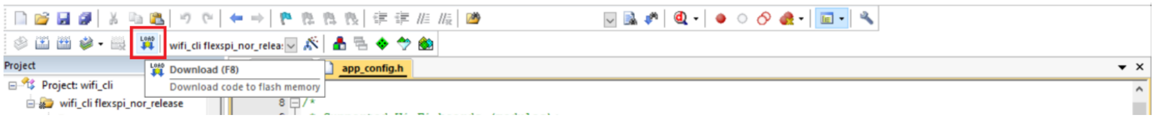
☐ Warn if outdated Executable is loaded

Manage Component Viewer Description Files ...

OK | Cancel | Defaults | Help

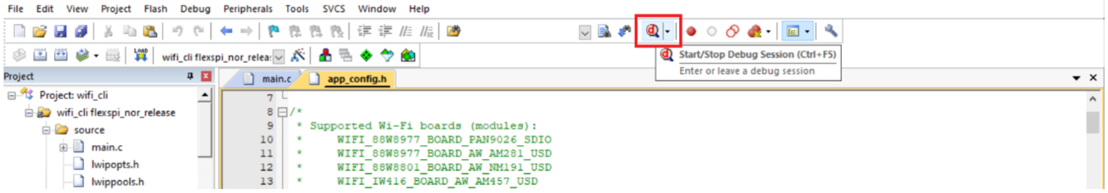
Figure 26: Debugger Selection in Keil

- **To start the application debug, click on the LOAD icon to download the application on the board then click on the Start/Stop Debug Session icon in the toolbar.**



wifi_cli flexspi_nor_relea
Download (F8)
Download code to flash memory
app_config.h
Project
Project: wifi_cli
wifi_cli flexspi_nor_release

Figure 27: Load the application

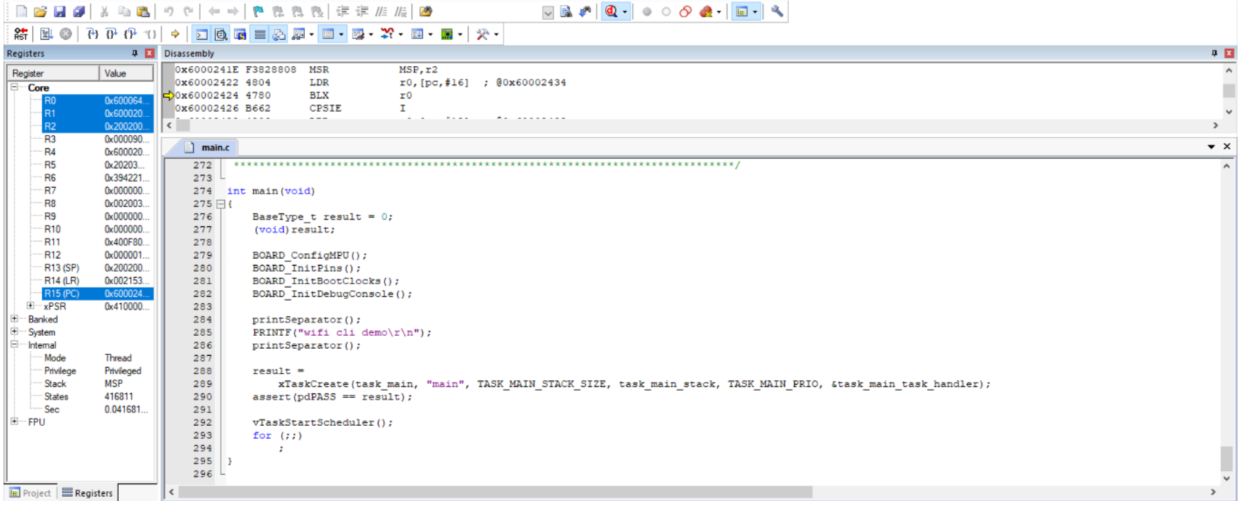


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
wifi_cli flexspi_nor_relea
Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session
Project
Project: wifi_cli
wifi_cli flexspi_nor_release
source
main.c
lwipopts.h
lwippools.h
main.c
app_config.h

```
/*
 * Supported Wi-Fi boards (modules):
 *     WIFI_88W8977_BOARD_PAN9026_SDIO
 *     WIFI_88W8977_BOARD_AW_AM281_USD
 *     WIFI_88W8801_BOARD_AW_NM191_USD
 *     WIFI_IW416_BOARD_AW_AM457_USD
```

Figure 28: Initiate Debug in Keil

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



Register	Value
Core	
R0	0x600064...
R1	0x600020...
R2	0x200200...
R3	0x000090...
R4	0x600020...
R5	0x20203...
R6	0x394221...
R7	0x000000...
R8	0x002003...
R9	0x000000...
R10	0x000000...
R11	0x400F80...
R12	0x000001...
R13 (SP)	0x200200...
R14 (LR)	0x002153...
R15 (PC)	0x600024...
xPSR	0x410000...
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	416811
Sec	0.041681...
FPU	

```
0x6000241E F3828808 MSR          MSP,r2
0x60002422 4804     LDR          r0,[pc,#16]  ; @0x60002434
0x60002424 4780     BLX          r0
0x60002426 B662     CPSIE        I
```

```
 ****************************************************************************/

int main(void)
{
    BaseType_t result = 0;
    (void)result;

    BOARD_ConfigMPU();
    BOARD_InitPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    printSeparator();
    PRINTF("wifi cli demo\r\n");
    printSeparator();

    result =
        xTaskCreate(task_main, "main", TASK_MAIN_STACK_SIZE, task_main_stack, TASK_MAIN_PRIO, &task_main_task_handler);
    assert(pdPASS == result);

    vTaskStartScheduler();
    for (;;)
        ;
}
```

Figure 29: Application Debugging 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.**

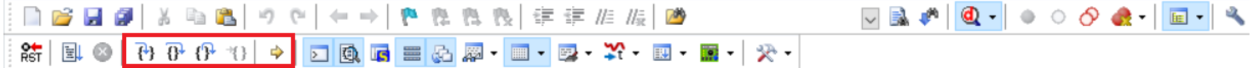
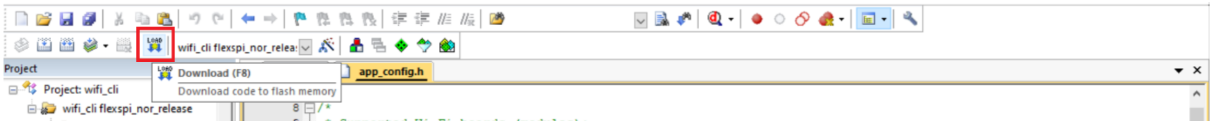


Figure 30: Application Debugging Features in Keil

3.1.4.6 Flash the application program (no debugging)

Please refer following steps to flash the application program.

- **Click on the Download icon in the toolbar to flash the required binary file.**



wifi_cli flexspi_nor_relea
Download (F8)
Download code to flash memory
Project
Project: wifi_cli
wifi_cli flexspi_nor_release
app_config.h

Figure 31: Binary Flashing in Keil

NOTE: *Please refer to section 3.1.5 to view the output on the console once the application is executed.*

3.1.5 *wifi_cli* Application Execution

3.1.5.1 Start-up logs

The following logs can be observed on the console once the devices (i.MX RT1060 EVK board and NXP-based Wireless module) are up and running and it shows that Wi-Fi module is ready for the operations. This section describes the available Wi-Fi commands, press Enter for the command prompt.

```
========================================
========================================
wifi cli demo
========================================
Initialize CLI
========================================
Initialize WLAN Driver
========================================
MAC Address: 70:66:55:26:8A:F3
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
ENHANCED WLAN CLIs are initialized
========================================
CLIs Available:
========================================

help
wlan-reset
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-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-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-sta-filter  <filter mode> [<mac address list>]
wlan-roaming <0/1> <rssi_threshold>
wlan-multi-mef <ping/arp/multicast/del> [<action>]
wlan-send-hostcmd
wlan-set-uap-bandwidth <1/2/3> 1:20 MHz 2:40MHz 3:80MHz
```

```
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-ft-roam <bssid> <channel>
wlan-set-antcfg <ant mode> [evaluate_time]
wlan-get-antcfg
wlan-scan-channel-gap <channel_gap_value>
wlan-get-regioncode
wlan-11d-enable <sta/uap> <0/1>
wlan-tx-ampdu-prot-mode <mode>
wlan-rssi-low-threshold <threshold_value>
wlan-get-signal
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-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-get-txpwrlimit <subband>
wlan-set-txpwrlimit
wlan-set-chanlist-and-txpwrlimit
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
wlan-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>
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6
address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
========================================
```

3.1.5.2 Help command

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

```
# help

help
wlan-reset
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-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
.
.
.
```

3.1.5.3 Reset Wi-Fi module

The reset command is used to reset and re-initialize the Wi-Fi module.

```
# wlan-reset
MAC Address: 70:66:55:7B:AC:84

# ========================================
app_cb: WLAN: received event 11
========================================
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
CLIs Available:
.
.
.
```

3.1.5.4 Scan command

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

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

# 1 network found:
38:E6:0A:C6:1A:EC  "nxp" Infra
        channel: 11
        rssi: -57 dBm
        security: WPA2
        WMM: YES
        802.11V: YES
        802.11W: NA
```

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

# 1 network found:
38:E6:0A:C6:1A:EC  "nxp" Infra
        channel: 11
        rssi: -54 dBm
        security: WPA2
        WMM: YES
        802.11V: YES
        802.11W: NA
```

Set time gap between two consecutive channels scan

Command usage:

```
# wlan-scan-channel-gap
Invalid arguments
```

```
Usage:
wlan-scan-channel-gap <scan_gap_value>
scan_gap_value: [2,500]
```

Set time gap to 5 sec

```
# wlan-scan-channel-gap 5
```

3.1.5.5 Add network profile

Before adding a network profile for Soft AP and Station mode, please check command usage.

```
# wlan-add
Usage:
For Station interface
  For DHCP IP Address assignment:
    wlan-add <profile_name> ssid <ssid> [wpa2 <psk/psk-sha256> <secret>] [mfpc
<1> mfpr <0>]
      If using WPA2 security, set the PMF configuration as mentioned above.
    wlan-add <profile_name> ssid <ssid> <owe_only> mfpc 1 mfpr 1
      If using OWE only security, always set the PMF configuration.
    wlan-add <profile_name> ssid <ssid> [wpa3 sae <secret> [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> <secret>] [owe_only] [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>]]
    [owe_only ]
    [mfpc <0/1>] [mfpr <0/1>]
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
```

3.1.5.6 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’s SSID and Passphrase in argument shown below:

```
# wlan-add abc ssid nxp wpa2 psk 12345678
Added “abc”
```

Connect to the AP network using the saved network profile:

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

```
# =========================================
app_cb: WLAN: received event 1
========================================
app_cb: WLAN: authenticated to network
========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [nxp]
IPv4 Address: [192.168.43.35]
IPv6 Address: Link-Local   :    FE80::7266:55FF:FE7B:AD1A (Preferred)
IPv6 Address: Global       :    2409:4041:E8A:AFD9:7266:55FF:FE7B:AD1A
(Preferred)
```

NOTE: *Once connected to the AP the console output will show Client successfully connected to AP with ssid “nxp” and got ip address “192.168.43.35” from AP.*

Get signal information of connected External AP

```
# wlan-get-signal
        BeaconLast      Beacon Average  Data Last       Data Average
RSSI    -32             -32             -33             -33
SNR     58              58              57              57
NF      -90             -90             -90             -90
```

WPA2 Station disconnection (from AP)

Disconnect from the AP network profile:

```
# wlan-disconnect

========================================
app_cb: WLAN: received event 9
========================================
app_cb: disconnected
```

Remove the saved network profile:

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

WPA3 Security

NOTE: *For WPA3 default mode is set to pwe 2 (both hunting-and-pecking loop and hash-to-element enabled)*

Usage for pwe and tr

```
SAE mechanism for PWE derivation
```

```
# 0 = hunting-and-pecking loop only (default without password identifier)
# 1 = hash-to-element only (default with password identifier)
# 2 = both hunting-and-pecking loop and hash-to-element enabled
```

Transition Disable indication

```
# 0 = transition mode (allow to connect WPA2-Personal)
# 1 = disable transition mode ((i.e., disable WPA2-Personal = WPA-PSK and only
allow SAE to be used))
```

WPA3 SAE (R1)

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 pwe 0 mfpc 1 mfpr 1
Added “nxp_test_1”
```

WPA3 SAE (R3)

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 pwe 1 mfpc 1 mfpr 1
Added “nxp_test_1”
```

OWE

Always set mfpc and mfpr to 1.

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

Connect to the AP network using the saved network profile:

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

========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [WPA3_AP]
IPv4 Address: [192.168.10.2]
```

NOTE: *Once connected to the AP the console output will show Client successfully connected to AP with ssid "*WPA3_AP*" and got ip address "192.168.10.2" from AP. For WPA3 R3, above configuration will also work.*

WPA3 Station disconnection (from AP)

Disconnect from the AP network profile:

```
# wlan-disconnect

========================================
app_cb: WLAN: received event 9
========================================
app_cb: disconnected
```

Remove the saved network profile:

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

3.1.5.7 Start Soft AP

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

WPA2 Security

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

Added "xyz"
```

WPA3 Security

Note: *Default value of pwe is 0 for Soft AP*

Default value of tr is 0 for Soft AP

WPA3 SAE (R1)

```
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 pwe 0 mfpc 1 mfpr 1
```

WPA3 SAE (R3)

```
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 pwe 1 mfpc 1 mfpr 1
```

WPA3 SAE (R3), with capability set to 11AX

```
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 pwe 1 mfpc 1 mfpr 1 capa 11ax
```

WPA3 SAE (R3), Transition Disable set

```
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 pwe 1 tr 1 mfpc 1 mfpr 1
```

OWE

Always set mfpc and mfpr to 1.

```
# wlan-add xyz ssid oweNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 36 owe_only mfpc 1 mfpr 1
```

Set ACS mode

The Automatic Channel Selection (ACS) mode can be enabled while adding the profile using wlan-add command. When channel parameter is set as 0 then it enables ACS mode.

Default value for ACS band is 0.

<acs_band> usage

```
# 0 = 2.4GHz
# 1 = 5GHz
```

AP with wpa2 psk security configured with 5 GHz ACS mode

```
# wlan-add xyz ssid NXPAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
channel 0 acs_band 1 wpa2 psk 12345678
```

AP with wpa2 psk security configured with 2.4 GHz ACS mode

```
# wlan-add xyz ssid NXPAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
channel 0 acs_band 0 wpa2 psk 12345678
```

Set Wi-Fi bandwidth

The following command is used to set Wi-Fi bandwidth (20MHz or 40MHz or 80MHz):

***NOTE:** Default bandwidth is set to 40MHz if not set by following command.*

***NOTE:** For 88W8801, default bandwidth is set to 20MHz and following command is not available.*

Command Usage:

```
# wlan-set-uap-bandwidth
Usage: wlan-set-uap-bandwidth <1/2/3>
Error: Specify 1 to set bandwidth 20MHz or 2 for 40MHz or 3 for 80MHz
```

Set bandwidth:

```
# wlan-set-uap-bandwidth 1
bandwidth set successfully
```

Start the AP using saved network profile:

```
# wlan-start-network xyz
[wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is
on.

========================================
app_cb: WLAN: received event 14
========================================
app_cb: WLAN: UAP Started
========================================
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:

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

Get the associated clients list:

```
# wlan-get-uap-sta-list
```

```
Number of STA = 1

STA 1 information:
=====================
MAC Address: 38:E6:0A:C6:1A:EC
Power mfg status: power save
Rssi : -58 dBm
```

Get the IP and MAC information for the associated clients:

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

SSID broadcast configuration:

User can control SSID IE configuration using this command.

It has 3 modes:

0: When user wants to enable SSID broadcast (default)

1: When user wants to disable SSID name(ASCII 0) and SSID length (Length = 0)

2: When user wants to disable only the SSID name (ASCII 0)

Command usage:

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

Set SSID broadcast control

```
# wlan-set-uap-hidden-ssid 1
SSID broadcast control set successfully
```

Stop Soft AP

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

3.1.5.8 IPerf Server/Client

The sample application implements the protocol used by iPerf performance measurement tool. The performance is measured between a single i.MX RT+NXP-based Wireless module and a computer running the iPerf tool. The instructions in this guide use an i.MX RT1060 EVK board. Yet the same steps apply to other i.MX RT products. The following figures show the setup overview to run the iPerf performance test.

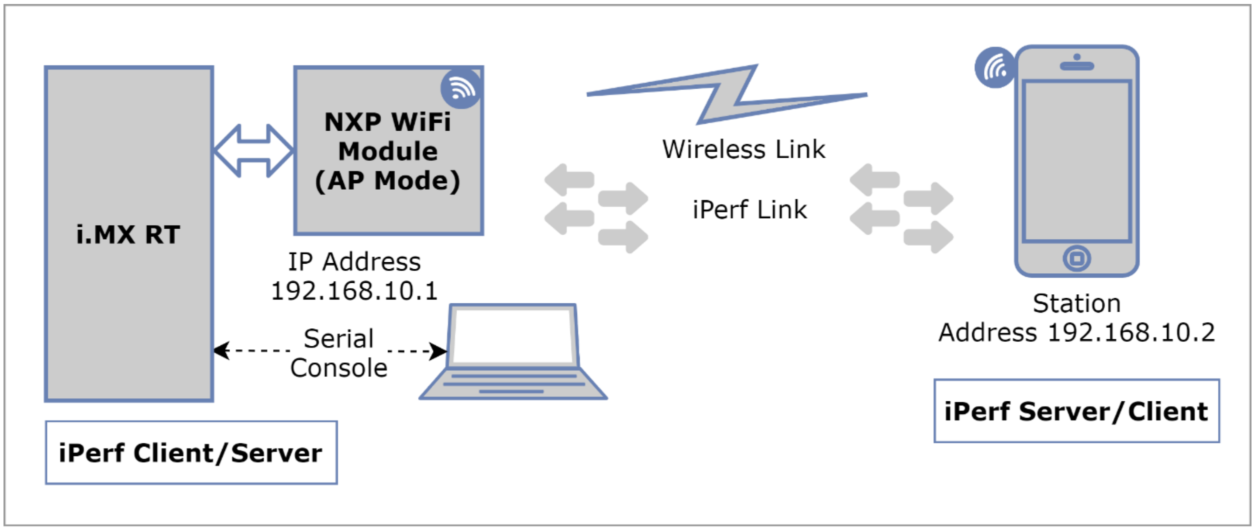


Figure 32: Hardware Setup for iPerf performance test with Soft AP Mode

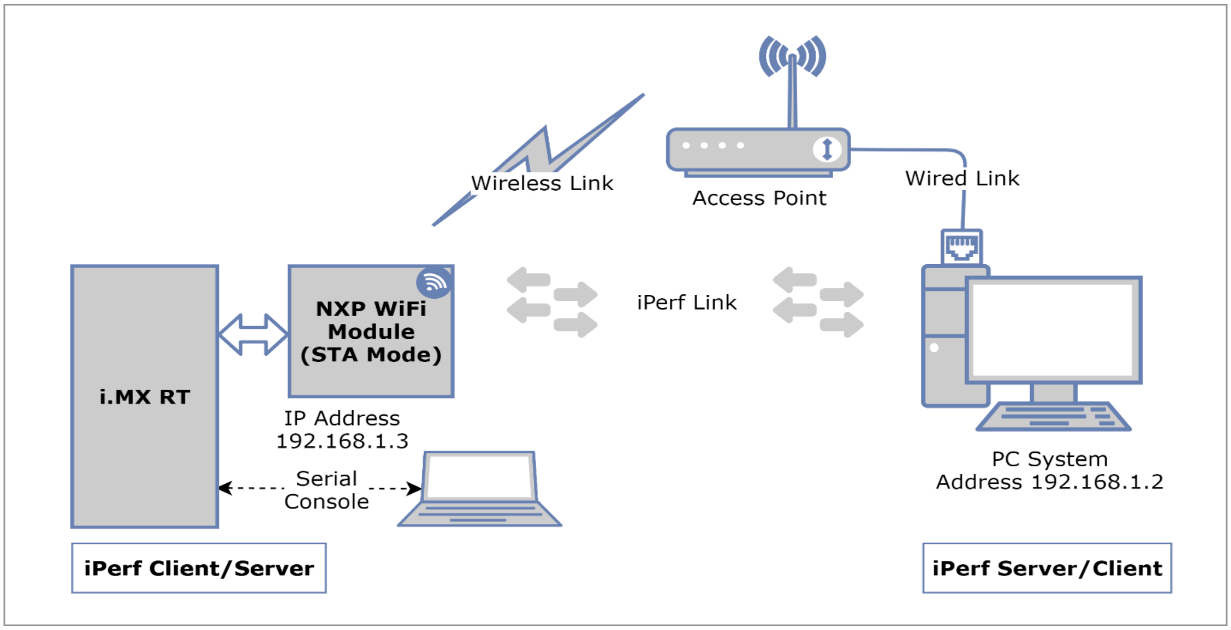


Figure 33: Hardware Setup for iPerf performance test with Station Mode

NOTE: *Please refer to the 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
           -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 Linux and Mobile application commands refer* Table 2 and Table 3 *respectively from section 2.3*

iPerf TCP

Start IPerf server:

```
# iperf -s

# IPERF initialization successful
New TCP client (settings flags 0x0)

-------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 36874
 Bytes Transferred XXXX
 Duration (ms) 10130
 Bandwidth (Mbitpsec) XX
```

Start IPerf Client (Tx Only):

```
# iperf -c 192.168.10.2

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

Start IPerf Server (Tx and Rx individual):

```
# iperf -s -r
IPERF initialization successful

# New TCP client (settings flags 0xc0010078)
                                            client requested transmission after
end of test
-------------------------------------------------
```

```
 TCP_DONE_SERVER (RX)
 Local address : 192.168.1.1  Port 5001
 Remote address : 192.168.1.2  Port 50496
 Bytes Transferred xxxx
 Duration (ms) 10177
 Bandwidth (Mbitpsec) xx

-------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.1.1  Port 54237
 Remote address : 192.168.1.2  Port 5001
 Bytes Transferred xxxx
 Duration (ms) 10001
 Bandwidth (Mbitpsec) xx
```

Start IPerf Client (Tx and Rx simultaneous):

```
# iperf -c 192.168.10.2 -d

IPERF initialization successful
New TCP client (settings flags 0x30313233)

-------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 49154
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXX
 Duration (ms) 10001
 Bandwidth (Mbitpsec) XX

-------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 36876
 Bytes Transferred XXXX
 Duration (ms) 10138
 Bandwidth (Mbitpsec) XX
```

Start IPerf Client (Tx and Rx individual):

```
# iperf -c 192.168.10.2 -r

# IPERF initialization successful
-------------------------------------------------
 TCP_DONE_CLIENT (TX)
 Local address : 192.168.10.1  Port 49155
 Remote address : 192.168.10.2  Port 5001
 Bytes Transferred XXXX
 Duration (ms) 10001
 Bandwidth (Mbitpsec) XX

New TCP client (settings flags 0x30313233)

-------------------------------------------------
 TCP_DONE_SERVER (RX)
 Local address : 192.168.10.1  Port 5001
 Remote address : 192.168.10.2  Port 36878
 Bytes Transferred XXXX
 Duration (ms) 10095
 Bandwidth (Mbitpsec) XX
```

iPerf UDP

For UDP tests please specify local interface ip address using -B option

Start IPerf server:

```
# iperf -s -u -B 192.168.10.1

# IPERF initialization successful
New UDP client (settings flags 0x0)

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

Start IPerf Client (Tx Only):

for UDP, bandwidth to send at in Mbps, default 100Mbps

```
# 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

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

Start IPerf Client with specific time (Tx Only):

for UDP, bandwidth to send at in Mbps, default 100Mbps

```
# iperf -c 192.168.10.2 -u -B 192.168.10.1 -b 50 -t 10

Ideal frame delay: 224 us

Send 4 frame(s) once per 1000 us

IPERF initialization successful

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

Start IPerf server with multicast ip (Tx Only):

```
# iperf -s -u -B 224.0.0.3
# IPERF initialization successful
# New UDP client (settings flags 0x0)
  Sending report back to client (0x80).
  Jitter 0.045, Lost 4/5 datagrams, OoO 2
  -------------------------------------------------
 UDP_DONE_SERVER (RX)
 Local address : 224.0.0.3  Port 5001
 Remote address : 192.168.1.2  Port 34218
 Bytes Transferred XXXX
 Duration (ms) 11033
 Bandwidth (Mbitpsec) XX
```

3.1.5.9 Wi-Fi Power Save

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

NOTE: *By default feature (IEEEPS and DEEP Sleep) is enabled, to disable need to configure macro CONFIG_WIFI_AUTO_POWER_SAVE 0 in* ***wifi_config.h***

- **IEEE Power Save (idle time is 10 msec)**

For IEEEPS mode Wi-Fi station should be connected with external AP and Soft AP should be de-activated.

IEEEPS Usage:

```
# wlan-ieee-ps
Usage: wlan-ieee-ps <0/1>
Error: Specify 0 to Disable or 1 to Enable
If <WNM> <sleep_interval> is specified, fw will enable WNM and use
sleep_interval
Example:
          wlan-ieee-ps 1 WNM 5
```

NOTE: *WNM is refer to Wireless Network Management sleep, if* ***CONFIG_WNM_PS*** *is enabled in* ***wifi_config.h*** *then ieee-ps command takes this extra argument of sleep duration, with this sleep duration can be extended.*

Enable IEEEPS:

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

Disable IEEEPS :

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

- **DeepSleep (idle time is 100 msec)**

Check Wi-Fi connection:

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

DeepSleep Usage:

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

Enable DeepSleep:

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

Disable DeepSleep:

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

3.1.5.10 Wi-Fi Host sleep/wowlan

The following commands are used to put the Wi-Fi in the sleep mode and wake up based on the provided conditions.

NOTE: *Define CONFIG_HOST_SLEEP macro in* ***wifi_config.h*** *to enable the feature.*
NOTE: *This command is only tested with i.MX RT1060 EVKC, i.MX RT1060 EVKB and i.MX RT1170 EVKB.*
For this command execution Wi-Fi station should be connected with external AP.

Host sleep Usage:

```
# wlan-host-sleep
Error: invalid number of arguments
Usage:
    wlan-host-sleep <1/0> <wowlan [wake_up_conds]/mef>
    [wake_up_conds] -- value for host wakeup conditions
               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
               All bit 1 cancel host sleep configuration
    mef     -- MEF host wakeup
Example:
    wlan-host-sleep <1/0> mef
    wlan-host-sleep <1/0> wowlan 0x1e
    wlan-host-sleep <1/0> wowlan 0x1e
```

MEF 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
Error: invalid number of arguments
```

Reset Previous configured Host sleep configuration

```
# wlan-host-sleep 0
Cancel Previous configured Host sleep configuration
```

Enable host sleep with one of the conditions like, Broadcast or Unicast or Multicast or Mac event or ARP Broadcast or Management frame. For example, device will wake up on ping request.

```
# wlan-multi-mef ping 3

# wlan-host-sleep 1 wowlan 2
Host sleep configuration successs with regular condition
```

Enable host sleep with no condition. Here device will not wake up based on any event/request like ping.

```
# wlan-host-sleep 1 wowlan 0
Host sleep configuration successs with regular condition
```

3.1.5.11 Set/Get Antenna Diversity Configuration

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

NOTE: *Make sure second antenna is connected before performing antenna configurations.*

Command Usage:

```
# wlan-set-antcfg
Usage:
wlan-set-antcfg <ant mode> [evaluate_time]

    <ant mode>:
         1   -- Tx/Rx antenna 1
         2   -- Tx/Rx antenna 2
         0xFFFF  -- Tx/Rx antenna diversity
    [evaluate_time]:
         if ant mode = 0xFFFF, SAD evaluate time interval,
         default value is 6000 milli seconds
```

3.1.5.12 Get Region Code

Note: *The region codes will be update from tx_pwr_limit region files.*

The following commands are used to get region code:

Get region code:

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

3.1.5.13 Roaming based on RSSI event

NOTE: *This feature is only enabled for IW611/612, and IW611/612 support is enabled in i.MX RT1170 EVKB and i.MX RT1060 EVKC.*

Command Usage:

```
# wlan-roaming
Usage:
    wlan-roaming <0/1> <rssi_threshold>
Example:
    wlan-roaming 1 40
Error: invalid number of arguments
```

Enable client to roam based on RSSI values. If AP1 crosses RSSI value, DUT will roam to AP2.

```
# wlan-roaming 1 40
```

Legacy roam sequence

```
# wlan-add abc ssid nxp wpa2 psk 12345678
# wlan-connect abc
# wlan-roaming 1 40
```

FT roam sequence

```
# wlan-add abc ssid nxp wpa2 ft-psk 12345678
# wlan-connect abc
# wlan-roaming 1 40
```

3.1.5.14 Roaming with 802.11k, 802.11r, and 802.11v

The following commands are used for client roaming using Wi-Fi network standards:

- **802.11K**

  The 802.11k standard helps devices search quickly for nearby APs that are available as roaming targets by creating an optimized list of channels. When the signal strength of the current AP weakens, STADUT will scan for target APs from this list.

  NOTE: For roaming, Ext.AP should be capable of 11k, 11v and 11r.

  Command Usage:

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

  Enable 11k:

  ```
  # wlan-host-11k-enable 1
  ```

  Send neighbor request and get nearby Aps list

  Command Usage:

  ```
  # wlan-host-11k-neighbor-req [ssid <ssid>]
  ```

  Send neighbor request with all nearby APs:

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

  Send neighbor request with particular APs with SSID name “11K_AP”:

  ```
  # wlan-host-11k-neighbor-req ssid 11K_AP
  ```

- **802.11r**

  When STADUT roams from one AP to another on the same network, 802.11r uses a feature called Fast Basic Service Set Transition (FT) to authenticate more quickly.

  Command Usage:

  bssid: MAC address of that AP to which user wants to roam

  channel: Channel number on which desired AP is active

```
# wlan-ft-roam
Usage:
Roam to new AP using FT:
    wlan-ft-roam <bssid> <channel>
Error: invalid number of arguments
```

  Roam through bssid and channel

```
wlan-ft-roam 00:e9:3a:b9:e0:35 1
```

- **802.11v**

  Trigger the bss transition query with specified status code from 0 to 16.

  Command Usage:

```
# wlan-host-11v-bss-trans-query
Usage: wlan-host-11v-bss-trans-query <query_reason[0..16]>
```

3.1.5.15 Other useful CLI commands

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

Get the Wi-Fi information:

```
# wlan-info
Station connected to:
"abc"
        SSID: nxp
        BSSID: 38:E6:0A:C6:1A:EC
        channel: 11
        role: Infra
        security: WPA2

        IPv4 Address
        address: DHCP
                IP:             192.168.43.156
                gateway:        192.168.43.233
                netmask:        255.255.255.0
                dns1:           192.168.43.233
                dns2:           0.0.0.0
        IPv6 Addresses
        Link-Local   :  FE80::2E9:3AFF:FEB9:E035 (Preferred)
        Global       :  2409:40C1:33:DCD:2E9:3AFF:FEB9:E035 (Preferred)
uAP started as:
"xyz"
        SSID: NXPAP
        BSSID: 00:13:43:6A:5A:ED
        channel: 11
        role: uAP
        security: WPA2

        IPv4 Address
        address: STATIC
                IP:             192.168.10.1
                gateway:        192.168.10.1
                netmask:        255.255.255.0
                dns1:           192.168.43.233
                dns2:           0.0.0.0
        IPv6 Addresses
        Link-Local   :  FE80::2E9:3AFF:FEB9:E035 (Tentative)
```

Get the Wi-Fi driver and firmware version:

```
# wlan-version
WLAN Driver Version   : vX.X.rXX.pX
WLAN Firmware Version : IW416-V0, RF878X, FP91, 16.91.21.p133, WPA2_CVE_FIX 1,
PVE_FIX 1
```

Get the Wi-Fi MAC address:

```
# wlan-mac
MAC address
00:13:43:6A:5A:ED
```

Get the list of Wi-Fi networks:

```
# wlan-list
2 networks:
"xyz"
        SSID: NXPAP
        BSSID: 00:00:00:00:00:00
        channel: 6
        role: uAP
        security: WPA2

        IPv4 Address
        address: STATIC
                IP:             192.168.10.1
                gateway:        192.168.10.1
                netmask:        255.255.255.0
                dns1:           192.168.43.1
                dns2:           0.0.0.0
"abc"
        SSID: nxp
        BSSID: 00:00:00:00:00:00
        channel: (Auto)
        role: Infra
        security: WPA2

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

Get the Wi-Fi stats:

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

Get the AP channel:

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

Ping the IP address:

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

Send RF Calibration host command:

```
# wlan-send-hostcmd
Hostcmd success, response is e0 80 18 0 29 0 0 0 1 0 0 0 38 2 8 0 5 1
```

This cli hardcodes a specific command and demonstrates usage of ***wlan_send_hostcmd*** API. This command can be changed to any other hostcmd, formed in the format mentioned here.
First 8 bytes of cmd_buf should have Command Header.

```
*  2 bytes : Command.
*  2 bytes : Size.
*  2 bytes : Sequence number.
*  2 bytes : Result.
*  Rest of buffer length is Command/Response Body
```

Default structure for hostcmd defined in ***wlan_tests.c*** cmd_buf[] = {0xe0, 0, 0x18, 0, 0x29, 0, 0, 0, 0x01, 0, 0, 0, 0x38, 0x02, 0x08, 0, 0x05, 0, 0x01, 0, 0x02, 0x01, 0, 0x01}; and differentiated as below.

```
cmd_buf[] = {
Command: 0xe0, 0,
Size: <2 bytes of size of entire data>,
Sequence number: 0, 0,
Result: 0, 0,
Set/Get: (for set 0x1 0x0, for get 0x0 0x0)
Revision: <Cal data format revision, 2 bytes>
Cal Data len: <length of cal data, 2 bytes>
Cal Data: <cal data byte array>
};
```

Please refer to AN13296 for more details about RF calibration Data commands.

Get the heap utilization

```
# heap-stat

Heap size ---------------------- : 32968
Largest Free Block size -------- : 32968
Smallest Free Block size ------- : 32968
Number of Free Blocks ---------- : 1
Total successful allocations --- : 49
Total successful frees --------- : 0
Min Free since system boot ----- : 32968
```

NOTE: *Add CONFIG_HEAP_STAT in wifi_config.h to include in cli option.*

Coexistence with External Radios for 88W8801

88W8801 Wi-Fi subsystem supports the packet traffic arbiter (PTA) protocol for coexistence with external radios (BT/802.15.4). The CLI command ***wlan-8801-enable-ext-coex*** is used to configure coexistence according to the external radio capabilities. Please refer to AN13612 for more details *about the Wi-Fi driver APIs*.

```
# wlan-8801-enable-ext-coex
8801 External Coex Config set successfully
```

Get External Radio Coex statistics

```
# wlan-8801-get-ext-coex-stats
BLE_EIP: 1, BLE_PRI: 1, WLAN_EIP: 1
```

Data encryption and decryption

wlan-eu-crypto command is used to encrypt and decrypt data based on FIPS (Federal Information Processing Standards). FIPS is the standard for the protection of sensitive or valuable data.

Usage:

```
# wlan-eu-crypto-aes-wrap
Usage:
Algorithm AES-WRAP encryption and decryption verification
wlan-eu-crypto-aes-wrap <EncDec>
EncDec: 0-Decrypt, 1-Encrypt
Error: invalid number of arguments
```

Encrypt Data:

```
# wlan-eu-crypto-aes-wrap 1
Raw Data:
**** Dump @ 2020C1B4 Len: 16 ****
12 34 56 78 90 12 34 56 78 90 12 34 56 78 90 12

******** End Dump *******
Encrypted Data:
**** Dump @ 2020C1D8 Len: 24 ****
fa da 96 53 30 97 4b 61 77 c6 d4 3c d2 0e 1f 6d
43 8a 0a 1c 4f 6a 1a d7
******** End Dump *******
```

Decrypt Data:

```
# wlan-eu-crypto-aes-wrap 0
Raw Data:
**** Dump @ 2020C19C Len: 24 ****
fa da 96 53 30 97 4b 61 77 c6 d4 3c d2 0e 1f 6d
43 8a 0a 1c 4f 6a 1a d7
******** End Dump *******
Decrypted Data:
**** Dump @ 2020C1D8 Len: 16 ****
12 34 56 78 90 12 34 56 78 90 12 34 56 78 90 12

******** End Dump *******
```

List of useful crypto commands

```
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>
```

3.1.6 Add CLIs in wifi_cli Sample Application

APIs 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.

The following example shows how to add a new command with arguments in the CLI application.

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 argv list and pass it to the relative APIs to be used by the driver/firmware.

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_setup Sample Application

This section describes *wifi_setup* sample application and its configuration along with the application execution. The *wifi_setup* sample application is used to demonstrate a Wi-Fi Station mode that connects to AP and starts pinging the IP address provided by the user.

Wi-Fi Features:

Table 7: wifi_setup Application Features

Features	Details
Wi-Fi	Wi-Fi Scan Wi-Fi Station mode Ping

3.2.1 wifi_setup Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console tool setup.

3.2.1.1 Run the application

The following logs can be observed on the console once the devices (i.MX RT1060 EVK board and NXP-based wireless module) are up and running.

```
Starting wifi_setup DEMO
MAC Address: 20:4E:F6:EC:1F:27
[i] WPL_Init: Success
[i] WPL_Start: Success
```

Once Wi-Fi module is initialized it'll try to scan nearby networks.

```
Initiating scan...

NXP_V10
     BSSID         : 5C:DF:89:0F:32:78
     RSSI          : -61dBm
     Channel       : 1
NXP_V10
     BSSID         : 5C:DF:89:0F:32:7C
     RSSI          : -64dBm
     Channel       : 36
ASUS_5G
     BSSID         : 7C:10:C9:02:DA:4C
     RSSI          : -58dBm
     Channel       : 36
Avaya_5G
     BSSID         : DC:33:3D:AB:E9:F8
     RSSI          : -73dBm
     Channel       : 11
nxp_netgear_2GHz
     BSSID         : 9C:C9:EB:F8:1F:71
     RSSI          : -76dBm
     Channel       : 6
NETGEAR-Guest
     BSSID         : A2:C9:EB:F8:1F:71
     RSSI          : -75dBm
     Channel       : 6

     BSSID         : DC:33:3D:AB:E9:FD
     RSSI          : -71dBm
     Channel       : 11
nxp_netgear_5G
     BSSID         : 9C:C9:EB:F8:1F:73
     RSSI          : -74dBm
     Channel       : 40
ASUS_2G
     BSSID         : 7C:10:C9:02:DA:48
     RSSI          : -58dBm
     Channel       : 11
nxp
     BSSID         : 22:49:3E:98:55:CF
     RSSI          : -52dBm
     Channel       : 11
```

It will ask details to connect preferred network

```
Please enter parameters of WLAN to connect

SSID:nxp
Password (for unsecured WLAN press Enter): ********
```

Once the connection is established successfully, it will ask a valid IPv4 address to ping. It will continuously ping the IP and print the received response time in ms(millisecond).

```
Please enter a valid IPv4 address to test the connection

IP address: 8.8.8.8

Starting ping task...
ping: send
8.8.8.8

ping: send
8.8.8.8


ping: recv
8.8.8.8
 278 ms

ping: send
8.8.8.8


ping: recv
8.8.8.8
 173 ms

ping: send
8.8.8.8

...
```

3.3 3.7wifi_webconfig Sample Application

This section describes *wifi_webconfig* sample application and its configuration along with the application execution. The *wifi_webconfig* sample application is used to demonstrate a commissioning procedure using the uAP with an HTTP server to configure client mode to connect to an AP.

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

The website in AP mode shows the available networks using scan. The desired network can be chosen by clicking on 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 consequently the website closes.

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.

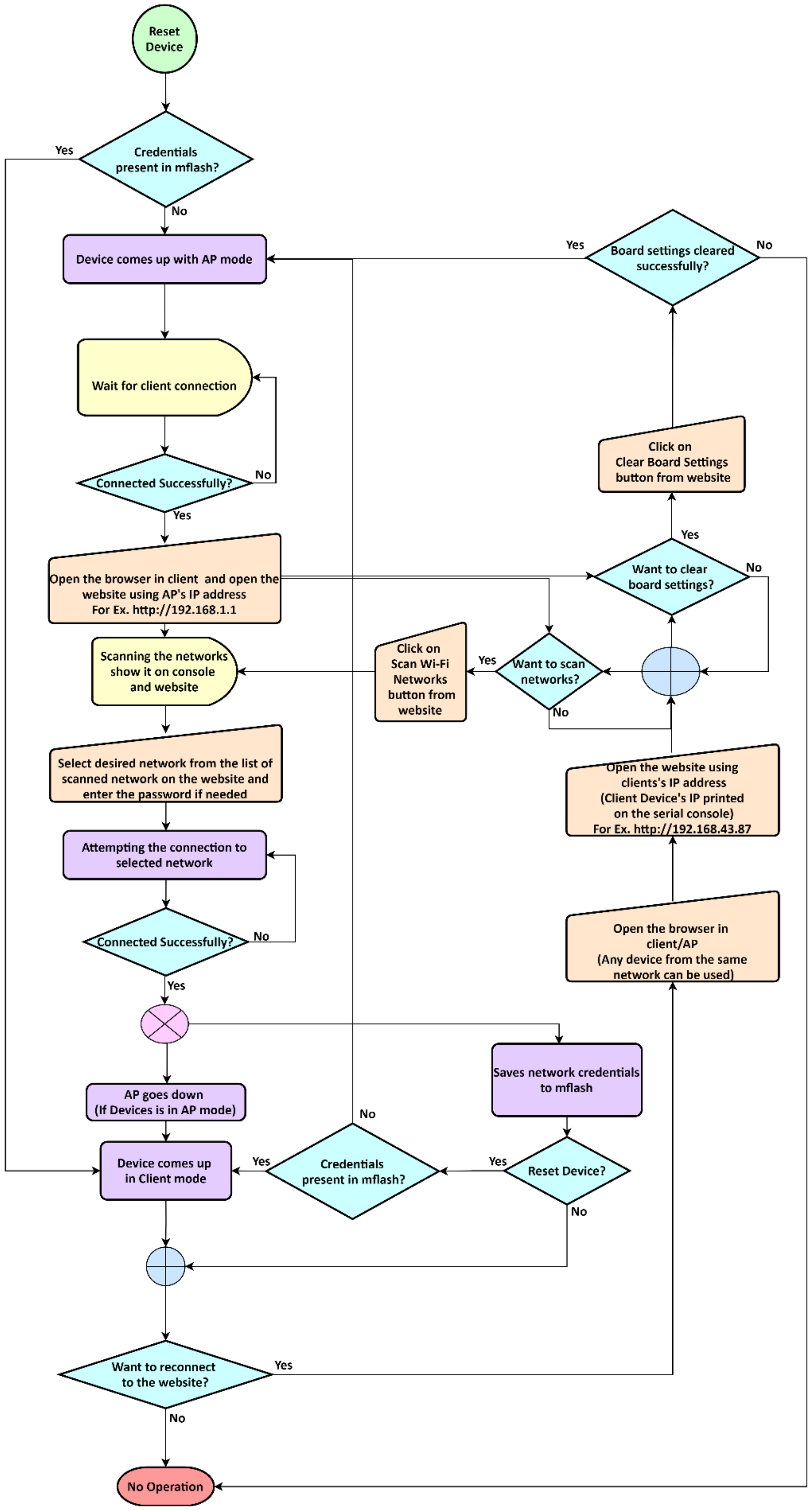

Reset Device
Credentials present in mflash?
Yes
No
Device comes up with AP mode
Wait for client connection
Connected Successfully?
No
Yes
Open the browser in client and open the website using AP's IP address For Ex. http://192.168.1.1
Scanning the networks show it on console and website
Select desired network from the list of scanned network on the website and enter the password if needed
Attempting the connection to selected network
Connected Successfully?
No
Yes
AP goes down (If Devices is in AP mode)
Device comes up in Client mode
Saves network credentials to mflash
Reset Device?
Yes
No
Credentials present in mflash?
Yes
No
Want to reconnect to the website?
Yes
No
No Operation
Board settings cleared successfully?
Yes
No
Click on Clear Board Settings button from website
Want to clear board settings?
Yes
No
Want to scan networks?
Yes
No
Click on Scan Wi-Fi Networks button from website
Open the website using clients's IP address (Client Device's IP printed on the serial console) For Ex. http://192.168.43.87
Open the browser in client/AP (Any device from the same network can be used)

Figure 34: wifi_webconfig flow diagram

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

Table 8: 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

3.3.1 User Configurations

Some of the Wi-Fi features and feature related macros that user can configure based on requirement are listed in below table along with source file name.

Wi-Fi configurations

Table 9: 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”		

3.3.2 wifi_webconfig Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console setup.

3.3.2.1 Start-up logs

The following logs can be observed on the console once the devices (i.MX RT1060 EVK board and NXP-based wireless module) are up and running. The *Wi-Fi FW version log* shows that the Wi-Fi module is ready to operate.

```
Starting webconfig DEMO
[i] Trying to load data from mflash.
[i] Nothing stored yet
[i] Initializing Wi-Fi connection...
MAC Address: 20:4E:F6:EC:1F:27
[i] Successfully initialized Wi-Fi module
Starting Access Point: SSID: nxp_configuration_access_point, Chnl: 1
[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
```

3.3.2.2 Connect the client to Soft AP

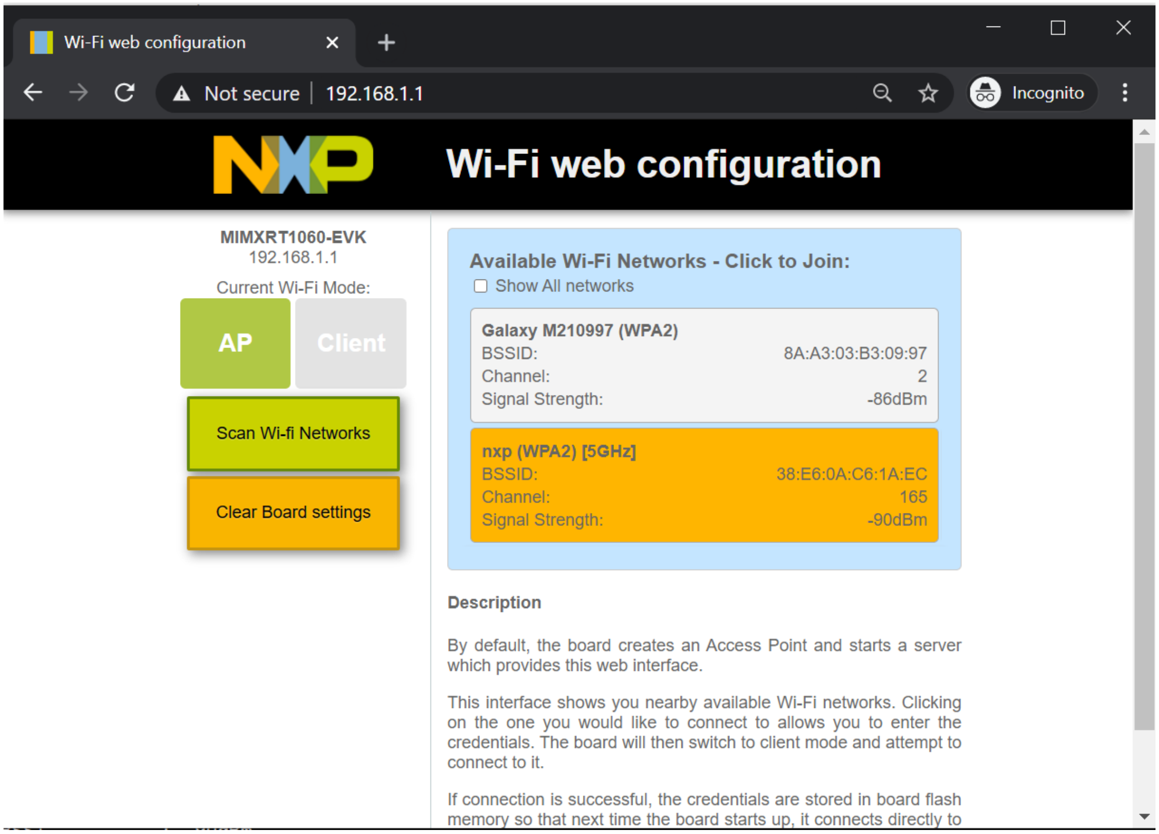
Connect the client to soft AP and observe the logs with the client mac address.

```
Client => 14:AB:C5:F4:C4:C3 Associated with Soft AP
```

3.3.2.3 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 mode AP is highlighted on the web page. See Figure 35.

```
Initiating scan...

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



Wi-Fi web configuration

MIMXRT1060-EVK
192.168.1.1
Current Wi-Fi Mode:

Available Wi-Fi Networks - Click to Join:

□ Show All networks

Galaxy M210997 (WPA2)
BSSID: 8A:A3:03:B3:09:97
Channel: 2
Signal Strength: -86dBm

nxp (WPA2) [5GHz]
BSSID: 38:E6:0A:C6:1A:EC
Channel: 165
Signal Strength: -90dBm

Description

By default, the board creates an Access Point and starts a server which provides this web interface.

This interface shows you nearby available Wi-Fi networks. Clicking on the one you would like to connect to allows you to enter the credentials. The board will then switch to client mode and attempt to connect to it.

If connection is successful, the credentials are stored in board flash memory so that next time the board starts up, it connects directly to

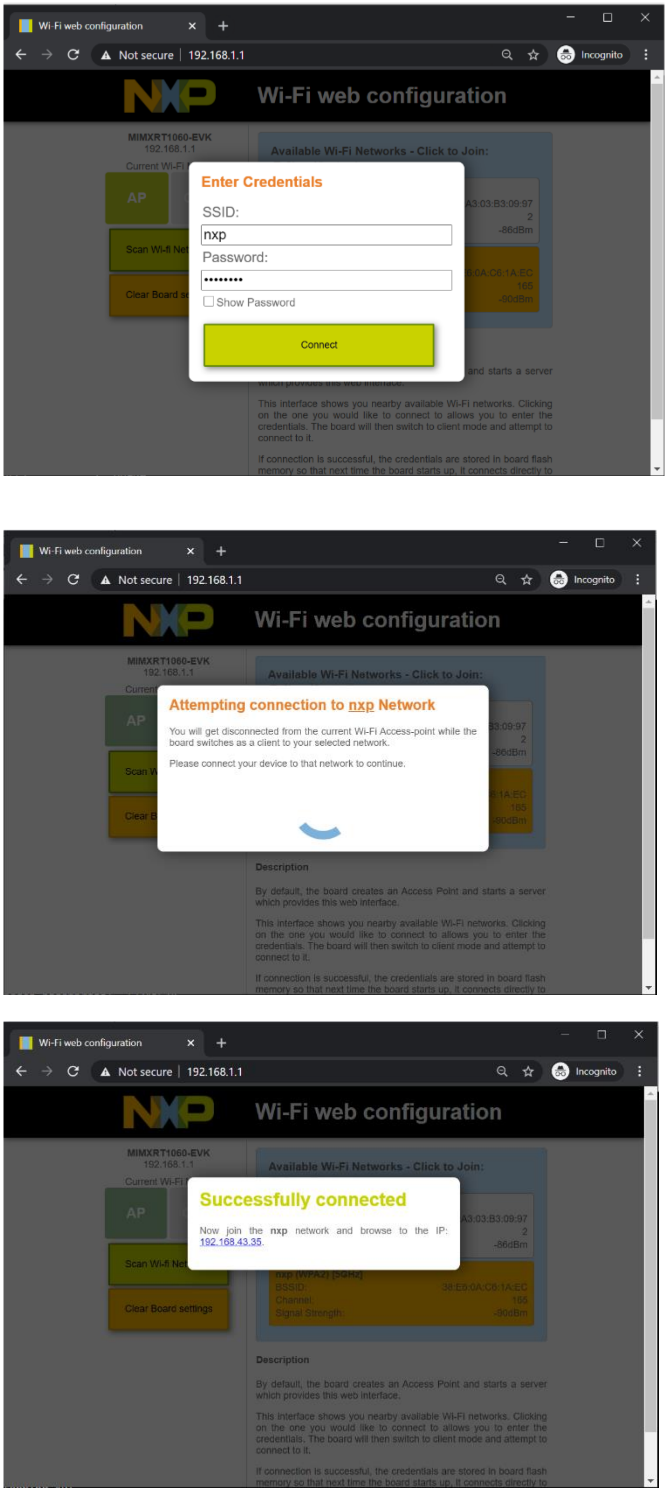
Figure 35: wifi_webconfig Website in AP Mode

3.3.2.4 Connect the device to the AP

Click on the desired SSID on the web page. 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.

***NOTE**: This demo will not allow to connect with WPA3 SSID*

```
[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!
```



Enter Credentials

SSID: nxp

Password: ••••••••

□ Show Password

Connect

Attempting connection to nxp Network

You will get disconnected from the current Wi-Fi Access-point while the board switches as a client to your selected network.

Please connect your device to that network to continue.

Successfully connected

Now join the nxp network and browse to the IP: 192.168.43.35.

Description

By default, the board creates an Access Point and starts a server which provides this web interface.

This interface shows you nearby available Wi-Fi networks. Clicking on the one you would like to connect to allows you to enter the credentials. The board will then switch to client mode and attempt to connect to it.

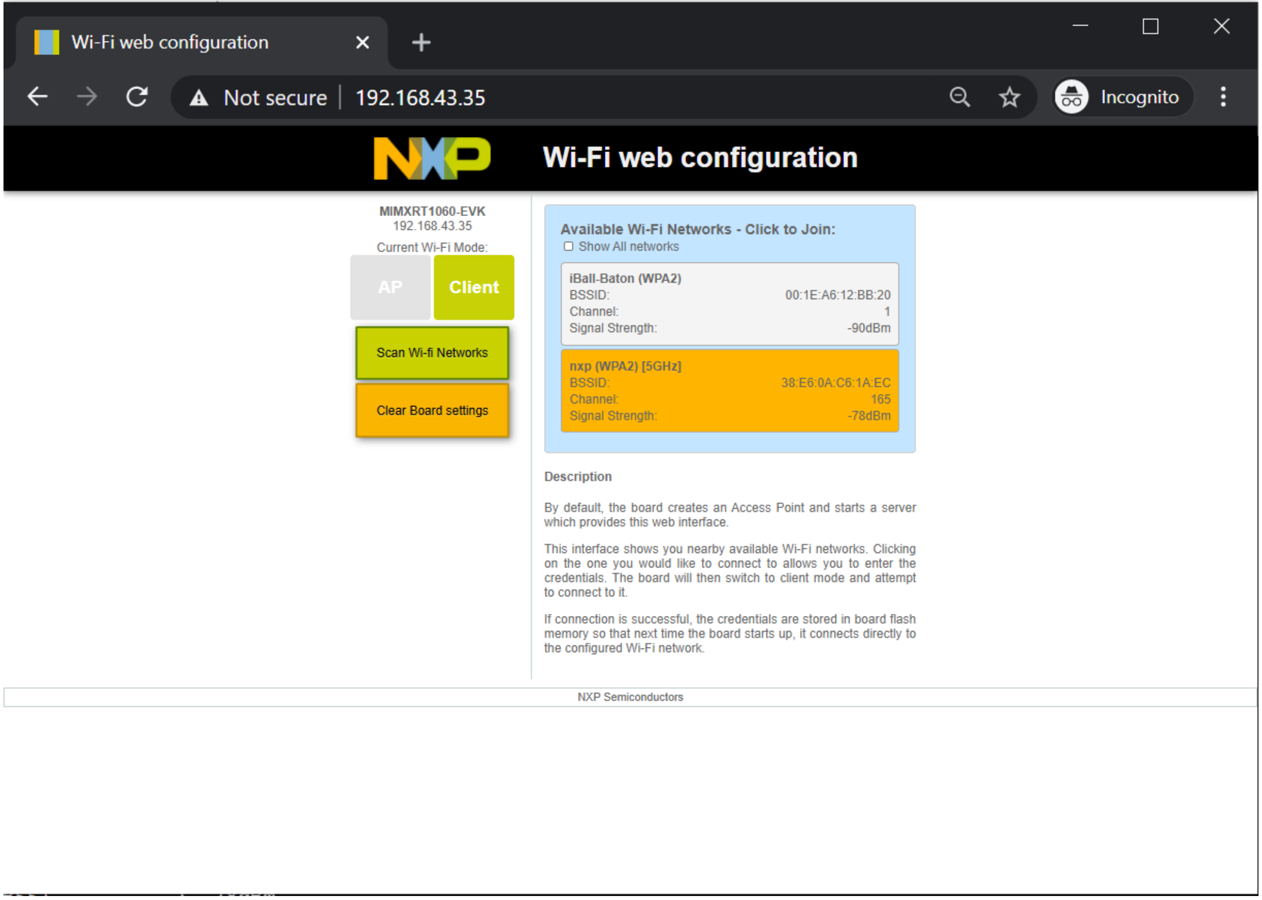
If connection is successful, the credentials are stored in board flash memory so that next time the board starts up, it connects directly to

Figure 36: Connection Attempt to AP using wifi_webconfig Application

***NOTE**: Once the configurations are successfully received by the device, soft AP goes down and the device switches to the client mode. To reconnect to the website, switch to the AP network and use the device (client mode) IP (printed on the console) to open the website.*

For example, Figure 37 shows http://192.168.43.35 to reconnect to website.

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



MIMXRT1060-EVK
192.168.43.35
Current Wi-Fi Mode:

Available Wi-Fi Networks - Click to Join:

□ Show All networks

iBall-Baton (WPA2)

BSSID:	00:1E:A6:12:BB:20
Channel:	1
Signal Strength:	-90dBm

nxp (WPA2) [5GHz]

BSSID:	38:E6:0A:C6:1A:EC
Channel:	165
Signal Strength:	-78dBm

Description

By default, the board creates an Access Point and starts a server which provides this web interface.

This interface shows you nearby available Wi-Fi networks. Clicking on the one you would like to connect to allows you to enter the credentials. The board will then switch to client mode and attempt to connect to it.

If connection is successful, the credentials are stored in board flash memory so that next time the board starts up, it connects directly to the configured Wi-Fi network.

Figure 37: wifi_webconfig Website in Client Mode

3.3.2.5 Device reboot with the configurations stored in mflash

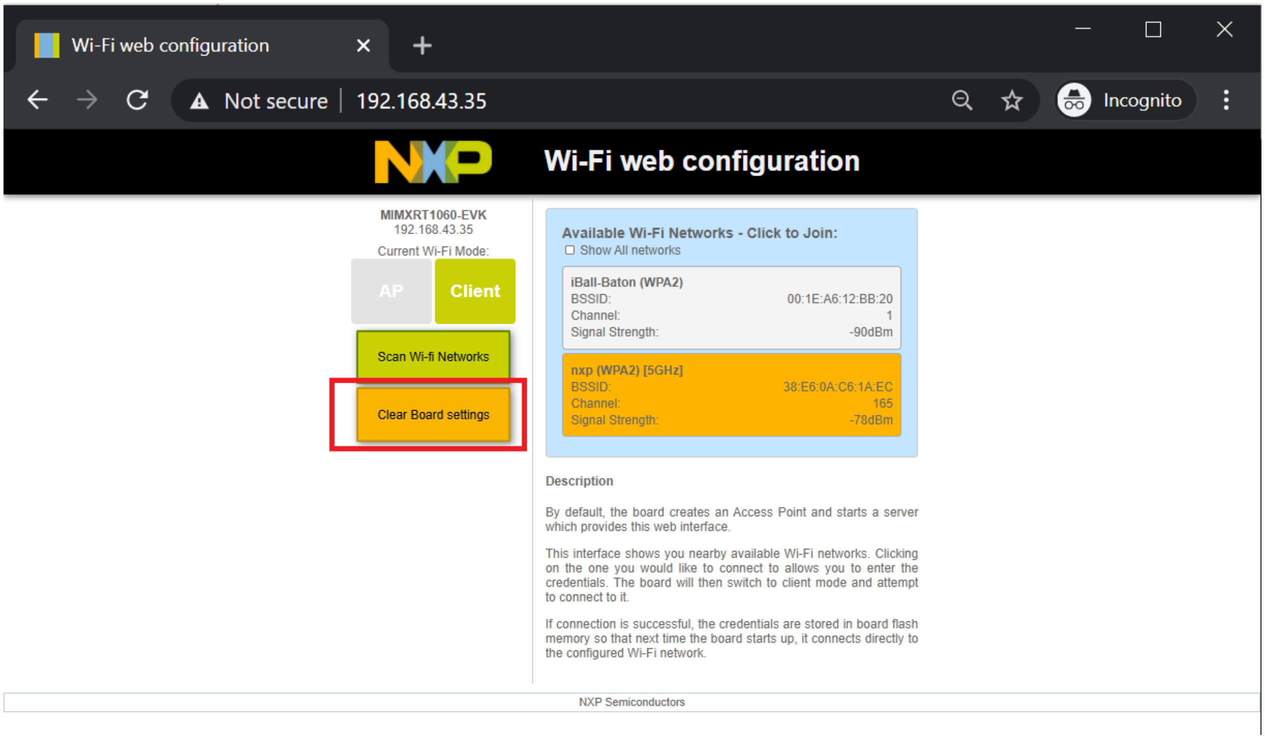
The following logs can be observed when the device has the client configuration saved in *mflash*. It reads the stored information and uses it to configure client mode after a reboot.

```
Starting webconfig DEMO
[i] Trying to load data from mflash.
[i] Saved SSID: nxp, Password: 12345678
[i] Initializing Wi-Fi connection...
MAC Address: 20:4E:F6:EC:1F:27
[i] Successfully initialized Wi-Fi module
Connecting as client to ssid: nxp with password 12345678
```

3.3.2.6 Clear the settings on the website

To clear the configurations saved in mflash, press the **Clear Board settings** button available on the webpage.

```
[i] mflash_save_file success
Starting Access Point: SSID: nxp_configuration_access_point, Chnl: 1
[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
```



iBall-Baton (WPA2)
BSSID: 00:1E:A6:12:BB:20
Channel: 1
Signal Strength: -90dBm

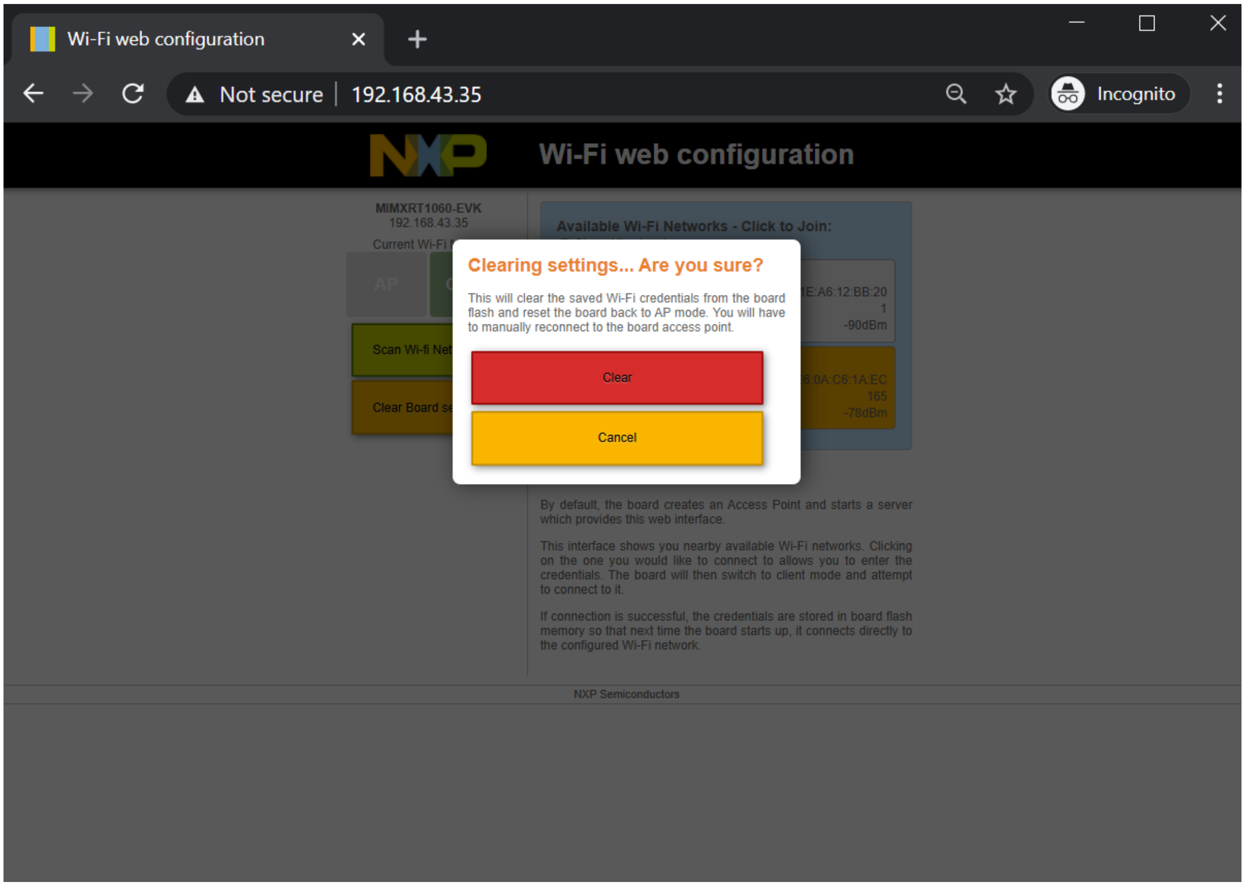
nxp (WPA2) [5GHz]
BSSID: 38:E6:0A:C6:1A:EC
Channel: 165
Signal Strength: -78dBm

Description

By default, the board creates an Access Point and starts a server which provides this web interface.

This interface shows you nearby available Wi-Fi networks. Clicking on the one you would like to connect to allows you to enter the credentials. The board will then switch to client mode and attempt to connect to it.

If connection is successful, the credentials are stored in board flash memory so that next time the board starts up, it connects directly to the configured Wi-Fi network.



Clearing settings... Are you sure?

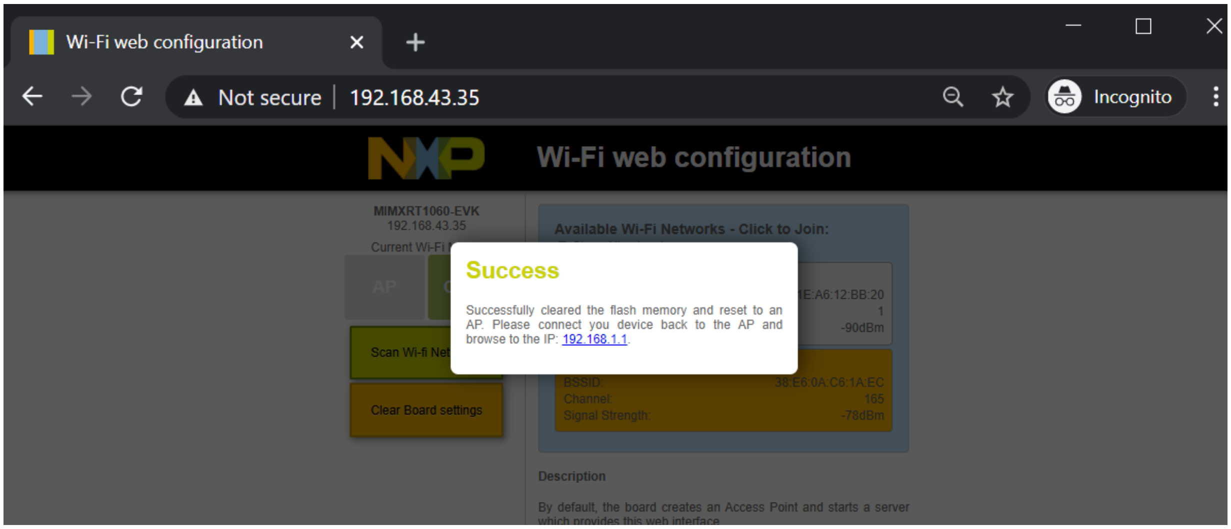
This will clear the saved Wi-Fi credentials from the board flash and reset the board back to AP mode. You will have to manually reconnect to the board access point.

By default, the board creates an Access Point and starts a server which provides this web interface.

This interface shows you nearby available Wi-Fi networks. Clicking on the one you would like to connect to allows you to enter the credentials. The board will then switch to client mode and attempt to connect to it.

If connection is successful, the credentials are stored in board flash memory so that next time the board starts up, it connects directly to the configured Wi-Fi network.

Figure 38: Clear Configurations saved in mflash using website (wifi_webconfig Application)



Success

Successfully cleared the flash memory and reset to an AP. Please connect you device back to the AP and browse to the IP: 192.168.1.1.

By default, the board creates an Access Point and starts a server which provides this web interface

Figure 39: Clear Configuration Success Message in wifi_webconfig Application

3.4 wifi_test_mode Sample Application

This section describes the wifi_test_mode application to demonstrate the CLI support to enable the user to control the Wi-Fi device to run various RF and regulatory compliance tests. This application enables RF testing for the Wi-Fi module. It helps to Measure RF parameters such as transmit power for both 2.4GHz and 5GHz, display RF packet counts, RF antenna configuration and transmit standard 802.11 packets.

3.4.1 wifi_test_mode Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console setup.

3.4.1.1 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 test mode 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 Test Mode CLIs are initialized
========================================
CLIs Available:
========================================

help
wlan-version
wlan-mac
wlan-set-rf-test-mode
wlan-set-rf-tx-antenna <antenna>
wlan-get-rf-tx-antenna
wlan-set-rf-rx-antenna <antenna>
wlan-get-rf-rx-antenna
wlan-set-rf-band <band>
wlan-get-rf-band
wlan-set-rf-bandwidth <bandwidth>
wlan-get-rf-bandwidth
wlan-set-rf-channel <channel>
wlan-get-rf-channel
wlan-set-rf-tx-power <tx_power> <modulation> <path_id>
wlan-set-rf-tx-cont-mode <enable_tx> <cw_mode> <payload_pattern> <cs_mode>
<act_sub_ch> <tx_rate>
wlan-set-rf-tx-frame <start> <data_rate> <frame_pattern> <frame_len>
<adjust_burst_sifs> <burst_sifs_in_us> <short_preamble> <act_sub_ch> <short_gi>
<adv_coding> <tx_bf> <gf_mode> <stbc> <bssid>
wlan-get-and-reset-rf-per
========================================
```

3.4.1.2 Prerequisite Commands

The following steps describe prerequisite commands to start Wi-Fi RF Test.

Wi-Fi RF test mode enable

The following command is used to set Wi-Fi mode to rf test mode:

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

Wi-Fi RF band set and get

The following commands are used to set and get Wi-Fi band:

Command Usage:

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

Set and Get RF band:

```
# wlan-set-rf-band 1
RF Band configuration successful
```

```
# wlan-get-rf-band
Configured RF Band is: 5G
```

Wi-Fi RF channel set and get

The following commands are used to set and get Wi-Fi channel:

Command Usage:

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

Set and Get RF channel:

```
# wlan-set-rf-channel 132
Channel configuration successful
```

```
# wlan-get-rf-channel
Configured channel is: 132
```

Wi-Fi RF bandwidth set and get

The following commands are used to set and get Wi-Fi bandwidth:

NOTE: *88W8987 supports 11ac 80MHz support*

Command Usage:

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

        <bandwidth>:
                0: 20MHz
                1: 40MHz
                4: 80MHz
```

Set and Ge RF Bandwidth:

For 20MHz

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

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

For 80MHz

```
# wlan-set-rf-bandwidth 4
Bandwidth configuration successful
```

```
# wlan-get-rf-bandwidth
Configured bandwidth is: 80MHz
```

3.4.1.3 Display and Clear Received Wi-Fi Packet Count

The following command clear the received packet count and displays the received multi-cast and error packet counts.

```
# wlan-get-and-reset-rf-per
PER is as below:
  Total Rx Packet Count                    : 9856
  Total Rx Multicast/Broadcast Packet Count: 1414
  Total Rx Packets with FCS error          : 685
```

3.4.1.4 Wi-Fi Antenna Configuration

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

Command Usage:

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

Set and Get TX antenna configuration:

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

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

Command Usage:

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

Set and Get RX antenna configuration:

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

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

3.4.1.5 Wi-Fi Tx Power configuration

The following command is used to set the transmitter output power at the antenna using stored calibration data. 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)
```

Set Tx Power:

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

3.4.1.6 Wi-Fi set transmitter in CW mode

The following command is used to set Wi-Fi transmitter to Continuous Wave (CW) mode.

Command Usage:

For different data rate values See Table 10bgn: Data rate parameter

```
# 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:
wlan-set-rf-tx-cont-mode 0
```

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
```

Disable CW mode:

```
# 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: *It is required to disable CW mode once test completed. CW mode test and TX frame test does not support parallel operation.*

Table 10bgn: Data rate parameter

ID (Hex value)	Data rate
00	1Mbits/sec
01	2Mbits/sec
02	5.5Mbits/sec
03	11Mbits/sec
04	22Mbits/sec
05	6Mbits/sec
06	9Mbits/sec
07	12Mbits/sec
08	18Mbits/sec
09	24Mbits/sec
0A	36Mbits/sec
0B	48Mbits/sec
0C	54Mbits/sec
0D	72Mbits/sec
0E	HT_MCS 0
0F	HT_MCS 1
10	HT_MCS 2
11	HT_MCS 3
12	HT_MCS 4
13	HT_MCS 5
14	HT_MCS 6
15	HT_MCS 7
2E	HT_MCS 32

Table 11: 11ac Data rate parameter

ID (Hex value)	Data rate
00	1Mbits/sec
01	2Mbits/sec
02	5.5Mbits/sec
03	11Mbits/sec
04	Reserved
05	6Mbits/sec
06	9Mbits/sec
07	12Mbits/sec
08	18Mbits/sec
09	24Mbits/sec
0A	36Mbits/sec
0B	48Mbits/sec

0C	54Mbits/sec
0D	Reserved
0E	HT_MCS 0
0F	HT_MCS 1
10	HT_MCS 2
11	HT_MCS 3
12	HT_MCS 4
13	HT_MCS 5
14	HT_MCS 6
15	HT_MCS 7
16	HT_MCS 8
17	HT_MCS 9
18	HT_MCS 10
19	HT_MCS 11
1A	HT_MCS 12
1B	HT_MCS 13
1C	HT_MCS 14
1D	HT_MCS 15
100	VHT_SS1_MCS0
101	VHT_SS1_MCS1
102	VHT_SS1_MCS2
103	VHT_SS1_MCS3
104	VHT_SS1_MCS4
105	VHT_SS1_MCS5
106	VHT_SS1_MCS6
107	VHT_SS1_MCS7
108	VHT_SS1_MCS8
109	VHT_SS1_MCS9

Table 12: 11ax Data rate parameter

ID (Hex value)	Data rate
00	1Mbits/sec
01	2Mbits/sec
02	5.5Mbits/sec
03	11Mbits/sec
04	22Mbits/sec
05	6Mbits/sec
06	9Mbits/sec
07	12Mbits/sec
08	18Mbits/sec
09	24Mbits/sec

0A	36Mbits/sec
0B	48Mbits/sec
0C	54Mbits/sec
0D	72Mbits/sec
0E	HT_MCS 0
0F	HT_MCS 1
10	HT_MCS 2
11	HT_MCS 3
12	HT_MCS 4
13	HT_MCS 5
14	HT_MCS 6
15	HT_MCS 7
1100	VHT_SS1_MCS0
1101	VHT_SS1_MCS1
1102	VHT_SS1_MCS2
1103	VHT_SS1_MCS3
1104	VHT_SS1_MCS4
1105	VHT_SS1_MCS5
1106	VHT_SS1_MCS6
1107	VHT_SS1_MCS7
1108	VHT_SS1_MCS8
1109	VHT_SS1_MCS9
2100	HE_SS1_MCS0
2101	HE_SS1_MCS1
2102	HE_SS1_MCS2
2103	HE_SS1_MCS3
2104	HE_SS1_MCS4
2105	HE_SS1_MCS5
2106	HE_SS1_MCS6
2107	HE_SS1_MCS7
2108	HE_SS1_MCS8
2109	HE_SS1_MCS9
210A	HE_SS1_MCS10
210B	HE_SS1_MCS11

3.4.1.7 Transmit standard 802.11 packets

The following command is used to continuously transmit packets, with an adjustable time gap of 0 to 250 microseconds between packets.

Command Usage:

For different data rate values See Table 10bgn: Data rate parameter

```
# wlan-set-rf-tx-frame
Usage:
```

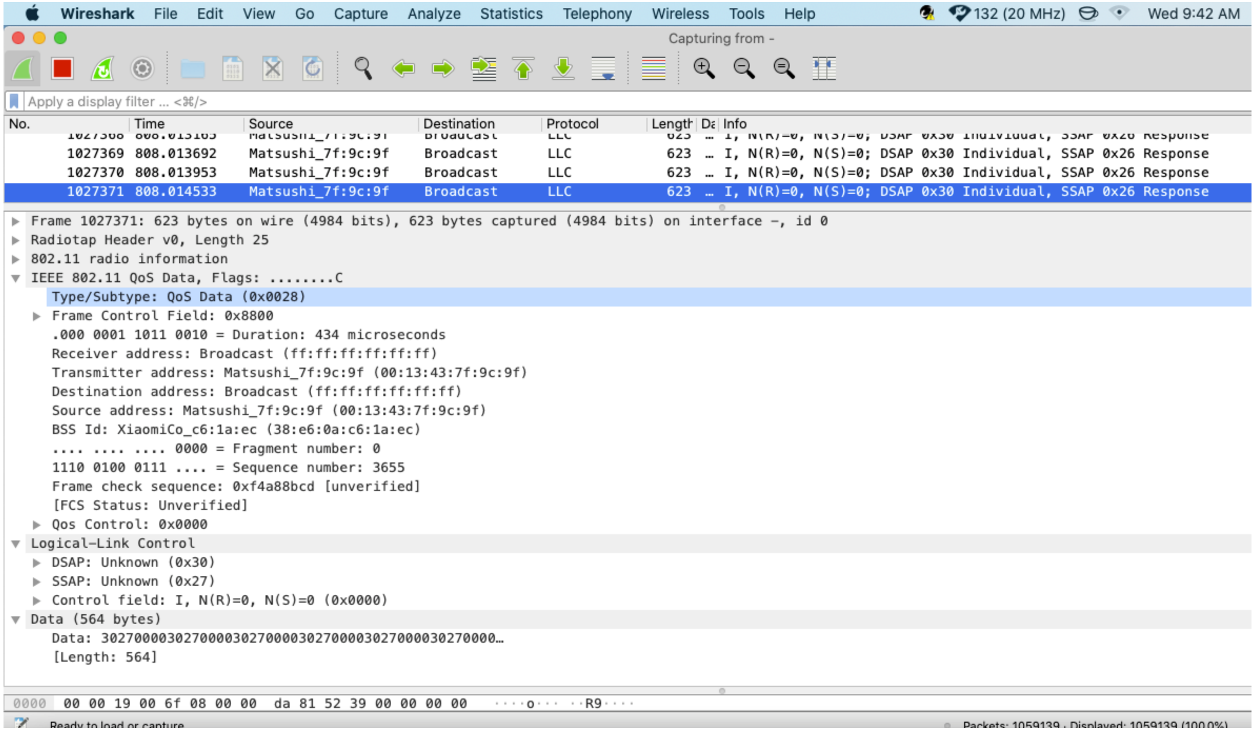
```
wlan-set-rf-tx-frame <start> <data_rate> <frame_pattern> <frame_len>
<adjust_burst_sifs> <burst_sifs_in_us> <short_preamble> <act_sub_ch> <short_gi>
<adv_coding> <tx_bf> <gf_mode> <stbc> <bssid>
Enable                  (0:disable, 1:enable)
Tx Data Rate            (Rate Index corresponding to legacy/HT/VHT 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
```

Enable Tx Frame:

```
# wlan-set-rf-tx-frame 1 0x7 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
```

Packet Capture:

Please refer section 2.2 for the Wireshark tool setup and start capturing packets for configured channel and bandwidth.



```
No.      Time        Source             Destination  Protocol  Length Da Info
1027369  808.013692  Matsushi_7f:9c:9f  Broadcast    LLC       623    …  I, N(R)=0, N(S)=0; DSAP 0x30 Individual, SSAP 0x26 Response
1027370  808.013953  Matsushi_7f:9c:9f  Broadcast    LLC       623    …  I, N(R)=0, N(S)=0; DSAP 0x30 Individual, SSAP 0x26 Response
1027371  808.014533  Matsushi_7f:9c:9f  Broadcast    LLC       623    …  I, N(R)=0, N(S)=0; DSAP 0x30 Individual, SSAP 0x26 Response
```

```
▶ Frame 1027371: 623 bytes on wire (4984 bits), 623 bytes captured (4984 bits) on interface –, id 0
▶ Radiotap Header v0, Length 25
▶ 802.11 radio information
▼ IEEE 802.11 QoS Data, Flags: ........C
    Type/Subtype: QoS Data (0x0028)
  ▶ Frame Control Field: 0x8800
    .000 0001 1011 0010 = Duration: 434 microseconds
    Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
    Transmitter address: Matsushi_7f:9c:9f (00:13:43:7f:9c:9f)
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: Matsushi_7f:9c:9f (00:13:43:7f:9c:9f)
    BSS Id: XiaomiCo_c6:1a:ec (38:e6:0a:c6:1a:ec)
    .... .... .... 0000 = Fragment number: 0
    1110 0100 0111 .... = Sequence number: 3655
    Frame check sequence: 0xf4a88bcd [unverified]
    [FCS Status: Unverified]
  ▶ Qos Control: 0x0000
▼ Logical-Link Control
  ▶ DSAP: Unknown (0x30)
  ▶ SSAP: Unknown (0x27)
  ▶ Control field: I, N(R)=0, N(S)=0 (0x0000)
▼ Data (564 bytes)
    Data: 302700003027000030270000302700003027000030270000…
    [Length: 564]

0000  00 00 19 00 6f 08 00 00  da 81 52 39 00 00 00 00   ····o··· ··R9····
```

Figure 40: TX Frame Packet Capture

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
```

3.4.1.8 Other useful CLI commands

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

Get the Wi-Fi driver and firmware version:

```
# wlan-version
WLAN Driver Version   : vX.X.rXX.pX
WLAN Firmware Version : IW416-V0, RF878X, FP91, 16.91.21.p133, WPA2_CVE_FIX 1,
PVE_FIX 1
```

Get the Wi-Fi MAC address:

```
MAC address
00:13:43:7F:9C:9F
```

3.4.1.9 Example command sequences for adjusting Tx power in 2.4GHz

The radio is configured as shown below.

- **2.4 GHz band**
- **Channel 6**
- **20 MHz bandwidth**
- **6 Mbps legacy data rate**
- **Test pattern transmitted is 0x00000AAA**
- **Output power set to +15 dBm. then adjusted to +14 dBm**
- **For different data rate values See Table 10bgn: Data rate parameter**

Table 13: Tx power command sequences for 2.4GHz

Step	Operation	Command
1	Set RF test mode	**# wlan-set-rf-test-mode** rf_test_mode set successfully
2	Set RF band	**# wlan-set-rf-band 0** RF Band configuration successful
3	Set RF bandwidth (switched order with step 4)	**# wlan-set-rf-bandwidth 0** Bandwidth configuration successful
4	Set RF channel	**# wlan-set-rf-channel 6** Channel configuration successful
5	Set Tx antenna	**# wlan-set-rf-tx-antenna 1** Tx antenna configuration successful
6	Get settings (optional)	**# wlan-get-rf-band** Configured RF band is: 2.4 G **# wlan-get-rf-channel** Configured channel is: 6 **# wlan-get-rf-bandwidth** Configured bandwidth is: 20MHz
7	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
8	Set continuous transmit mode	**# wlan-set-rf-tx-cont-mode 1 0 0xAAA 0 3 5** Tx continous mode successful Enable : enable CW 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**
10	Set output power to +14 dBm	**# wlan-set-rf-tx-power 14 1 0** Tx Power configuration successful Power : 14 dBm Modulation : OFDM Path ID : PathA

11	Restart transmission	**# wlan-set-rf-tx-cont-mode 1 0 0xAAA 0 3 5** Tx continous mode successful Enable : enable CW 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**

3.4.1.10 Example command sequences for adjusting Tx power in 5GHz

The radio is configured as shown below.

- **5 GHz band**
- **Channel 44/48**
- **40 MHz bandwidth**
- **MCS0 HT data rate**
- **Test pattern transmitted is 0x00BBBAAA**
- **Output power set to +9 dBm, then adjusted to +8 dBm.**
- **For different data rate values See Table 10bgn: Data rate parameter**

Table 14: Tx power command sequences for 5GHz

Step	Operation	Command
1	Set RF test mode	**# wlan-set-rf-test-mode** RF Test Mode configuration successful
2	Set RF band	**# wlan-set-rf-band 1** RF Band configuration successful
3	Set RF bandwidth (switched order with step 4)	**# wlan-set-rf-bandwidth 1** Bandwidth configuration successful
4	Set RF channel	**# wlan-set-rf-channel 48** Channel configuration successful
5	Set Tx antenna	**# wlan-set-rf-tx-antenna 1** Tx antenna configuration successful
6	Get settings (optional)	**# wlan-get-rf-band** Configured RF band is: 5 G **# wlan-get-rf-channel** Configured channel is: 48 **# wlan-get-rf-bandwidth** Configured bandwidth is: 40MHz
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 continous mode successful Enable : enable CW mode : disable Payload pattern : 0x00BBBAAA CS mode : disable Active SubChannel : both Tx Data Rate : 14
9	Stop transmission	**# wlan-set-rf-tx-cont-mode 0**
10	Set output power to +9 dBm	**# wlan-set-rf-tx-power 9 1 0** Tx Power configuration successful Power : 9 dBm Modulation : OFDM Path ID : PathA
11	Restart transmission	**# wlan-set-rf-tx-cont-mode 1 0 0xBBBAAA 0 3 14** Tx continous mode successful Enable : enable CW 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**
13	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
14	Restart transmission	**# wlan-set-rf-tx-cont-mode 1 0 0xBBBAAA 0 3 14** Tx continous mode successful Enable : enable CW mode : disable Payload pattern : 0x00BBBAAA CS mode : disable Active SubChannel : both Tx Data Rate : 14
15	Stop transmission	**# wlan-set-rf-tx-cont-mode 0**

3.5 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.

NOTE: *Support for this application is available only for i.MX RT1060 EVK/EVKC board.*

Table 15: 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 ED MAC Wi-Fi host sleep/wowlan Wi-Fi RF Calibration Wi-Fi coexistence with external radios (for 88W8801)
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)

3.5.1 wifi_cert Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console 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.

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

help
wlan-version
wlan-mac
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-deep-sleep-ps <0/1>
wlan-host-sleep <0/1> wowlan_test <0/1>
wlan-send-hostcmd
wlan-8801-enable-ext-coex
wlan-get-regioncode
wlan-get-txpwrlimit <subband>
wlan-set-txpwrlimit
wlan-set-chanlist-and-txpwrlimit
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting>
wlan-get-txratecfg
wlan-get-data-rate
wlan-set-pmfcfg <mfpc> <mfpr>
wlan-get-pmfcfg
wlan-set-antcfg <ant mode> [evaluate_time]
wlan-get-antcfg
wlan-set-ed-mac-mode <ed_ctrl_2g> <ed_offset_2g>
wlan-get-ed-mac-mode
```

```
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4
address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
========================================
```

NOTE: Please refer sections 3.1.5.4 to 0 for basic Wi-Fi features like Wi-Fi Scan, Wi-Fi AP mode, Wi-Fi Station mode, IPerf etc.

3.5.1.2 Get Region Code

Note: *The region codes will be update from tx_pwr_limit region files.*

Get region code:

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

3.5.1.3 Set/Get Tx Power Limit

The following commands are used to get and set tx power limit:

Command Usage:

```
# wlan-get-txpwrlimit
Usage:
wlan-get-txpwrlimit <subband>

         Where subband is:
               0x00 2G subband  (2.4G: channel 1-14)
               0x10 5G subband0 (5G: channel 36,40,44,48,
                                             52,56,60,64)
               0x11 5G subband1 (5G: channel 100,104,108,112,
                                             116,120,124,128,
                                             132,136,140,144)
               0x12 5G subband2 (5G: channel 149,153,157,161,165,172)
               0x13 5G subband3 (5G: channel 183,184,185,187,188,
                                             189, 192,196;
                                 5G: channel 7,8,11,12,16,34)
```

Get Tx Power Limit:

```
# wlan-get-txpwrlimit 00
-------------------------------------------------------------------------------
Get txpwrlimit: sub_band=0
StartFreq: 2407
ChanWidth: 20
ChanNum:   1
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   2
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   3
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   4
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   5
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   6
```

```
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   7
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   8
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   9
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   10
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   11
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   12
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   13
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2414
ChanWidth: 20
ChanNum:   14
Pwr:0,0,1,0,2,0,3,0,4,0,5,0,6,0
```

Set Tx Power Limit:

NOTE: *This command will first set tx power configurations for the band 2GHz or both 2GHz and 5GHz based on the Wi-Fi module selection and then prints saved tx power configurations of all the sub-bands on the console output.*

```
# wlan-set-txpwrlimit
-------------------------------------------------------------------------------
-
StartFreq: 2407
ChanWidth: 20
ChanNum:   1
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   2
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   3
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   4
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   5
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   6
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   7
…
```

3.5.1.4 Set/Get Active/Passive Channel List

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

Set Channel List:

```
# wlan-set-chanlist
-------------------------------------------------------------------------------
-
Number of channels configured: 39

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  Passive
ChanNum: 13     ChanFreq: 2472  Passive
ChanNum: 14     ChanFreq: 2484  Passive
ChanNum: 36     ChanFreq: 5180  Active
…
```

Get Channel List:

```
# wlan-get-chanlist
-------------------------------------------------------------------------------
-
Number of channels configured: 39

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  Passive
ChanNum: 13     ChanFreq: 2472  Passive
ChanNum: 14     ChanFreq: 2484  Passive
ChanNum: 36     ChanFreq: 5180  Active
…
```

3.5.1.5 Set Channel List and Tx Power Limit

The following command is used to set channel list as well as tx power limit.

NOTE: *This command will first set defined configuration of channel list and tx power limit and then prints saved configuration on the console output.*

```
# wlan-set-chanlist-and-txpwrlimit
-------------------------------------------------------------------------------
-
Get txpwrlimit: sub_band=0
StartFreq: 2407
ChanWidth: 20
ChanNum:   1
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   2
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   3
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   4
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   5
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   6
Pwr:0,8,1,8,2,8,3,8,4,8,5,8,6,8
StartFreq: 2407
ChanWidth: 20
ChanNum:   7
…
Number of channels configured: 39
```

```
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  Passive
ChanNum: 13     ChanFreq: 2472  Passive
ChanNum: 14     ChanFreq: 2484  Passive
ChanNum: 36     ChanFreq: 5180  Active
…
```

3.5.1.6 Set/Get Tx Rate Configuration

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

Command Usage:

```
# wlan-set-txratecfg
Invalid arguments
Usage:
wlan-set-txratecfg <sta/uap> <format> <index> <nss> <rate_setting>
        Where
        <format> - This parameter specifies the data rate format used in this
command
                0:    LG
                1:    HT
                2:    VHT
                0xff: Auto
        <index> - This parameter specifies the rate or MCS index
        If <format> is 0 (LG),
                0       1 Mbps
                1       2 Mbps
                2       5.5 Mbps
                3       11 Mbps
                4       6 Mbps
                5       9 Mbps
                6       12 Mbps
                7       18 Mbps
                8       24 Mbps
                9       36 Mbps
                10      48 Mbps
                11      54 Mbps
        If <format> is 1 (HT),
                0       MCS0
                1       MCS1
                2       MCS2
                3       MCS3
                4       MCS4
                5       MCS5
                6       MCS6
                7       MCS7
        If <format> is 2 (VHT),
                0       MCS0
                1       MCS1
                2       MCS2
                3       MCS3
                4       MCS4
                5       MCS5
                6       MCS6
                7       MCS7
                8       MCS8
                9       MCS9
        <nss> - This parameter specifies the NSS. It is valid only for VHT and
HE
        If <format> is 2 (VHT) or 3 (HE),
                1       NSS1
                2       NSS2
        <rate_setting> - This parameter can only specifies the GI types now.
        If <format> is 1 (HT),
                0x0000  Long GI
                0x0020  Short GI
        If <format> is 2 (VHT),
                0x0000  Long GI
                0x0020  Short GI
                0x0060  Short GI and Nsym mod 10=9
```

Set Tx Rate:

```
# wlan-set-txratecfg ff 0
Configured txratecfg as below:
Tx Rate Configuration:
    Type:       0 (LG)
    Rate Index: 0 (1 Mbps)
```

Get Tx Rate:

```
# wlan-get-txratecfg
Tx Rate Configuration:
    Type:       0xFF (Auto)
    Rate Index: 0 (1 Mbps)
```

Get Data Rate:

```
# wlan-get-data-rate
Data Rate:
  TX:
    Type: LG
    Rate: 1 Mbps
  RX:
    Type: LG
    Rate: 1 Mbps
```

3.5.1.7 Set/Get Management Frame Protection Capability

The following commands are used to set and get MFP capability:

Command Usage:

```
# wlan-set-pmfcfg
Usage:
wlan-set-pmfcfg <mfpc> <mfpr>


    <mfpc>:   Management Frame Protection Capable (MFPC)
         1: Management Frame Protection Capable
         0: Management Frame Protection not Capable
    <mfpr>:   Management Frame Protection Required (MFPR)
         1: Management Frame Protection Required
         0: Management Frame Protection Optional
    Default setting is PMF not capable.
    mfpc = 0, mfpr = 1 is an invalid combination
```

Set MFP capability:

```
# wlan-set-pmfcfg 1 1
PMF configuration successful
```

Get MFP Capability:

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

3.5.1.8 Set/Get Antenna Diversity Configuration

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

NOTE: *Make sure second antenna is connected before performing antenna configurations.*

Command Usage:

```
# wlan-set-antcfg
Usage:
wlan-set-antcfg <ant mode> [evaluate_time]
    <ant mode>:
            Bit 0   -- Tx/Rx antenna 1
            Bit 1   -- Tx/Rx antenna 2
            0xFFFF  -- Tx/Rx antenna diversity
    [evaluate_time]:
            if ant mode = 0xFFFF, SAD evaluate time interval,
            default value is 6s(0x1770)
```

3.5.1.9 Set/Get 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 Energy Detection (ED) threshold offset (ed_ctrl_2g.offset and ed_ctrl_5g.offset) needs to be adjusted. The ED threshold offset can be adjusted in steps of 1 dB.

This section includes definitions of the commands and examples which shows how to adjust ED MAC. Below are the get and set commands for ED-MAC adjustment.

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

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

Where:

Table 16: ED MAC Parameters

Parameter	Description
interface	0 = STA 1 = uAP
ed_ctrl_2_g	0 = disable ED MAC threshold for 2.4GHz band 1 = enable ED MAC threshold for 2.4GHz 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 5GHz band 1 = enable ED MAC threshold for 5GHz 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 2.4GHz band:

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

Table 17: ED MAC 2.4 GHz Command Operations

Step	Operation	Command
1	Get ED-MAC status	**#wlan-get-ed-mac-mode 0** EU adaptivity for 2.4GHz band : Enabled Energy Detect threshold offset : 0X9
2	Set ED-MAC threshold	**#wlan-set-ed-mac-mode 0 1 0x8** ED MAC MODE settings configuration successful

For 5GHz band:

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

Table 18: ED MAC 5 GHz Command Operations

Step	Operation	Command
1	Get ED-MAC status	**#wlan-get-ed-mac-mode 0** EU adaptivity for 2.4GHz band : Enabled Energy Detect threshold offset : 0X9 EU adaptivity for 5GHz band : Enabled Energy Detect threshold offset : 0Xc
2	Set ED-MAC threshold	**#wlan-set-ed-mac-mode 0 1 0x9 1 0x3** ED MAC MODE settings configuration successful

3.6 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 something to this (demo) address from the remote computer need to specify the interface over which the demo is reachable by appending % followed by zone index. Please refer to section 2.4 for more details about zone index.

3.6.1 wifi_ipv4_ipv6_echo Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console setup and section 2.4 for ipv4/6 tool setup.

3.6.1.1 Run the application

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

```
========================================
Initialize WLAN Driver
========================================
MAC Address: 48:E7:DA:9A:CE:39
Initialize CLI
========================================

Copyright  2020  NXP
```

3.6.1.2 Help command

```
SHELL>> help

"help": List all the registered commands

"exit": Exit program

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

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

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

"end": Ends echo_* command.

"print_ip_cfg": Prints IP configuration.
```

```
       "wlan_scan": Scans networks.

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

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

3.6.1.3 Scan command

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

```
SHELL>> wlan_scan
Scanning
SHELL>>
Initiating scan...
NXP_V10
     BSSID          : 5C:DF:89:0F:32:78
     RSSI           : -67dBm
     Channel        : 1
nxp
     BSSID          : 8E:36:15:52:42:0C
     RSSI           : -51dBm
     Channel        : 11
…
```

3.6.1.4 Connect to available access point

Connect to the network using one of the following commands:

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

NOTE: *SSID (the name of the network) or BSSID (it's mac)*

```
wlan_connect_with_password nxp 12345678
Joining: nxp
Network joined
```

3.6.1.5 Print IP Configuration

This command will print IPv4 and IPv6 address of the board received from the external access point

```
SHELL>> print_ip_cfg
************************************************
 IPv4 Address     : 192.168.148.150
 IPv4 Subnet mask : 255.255.255.0
 IPv4 Gateway     : 192.168.148.93
 IPv6 Address0    : FE80::224E:F6FF:FEEC:1F27
 IPv6 Address1    : 2409:40C1:4E:7AFA:224E:F6FF:FEEC:1F27
 IPv6 Address2    : -
************************************************
```

NOTE: *It is necessary to have installed tools capable of sending and receiving data over TCP or UDP to interact with the demo. Please refer to the section 2.4 for tool setup.*

3.6.1.6 TCP client echo

Run ncat on 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 demo shell.

```
SHELL>> echo_tcp_client 192.168.148.80 10001

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

Verify connection from Remote host console. Type some text and hit enter, the demo will send line back.

```
C:\Users\nxp>
Ncat: Connection from 192.168.148.150.
Ncat: Connection from 192.168.148.150:49153.

hello
hello
```

Check console logs which shows number of bytes sent back to 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> 10001 in demo shell.

```
SHELL>> echo_tcp_client fe80::8f3d:b4b4:b64f:764d 10001

Creating new socket.
Connecting...
Connected.

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

Verify connection from Remote host console. Type some text and hit enter, the demo will send line back.

```
C:\Users\nxp>
Ncat: Connection from fe80::224e:f6ff:feec:1f27.
Ncat: Connection from fe80::224e:f6ff:feec:1f27:49153.

hello
hello
```

Check console logs which shows number of bytes sent back to Remote Host PC

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

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

3.6.1.7 TCP server echo

Run the command echo_tcp_server 10001 in demo shell.

```
SHELL>> echo_tcp_server 10001

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

IPv4

Run the command ncat -v <Demo IPv4 addr> 10001 on Remote host PC to connect with TCP server

```
C:\Users\nxp>ncat -v 192.168.148.150 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
```

Verify connection from Remote host console. Type some text and hit enter, the demo will send line back.

```
C:\Users\nxp>
Ncat: Connected to 192.168.148.150:10001.

hello
```

Check console logs which shows number of bytes sent back to Remote Host PC

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

ECHO_TCP_SERVER>>
6B sent back.
```

IPv6

Run the command ncat -v <Demo IPv6 addr *FE80::**** >> 10001 on Remote host PC to connect with TCP server

```
C:\Users\nxp>ncat -v FE80::224E:F6FF:FEEC:1F27 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to FE80::224E:F6FF:FEEC:1F27:10001.
```

Verify connection from Remote host console. Type some text and hit enter, the demo will send line back.

```
C:\Users\nxp>
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to FE80::224E:F6FF:FEEC:1F27:10001.

hello
```

Check console logs which shows number of bytes sent back to Remote Host PC

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

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

3.6.1.8 UDP echo

Run the command echo_udp 10001 in demo shell.

```
SHELL>> echo_udp 10001

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

IPV4

Run the command ncat -v -u <Demo IPv4 addr> 10001 on Remote host PC to connect with UDP server

```
C:\Users\nxp>ncat -v -u 192.168.148.150 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
```

Verify connection from Remote host console. Type some text and hit enter, the demo will send line back.

```
Ncat: Connected to 192.168.148.150:10001.

hello
```

Check console logs which shows number of bytes sent back to Remote Host PC

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

IPV6

Run the command ncat -v -u <Demo IPv6 addr *FE80::**** >> 10001 on Remote host PC to connect with UDP server

```
C:\Users\nxp>ncat -v -u FE80::224E:F6FF:FEEC:1F27 10001
Ncat: Version 7.92 ( https://nmap.org/ncat )
```

Verify connection from Remote host console. Type some text and hit enter, the demo will send line back.

```
Ncat: Connected to FE80::224E:F6FF:FEEC:1F27:10001.
hello
```

Check console logs which shows number of bytes sent back to Remote Host PC

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

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

3.7 uart_wifi_bridge Sample Application

This section describes the application to demonstrate bridging between Labtool and UART communication for i.MX RT 1170 EVKB host using NXP Wireless module IW611/612.

The following block diagram represents the application setup.

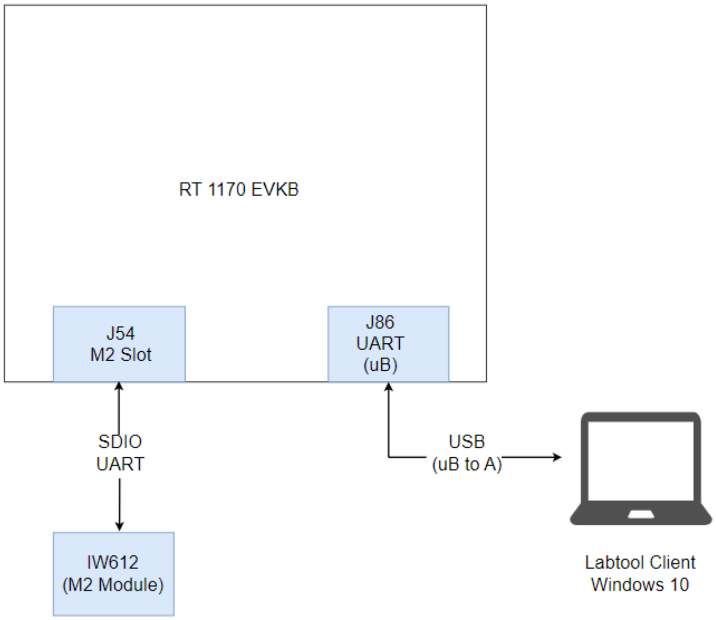


Figure 41: RT1170 EVKB Labtool setup

3.7.1 uart_wifi_bridge Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console tool setup.

3.7.1.1 Run the application

Labtool Setup on Windows PC

Step 1: Download the latest MFG Labtool release for IW612 from nxp.com

(for 8987 from nxp.com and for IW416 from nxp.com)

Step 2: Terminate all COM terminal programs connected to the UART port of the target.

Step 3: Connect the UART cable with target and get the UART COM port from Device Manager.

Step 4: Update the `<MFG-IW61X-MF-RTOS-BRG-WIN-X86>\bin\labtool\SetUp.ini` to reflect the COM port settings

```
[COMSET]
ComNo      = 3
BaudRate   =  115200
byParity   =  0
byStopBits = 1
byByteSize = 8
```

Step 5: Launch `<MFG-IW61X-MF-RTOS-BRG-WIN-X86>\bin\labtool\DutApiSisoApApp_W9177Uart.exe` and interact with DUT

Demo Execution

Step 1: Provide option 1 to check Labtool connection with DUT

```
Date:           Jul 20 2023 (02:39:35)

NOTE:

1. =========WiFi tool=============
2. =========BT   tool=============
3. =========15_4   tool=============

Enter CMD 99 to Exit

Enter option: 1
Name:           DutApiClass
Interface:      EtherNet
Version:        1.0.0.45.5
Date:           Jul 20 2023 (02:39:20)

NOTE:

 DutIf_InitConnection: 0
--------------------------------------------------------
                W9177 (802.11a/g/b/n/ac/ax) TEST MENU
--------------------------------------------------------
Enter option:
```

NOTE: *In above output, W9177 represents IW612*

Step 2: Get FW version with option 88

```
Enter option: 88
DLL Version : 1.0.0.45.5
LabTool Version: 1.0.0.45.5
FW Version:  18.80.2.49 Mfg Version: 2.0.0.63
SFW Version:  0.0.0.09  SHAL Version: 0.0.0.0
SOC OR Version: 1.2      Customer ID:   0
RF OR Version:  1.2      Customer ID:   0
Enter option:
```

Important Option values for all Wi-Fi feature related to labtool. (This is for quick reference)
In case of option not working refer step 1 for updated list.

Command Number	Description
5	Get Radio mode
6	Set Radio mode
9	Get Antenna
10	Set Antenna
11	Get RF Channel
12	Set RF Channel
13	Get RF Data Rate
22	Load Calibration Data File
29	Get RF Band
31	Clear received packet Count
32	Get received packet Count
35	Duty cycle Tx with SIFS gap
44	Get/Set Storage Type
45	Read MAC Address From OTP
46	Write MAC Address in OTP
53	Write calibration data from text files to OTP/.conf file
54	Get Calibration from OTP/.conf file into text files

88	Get Firmware and Labtool version
95	Get RF Crystal calibration offset
96	Set RF Crystal calibration offset
99	Exit
111	Get Channel BW
112	Set Channel BW
120	Get Thermal Sensor Reading
122	Enable IMD3 Calibration
123	Get IMD3 Calibration Data
198	Start RSSI Data Collection
199	Stop RSSI Data Collection and Report Result
225	HE rate Tx command (to enable sending trigger frame)
231	Configure HE TB (trigger-based) Tx
235	Generate Trigger frame using configuration file

3.8 wifi_wpa_supplicant Sample Application

The *wifi_wpa_supplicant* application demonstrates CLI support usage using wpa supplicant (host based). This application includes similar commands to *wifi_cli* application, some new commands/features which related to host based supplicant are covered in this section i.e WPA Enterprise, WPS.

Wi-Fi and iPerf Features:

Table 19: 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 IEEEPS power saving mode Wi-Fi deep-sleep power saving mode Wi-Fi host sleep/wowlan Wi-Fi RF Calibration Wi-Fi coexistence with external radios (for 88W8801) WPA Enterprise WPS Wi-Fi 11r roaming 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)

3.8.1 wifi_wpa_supplicant Application Execution

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console tool setup.

3.8.1.1 Start-up logs

The following logs can be observed on the console once the devices (*i.MX RT1060 EVK board and NXP-based Wireless module*) are up and running and it shows that Wi-Fi module is ready for the operations. This section describes the available Wi-Fi commands, press Enter for the command prompt.

```
========================================
wifi wpa supplicant demo
========================================
Initialize CLI
========================================
Initialize WLAN Driver
========================================
Setting up new cal data
MAC Address: 20:4E:F6:EC:1F:27
Info: supplicant_main_task: 241 Starting wpa_supplicant thread with debug
level: 3

Info: Successfully initialized wpa_supplicant
Info: iface_cb: iface ml1 ifindex 2 20:4e:f6:ec:1f:27
```

```
Info: Using interface ml1

Info: Initializing interface 0: ml1

========================================
app_cb: WLAN: received event 11
========================================
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
CLIs Available:
========================================

help
wlan-reset
wlan-version
wlan-mac
wlan-set-mac <MAC_Address>
wlan-scan
wlan-scan-opt ssid <ssid> bssid ...
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect <profile_name>
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-get-uap-channel
wlan-get-uap-sta-list
wlan-ieee-ps <0/1>
wlan-deep-sleep-ps <0/1>
wlan-host-11k-enable <0/1>
wlan-host-11k-neighbor-req [ssid <ssid>]
wlan-host-11v-bss-trans-query <0..16>
wlan-pmksa-list
wlan-pmksa-flush
wlan-set-scan-interval <scan_int: in seconds>
wlan-roaming <0/1>
wlan-host-sleep <0/1> wowlan <wake_up_conds>
wlan-send-hostcmd
wlan-set-uap-bandwidth <1/2> 1:20 MHz 2:40MHz
wlan-set-uap-hidden-ssid <0/1/2>
wlan-ft-roam <bssid> <channel>
wlan-set-antcfg <ant mode> [evaluate_time]
wlan-get-antcfg
wlan-scan-channel-gap <channel_gap_value>
wlan-get-regioncode
wlan-rssi-low-threshold <threshold_value>
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-get-signal
wlan-cloud-keep-alive <start/stop/reset>
```

```
ping [-s <packet_size>] [-c <packet_count>] [-W <timeout in sec>] <ipv4/ipv6
address>
iperf [-s|-c <host>|-a|-h] [options]
dhcp-stat
========================================
```

3.8.1.2 Add network profile

Before adding a network profile for Soft AP and Station mode, please check command usage for different EAP methods.

```
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> <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>] [owe_only] [wpa3 sae/ft-sae
<secret>] [mfpc <0/1> mfpr <0/1>]
For Micro-AP interface
    wlan-add <profile_name> ssid <ssid>
    ip:<ip_addr>,<gateway_ip>,<netmask>
    role uap [bssid <bssid>]
    [channel <channelnumber>]
    [wpa2 <psk/psk-sha256> <secret>] [wpa3 sae <secret> [sg <"19 20 21">] [pwe
<0/1/2>] [tr <0/1/2/4/8>]]
    [ft-psk <secret>] [wpa3 ft-sae <secret>]
    [owe_only [og <"19 20 21">]]
    [mfpc <0/1>] [mfpr <0/1>]

    [capa <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
```

3.8.1.3 Station mode (connect to External AP)

WPA3 Security

NOTE: *For WPA3 default mode is set to pwe 2 (both hunting-and-pecking loop and hash-to-element enabled)*

Usage for pwe and tr

```
SAE mechanism for PWE derivation
# 0 = hunting-and-pecking loop only (default without password identifier)
# 1 = hash-to-element only (default with password identifier)
# 2 = both hunting-and-pecking loop and hash-to-element enabled
```

WPA3 SAE (R1)

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 pwe 0 mfpc 1 mfpr 1
Added “nxp_test_1”
```

WPA3 SAE (R3)

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 pwe 1 mfpc 1 mfpr 1
Added “nxp_test_1”
```

WPA3 SAE (R3), with SAE group 20,21

```
# wlan-add nxp_test_1 ssid WPA3_AP wpa3 sae 12345678 sg “19 20 21” pwe 1 mfpc 1
mfpr 1
Added “nxp_test_1”
```

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

NOTE*: Here we make another RT as an External AP on which radius server is running. To generate own certificates please refer to the section 3.7.1.5.*

WPA2 Enterprise Security

Use the following command to add the network profile to configure the device in station mode using ***EAP-TLS*** method. Provide any profile name, external AP’s SSID, User ID and Password to authenticate with the server in argument shown below:

```
# wlan-add EapNet ssid EapNet_AP eap-tls id client1 key_passwd whatever
Added “abc”
```

Connect to the AP network using the saved network profile:

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

# Info: ml1: SME: Trying to authenticate with d8:c0:a6:0f:d6:89
(SSID='EapNet_AP' freq=2437 MHz)
Info: ml1: Trying to associate with d8:c0:a6:0f:d6:89 (SSID='EapNet_AP'
freq=2437 MHz)
Info: ml1: Associated with d8:c0:a6:0f:d6:89
Info: ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
Info: ml1: CTRL-EVENT-EAP-STARTED EAP authentication started
Info: ml1: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=13
Info: ml1: CTRL-EVENT-EAP-METHOD EAP vendor 0 method 13 (TLS) selected
Info: 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
Info: 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
Info: ml1: CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
Info: ml1: PMKSA-CACHE-ADDED d8:c0:a6:0f:d6:89 0
========================================
app_cb: WLAN: received event 1
========================================
app_cb: WLAN: authenticated to network
Info: ml1: WPA: Key negotiation completed with d8:c0:a6:0f:d6:89 [PTK=CCMP
GTK=CCMP]
Info: ml1: CTRL-EVENT-CONNECTED - Connection to d8:c0:a6:0f:d6:89 completed
[id=0 id_str=]
========================================
app_cb: WLAN: received event 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 will show Client successfully connected to AP with ssid "EapNet_AP" and got ip address "192.168.10.2" from AP.*

Get signal information of connected External AP

```
# wlan-get-signal
        BeaconLast      Beacon Average  Data Last       Data Average
RSSI    -32             -32             -33             -33
SNR     58              58              57              57
NF      -90             -90             -90             -90
```

Get PMKSA list

```
# wlan-pmksa-list
PMKSA list
Index / AA / PMKID / expiration (in seconds) / opportunistic
1 d8:c0:a6:0f:d6:89 9ca541d20dcc1cbc3ae0834d54c816b4 43187 0
```

To flush the PMKSA entries

```
# wlan-pmksa-flush
Info: ml1: PMKSA-CACHE-REMOVED d8:c0:a6:0f:d6:89 0
Info: ml1: CTRL-EVENT-DISCONNECTED bssid=d8:c0:a6:0f:d6:89 reason=1
locally_generated=1
Info: ml1: CTRL-EVENT-DSCP-POLICY clear_all
Flushed PMKSA cache
```

WPA2 Station disconnection (from AP)

Disconnect from the AP network profile:

```
# wlan-disconnect

========================================
app_cb: WLAN: received event 9
========================================
app_cb: disconnected
```

Remove the saved network profile:

```
# wlan-remove EapNet
Info: ml1: PMKSA-CACHE-REMOVED d8:c0:a6:0f:d6:89 0
Removed "EapNet"
```

3.8.1.3.1 Channel State Information (CSI)

The CSI feature provides a method to send information about channel properties from Wi-Fi firmware to Host periodically. Once the CSI information is generated by the firmware, it will forward the CSI record (CSI header + CSI data) on a separate path from the actual packet received by the firmware. The header for the CSI record is extracted from the actual packet received.

***NOTE:** Define CONFIG_CSI macro in **wifi_config.h** to enable the feature.*

Set CSI config info

Usage:

```
# wlan-set-csi-param-header
Error: invalid number of arguments
```

```
Usage: wlan-set-csi-param-header <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
```

```
# wlan-set-csi-param-header 1 66051 66051 170 0 6 1 1

The current csi_param is:
csi_enable    : 1
head_id       : 66051
tail_id       : 66051
csi_filter_cnt: 0
chip_id       : 170
band_config   : 0
channel       : 6
csi_monitor_enable : 1
ra4us         : 1
```

Set CSI filter

Usage:

```
# 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
```

Issues the CSI command to Wi-Fi firmware

```
wlan-csi-cfg
```

Wi-Fi firmware receives the CSI packet, convert the CSI header based on the software definition, and passes it to the Host driver through the CSI event. The driver broadcasts the events with CSI header and data.

```
# CSI user callback: Event CSI data
**** Dump @ 2020F504 Len: 156 ****
27 00 cd ab 03 02 01 00 00 00 94 00 7c 05 32 6d
00 00 00 00 70 66 55 26 8a 6b 26 18 1d 56 65 0a
cb 01 a3 d9 28 06 02 aa 00 00 00 00 1b 00 00 00
```

```
00 00 00 00 10 eb f8 e8 ea f8 ee 0a fc 11 07 03
03 00 02 04 07 07 0f 01 11 f5 07 e8 f4 e5 e2 f2
de 08 eb 1d 02 24 17 19 14 f2 05 ee fa f6 fb 02
06 06 12 f9 fc da e0 e8 d8 02 e2 1c fa 28 15 22
1d f3 0a e8 f8 ed f2 fb f9 07 05 06 09 fc 02 f2
f3 f3 ea 01 f0 13 03 1b 16 10 0b e8 f5 ea eb fd
f5 0e 06 0b 00 00 00 00 03 02 01 00
******** End Dump *******
```

Steps to get CSI data in STA mode

Configure Ex-AP in 2.4GHz/5GHz with wpa2 psk security.
Connect STAUT to Ex-AP.
Enable CSI on STA: (**bold 36** is the channel on which AP is present)

```
#wlan-set-csi-param-header sta 1 66051 66051 170 1 36 0 1
```

Set CSI filter via below command:

```
#wlan-set-csi-filter add <ext-AP's MAC address> 255 255 0
```

Start CSI

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

Disable CSI on STA

```
#wlan-set-csi-param-header sta 2 66051 66051 170 1 36 0 1
```

Stop CSI

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

Steps to get CSI data in Soft AP mode

Configure DUT in 2.4Ghz/5GHz with wpa2 psk security.

Connect ext-STA to UAP.

Enable CSI on UAP: (**bold 36** is the channel on which AP is configured)

```
#wlan-set-csi-param-header uap 1 66051 66051 170 1 36 0 1
```

Set CSI filter

```
#wlan-set-csi-filter add <ext-STA's MAC address> 255 255 0
```

Start CSI

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

Disable CSI:

```
#wlan-set-csi-param-header uap 2 66051 66051 170 1 36 0 1
```

Stop CSI

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

3.8.1.3.2 WPA3- Enterprise

To use WPA3 Suite B or Suite B 192 bit enterprise security, add wpa3-sb or wpa3-sb-192 before EAP security type. Applicable for all EAP securities.

WPA3 EAP TLS (Suite B)

```
wlan-add EapNet ssid EapNet_AP wpa3-sb eap-tls id client2 key_passwd whatever
mfpc 1 mfpr 1
```

WPA3 EAP (Suite B 192 bit)

TLS

```
wlan-add EapNet ssid EapNet_AP wpa3-sb-192 eap-tls id client4 key_passwd
whatever mfpc 1 mfpr 1
```

TTLS

```
wlan-add EapNet ssid EapNet_AP wpa3-sb-192 eap-ttls-mschapv2 aid Client id
Client pass whatever key_passwd whatever mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc
0x1000
```

PEAP-v0-mschapv2

```
wlan-add EapNet ssid EapNet_AP wpa3-sb-192 eap-peap-mschapv2 ver 0 aid Client
id Client pass whatever key_passwd whatever mfpc 1 mfpr 1
```

PEAP-v1-mschapv2

```
wlan-add EapNet ssid EapNet_AP wpa3-sb-192 eap-peap-mschapv2 ver 1 aid Client
id Client pass whatever key_passwd whatever mfpc 1 mfpr 1
```

PEAP-v1-gtc

```
wlan-add EapNet ssid EapNet_AP wpa3-sb-192 eap-peap-gtc ver 1 aid Client id
Client pass whatever key_passwd whatever mfpc 1 mfpr 1
```

Connection Establish time

Asymmetric crypto is supported for i.MX RT117x platforms while it is not supported for i.MX RT10xx platforms, so the initial connection time for WPA3 Enterprise RT10xx is high compared to RT117x.

3.8.1.3.3 Other Security options:

OWE

Always set mfpc and mfpr to 1

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

WPS-PIN

```
# wlan-start-wps-pin 96288863
Start WPS PIN session with 96288863 pin
Info: ml1: WPS-PIN-ACTIVE
```

WPS-PBC

```
# wlan-start-wps-pbc
Info: ml1: WPS-PBC-ACTIVE
# Info: ml1: SME: Trying to authenticate with d8:c0:a6:0f:d6:89 (SSID='NXPAP'
freq=2437 MHz)
Info: ml1: Trying to associate with d8:c0:a6:0f:d6:89 (SSID='NXPAP' freq=2437
MHz)
Info: ml1: Associated with d8:c0:a6:0f:d6:89
```

AKA_PRIME_WPA2:

```
wlan-add 2 ssid RR1 eap-aka-prime id 6555444333222111 pass
5122250214c33e723a5dd523fc145fc0:981d464c7c52eb6e5036234984ad0bcf:000000000123
```

EAP_SIM_WPA2:

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

EAP_AKA_WPA2:

```
wlan-add 1 ssid EAP eap-aka id 0232010000000000 pass
90dca4eda45b53cf0f12d7c9c3bc6a89:cb9cccc4b9258e6dca4760379fb82581:000000000123
```

EAP_AKA_WPA3:

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

AKA_PRIME_WPA3:

```
wlan-add 2 ssid Suite-B-192  wpa3-sb-192 eap-aka-prime id 6555444333222111 pass
5122250214c33e723a5dd523fc145fc0:981d464c7c52eb6e5036234984ad0bcf:000000000123
mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

EAP_SIM_WPA3:

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

FAST-GTC:

```
wlan-add EapNet ssid EapNet_AP eap-fast-gtc aid client1 id user2 pass password2
key_passwd whatever mfpc 1 mfpr 0
```

FAST-mschapv2:

```
wlan-add EapNet ssid EapNet_AP eap-fast-mschapv2 aid client1 id user2 pass
password2 key_passwd whatever mfpc 1 mfpr 0
```

3.8.1.4 Soft AP mode

Use the following command to add the network profile to configure the device in Enterprise AP mode. Use your AP’s SSID, IP details, role, channel, security, user id and password in argument shown below.

NOTE*: To generate own certificates please refer to the section 3.7.1.5.*

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 id client2 id client3 id client4
key_passwd whatever

Added "EapNet"
```

Set ACS mode

The Automatic Channel Selection (ACS) mode can be enabled while adding the profile using wlan-add command. When channel parameter is set as 0 then it enables ACS mode.

Default value for ACS band is 0.

<acs_band> usage

```
# 0 = 2.4GHz
# 1 = 5GHz
```

AP with wpa2 psk security configured with 5 GHz ACS mode

```
# wlan-add xyz ssid NXPAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
channel 0 acs_band 1 wpa2 psk 12345678
```

AP with wpa2 psk security configured with 2.4 GHz ACS mode

```
# wlan-add xyz ssid NXPAP ip:192.168.10.1,192.168.10.1,255.255.255.0 role uap
channel 0 acs_band 0 wpa2 psk 12345678
```

Set Wi-Fi bandwidth

The following command is used to set Wi-Fi bandwidth (20MHz or 40MHz):

NOTE: *Default bandwidth is set to 40MHz if not set by following command.*

NOTE: *For 88W8801, default bandwidth is set to 20MHz and following command is not available.*

Command Usage:

```
# wlan-set-uap-bandwidth
Usage: wlan-set-uap-bandwidth <1/2/3>
Error: Specify 1 to set bandwidth 20MHz or 2 for 40MHz or 3 for 80MHz
```

Set bandwidth:

```
# wlan-set-uap-bandwidth 1
bandwidth set successfully
```

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.
Info: ua2: interface state UNINITIALIZED->COUNTRY_UPDATE

========================================
app_cb: WLAN: received event 15
========================================
app_cb: WLAN: UAP Started
========================================
Soft AP "EapNet_AP" started successfully
========================================
DHCP Server started successfully
========================================
```

Check created network details

```
# wlan-info
Station not connected
uAP started as:
"EapNet"
        SSID: EapNet_AP
        BSSID: D8:C0:A6:0F:D6:89
        mode: 802.11AC
        channel: 6
        role: uAP
        security: WPA2 Enterprise EAP-TLS
        wifi capability: 11ac
        user configure: 11ac

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

        IPv6 Addresses
        Link-Local   :  FE80::DAC0:A6FF:FE0F:D689 (Tentative)
```

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

```
# Info: ua2: STA 20:4e:f6:ec:1f:27 IEEE 802.11: associated (aid 1)
Info: : CTRL-EVENT-EAP-STARTED 20:4e:f6:ec:1f:27
Info: : CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
Info: : CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=13
Info: : CTRL-EVENT-EAP-RETRANSMIT 20:4e:f6:ec:1f:27
Info: ml1: CTRL-EVENT-EAP-PEER-CERT depth=1 subject='C=IN, ST=MH, L=PUNE,
O=NXP, CN=CA, emailAddress=ca@nxp.com'
hash=4f7f0a703ca723e3f0e5c7d11f7f5e0ec5d68975791370354f2a02
Info: ml1: CTRL-EVENT-EAP-PEER-CERT depth=0 subject='C=IN, ST=MH, L=PUNE,
O=NXP, CN=Client, emailAddress=client@nxp.com'
hash=8bb701aedec525fbc4934c3a53a00adbcfb86f8c30750b
Info: : CTRL-EVENT-EAP-SUCCESS 20:4e:f6:ec:1f:27
Info: ua2: STA 20:4e:f6:ec:1f:27 WPA: pairwise key handshake completed (RSN)
Info: : EAPOL-4WAY-HS-COMPLETED 20:4e:f6:ec:1f:27
Info: : AP-STA-CONNECTED 20:4e:f6:ec:1f:27
========================================
app_cb: WLAN: received event 16
========================================
app_cb: WLAN: UAP a Client Associated
```

```
========================================
Client => 20:4E:F6:EC:1F:27 Associated with Soft AP
========================================
Info: ua2: STA 20:4e:f6:ec:1f:27 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: 20:4E:F6:EC:1F:27
Power mfg status: active
Rssi : -69 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    20:4E:F6:EC:1F:27
```

SSID broadcast configuration:

User can control SSID IE configuration using this command.

It has 3 modes:

0: When user wants to enable SSID broadcast (default)

1: When user wants to disable SSID name(ASCII 0) and SSID length (Length = 0)

2: When user wants to disable only the SSID name (ASCII 0)

Command usage:

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

Set SSID broadcast control

```
# wlan-set-uap-hidden-ssid 1
SSID broadcast control set successfully
```

3.8.1.4.1 Other Security options

WPA3 Security

Note: *Default value of pwe is 0 for Soft AP*

Default value of tr is 0 for Soft AP

WPA3 SAE (R1)

```
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 pwe 0 mfpc 1 mfpr 1
```

WPA3 SAE (R3)

```
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 pwe 1 mfpc 1 mfpr 1
```

WPA3 SAE (R3), with capability set to 11AX

```
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 pwe 1 mfpc 1 mfpr 1 capa 11ax
```

WPA3 SAE (R3), Transition Disable set

```
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 pwe 1 tr 1 mfpc 1 mfpr 1
```

WPA3 SAE (R3), SAE group 20, 21

```
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 sg “19 20 21” pwe 1 mfpc 1 mfpr 1
```

OWE

Always set mfpc and mfpr to 1.

```
wlan-add oweNet ssid oweNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 36 owe_only mfpc 1 mfpr 1
```

WPA3 Enterprise

To use WPA3 Suite B or Suite B 192 bit enterprise security, add wpa3-sb or wpa3-sb-192 before EAP security type. Applicable for all EAP securities.

WPA3 EAP TLS (Suite B)

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap wpa3-sb eaptls id client1 id client2 id client3 id client4 key_passwd
whatever mfpc 1 mfpr 1
```

WPA3 EAP (Suite B 192 bit)

TLS

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap wpa3-sb-192 eap-tls id client1 id client2 id client3 id client4 key_passwd
whatever mfpc 1 mfpr 1
```

TTLS

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 36 wpa3-sb-192 eap-ttls-mschapv2 aid Client id Client pass whatever
key_passwd whatever mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

PEAP-v0-mschapv2

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 36 wpa3-sb-192 eap-peap-mschapv2 ver 0 aid Client id Client pass
whatever key_passwd whatever mfpc 1 mfpr 1
```

PEAP-v1-mschapv2

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 36 wpa3-sb-192 eap-peap-mschapv2 ver 1 aid Client id Client pass
whatever key_passwd whatever mfpc 1 mfpr 1
```

PEAP-v1-gtc

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 36 wpa3-sb-192 eap-peap-gtc ver 1 aid Client id Client pass
whatever key_passwd whatever mfpc 1 mfpr 1
```

WPS (Wi-Fi Protected Setup)

There are two primary approaches to network setup within Wi-Fi Protected Setup: push-button and PIN entry.

WPS-PIN

```
# wlan-add abc ssid NXPAP ip:192.168.81.100,192.168.81.100,255.255.255.0 role
uap channel 6 wpa2 12345678
Added "abc"
```

```
# wlan-start-network abc
[wlcm] Warn: NOTE: uAP will automatically switch to the channel that station is
on.
Info: ua2: interface state UNINITIALIZED->COUNTRY_UPDATE
```

```
# ========================================
app_cb: WLAN: received event 15
========================================
app_cb: WLAN: UAP Started
========================================
Soft AP "NXPAP" started successfully
========================================
DHCP Server started successfully
========================================
Info: ua2: interface state COUNTRY_UPDATE->ENABLED
Info: : AP-ENABLED
```

```
# wlan-generate-wps-pin
WPS PIN is: 96288863
```

```
# wlan-start-ap-wps-pin 96288863
Start AP WPS PIN session with 96288863 pin
[uap] Warn: Overwriting previous configuration
```

WPS-PBC

```
wlan-add abc ssid NXPAP ip:192.168.81.100,192.168.81.100,255.255.255.0 role uap
channel 6 wpa2 12345678
```

```
# wlan-start-ap-wps-pbc add
[uap] Warn: Overwriting previous configuration
Info: : WPS-PBC-ACTIVE
```

EAP-Sim, AKA and AKA-prime

For eap-sim/eap-aka/eap-aka-prime use command ***read_gsm_triplets*** to add GSM authentication triplets and ***read_milenage*** to add Milenage keys and ***hlr_cli*** to start hlr_auc_gw

Usage:

```
read_gsm_triplets <imsi> <kc> <sres> <rand>
read_milenage <imsi> <ki> <opc> <amf> <sqn>
hlr_cli <standard hlr cli options>
```

Example:

```
# read_gsm_triplets 234567898765432 A0A1A2A3A4A5A6A7 D1D2D3D4
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
# read_gsm_triplets 234567898765432 B0B1B2B3B4B5B6B7 E1E2E3E4
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

```
# read_gsm_triplets 234567898765432 C0C1C2C3C4C5C6C7 F1F2F3F4
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
# read_milenage 232010000000000 90dca4eda45b53cf0f12d7c9c3bc6a89
cb9cccc4b9258e6dca4760379fb82581 61df 000000000000
```

```
# read_milenage 555444333222111 5122250214c33e723a5dd523fc145fc0
981d464c7c52eb6e5036234984ad0bcf c3ab 16f3b3f70fc1
```

```
# hlr_cli
```

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
```

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
```

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
```

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 mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

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 mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

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 mfpc 1 mfpr 1 gc 0x100 pc 0x100 gmc 0x1000
```

FAST-GTC:

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap eap-fast-mschapv2 aid client1 id user1 pass password1 id user2 pass
password2 id user3 pass password3 id user4 pass password4 key_passwd whatever
pac_opa_enc_key 000102030405060708090a0b0c0d0e0f a_id 0123456789abcd01
fast_prov 2 mfpc 1 mfpr 0
```

FAST-mschapv2:

```
wlan-add EapNet ssid EapNet_AP ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap eap-fast-mschapv2 aid client1 id user1 pass password1 id user2 pass
password2 id user3 pass password3 id user4 pass password4 key_passwd whatever
pac_opa_enc_key 000102030405060708090a0b0c0d0e0f a_id 0123456789abcd01
fast_prov 2 mfpc 1 mfpr 0
```

Stop Soft AP

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

3.8.1.5 Certificates/Key configurations for WPA2/3 Enterprise

For enterprise security it is mandatory to have a radius server (hostapd radius server) and server/client certificates. This section describes how user can configure their own CA certificate, Client/Server certificate, Client/Server private key for WPA2/3 Enterprise.

RT SDK supports certificates in .h format and already configured server and client certificates available at the location ***<SDK_PATH>/middleware/wifi_nxp/certs***. User need to replace ca-cert.h, client-cert.h and client-key.h files with newly created files.
Follow below steps for certificate conversion.

NOTE: *Below commands should be executed from 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 a PEM private key to a 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 array name and size inside .h as below:

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

client-cert

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

change array name and size inside .h as below:

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

client-key

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

change array name and size inside .h as below:

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

3.8.1.6 Independent Reset (IR)

The IR feature intended to be used to reset Wi-Fi and Bluetooth firmware, when it encounters a firmware fatal error. The following commands are used to trigger firmware fatal error manually to verify the feature.

NOTE: *Define CONFIG_WIFI_IND_DNLD and CONFIG_WIFI_IND_RESET macros in* ***wifi_config.h*** *to enable the feature. This feature is only enabled for i.MX RT1060 EVKC and RT1170 EVKB*

Following commands are used to reset Wi-Fi and Bluetooth firmware over SDIO interface.

Default mode set to in-band.

Usage :

```
# wlan-set-indrstcfg
Usage :
         wlan-set-indrstcfg <ir_mode> [gpio_pin]
         ir_mode   : 0 -- Disable
                     1 -- Enable out band reset, disable in band
                     2 -- Enable in band, disable out band
         gpio_pin  : 255 -- Default pin for reset
                     any other number for changing the gpio for reset.
Example :
         wlan-set-indrstcfg 1 255   : Set default pin as reset pin
         wlan-set-indrstcfg 0       : Disable the independent reset
         wlan-set-indrstcfg 2       : Enable in band reset mode
```

Set via In-band

In-Band Independent Reset(IB-IR) sends FW module reset signal over SDIO interface itself.

```
wlan-set-indrstcfg 2
```

Get current mode

```
# wlan-get-indrstcfg
Independent Reset Mode = In Band
```

Trigger manual FW crash using independent reset command

```
# wlan-independent-reset
[wifi] Warn: Command response timed out. command 0x8b, len 12, seqno 0x0
Independent reset success
```

Set via Out-of-band

Out of Band Independent Reset(OoB-IR) feature allows user to reset FW module over external signal(GPIO) rather than the default SDIO interface.

Note: *For 1XK and 1ZM M.2 module connect Fly-Wire between J16.1 and J108.4 of i.MX RT1060EVKC, J108 is routed on M2.P48 which internally routed on IR GPIO[15] of Controller 1XK/1ZM. For 2EL-M2, No fly-wire connection required.*

GPIO for 2EL is 1 and for 1XK/1ZM it is 14

```
wlan-set-indrstcfg 1 1
```

Trigger manual FW crash using independent reset command

```
# wlan-independent-reset 1
```

3.8.1.7 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.

NOTE: *This feature is only supported on IW612.*

This section describes:

- **The test procedure of Wi-Fi easy connect (DPP) using CLI commands supported in *wifi_wpa_supplicant* sample app**
- **Configuration of Wi-Fi devices in STA and AP modes**
- **Connection of STA and AP devices using DPP**

DPP QR code test setup:

- **DUT (STA) act as Enrollee, Initiator(Authentication)**
- **Device1 (External STA) act as configurator**
- **Device2 (External AP) acts as responder and enrollee**

Step 1: Start the Soft AP on Device2

```
# wlan-set-mac 00:50:43:02:11:01
```

```
# wlan-add testAP ssid DPPNET01 ip:192.168.10.1,192.168.10.1,255.255.255.0 role
uap channel 11 wpa2 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=WW

# ua2: interface state COUNTRY_UPDATE->ENABLED
: AP-ENABLED
========================================
app_cb: WLAN: received event 16
========================================
app_cb: WLAN: UAP Started
========================================
Soft AP "DPPNET01" started successfully
========================================
DHCP Server started successfully
========================================
```

Step 2: Generate QR code on Device2

```
# wlan-dpp-bootstrap-gen "type=qrcode chan=81/11 mac= A0:CD:F3:77:E7:1C"

bootstrap generate id = 1
```

NOTE: *MAC address of Device2 should input in above command and returned value "1" is bootstrap info id which require to get QR code string*

Get QR code URI

```
# wlan-dpp-bootstrap-get-uri 1

Bootstrapping QR Code URI:
```

```
DPP:C:81/11;M:a0cdf377e71c;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADMgoJ7zgcGN
PpoWKZtaapts0wBjJLUFTL9dgxqj3nb68=;;
```

NOTE: *This QR code will be use on Device1 with command wlan-dpp-qr-code.*

Step 3: Configure Device1 as configurator

Set MAC address

```
# wlan-set-mac 00:50:43:02:11:03
```

Add as a configurator

```
# wlan-dpp-configurator-add

conf_id = 1
```

Step 4: Authenticate Device1 with Device2

```
wlan-dpp-qr-code DPP:C:81/11;M:a0cdf377e71c;V:3;K:M…

DPP qr code id = 1
```

NOTE: *On successfully adding QR Code, a bootstrapping info id is returned as shown 1 in above command and should input in below command DPP_AUTH_INIT*

```
# wlan-dpp-auth-init " peer=1 conf=ap-dpp ssid=4450504e45543031 configurator=1"

[wlcm] Warn: ieee ps not enabled yet: 0
[wlcm] Warn: deep sleep ps not enabled yet: 0
ml1: DPP-TX dst=a0:cd:f3:77:e7:1c freq=2462 type=0

 DPP Auth Init OK!

# ml1: DPP-TX-STATUS dst=a0:cd:f3:77:e7:1c freq=2462 result=SUCCESS
ml1: DPP-RX src=a0:cd:f3:77:e7:1c freq=2462 type=1
ml1: DPP-AUTH-DIRECTION mutual=0
ml1: DPP-TX dst=a0:cd:f3:77:e7:1c freq=2462 type=2
ml1: DPP-TX-STATUS dst=a0:cd:f3:77:e7:1c freq=2462 result=SUCCESS
ml1: DPP-AUTH-SUCCESS init=1
ml1: DPP-CONF-REQ-RX src=a0:cd:f3:77:e7:1c
ml1: DPP-RX src=a0:cd:f3:77:e7:1c freq=2462 type=11
ml1: DPP-CONF-SENT
```

NOTE: *ssid should be hex string, here ssid=4450504e45543031 is hex string of DPPNET01*

Output on Device2

```
: DPP-RX src=a0:cd:f3:77:e4:36 freq=2462 type=0
: DPP-TX dst=a0:cd:f3:77:e4:36 freq=2462 type=1
: DPP-TX-STATUS dst=a0:cd:f3:77:e4:36 result=SUCCESS
: DPP-RX src=a0:cd:f3:77:e4:36 freq=2462 type=2
: DPP-AUTH-SUCCESS init=0
: GAS-QUERY-START addr=a0:cd:f3:77:e4:36 dialog_token=0 freq=2462
: GAS-QUERY-DONE addr=a0:cd:f3:77:e4:36 dialog_token=0 freq=2462 status_code=0
result=SUCCESS
: DPP-CONF-RECEIVED
: DPP-CONFOBJ-AKM dpp
: DPP-CONFOBJ-SSID DPPNET01
: DPP-CONNECTOR
eyJ0eXAiOiJkcHBDb24iLCJraWQiOiJ5dXhXNEFEVzdEcEowazhDbUVtenVmZzN5ZldtTW5lS1pVamF
rWXRXTjFJIiwiYWxnIjoiRVMyNTYifQ.eyJncm91cHMiOlt7Imdyb3VwSWQiOiIqIiwibmV0Um9sz
: DPP-C-SIGN-KEY
3039301306072a8648ce3d020106082a8648ce3d030107032200002939ea2def528cf4556c737f36
8bfb4346aa3ef4a86c836c301d036c5394e3925
```

```
: DPP-NET-ACCESS-KEY
307702010104205d7f4e0e0723ae7d4998115b73a00b5ed31e3da542ef8da3ab735698884a7f46a
00a06082a8648ce3d030107a1440342000434lb65763b3fafb301587fd383cdd8f2fa862
: DPP-TX dst=a0:cd:f3:77:e4:36 freq=2462 type=11
: DPP-TX-STATUS dst=a0:cd:f3:77:e4:36 result=SUCCESS
```

Step 5: Generate QR code on Device1 (configurator)

```
# wlan-dpp-configurator-params " conf=sta-dpp ssid=<hex_ascii> configurator=1"
```

NOTE: *space character exists between “ and conf.*

```
# wlan-dpp-bootstrap-gen "type=qrcode chan=81/11 mac=A0:CD:F3:77:E4:36"

 bootstrap generate id = 2
```

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

Bootstrapping QR Code URI:

DPP:C:81/11;M:a0cdf377e436;V:3;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACHsnUedxM3b
Gf6rXR0hETPAebTy8hHvKR1CRb1D6QqfA=;;
```

NOTE: *This QR code will be use on DUT with command DPP_QR_CODE.*

Step 6: Put Device1 in listening mode on specified channel

```
# wlan-dpp-listen "2462 role=configurator"

[wlcm] Warn: ieee ps not enabled yet: 0
[wlcm] Warn: deep sleep ps not enabled yet: 0

 DPP Listen OK!
```

Step 7: Authenticate DUT(STA) on Device1(STA)

```
# wlan-set-mac 00:50:43:02:11:02

# wlan-dpp-qr-code DPP:C:81/11;M:a0cdf377e436;V:3;K:…

DPP qr code id = 1
```

NOTE: *On successfully adding QR Code, a bootstrapping info id is returned as shown 1 in above command and should input in below command DPP_AUTH_INIT*

```
# wlan-dpp-auth-init " peer=1 role=enrollee"
```

Successful connection between DUT (STA) and Device2 (AP) can be verify using ping command.

3.8.1.8 wlan-cloud-keep-alive

The cloud keep alive feature provides a method to send keep-alive packets from Wi-Fi to cloud server periodically. In Host suspend state, Wi-Fi firmware will send keep-alive packets to cloud server periodically. For every keep-alive packet sent, firmware will receive the ACK from cloud server, if no ACK from server on 3 packets continuously, it indicates keep alive failure.

This section describes:

- **The test procedure of cloud keep alive (TCP keep alive) using CLI commands on IW612 with i.MX RT1170 EVKB board**
- **Configuration of keep-alive parameters (TCP/IP header info. etc.) in Wi-Fi firmware for WoWLAN.**

Test Setup:

- **DUT act as STA**
- **Configure external AP with open security**

- **Cloud server running in AP backend**
- **Connect probe between pin 2 of J9 port (on RT1170 EVKB) with HD3 GPIO 17 (on Murata uSD M.2) using probe.**

Step 1: Configure DUT in STA mode

```
# wlan-add abc ssid ASUS_2G
Added "abc"
```

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 7c:10:c9:02:da:48 (SSID='ASUS_2G'
freq=2412 MHz)
ml1: Trying to associate with 7c:10:c9:02:da:48 (SSID='ASUS_2G' freq=2412 MHz)
ml1: Associated with 7c:10:c9:02:da:48
========================================
app_cb: WLAN: received event 1
========================================
app_cb: WLAN: authenticated to network
ml1: CTRL-EVENT-CONNECTED - Connection to 7c:10:c9:02:da:48 completed [id=0
id_str=]
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [ASUS_2G]
IPv4 Address: [192.168.0.123]
```

Step 3: Start server in AP backend

Step 4: Run cloud keep alive command on DUT

Command Usage:

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

Table 20: cloud keep alive command usage

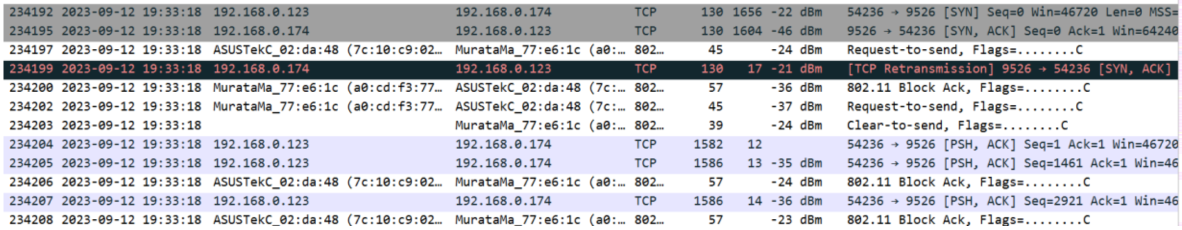
Command Parameters	Description
<dst_mac>	Destination MAC address
<dst_ip>	Destination IP
<dst_port>	Destination port

```
# wlan-cloud-keep-alive start id 0 dst_mac a4:fc:77:49:81:e7 dst_ip
192.168.0.174 dst_port 9526
```

Step 5: Make TCP connection

```
# wlan_tcp_client dst_ip 192.168.0.174  src_port 54236 dst_port 9526
```

Step 6: Verify the TCP-connection on sniffer



```
234192 2023-09-12 19:33:18  192.168.0.123                      192.168.0.174              TCP      130 1656 -22 dBm   54236 → 9526 [SYN] Seq=0 Win=46720 Len=0 MSS=
234195 2023-09-12 19:33:18  192.168.0.174                      192.168.0.123              TCP      130 1604 -46 dBm   9526 → 54236 [SYN, ACK] Seq=0 Ack=1 Win=64240
234197 2023-09-12 19:33:18  ASUSTekC_02:da:48 (7c:10:c9:02…  MurataMa_77:e6:1c (a0:…  802…      45      -24 dBm   Request-to-send, Flags=........C
234199 2023-09-12 19:33:18  192.168.0.174                      192.168.0.123              TCP      130   17 -21 dBm   [TCP Retransmission] 9526 → 54236 [SYN, ACK]
234200 2023-09-12 19:33:18  MurataMa_77:e6:1c (a0:cd:f3:77…  ASUSTekC_02:da:48 (7c:…  802…      57      -36 dBm   802.11 Block Ack, Flags=........C
234202 2023-09-12 19:33:18  MurataMa_77:e6:1c (a0:cd:f3:77…  ASUSTekC_02:da:48 (7c:…  802…      45      -37 dBm   Request-to-send, Flags=........C
234203 2023-09-12 19:33:18                                     MurataMa_77:e6:1c (a0:…  802…      39      -24 dBm   Clear-to-send, Flags=........C
234204 2023-09-12 19:33:18  192.168.0.123                      192.168.0.174              TCP     1582   12            54236 → 9526 [PSH, ACK] Seq=1 Ack=1 Win=46720
234205 2023-09-12 19:33:18  192.168.0.123                      192.168.0.174              TCP     1586   13 -35 dBm   54236 → 9526 [PSH, ACK] Seq=1461 Ack=1 Win=46
234206 2023-09-12 19:33:18  ASUSTekC_02:da:48 (7c:10:c9:02…  MurataMa_77:e6:1c (a0:…  802…      57      -24 dBm   802.11 Block Ack, Flags=........C
234207 2023-09-12 19:33:18  192.168.0.123                      192.168.0.174              TCP     1586   14 -36 dBm   54236 → 9526 [PSH, ACK] Seq=2921 Ack=1 Win=46
234208 2023-09-12 19:33:18  ASUSTekC_02:da:48 (7c:10:c9:02…  MurataMa_77:e6:1c (a0:…  802…      57      -23 dBm   802.11 Block Ack, Flags=........C
```

Step 7: Add arp entry on AP backend.

```
 arp -s <STAUT ip address> <STAUT mac address>
```

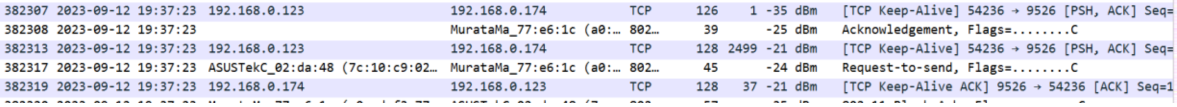
Step 8: Run host-sleep command (16 is for all ARP Broadcast Condition wherein DUT should only wakeup on Broadcast ping)

```
# wlan-host-sleep 1 wowlan 16
```

Step 9: Put HOST on suspend state

```
# mcu-suspend
```

Once the DUT entered into sleep state, following packets can be observed on sniffer



```
382307 2023-09-12 19:37:23  192.168.0.123                      192.168.0.174              TCP      126    1 -35 dBm   [TCP Keep-Alive] 54236 → 9526 [PSH, ACK] Seq=
382308 2023-09-12 19:37:23                                     MurataMa_77:e6:1c (a0:…  802…      39      -25 dBm   Acknowledgement, Flags=........C
382313 2023-09-12 19:37:23  192.168.0.123                      192.168.0.174              TCP      128 2499 -21 dBm   [TCP Keep-Alive] 54236 → 9526 [PSH, ACK] Seq=
382317 2023-09-12 19:37:23  ASUSTekC_02:da:48 (7c:10:c9:02…  MurataMa_77:e6:1c (a0:…  802…      45      -24 dBm   Request-to-send, Flags=........C
382319 2023-09-12 19:37:23  192.168.0.174                      192.168.0.123              TCP      128   37 -21 dBm   [TCP Keep-Alive ACK] 9526 → 54236 [ACK] Seq=1
```

Step 10: Stop or reset cloud keep-alive connection

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

OR

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

3.8.1.9 Wireless Location Service (WLS) using IEEE 802.11mc and IEEE 802.11az

WLS used to measure distance between two devices using Round Trip time of Flight(TOF) of RF signals. It meant to operate within Wi-Fi infrastructure to deliver precise location determination up to 1 to 2 meter ranging accuracy. This feature is based on the Fine Timing Measurement (FTM) protocol and enables a Wi-Fi station (STA) to estimate its distance relative to one or more fixed position Wi-Fi access points (APs) in the network.

This section explains the steps for a STA to use Wi-Fi Location™ to measure the distance from a single fixed position AP using the IEEE 802.11mc or 802.11az standard.

In addition, both 802.11mc and 802.11az support the following two modes:

Unassociated:

Initiator (STA) and Responder (AP) are not connected. Both take measurements given the MAC address and channel.

Associated:

Initiator (STA) and Responder (AP) are connected and take measurements.

NOTE: *Define CONFIG_11MC, CONFIG_11AZ, CONFIG_CSI, and CONFIG_WLS_CSI_PROC to enable the feature.*

3.8.1.9.1 IEEE 802.11mc

This is known as enhanced distributed channel access (EDCA) ranging. The distance is measured during the FTM session.

NOTE*: EDCA 802.11mc ranging is supported as a legacy mode.*

Configuration commands and usage:

# wlan-civ-cfg

This command used to set CIVC information configuration, which means user's civil information i.e address, country etc..

Usage:

```
wlan-civ-cfg civ_req <civ_req> loc_type <loc_type> country_code <country_code>
addr_type <addr_type>

civ_req: 0 or 1
loc_type: 1
country_code: 0 for USA
addr_type: 22
Example:
wlan-civ-cfg civ_req 1 loc_type 1 country_code 0 addr_type 22
```

# wlan-loc-cfg

This command used to set global user location

Usage:

```
wlan-loc-cfg lci_req <lci request> latit <latitude> longi <longitude> altit
<altitude> lat_uncert <latitude uncertainity> lon_uncert <longitude
uncertainity> alt_uncert <altitude uncertainity>

lci_req: 0 or 1
latitude: -180.0 to 180.0
longitude: -180.0 to 180.0
altitude: -180.0 to 180.0
latitude uncertainity: 0 to 255
longitude uncertainity: 0 to 255
altitude uncertainity: 0 to 255
Example:
wlan-loc-cfg lci_req 1 latit -111.111 longi 222.222 altit 33.333 lat_uncert 1
lon_uncert 2 alt_uncert 3
```

# wlan-11mc-nego-cfg

This command used to configure 11mc negotiation parameters.

Usage:

```
wlan-11mc-nego-cfg burst_dur <burst_dur> min_delta <min_delta> asap <asap>
ftm_per_burst <ftm_per_burst> BW <bw> burst_period <burst_period>

burst_dur: 2 to 11
min_delta: 1 to 63
asap: 0 or 1
ftm_per_burst: 2 to 10
BW: 9 to 13
burst_period: 1 to 10
Example:
wlan-11mc-nego-cfg burst_dur 11 min_delta 60 asap 1 ftm_per_burst 5 BW 13
burst_period 10
```

# wlan-ftm-ctrl

This command used to start and stop FTM session

Usage:

```
wlan-ftm-ctrl <action> loop_cnt <count> channel <channel> mac <peer_mac>

action:
 1: Start non-secure 11mc/11az FTM with associated Peer AP
 2: Stop FTM session
 3: Start secure 11az FTM with associated Peer AP
 4: Start non-secure 11az/11mc FTM with unassoc Peer
 5: Start secure 11az FTM with unassociated & pre-authenticated Peer
loop_cnt: number of ftm sessions to run repeatedly (default:1, 0:non-stop,
n:times>)
channel: Channel on which FTM must be started
mac: Mac address of the peer with whom FTM session is required

Example:
Run non-secure FTM session:
wlan-ftm-ctrl 1 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Runs secure 11az FTM session:
wlan-ftm-ctrl 3 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Runs non-secure FTM session with unassoc peer until user terminate:
wlan-ftm-ctrl 4 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Runs Secure FTM session with unassociated Peer AP:
wlan-ftm-ctrl 5 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Stop the FTM session:
wlan-ftm-ctrl 2
```

Output:

```
# wlan-civ-cfg civ_req 1 loc_type 1 country_code 0 addr_type 22
# wlan-loc-cfg lci_req 1 latit -111.111 longi 222.222 altit 33.333 lat_uncert 1
lon_uncert 2 alt_uncert 3
# wlan-11mc-nego-cfg burst_dur 11 min_delta 60 asap 1 ftm_per_burst 5 BW 13
burst_period 10
# wlan-ftm-ctrl 1 loop_cnt 1 channel 44 mac A2:CD:F3:77:E5:70

FTM Session Complete:
=====================
Average RTT: 63633431 ns
Average Clockoffset:34484 ns
Distance: 5.172600 meters
```

3.8.1.9.2 IEEE 802.11az

This is known as Next Generation Positioning.

Configuration commands and usage:

# wlan-11az-rang-cfg

Usage:

```
wlan-11az-rang-cfg <protocol> format_bw <format_bw> num_measurements
<num_measurements> measurement_freq <measurement_freq> i2r_sts <i2r_sts>
r2i_sts <r2i_sts> i2r_lmr <i2r_lmr>

protocol: 1
format_bw: 0 to 2
num_measurements: 1 to 10
measurement_freq: 1 to 10
i2r_sts: 0/1 - Num of antennas: 0=>1 antenna and 1=>2 antennas
r2i_sts: 0/1 - Num of antennas: 0=>1 antenna and 1=>2 antennas
i2r_lmr: 0 never, 1 always, 2 up to RSTA
Example:
wlan-11az-rang-cfg 1 format_bw 2 num_measurements 5 measurement_freq 4 i2r_sts
0 r2i_sts 0 i2r_lmr 0
```

# wlan-ftm-ctrl

Usage:

```
wlan-ftm-ctrl <action> loop_cnt <count> channel <channel> mac <peer_mac>

action:
 1: Start non-secure 11mc/11az FTM with associated Peer AP
 2: Stop FTM session
 3: Start secure 11az FTM with associated Peer AP
 4: Start non-secure 11az/11mc FTM with unassoc Peer
 5: Start secure 11az FTM with unassociated & pre-authenticated Peer

loop_cnt: number of ftm sessions to run repeatedly (default:1, 0:non-stop,
n:times>)

channel: Channel on which FTM must be started

mac: Mac address of the peer with whom FTM session is required

Example:
Run non-secure FTM session:
wlan-ftm-ctrl 1 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Runs secure 11az FTM session:
wlan-ftm-ctrl 3 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Runs non-secure FTM session with unassoc peer until user terminate:
wlan-ftm-ctrl 4 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Runs Secure FTM session with unassociated Peer AP:
wlan-ftm-ctrl 5 loop_cnt 1 channel 36 mac 00:50:43:20:bc:44
Stop the FTM session:
wlan-ftm-ctrl 2
```

Output:

```
# wlan-11az-rang-cfg 1 format_bw 2 num_measurements 5 measurement_freq 4
i2r_sts 0 r2i_sts 0 i2r_lmr 0
# wlan-ftm-ctrl 3 loop_cnt 1 channel 44 mac A2:CD:F3:77:E5:70
On giving the above commands, following output should be displayed on console:
FTM Session Complete:
=====================
Average RTT: 63633431 ns
Average Clockoffset:34484 ns
Distance: 5.172600 meters
```

3.8.1.10 WLAN offload feature

In this feature host can go in low power mode and FW will handle the reply (without waking up the host) to the configured frames like ARP, NS frame, TCP keepalive frames.

3.8.1.10.1 ARP Offload

For ARP offload, steps are as follows:

Step 1: Configure DUT in STA mode

```
# wlan-add abc ssid ASUS_2G
Added "abc"
```

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 7c:10:c9:02:da:48 (SSID='ASUS_2G'
freq=2412 MHz)
ml1: Trying to associate with 7c:10:c9:02:da:48 (SSID='ASUS_2G' freq=2412 MHz)
ml1: Associated with 7c:10:c9:02:da:48
========================================
app_cb: WLAN: received event 1
========================================
app_cb: WLAN: authenticated to network
ml1: CTRL-EVENT-CONNECTED - Connection to 7c:10:c9:02:da:48 completed [id=0
id_str=]
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
```

```
========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [ASUS_2G]
IPv4 Address: [192.168.0.123]
```

Step 3: Configure auto arp offload feature on DUT using below command

```
# wlan-auto-arp
```

Step 4: Configure host-sleep params on DUT using below command

(For wlan-host-sleep option please refer section 3.1.5.10)

```
# wlan-host-sleep 1 wowlan 1
```

Step 5: Put HOST on suspend state

```
# mcu-suspend
```

Step 6: Run arping to DUT from AP backend. One or more than one arp response should be seen from DUT after sending broadcast arping. DUT should not wakeup.

3.8.1.10.2 ARP Offload

For NS(Neighbor Solicitation) offload, steps are as follows:

Step 1: Configure DUT in STA mode

```
# wlan-add abc ssid ASUS_2G
Added "abc"
```

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 7c:10:c9:02:da:48 (SSID='ASUS_2G'
freq=2412 MHz)
ml1: Trying to associate with 7c:10:c9:02:da:48 (SSID='ASUS_2G' freq=2412 MHz)
ml1: Associated with 7c:10:c9:02:da:48
========================================
app_cb: WLAN: received event 1
========================================
app_cb: WLAN: authenticated to network
ml1: CTRL-EVENT-CONNECTED - Connection to 7c:10:c9:02:da:48 completed [id=0
id_str=]
ml1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
========================================
app_cb: WLAN: received event 0
========================================
app_cb: WLAN: connected to network
Connected to following BSS:
SSID = [ASUS_2G]
IPv4 Address: [192.168.0.123]
```

Step 3: Configure NS offload feature on DUT using below command

```
# enable-ns-offload
```

Step 4: From AP's backend server run ipv6 ping command. ipv6 ping request should come from AP and ipv6 ping reply should come from DUT.

Step 5: Configure mef params for NS using below command:

```
# wlan-multi-mef ns 0
```

Step 6: Configure host sleep and parameters using below command:

```
# wlan-host-sleep 1 mef
```

Step 7: Put HOST on suspend state

```
# mcu-suspend
```

Step 8: Only ping request from AP is observed due to DUT's suspend state. AP should send NS (Neighbor Solicitation) packet when DUT is in suspend state and DUT should respond to NS packet with NA (Neighbor Advertisement) packet. DUT should not wakeup.

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*: Please refer to* ***wifi_cert*** *demo from section 3.5, which has support for these APIs.* Please refer to MCUXSDKGSUG for more details *about the Wi-Fi driver APIs*.

4.1 Set/Get 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 Energy Detection (ED) threshold offset (ed_ctrl_2g.offset and ed_ctrl_5g.offset) needs to 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 21: Set ED MAC API argument

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

Return Value:

`WM_SUCCESS` if the call is successful, -`WM_FAIL` if the call failed

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 22: Get ED MAC API argument

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

Return Value:

`WM_SUCCESS` if the call is successful, -`WM_FAIL` if the call failed

4.1.3 Usage and Output

This section includes the output console logs and code snippets for the reference and it can be used to add the feature-related commands in the user application.

To add new CLI command in the existing *wifi_cli* sample application, please refer to section 0.

Usage:

To add `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
```

To print the usage regarding `set-ed-mac`

```
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
}
```

To 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();
    }
}
```

To add `get` command to the command list,

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

To 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");
}
```

To 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
```

4.2 Enable Host based WPA supplicant Feature for Wi-Fi application

This section describes changes required to enable host based wpa supplicant for Wi-Fi applications on i.MX RT1060 + IW416/88W8987/IW612 NXP-based wireless modules.

Host based supplicant is an open source software which requires increase in memory, mainly, HEAP (90~180 KB) and Increase number of clients store data in netif (LWIP).

To enable and support host based wpa supplicant Following files need updates.

NOTE: *The file paths given in this section refers to application **wifi_setup**.*

```
<SDK_PATH>/ boards/evkmimxrt1060/wifi_examples/wifi_setup/FreeRTOSConfig.h
<SDK_PATH>/ boards/evkmimxrt1060/wifi_examples/wifi_setup/lwipopts.h
<SDK_PATH>/ boards/evkmimxrt1060/wifi_examples/wifi_setup/wifi_config.h
```

4.2.1 FreeRTOSConfig.h

Increase Stack size:

In the following example stack is increased from 128 to 160 bytes.

```
- #define configMINIMAL_STACK_SIZE                ((unsigned short)128)
+ #define configMINIMAL_STACK_SIZE                ((unsigned short)160)
```

Increase heap size:

In following example, heap is increased from 60K to 120K bytes. 120KB is minimum required heap. If you enable CONFIG_WPA_SUPP_CRYPTO_ENTERPRISE, then 180 KB of heap is required.

```
- #define configTOTAL_HEAP_SIZE            ((size_t)(60 * 1024))
+ #define configTOTAL_HEAP_SIZE            ((size_t)(120 * 1024))
```

Increase stack for software timer task

Following example is increasing stack and making it twice the earlier size.

```
- #define configTIMER_TASK_STACK_DEPTH (configMINIMAL_STACK_SIZE)
+ #define configTIMER_TASK_STACK_DEPTH (configMINIMAL_STACK_SIZE * 2U)
```

4.2.2 lwipopts.h

Add following definitions at the end of file before “#endif /* __LWIPOPTS_H__ */”

```
/**
 * LWIP_CHECKSUM_ON_COPY==1: Calculate checksum when copying data from
 * application buffers to pbufs.
 */
#define LWIP_CHECKSUM_ON_COPY 1

/**
 * LWIP_CHKSUM_ALGORITHM==3: Use the optimised checksum algorithm.
 */
#define LWIP_CHKSUM_ALGORITHM 3

#if (LWIP_DNS || LWIP_IGMP || LWIP_IPV6) && !defined(LWIP_RAND)
/* When using IGMP or IPv6, LWIP_RAND() needs to be defined to a random-
function returning an u32_t random value*/
#include "lwip/arch.h"
u32_t lwip_rand(void);
#define LWIP_RAND() lwip_rand()
#endif

#define LWIP_NETIF_TX_SINGLE_PBUF   1

#if (LWIP_NETIF_TX_SINGLE_PBUF)
#define PBUF_LINK_ENCAPSULATION_HLEN 26
#endif

#define LWIP_NUM_NETIF_CLIENT_DATA  2
```

```
/* ---------- Core locking ---------- */
```

4.2.3 wifi_config.h

Add following definitions to enable various WPA modes. NOTE that Enterprise mode is enabled only if needed.

```
/*
 * Config options for wpa supplicant
 */
#define CONFIG_WPA_SUPP 1

#ifdef CONFIG_WPA_SUPP
#define CONFIG_WPA_SUPP_WPS               1
#define CONFIG_WPA_SUPP_WPA3              1
//#define CONFIG_WPA_SUPP_CRYPTO_ENTERPRISE 1
#endif
#endif/*
 * wpa supplicant debug options
 */
#define CONFIG_WPA_SUPP_DEBUG_LEVEL 3
```

4.2.4 Adding components

For those users who are using zip package, following files need to be updated at board_MIMXRT1060-EVK/boards/evkbmimxrt1060/wifi_examples/wifi_<app name>/armgcc

Add dependent component needed for wpa supplicant, mbedtls and hardware drivers.

Update config.cmake and append following

```
set(CONFIG_USE_middleware_mbedtls true)
set(CONFIG_USE_middleware_wireless_wpa_supplicant_rtos true)
set(CONFIG_USE_middleware_mbedtls_port_ksdk true)
set(CONFIG_USE_middleware_mbedtls_template true)
set(CONFIG_USE_driver_dcp true)
set(CONFIG_USE_driver_trng true)
```

Update CMakeLists.txt and add following

```
if(CMAKE_BUILD_TYPE STREQUAL flexspi_nor_debug)
     target_compile_definitions(${MCUX_SDK_PROJECT_NAME}  PRIVATE
MBEDTLS_CONFIG_FILE="wpa_supp_mbedtls_config.h")
endif(CMAKE_BUILD_TYPE STREQUAL flexspi_nor_debug)

if(CMAKE_BUILD_TYPE STREQUAL flexspi_nor_release)
     target_compile_definitions(${MCUX_SDK_PROJECT_NAME}  PRIVATE
MBEDTLS_CONFIG_FILE="wpa_supp_mbedtls_config.h")
endif(CMAKE_BUILD_TYPE STREQUAL flexspi_nor_release)
```

4.2.5 Memory Overflow Issue Handling

It is observed that for certain RT boards such as RT1020, RT1040 and RT1050 with lower memory footprint, the memory overflow error occurs during compiling with certain compiler tool. Following is the solution to such issues.

Linker file update can help if we get memory overflow during linking stage for low end platforms on certain compilers.

For mcuxpresso projects

create bss.ldt file at location evkbmimxrt1060/wifi_examples/common/linkscripts/bss.ldt

and add following in bss.ldt

```
        <#if memory.name=="SRAM_OC">
        *(.bss*)
        </#if>
```

For armgcc, iar and mdk, we need to move bss to m_data2. To do this update following files.

/wifi_examples/common/linker/MIMXRT1062xxxxx_flexspi_nor.icf

```
-place in DATA_region                            { block ZI };
+place in DATA2_region                           { block ZI };
```

evkbmimxrt1060/wifi_examples/common/linker/MIMXRT1062xxxxx_flexspi_nor.ld

In the .bss block, update '> m_data' to '> m_data2'. Do not change rest of the .bss block.

```
  /* Uninitialized data section */
  .bss :
  {
    /* This is used by the startup in order to initialize the .bss section */
    . = ALIGN(4);
    __START_BSS = .;
    __bss_start__ = .;
    *(m_usb_dma_noninit_data)
    *(.bss)
    *(.bss*)
    *(COMMON)
    . = ALIGN(4);
    __bss_end__ = .;
    __END_BSS = .;
  -} > m_data
  +} > m_data2
```

evkbmimxrt1060/wifi_examples/common/linker/MIMXRT1062xxxxx_flexspi_nor.scf

```
-  RW_m_ncache m_data2_start EMPTY 0 {
+  RW_m_data2 m_data2_start m_data2_size { ; RW data2
+    .ANY (+RW +ZI)
+  }
+  RW_m_ncache +0 EMPTY 0 {
  }
-  RW_m_ncache_unused +0 EMPTY m_data2_size-ImageLength(RW_m_ncache) { ; Empty
region added for MPU configuration
+  RW_m_ncache_unused +0 EMPTY m_data2_size-ImageLength(RW_m_data2)-
ImageLength(RW_m_ncache) { ; Empty region added for MPU configuration
```

5 Bluetooth Classic/Low Energy Applications

This chapter describes the Bluetooth Classic/Low Energy example applications that are available in the SDK, and the steps 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 HCI UART interface and PCM interface for voice.

Please refer to "*Hardware Rework Guide for EdgeFast BT PAL.pdf"* guide referenced in the Section 1.3 "References" for details to enable the UART and PCM interfaces.

The setup is done between the single i.MX RT+ IW416 NXP-based wireless module and remote Bluetooth devices. The instructions in this guide use an i.MXRT1060 EVK board. Yet the same steps apply to the other i.MX RT products.

The table lists the Bluetooth module specific preprocessor macro that is common to all Bluetooth examples.

Table 23: Preprocessor Macros for Bluetooth Modules

Module	Chipset	Macro
AzureWave AW-AM457-S-uSD	IW416	WIFI_IW416_BOARD_AW_AM457_USD
AzureWave AW-CM358-uSD	88W8987	WIFI_88W8987_BOARD_AW_CM358_USD
AzureWave AW-AM510-uSD	IW416	WIFI_IW416_BOARD_AW_AM510_USD
Murata Type 1XK	IW416	WIFI_IW416_BOARD_MURATA_1XK_USD WIFI_IW416_BOARD_MURATA_1XK_M2
Murata Type 1ZM	88W8987	WIFI_88W8987_BOARD_MURATA_1ZM_USD WIFI_88W8987_BOARD_MURATA_1ZM_M2
Murata Type 2EL	IW612	WIFI_IW61x_BOARD_MURATA_2EL_USD

USD = microSD interface

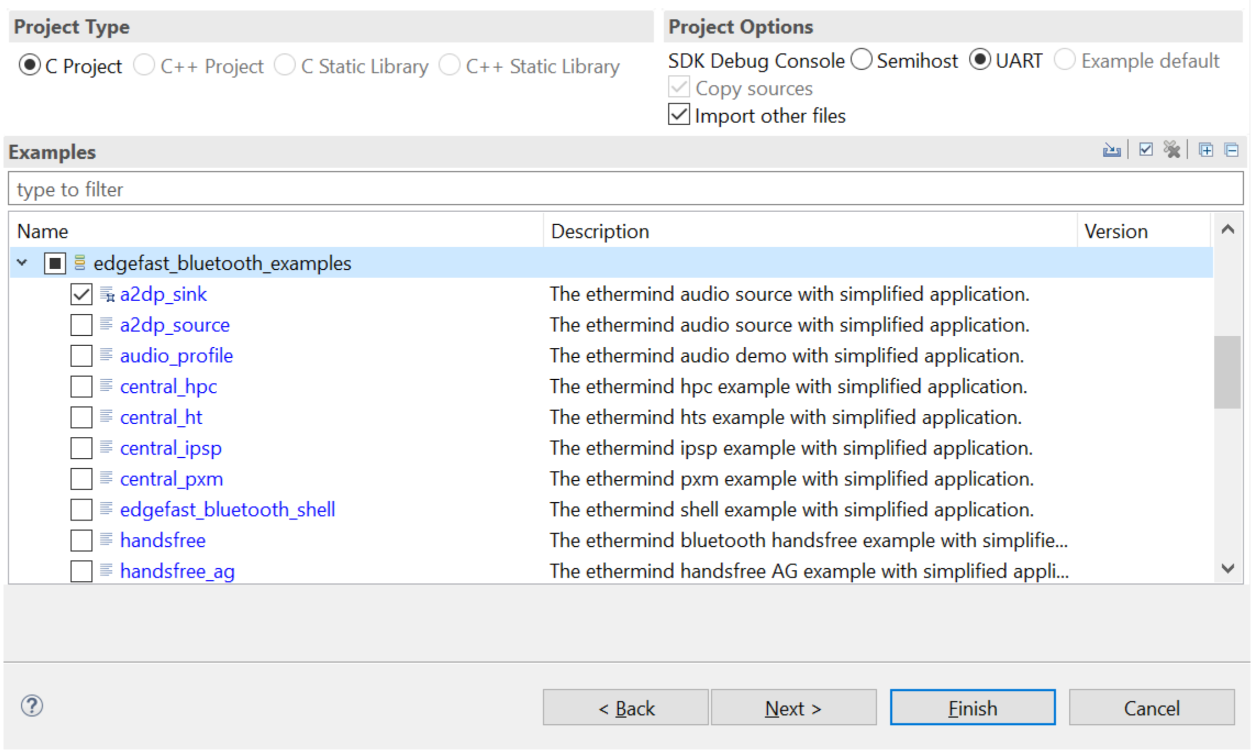
M2 = M.2 interface

5.1 a2dp_sink Sample Application

This section describes the steps to configure the i.MX RT1060 EVK board and IW416 wireless module as an A2DP Sink device.

5.1.1 a2dp_sink Application Execution

Please refer to the previous section 3.1.1 to run the demo using MCUXpresso IDE. Refer below image for selection of Bluetooth example.



Project Type

● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options

SDK Debug Console ○ Semihost ● UART ○ Example default

☑ Copy sources

☑ Import other files

Examples

type to filter

Name	Description	Version
■ edgefast_bluetooth_examples		
☑ a2dp_sink	The ethermind audio source with simplified application.	
□ a2dp_source	The ethermind audio source with simplified application.	
□ audio_profile	The ethermind audio demo with simplified application.	
□ central_hpc	The ethermind hpc example with simplified application.	
□ central_ht	The ethermind hts example with simplified application.	
□ central_ipsp	The ethermind ipsp example with simplified application.	
□ central_pxm	The ethermind pxm example with simplified application.	
□ edgefast_bluetooth_shell	The ethermind shell example with simplified application.	
□ handsfree	The ethermind bluetooth handsfree example with simplifie...	
□ handsfree_ag	The ethermind handsfree AG example with simplified appli...	

Please refer to the previous sections 3.1.1-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.1.1.1 Run the Application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
BR/EDR set connectable and discoverable done
```

Discover the device "a2dp sink" from peer mobile phone and connect to it. The following logs should be displayed on the console.

```
Connected
Security changed: 7E:5A:23:AE:9E:C3 (0xed) level 2
a2dp connected success
a2dp configure sample rate 44100Hz
```

Now, user can play music from the cell phone connected and listen on the audio jack of the i.MX RT 1060 EVK board.

Following logs will appear on the console:

```
a2dp start playing
```

Stop playing music from the cell phone.

Following logs will appear on the console

```
a2dp stop playing
```

Disconnect the device from peer cell phone.

```
a2dp deconfigure
Disconnected (reason 0x13)
```

5.2 a2dp_source Sample Application

This section describes the steps to configure the i.MX RT1060 EVK board and IW416 wireless module as an A2DP Source device.

5.2.1 a2dp_source Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.2.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized

Copyright  2022  NXP

>>
```

Input “help” to show the available list of commands:

```
>>
help

"help": List all the registered commands

"exit": Exit program

"bt": BT related function
  USAGE: bt [discover|connect|disconnect|delete]
    discover    start to find BT devices
    connect     connect to the device that is found, for example: bt
connectdevice n (from 1)
    disconnect  disconnect current connection.
    delete      delete all devices. Ensure to disconnect the HCI link
connection with the peer device before attempting to delete the bonding
information.
>>
```

Input “bt discover” to scan connectable nearby Bluetooth devices.

```
>> bt discover
Discovery started. Please wait ...
>> BR/EDR discovery complete
[1]: 20:39:56:C6:6C:6C, RSSI -89 Nokia 6.1 Plus
[2]: B8:F6:53:E8:BF:B7, RSSI -84 JBL Flip 5
[3]: 70:F0:87:C0:FC:0E, RSSI -100
>>
```

Input "bt connect [index]" to create Bluetooth connection with the discovered device. The music starts playing on successful connection with the Bluetooth device.

```
>> bt connect 2
Connection pending
>> SDP discovery started
Connected
sdp success callback
```

```
A2DP Service found. Connecting ...
Security changed: 7E:5A:23:AE:9E:C3 (0xed) level 2
a2dp connected success
a2dp start playing
```

Input “bt disconnect” to disconnect the current connection.

```
>> bt disconnect
>> a2dp disconnected
Disconnected (reason 0x16)
```

Input “bt delete” to delete the bonding information of all the devices.

NOTE: *Disconnect the HCI link connection with the peer device before attempting to delete the bonding information**.***

```
>> bt delete
success
>>
```

5.3 handsfree Sample Application

This section describes the steps to configure the i.MX RT1060 EVK board and IW416 wireless module as an HF Unit.

5.3.1 handsfree Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.3.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
BR/EDR set connectable and discoverable done

Copyright  2022  NXP
```

Discover the device “edgefast hfp” from peer mobile phone and connect to it. The following logs should be displayed on the console.

```
HFP HF Connected!
Wideband Config at Controller: Disabled
Sending Vendor command 0028
Sending Vendor command 0007 now
Sending Vendor command 0029 now
Sending Vendor command 001d now
Sending Vendor command 0070 now
Sending Vendor command 0073 with WBS disabled
Service indicator value: 1
Signal indicator value: 5
```

Make an incoming call to the mobile phone which is connected to the setup:

```
Call Setup indicator value: 1
Incoming Call...
Init Audio CODEC for RingTone
```

Type help command to check all calling options.

```
"help": List all the registered commands

"exit": Exit program

"bt": BT related function
```

```
  USAGE: bt [dial|aincall|eincall]
    dial           dial out call.
    aincall        accept the incoming call.
    eincall        end an incoming call.
    svr            start voice recognition.
    evr            stop voice recognition.
    clip           enable CLIP notification.
    disclip        disable CLIP notification.
    ccwa           enable call waiting notification.
    disccwa        disable call waiting notification.
    micVolume      Update mic Volume.
    speakerVolume  Update Speaker Volume.
    lastdial       call the last dial number.
    voicetag       Get Voice-tag Phone Number (BINP).
    multipcall     multiple call option.
```

Input “bt aincall” to answer the incoming call:

```
Setup for SCO audio: Success
Sending Vendor command 006f now
Call indicator value: 1
Call Setup indicator value: 0
Init Audio SCO SAI and CODEC samplingRate :8000  bitWidth:16
mic dummy
```

Input “bt eincall” to end the incoming call:

```
sco_audio_stop_pl: Sending Vendor command 0073 with WBS disabled
Call indicator value: 0
```

5.4 handsfree_ag Sample Application

This application demonstrates the HFP audio gateway basic functionality. Currently, the support simulates an incoming call, and the call could be answered and ended.

The HFP audio gateway can be connected to a HFP HF device like headphone or device running HFP HF device.

5.4.1 handsfree_ag Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.4.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized

Copyright  2022  NXP

>>
```

Input “help” to show the available list of commands:

```
>> help

"help": List all the registered commands

"exit": Exit program

"bt": BT related function
  USAGE: bt [discover|connect|disconnect|delete]
    discover    start to find BT devices
    connect     connect to the device that is found, for example: bt connect n
(from 1)
    sincall     start an incoming call.
    aincall     accept the incoming call.
    eincall     end an incoming call.
    disconnect  disconnect current connection.
    delete      delete all devices. Ensure to disconnect the HCI link
connection with the peer device before attempting to delete the bonding
information.
>>
```

Input “bt discover” to scan the nearby Bluetooth devices:

```
>> bt discover
Discovery started. Please wait ...
>> BR/EDR discovery complete
[1]: 50:82:D5:78:31:DA, RSSI -65 iPhone 6
[2]: 80:CE:B9:22:B3:FD, RSSI -79 Galaxy S8
[3]: BC:17:B8:74:2C:9F, RSSI -58 LAPTOP
[4]: 70:F0:87:C0:FC:0E, RSSI -66 iPhone
[5]: 00:00:AB:CD:87:D6, RSSI -38 Airdopes 441
[6]: C8:3D:D4:43:5A:14, RSSI -93 LAPTOP-NLHDBT0E

>>
```

Input “bt connect <number>” to connect to the peer device.

```
>> bt connect 5
Connection pending
>> SDP discovery started
Connected
Security changed: 7A:D6:2C:9B:F7:A3 (0x5e) level 2
HFP AG Connected!
```

Input “bt disconnect” to disconnect from the peer device.

```
>> bt disconnect
>> HFP AG Disconnected!
Disconnected (reason 0x16)
```

Input “bt delete” to delete the bonding information of all the devices.

NOTE: *Disconnect the HCI link connection with the peer device before attempting to delete the bonding information***.**

```
>> bt delete
success
>>
```

5.5 spp Sample Application

This application demonstrates the Serial Port Profile on i.MX RT1060 EVK board and IW416 wireless module.

5.5.1 spp Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.5.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
```

```
BR/EDR set connectable and discoverable done

Copyright  2022  NXP

>>
```

Input “help” to display the available options:

```
>> help
"help": List all the registered commands

"exit": Exit program

"bt": BT related function
  USAGE: bt <discover|connect|disconnect|delete>
    bt conns          print all active bt connection
    bt switch <index> switch a bt connection
    bt discover       start to find BT devices
    bt connect        connect to the device that is found, for example: bt 
connectdevice n (from 1)
    bt disconnect     disconnect current connection.
    bt delete         delete all devices. Ensure to disconnect the HCI link 
connection with the peer device before attempting to delete the bonding 
information.

"spp": SPP related function
  USAGE:
    spp handle                display active spp handle list
    spp switch <hanlde>       switch spp handle
    spp register <cid>        register a spp server channel(cid)
    spp discover              discover spp server channel on peer device
    spp connect <cid>         create spp connection
    spp disconnect            disconnect current spp connection.
    spp send <1|2|3|4>        send data over spp connection.
    spp get_port <s|c> <cid>  get spp port setting of server/client 
channel(cid).
    spp set_port <s|c> <cid>  set spp port setting of server/client 
channel(cid).
    spp set_pn <s|c> <cid>    set pn of server/client channel(cid).
    spp get_pn <s|c> <cid>    get local pn of server/client channel(cid).
    spp send_rls              send rls.
    spp send_msc              send msc.
>>
```

5.5.1.2 Serial Port Profile Server Configuration

This section describes the steps to configure the i.MX RT1060 EVK board and IW416 wireless module as an SPP Server.

Register a SPP server channel.

```
>> spp register 5
SPP channel 5 register successfully, waiting for connected callback!
>>
```

Connect to the device “edgefast spp” from the smartphone Bluetooth pairing settings and enable the pairing.

Following logs will appear on console:

```
>> Connected
Security changed: 48:74:12:C2:F2:82 (0x6d) level 2
Disconnected (reason 0x13)
```

Now, open the “*Serial Bluetooth Terminal”* smartphone application and go to settings > devices.

Select the device “edgefast spp”. The connection will be established and following logs will appear on console:

```
BR connection with 48:74:12:C2:F2:82 is created successfully!
Security changed: 48:74:12:C2:F2:82 level 2
Security changed: 48:74:12:C2:F2:82 level 2
spp handle 0: server, channel = 5, connected with device 82:F2:C2:12:74:48.
SPP appl handle 0 is connected successfully and becomes current spp appl
handle!
```

Write data in the smartphone application and send:

```
SPP appl handle 0 received 31 data callback, dumped here:
----------------CHAR DUMP-----------------------
  h e l l o   f r o m   s m a r t p h o n e
° ã õ  ÿ ÿ ÿ
----------------------------------------------------

----------------HEX DUMP------------------------
20 68 65 6C 6C 6F 20 66 72 6F 6D 20 73 6D 61 72 74 70 68 6F 6E 65 0D 0A B0 E3
F5 81 FF FF FF
----------------------------------------------------
```

Input “spp send [n]” to send data to peer device.

```
>> spp send 1
SPP appl handle 0 send string successfully, waiting for data sent callback.
>>
SPP appl handle 0 sent 11 data callback, dumped here:
----------------CHAR DUMP-----------------------
A T + C I N D = ? \ r
-----------------------------------------------------
```

Input “spp disconnect” to disconnect with peer device.

```
>> spp disconnect
SPP appl handle 0 disconnect successfully, waiting for disconnected callback.
SPP appl handle 0 is disconnected successfully.
BR connection with 48:74:12:C2:F2:82 is disconnected (reason 0x13)
```

5.5.1.3 Serial Port Profile Client Configuration

This section describes the steps to configure the i.MX RT1060 EVK board and IW416 wireless module as an SPP Client. Here, another setup of i.MX RT1060 EVK board and IW416 wireless module is used as SPP Server.

Start SPP server first then follow the steps to configure SPP client.

Input “bt discover” to start find the nearby Bluetooth devices.

```
>> bt discover
Discovery started. Please wait ...
>> BR/EDR discovery complete
[1]: C8:3D:D4:43:5A:14, RSSI -86 LAPTOP-NLHDBT0E
[2]: 04:D1:3A:82:B5:FE, RSSI -65 LAPTOP-NLHDBT0F
[3]: 18:5E:0F:96:8C:31, RSSI -69 BUILD0
[4]: 20:4E:F6:EC:1F:26, RSSI -79 edgefast spp

>>
```

Input “bt connect <n>” to connect to the device that is found.

```
>> bt connect 4
Connection pending
>> Connected

>>
```

Input “spp discover” to discover the registered SPP server channel in peer device.

```
>> spp discover
>> Discover 1 SPP server channel from device 26:1F:EC:F6:4E:20!
0x0005

>>
```

Input “spp connect [channel]" to create SPP connection with peer SPP server channel.

```
>> spp connect 5
Connect SPP Successful!
>> Security changed: 84:17:25:08:64:F9 (0xef) level 2
SPP connection is created successfully!

>>
```

Input “spp send [1|2|3|4]” to send data over SPP.

```
>> spp send 1
>>
Status of SPP data sent callback: 0x0000.
Sent 11 data, dumped here:

----------------CHAR DUMP-----------------------
A T + C I N D = ? \ r
------------------------------------------------
>>
```

Input “spp disconnect” to disconnect with peer device.

```
>> spp disconnect
>> SPP connection is disconnected successfully!
```

5.6 PBAP-PCE Sample Application

This application demonstrates the Phone Book Access Profile (PBAP) on i.MX RT1170 EVKB board as a Phone Book Client Equipment (PCE).

The Phone Book Access Profile (PBAP) defines the procedures and protocols to exchange Phone Book objects between devices.

The Phone Book Client Equipment (PCE) is the device that retrieves phone book objects from the Phone Book Server Equipment (PSE) device.

5.6.1 Pbap-pce Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.6.1.1 Run the application

Press the power reset button on i.MX RT1170 EVKB board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth PBAP PCE demo start...
Bluetooth initialized
BR/EDR set connectable and discoverable done
```

The demo does not require user interaction.

The application will automatically starts the BR/EDR discovery. The user needs to place the PSE device that wants to be connected as close as possible to the PCE.

This demo application will automatically connects to the PSE device which has strongest RSSI (more Nearby) and has Class of device (COD) of computer or Phone.

```
Discovery started. Please wait ...
BR/EDR discovery complete
[1]: A0:CD:F3:77:F2:DC, RSSI -72
[2]: 04:6C:59:78:5E:DC, RSSI -71 NXL53507
[3]: F8:E5:CE:19:75:D3, RSSI -69 Chang
[4]: 24:2E:02:C3:75:45, RSSI -41 test
[5]: CC:EB:5E:10:3E:79, RSSI -70 Redmi K700000
[6]: 30:BB:7D:8F:43:34, RSSI -61 AlisaS
[7]: 94:87:E0:83:99:61, RSSI -70 AnChangMI 8 SE
[8]: 98:43:FA:27:41:CD, RSSI -93

Connect 4
Connection pending

bt_connected
SDP discovery started

sdp success callback
pbap version is 101
pbap pse supported repositories is 1
supported feature not found, use default feature_config : 3
rfcomm channel found. Connecting ...
Successfully START PBAP PCE entities
Security changed: 24:2E:02:C3:75:45 level 2
PABP CONNECT SUCCESS

============== BODY ==============
BEGIN:VCARD
VERSION:2.1
FN;CHARSET=UTF-8:descvs
N;CHARSET=UTF-8:descvs
```

```
END:VCARD
BEGIN:VCARD
VERSION:2.1
N:;cc;;;
FN:cc
TEL;CELL:154555845
END:VCARD
BEGIN:VCARD
VERSION:2.1
N:;qwe;;;
FN:qwe
X-ANDROID-CUSTOM:vnd.android.cursor.item/nickname;147;1;;;;;;;;;;;;;
TEL;CELL:151865216
TEL;CELL:153464856
EMAIL;HOME:wudhxjsjd@qq.com
ADR;HOME:;;123456789;;;;
NOTE:old
BDAY:1904-05-24
X-AIM:@qq.com
END:VCARD
============ END BODY ============

pbap pse path set success

============== BODY ==============
<?xml version="1.0"?><!DOCTYPE vcard-listing SYSTEM "vcard-listing.dtd"><vCard-
listing version="1.0">
<card handle="1.vcf" name="qwe"/><card handle="2.vcf" name="qwe"/><card
handle="3.vcf" name="qwe"/>
<card handle="4.vcf" name="1155"/><card handle="5.vcf"
name="051295205593"/><card handle="6.vcf"
name="130"/><card handle="7.vcf" name="051295205593"/><card handle="8.vcf"
name="130"/><card handle
="9.vcf" name="11100000000"/><card handle="10.vcf" name="130"/><card
handle="11.vcf" name="051295205
593"/><card handle="12.vcf" name="15850182986"/><card handle="13.vcf"
name="15850182986"/><ca
============ END BODY ============
============== BODY ==============
rd handle="14.vcf" name="15850182986"/><card handle="15.vcf"
name="15850182986"/></vCard-listing>
============ END BODY ============

pbap pse path set success

============== BODY ==============
BEGIN:VCARD
VERSION:2.1
FN:
N:
TEL;X-0:1155
X-IRMC-CALL-DATETIME;DIALED:20220913T110607
END:VCARD
============ END BODY ============

pbap pse path set success

PABP DISCONNECT SUCCESSL : a0
Disconnected (reason 0x13)
```

5.7 PBAP-PSE Sample Application

This application demonstrates the Phone Book Access Profile (PBAP) on i.MX RT1170 EVKB board as a Phone Book Server Equipment (PSE).

The Phone Book Access Profile (PBAP) defines the procedures and protocols to exchange Phone Book objects between devices.

The Phone Book Server Equipment (PSE) is the device that contains the source phone book objects.

5.7.1 Pbap-pse Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.7.1.1 Run the application

Press the power reset button on i.MX RT1170 EVKB board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth PBAP PSE demo start...
Bluetooth initialized
BR/EDR set connectable and discoverable done
```

The demo does not require user interaction. The application will automatically start the Bluetooth discovery.

Now prepare the Phone Book Client Equipment (PCE) device and connect with this PSE device. Then initiate PBAP profile level connection from PCE device.

```
bt_connected
Security changed: A0:CD:F3:77:E4:37 level 2
PABP connect successfully

appl params max list count : 65535

send response : 90

send response : 90

send response : a0
```

Refer section “PBAP-PCE Sample Application” if you want to setup a IMX RT1170 EVKB as a PCE device.

After a successful PBAP connection, following commands can be sent from the PCE device which will be responded by PSE device.

- **pull phonebook - This example command will send phonebook object but not parse/send all application parameters from/to PCE.**

  ```
  appl params max list count : 65535

  send response : 90

  send response : 90

  send response : a0

  pse current path is root
  ```

- **set phonebook path - This example command will set phonebook path correctly.**

  ```
  set path to child telecom

  pse set current path is root/telecom
  ```

- **pull vcard listing - This example command will send vcard listing object but not parse/send all application parameters from/to PCE.**

```
appl params max list count : 65535

send response : a0

pse current path is root/telecom
```

- **get vcard entry - This example command will send vcard entry object but not parse/send all application parameters from/to PCE.**

```
set path to child cch

pse set current path is root/telecom/cch

send response : a0

pse current path is root/telecom/cch
```

5.7.1.2 Limitations

- **This example only supports one PBAP connection.**
- **This example doesn't supports all application parameters and only supports to parse/send the part of application parameters from/to PCE.**

5.8 MAP-MCE Sample Application

This application demonstrates the Message Access Profile (MAP) on i.MX RT1170 EVKB board as a Messaging Client Equipment (MCE).

The Message Access Profile (MAP) defines a set of features and procedures to exchange messages between devices.

The Messaging Client Equipment (MCE) is the device that uses the message repository engine of the Messaging Server Equipment (MSE) for browsing and displaying existing messages and to upload messages created on the MCE to the MSE.

5.8.1 Map-mce Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.8.1.1 Run the application

Press the power reset button on i.MX RT1170 EVKB board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth MAP MCE demo start...
Bluetooth initialized
BR/EDR set connectable and discoverable done
```

The demo does not require user interaction.

The application will automatically starts the BR/EDR discovery. The user needs to place the MSE device that wants to be connected as close as possible to the MCE.

This demo application will automatically connects to the MSE device which has strongest RSSI (more Nearby) and has Class of device (COD) of computer or Phone.

```
Discovery started. Please wait ...
BR/EDR discovery complete
```

```
[1]: A0:CD:F3:77:F2:DC, RSSI -72
[2]: 04:6C:59:78:5E:DC, RSSI -71 NXL53507
[3]: F8:E5:CE:19:75:D3, RSSI -69 Chang
[4]: CC:EB:5E:10:3E:79, RSSI -50 test

Connect 4
Connection pending

SDP discovery started
Connected
sdp success callback
REFCOMM channel number 26
L2CAP PSM  0x1029
MAP version 0x0104
MAP supported features 0x000603FF
MAS instance ID 0
Supported message type 0x00
Service name SMS/MMS
Message Access Server found. Connecting ...
Security changed: CC:EB:5E:10:3E:79 level 2
MCE MAS connection
MAX Packet Length - 509
[1]: GET_FOLDER_LISTING_ROOT
MAP Get Folder Listing
MAP Get Folder Listing CNF - 0xA0
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<folder-listing version="1.0">
    <folder name="telecom" />
</folder-listing>

[2]: GET_FOLDER_LISTING_ROOT Complete
[3]: SET_FOLDER_TELECOM
MAP Set Folder
Name - telecom
MAP Set Folder CNF - 0xA0
[4]: SET_FOLDER_TELECOM Complete
[5]: SET_FOLDER_MSG
MAP Set Folder
Name - msg
MAP Set Folder CNF - 0xA0
[6]: SET_FOLDER_MSG Complete
[7]: SET_FOLDER_INBOX
MAP Set Folder
Name - inbox
MAP Set Folder CNF - 0xA0
[8]: SET_FOLDER_INBOX Complete
[9]: UPDATE_INBOX
MAP Update Inbox
MAP Update Inbox CNF - 0xD1
[10]: UPDATE_INBOX Complete
[11]: GET_MSG_LISTING
MAP Get MSG Listing
MAX List Count - 10
SRMP Wait Count - 2
MAP Get MSG Listing CNF - 0x90
New Message - 1
Listing Size - 10
MSE Time - 20240716T155258+0800
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<MAP-msg-listing version="1.0">
    <msg handle="0400000000001577" subject="Bluetooth MAP Test Bluetooth MAP
Test Bluetooth MAP Test Bluetooth MAP Test Bluetooth MAP Test Bluetooth MAP
Test Bluetooth MAP Test Bluetooth MAP Test Bluetooth MAP Test Bluetooth MAP
Test Bluetooth MAP Test Bluetooth MAP Test Bluetooth MAP Test Bluetooth"
datetime="20240716T154618" sender_name="" sender_addressing="+XXXXXXXXXXXX"
MAP Get MSG Listing CNF - 0xA0
 recipient_name="" recipient_addressing="+XXXXXXXXXXXX" type="SMS_GSM"
size="588" text="yes" reception_status="complete" attachment_size="0"
priority="no" read="no" sent="no" protected="no" />
</MAP-msg-listing>

[12]: GET_MSG_LISTING Complete
[13]: GET_MSG
MAP Get MSG
Name - 0400000000001577
Attachment - 0
Charset - 0
SRMP Wait Count - 0
MAP Get MSG CNF - 0x90
BEGIN:BMSG
VERSION:1.0
STATUS:UNREAD
TYPE:SMS_GSM
FOLDER:telecom/msg/inbox
BEGIN:VCARD
VERSION:3.0
FN:
N:
TEL:+XXXXXXXXXXXX
END:VCARD
BEGIN:BENV
BEGIN:BBODY
ENCODING:G-7BIT
LENGTH:1334
BEGIN:MSG
00440d91688120218956f1000042706151648123a0050003a8040184ec7a99fe7ed3d1a066100aa
296e77490905d2fd3df6f3a1ad40c4241d4f29c0e12b2eb65fafb4d47839a4128885a9ed3414276
bd4c7fbfe9685033080551cb733a48c8ae97e9ef371d0d6a06a1206a794e0709d9f532fdfda6a34
1cd2014442dcfe92021bb5ea6bfdf7434a8198482a8e5391d2
MAP Get MSG CNF - 0xA0
dfda6a341cd2014442dcfe9
END:MSG
END:BBODY
END:BENV
END:BMSG

[14]: GET_MSG Complete
[15]: SET_MSG_STATUS
MAP Set MSG Status
Name - 0400000000001577
Status Indicator - 0
Status Value - 0
MAP Set MSG Status CNF - 0xA0
[16]: SET_MSG_STATUS Complete
[17] ~ [18] Skip, BT_MAP_CONVO_LISTING is not supported
[19]: GET_MAS_INST_INFO
MAP Get MAS Instance Info
MAS Instance ID - 0
SRMP Wait Count - 0
MAP Get MAS Instance Info CNF - 0xA0
SMS/MMS
[20]: GET_MAS_INST_INFO Complete
[21]: SET_NTF_FILTER
MAP Set Notification Filter
```

```
Notification Filter Mask - 0
MAP Set NTF Filter CNF - 0xA0
[22]: SET_NTF_FILTER Complete
[23]: SET_NTF_REG_ON
MAP Set Notification Registration
Notification Status - 1
MAP Set Notification Registration CNF - 0xA0
MCE MNS connection
MAX Packet Length - 1790
[24]: SET_NTF_REG_ON Complete
[25]: SET_NTF_REG_OFF
MAP Set Notification Registration
Notification Status - 0
MAP Set Notification Registration CNF - 0xA0
MCE MNS disconnection - 0xA0
[26]: SET_NTF_REG_OFF Complete
[27] ~ [30] Skip, BT_MAP_OWNER_STATUS is not supported
[31]: SET_FOLDER_PARENT
MAP Set Folder
Name - ../
MAP Set Folder CNF - 0xA0
[32]: SET_FOLDER_PARENT Complete
[33]: SET_FOLDER_OUTBOX
MAP Set Folder
Name - outbox
MAP Set Folder CNF - 0xA0
[34]: SET_FOLDER_OUTBOX Complete
[35]: PUSH_MSG
MAP Push MSG
Charset - 0
MAP Push MSG CNF - 0xCC
[36]: PUSH_MSG Complete
[37]: MCE_MAS_DISCONNECT
MAP MCE MAS Disconnect
MCE MAS disconnection - 0xA0
Disconnected (reason 0x13)
```

5.9 MAP-MSE Sample Application

This application demonstrates the Message Access Profile (MAP) on i.MX RT1170 EVKB board as a Messaging Server Equipment (MSE).

The Message Access Profile (MAP) defines a set of features and procedures to exchange messages between devices.

The Messaging Server Equipment (MSE) is the device that provides the message repository engine i.e. has the ability to provide a client unit with messages that are stored in this device and notifications of changes in its message repository.

5.9.1 map-mse Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.9.1.1 Run the application

Press the power reset button on i.MX RT1170 EVKB board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth MAP MSE demo start...
Bluetooth initialized
BR/EDR set connectable and discoverable done
```

The demo does not require user interaction. The application will automatically start the Bluetooth discovery.

Now prepare the Phone Messaging Equipment (MCE) device and connect with this MSE device. Then initiate MAP profile level connection from MCE device.

```
Connected
Security changed: A0:CD:F3:77:E5:01 level 2
File system mounted

Total drive space - 48128B

Free drive space - 43008B

MSE MAS connection

MAX Packet Length - 509
```

Refer section “MAP-MCE Sample Application” if you want to setup a IMX RT1170 EVKB as a MCE device.

After a successful MAP connection, following commands can be sent from the MCE device which will be responded by MSE device.

- **get folder listing - This example will send Folder-listing object but not parse/send application parameters from/to MCE.**
- **set folder - This example will set folder correctly.**
- **get message listing - This example will send Messages-listing object with NewMessage, MSETime and ListingSize but not parse application parameters from MCE.**
- **get message - This example will send bMessage object but not parse/send application parameters from/to MCE.**
- **set message status - This example will set the read status and the deleted status correctly and save the extended data to the local buffer.**
- **push message - This example will save the message and return a message handle but not parse application parameters from MCE.**
- **set notification registration - When Notification Status is ON, this example will initiates a MNS OBEX connection.**
- **update inbox - This example always send success when receiving update inbox request.**
- **get mas instance information - This example will send MAS Instance Information but not send application parameters to MCE.**
- **set owner status - This example will save the application parameters to the local buffer that is used to respond to get owner status.**
- **get owner status - This example will respond to get owner status with the application parameters saved in set owner status.**
- **get conversation listing - This example will send Conversation-Listing object but not parse/send application parameters from/to MCE.**
- **set notification filter - This example always send success when receiving set notification filter request.**

5.9.1.2 Limitations

- **This example only supports one MAS and MNS OBEX connection.**
- **This example doesn't supports all application parameters and only supports to parse/send the part of application parameters from/to MCE.**
- **This example is based on Fatfs RAM disk. There is a limited memory to store the incoming message from MCE.**

5.10 peripheral_hps Sample Application

This application demonstrates the Bluetooth LE Peripheral role, except that this application specifically exposes the HTTP Proxy GATT Service.

5.10.1 peripheral_hps Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.10.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Advertising successfully started
```

The demo does not require user interaction.

The application will automatically start advertising the HTTP Proxy Service and it will accept the first connection request it receives. The application is then ready to process HTTP requests from the peer.

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

```
Connected to peer: 70:E8:F5:6B:FA:96 (random)
Passkey for 70:E8:F5:6B:FA:96 (random): 012896
Security changed: 90:78:B2:B6:A7:6C (public) level 4 (error 0)
```

5.11 central_hpc Sample Application

This application demonstrates very basic Bluetooth LE Central role functionality on i.MX RT1060 EVK board and IW416 wireless module by scanning for other Bluetooth LE devices and establishing a connection to the first one with a strong enough signal.

Except that this application specifically looks for HPS Server and programs a set of characteristics that configures a Hyper Text Transfer Protocol (HTTP) request, initiate this request, and then read the response once connected.

Here, another setup of i.MX RT1060 EVK board and IW416 wireless module is used as *peripheral_hps*.

5.11.1 central_hpc Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.11.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Scanning started
[DEVICE]: 20:4E:F6:25:F3:18 (public), AD evt type 0, AD data len 7, RSSI -71
```

The demo does not require user interaction.

The application will automatically start scanning and will connect 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 will perform a GET for the URI http://nxp.com by writing the URI and the Control Point characteristics of the HTTP Proxy Service.

The application will display the received response in the console after it gets notified through the HTTP Status Code characteristic.

```
Found device: Connected to peer: 20:4E:F6:25:F3:18 (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
```

5.12 peripheral_pxr Sample Application

This application demonstrates the BLE Peripheral role on i.MX RT1060 EVK board and IW416 wireless module. Except that this application specifically exposes the Proximity Reporter (including LLS, IAS, and TPS) GATT Service.

5.12.1 peripheral_pxr Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.12.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Advertising successfully started
```

The demo does not require user interaction.

The application will automatically start advertising the Link Loss Service and it will accept the first connection request it receives. The application is then ready to process operations from the peer.

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

```
Connected to peer: 20:4E:F6:25:F3:18 (public)
Locally setting Link Loss Alert Level to OFF
Locally setting Immediate Alert...

ALERT: OFF
```

The Proximity Monitor peer will trigger or stop the Immediate Alert on the application depending on the connection RSSI.

```
Monitor is setting Link Loss Alert Level to HIGH
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 will be triggered with the level previously set by the Monitor.

```
Link Loss Alert Triggered...
ALERT: HIGH
```

5.13 central_pxm Sample Application

This application demonstrates very basic Bluetooth LE Central role functionality on i.MX RT1060 EVK board and IW416 wireless module by scanning for other Bluetooth LE devices and establishing a connection to the first one with a strong enough signal.

Except that this application specifically looks for Proximity Reporter.

Here, another setup of i.MX RT1060 EVK board and IW416 wireless module is used as *peripheral_pxr*.

5.13.1 central_pxm Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.13.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Scanning started
```

The application will automatically start scanning and will connect 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, as well as additional services and characteristics specified by the Proximity Profile, such as Immediate Alert and Tx Power services.

```
[DEVICE]: 20:4E:F6:25:F3:18 (public), AD evt type 0, AD data len 11, RSSI -83
Found device: Connected to peer: 20:4E:F6:25:F3:18 (public)
Starting service discovery
GATT Write successful
Connection RSSI: -81
```

If the Tx Power service and its characteristics have been discovered, the application will read the peer's Tx power and display it.

```
Read successful - Tx Power Level: 0
```

If the Immediate Alert service and its characteristics have been discovered, the application will continuously monitor the connection RSSI and will trigger or stop the Immediate Alert on the peer when the value is crossing a preset threshold in either direction.

```
Connection RSSI: -81
Connection RSSI: -81
Connection RSSI: -81
Connection RSSI: -81
Connection RSSI: -81
```

After the mandatory Link Loss service is discovered, the application will write the Link Loss Alert Level on the peer as HIGH_ALERT.

To trigger the Link Loss Alert on the peer, the connection will have to be timed out. The user can trigger this by simply resetting the board (press the RST button).

5.14 peripheral_ht Sample Application

This application demonstrates the BLE Peripheral role on i.MX RT1060 EVK board and IW416 wireless module. Except that this application specifically exposes the HT (Health Thermometer) GATT Service.

Once a device connects it will generate dummy temperature values.

5.14.1 peripheral_ht Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.14.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board.

When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Advertising successfully started
```

The application does not require user interaction.

The application will automatically start advertising the Health Thermometer Service and it will accept the first connection request it receives. If the peer subscribes to receive temperature indications, these will be sent every 1 second.

The temperature readings are simulated with values between 20 and 25 degrees Celsius.

```
Connected to peer: 90:78:B2:B6:A7:6C (public)
Passkey for 90:78:B2:B6:A7:6C (public): 413238
Security changed: 90:78:B2:B6:A7:6C (public) level 4 (error 0)
temperature is 21C
Indication success
temperature is 22C
Indication success
```

5.15 central_ht Sample Application

This application demonstrates very basic Bluetooth LE Central role functionality on i.MX RT1060 EVK board and IW416 wireless module by scanning for other Bluetooth LE devices and establishing a connection to the first one with a strong enough signal.

Except that this application specifically looks for health thermometer sensor and reports the temperature readings once connected.

Here, another setup of i.MX RT1060 EVK board and IW416 wireless module is used as *peripheral_ht*.

5.15.1 central_ht Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.15.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Scanning started
```

The demo does not require user interaction.

The application will automatically start scanning and will connect 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 will subscribe to receive temperature indications from the peer.

The application will display the received indications in the console.

```
[DEVICE]: 20:4E:F6:25:F3:18 (public), AD evt type 0, AD data len 9, RSSI -84
Found device: Connected to peer: 20:4E:F6:25:F3:18 (public)
Starting service discovery
Subscribed to HTS
Temperature 21 degrees Celsius
Temperature 22 degrees Celsius
```

5.16 peripheral_ipsp Sample Application

This application demonstrates the BLE Peripheral role on i.MX RT1060 EVK board and IW416 wireless module. Except that this application specifically exposes the Internet Protocol Support GATT Service.

5.16.1 peripheral_ipsp Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.16.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board.

When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Advertising successfully started
IPSS Service ready
```

The demo does not require user interaction.

The application will automatically start advertising the IPSP Service and it will accept the first connection request it receives.

The application will perform the required setup for the L2CAP credit-based channel specified by the IPSP Profile. The application will display in console any message it receives from the peer through the L2CAP channel.

```
Connected to peer: 90:78:B2:B6:A7:6C (random)
Passkey for 90:78:B2:B6:A7:6C (random): 214634
Security changed: 90:78:B2:B6:A7:6C (public) level 4 (error 0)
Received message: hello
Received message: hello
```

5.17 central_ipsp Sample Application

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

Except that this application specifically looks for IPSP Service and communicates between the devices that support IPSP is done using IPv6 packets over the Bluetooth Low Energy transport once connected.

Here, another setup of i.MX RT1060 EVK board and IW416 wireless module is used as *peripheral_ipsp*.

5.17.1 central_ipsp Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.17.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Bluetooth initialized
Scanning started
```

The demo does not require user interaction.

The application will automatically start scanning and will connect 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 will send a predefined test message every 5 seconds through the channel.

```
[DEVICE]: 20:4E:F6:25:F3:18 (public), AD evt type 0, AD data len 7, RSSI -82
Found device: Connected
Starting service discovery
Sending message...
Sending message...
```

5.18 Broadcast media sender

This section describes the application to demonstrate on how to use the broadcast media sender example of the LE audio feature.

The Broadcast Media Sender (BMS) role is defined for LE devices that send media audio content to any number of receiving devices. Typical devices implementing the BMS role include smartphones, media players, TVs, laptops, tablets, and PCs.

Run and connect the Broadcast media receiver (BMR) with the this BMS device to verify the BMS audio.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.18.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the Application

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Save a wav music file to a USB drive and name it as "music_16_2.wav"

Step 3: Connect the same USB drive to USB OTG1 port of the i.MX RT EVK board

Step 4: Apply a power reset on i.MX RT EVK board

Step 5: Check the console on the connected computer screen to see the application start-up logs

5.18.2 Application execution

This section describes the steps for application execution.

Step 1: Press RESET button and restart the i.MX RT EVK board

When the demo starts, the media broadcast starts automatically and following message about the demo would appear on the console.

```
Broadcast Media Sender.
Bluetooth initialized
wav file info:
        sample_rate: 16000
        channels: 2
        bits: 16
```

```
       size: 3827552
       samples: 956888
LC3 encoder setup done!
Codec setup done!
Qos setup done!
Creating broadcast source
Creating broadcast source with 1 subgroups with 2 streams
Starting broadcast source
Broadcast source started
```

Step 2: Input "help" command to get the list of commands

```
BMS>> help
Help

"help": List all the registered commands
"exit": Exit program
play  :resume broadcast.
pause :stop broadcast.
BMS>>
```

Step 3: Input "pause" command to stop broadcast

```
BMS>> pause
pause

BMS>> Broadcast source stopped
```

Step 4: Input "play" command to start broadcast

```
BMS>> play
play

BMS>> Broadcast source started
```

5.19 Broadcast media receiver

This section describes the application to demonstrate on how to use the broadcast media receiver example of the LE audio feature.

The Broadcast Media Receiver (BMR) role is defined for devices that receive media audio content from a source device in a broadcast Audio Stream. Typical devices implementing the BMR role include headphones, earbuds, and speakers. A smartphone may also support this role to receive broadcast Audio Streams from a BMS.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.19.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the example project

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Connect a speaker/headphone to the 3.5mm audio jack of i.MX RT EVK board

Step 3: Apply a power reset on i.MX RT EVK board

Step 4: Check the console on the connected computer screen to see the application start-up logs

5.19.2 Application execution

This section describes the steps for application execution.

Step 1: Press RESET button and restart the i.MX RT EVK board

When the demo starts, following message about the demo would appear on the console.

```
Copyright  2022  NXP
```

```
BMR>>
Broadcast Media Receiver.

Please select sink role "left"|"right" use "init" command.
```

Step 2: Select the sink role

Once the sink role is selected, the application automatically start receiving media samples and following type of message would appear on the console.

```
BMR>> init left
init left

BMR@left>> BMR@left>> Bluetooth initialized
Scanning for broadcast sources

[device name]:broadcast_media_sender
connect...
Broadcast source found, waiting for PA sync
PA synced for broadcast sink 202DBE48 with broadcast ID 0xD789F0
Received BASE with 1 subgroups from broadcast sink 202DBE48
Broadcast source PA synced, waiting for BASE
BASE received, waiting for syncable
       Codec: freq 16000, channel count 1, duration 10000, channel alloc
0x00000001, frame len 40, frame blocks per sdu 1
Audio codec configed, waiting for syncable
Syncing to broadcast
Stream 202E3164 started
BMR@left>>
BMR@left>>
BMR@left>>
```

Step 3: Input "pause" command to stop playing

```
BMR@left>> pause
pause

BMR@left>> Stream 2025554C stopped
Broadcast sink stoped!
```

Step 4: Input "play" command to start playing

```
BMR@left>> play
play

BMR@left>> Syncing to broadcast
Stream 2025554C started
```

Step 5: Input "vol_set 100" to set the volume to level 100

```
BMR@left>> vol_set 100
vol_set 100

BMR@left>>
```

Step 6: Input "vol_up" command and increase the volume

```
BMR@left>> vol_up
vol_up

vol: 124
BMR@left>>
```

Step 7: Input "vol_down" command and decrease the volume

```
BMR@left>> vol_down
vol_down

vol: 99
BMR@left>>
```

Step 8: Input "vol_mute" command and mute the volume

```
BMR@left>> vol_mute

vol_mute

BMR@left>>
```

Step 9: Input "vol_unmute" command and unmute the volume

```
BMR@left>> vol_unmute
vol_unmute

vol: 99
BMR@left>>
```

5.20 Telephony and Media Audio Profile (TMAP) Peripheral Application

This section describes the application to demonstrate how to use the Media Audio Profile (TMAP) on the peripheral device.
The Telephony and Media Audio Service (TMAS) defines a characteristic to enable discovery of supported TMAP profile roles.

***NOTE:** This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.20.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the example project

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Apply a power reset on i.MX RT EVK board

Step 3: Check the console on the connected computer screen to see the application start-up logs

5.20.2 Application execution

When demo application starts, It automatically connects with TMAP central devices which scan it.

```
Bluetooth initialized
Initializing TMAP and setting role
VCP initialized
BAP initialized
Advertising successfully started
Connected: A0:CD:F3:77:E4:11 (public)
Security changed: 0
TMAP discovery done
CCP: Discovered GTBS
CCP: Discovered remote URI: skype
CCP initialized
ASE Codec Config: conn 202C37D8 ep 202C1D5C dir 1
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
```

```
64
  Frequency: 48000 Hz
  Frame Duration: 10000 us
  Channel allocation: 0x1
  Octets per frame: 100 (negative means value not pressent)
  Frames per SDU: 1
ASE Codec Config stream 202C208C
QoS: stream 202C208C qos 202C1DCC
QoS: interval 10000 framing 0x00 phy 0x02 sdu 100 rtn 5 latency 20 pd 40000
Enable: stream 202C208C meta_count 1
MCP: Discovered MCS
MCP initialized
Incoming audio on stream 202C208C len 100
CCP: Call originate successful
MCP: Successfully sent command (0) - opcode: 1, param: 0
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
CCP: Call with id 1 terminated
MCP: Successfully sent command (0) - opcode: 2, param: 0
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
Incoming audio on stream 202C208C len 100
```

5.21 Telephony and Media Audio Profile (TMAP) Central Application

This section describes the application to demonstrate how to use the Media Audio Profile (TMAP) on the central device.
The Telephony and Media Audio Service (TMAS) defines a characteristic to enable discovery of supported TMAP profile roles.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.21.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the example project

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Apply a power reset on i.MX RT EVK board

Step 3: Check the console on the connected computer screen to see the application start-up logs

5.21.2 Application execution

When demo application starts, It automatically scan for the TMAP peripheral device and connects with it.

```
Bluetooth initialized
Initializing TMAP and setting role
CAP initialized
VCP initialized
MCP initialized
CCP initialized
Scanning successfully started
[DEVICE]: 41:77:CE:FE:99:1B (random), [AD]: 1 data_len 1
[AD]: 255 data_len 22
[AD]: 3 data_len 2
[DEVICE]: 56:CF:9F:4C:98:40 (random), [AD]: 1 data_len 1
[AD]: 25 data_len 2
[AD]: 2 data_len 2
[AD]: 46 data_len 6
[AD]: 22 data_len 4
Found TMAS in peer adv data!
Attempt to connect!
MTU exchanged: 23/23
Connected: 56:CF:9F:4C:98:40 (random)
MTU exchanged: 65/65
Security changed: 0
TMAS discovery done
Error sending mute command!
Found CAS
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
1e
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1
data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
3c
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
50
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
4b
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
64
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
5a
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1
data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
78
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
75
meta #0: type 0x02 len 2
07
codec_capabilities 202CE99C dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 4
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
9b
meta #0: type 0x02 len 2
07
Sink #0: ep 202C2F44
Sink discover complete
Discover sources complete: err 0
Created group
Configured stream 202C83DC
QoS set stream 202C83DC
Enabled stream 202C83DC
Started stream 202C83DC
Sending mock data with len 100
CCP: Placing call to remote with id 1 to skype:friend
Sending mock data with len 100
Sending mock data with len 100
CCP: Call terminated for id 1 with reason 6
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
Sending mock data with len 100
```

5.22 Unicast media sender

This section describes the application to demonstrate on how to use the unicast media sender example of the LE audio feature.

The Unicast Media Sender (UMS) role is defined for devices that send media audio content in one or more Unicast Audio Streams. Typical devices implementing the UMS role include smartphones, media players, TVs, laptops, tablets, and PCs.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.22.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the Application

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Save a wav music file to a USB drive and name it as "music_16_2.wav"

Step 3: Connect the same USB drive to USB OTG1 port of the i.MX RT EVK board

Step 4: Apply a power reset on i.MX RT EVK board

Step 5: Check the console on the connected computer screen to see the application start-up logs

5.22.2 Application execution

This section describes the steps for application execution.

Step 1: Press RESET button and restart the i.MX RT EVK board

When the demo starts, it automatically starts scanning for the left and right profiles and following message about the demo would appear on the console.

```
Copyright  2022  NXP

UMS>>
Unicast Media Sender.
Initializing
Initialized
wav file info:
       sample_rate: 16000
       channels: 2
       bits: 16
       size: 3827552
       samples: 956888
LC3 encoder setup done!
Creating unicast group
Unicast group created
Waiting for connection
Scan & connect left cis sink:
Scanning successfully started
Device found: D0:17:69:EE:69:1B (public) (RSSI -70)
[AD]: 1 data_len 1
[AD]: 3 data_len 2
[AD]: 22 data_len 8
[AD]: 9 data_len 28
[device name]: unicast_media_receiver_right
Device found: D0:17:69:EE:68:9D (public) (RSSI -74)
Device found: D0:17:69:EE:68:9D (public) (RSSI -75)
Device found: 9C:19:C2:3E:D7:9B (public) (RSSI -102)
Device found: D0:17:69:EE:68:9D (public) (RSSI -75)
Device found: D0:17:69:EE:69:1B (public) (RSSI -71)
Device found: D0:17:69:EE:69:1B (public) (RSSI -70)
[AD]: 1 data_len 1
[AD]: 3 data_len 2
[AD]: 22 data_len 8
```

```
[AD]: 9 data_len 28
[device name]: unicast_media_receiver_right
Device found: D0:17:69:EE:68:9D (public) (RSSI -75)
Device found: D0:17:69:EE:68:9D (public) (RSSI -75)
Device found: CB:6D:B6:97:27:B7 (random) (RSSI -89)
Device found: CE:1D:86:26:FE:B7 (random) (RSSI -93)
Device found: D5:70:14:34:A1:B4 (random) (RSSI -96)
Device found: 6F:AD:70:37:8D:1E (random) (RSSI -92)
Device found: 70:33:9B:81:F7:C0 (random) (RSSI -96)
Device found: 5C:54:63:FD:86:66 (random) (RSSI -97)
Device found: C9:0F:39:26:C1:CC (random) (RSSI -83)
Device found: CE:1D:86:26:FE:B7 (random) (RSSI -96)
Device found: 6D:A4:6E:CC:F1:18 (random) (RSSI -86)
Device found: 7A:A1:AC:F1:DA:62 (random) (RSSI -86)
Device found: A0:CD:F3:77:E6:15 (public) (RSSI -72)
Device found: 7A:A1:AC:F1:DA:62 (random) (RSSI -87)
Device found: CB:6D:B6:97:27:B7 (random) (RSSI -92)
Device found: 9C:19:C2:3E:D7:9B (public) (RSSI -77)
Device found: 6D:A4:6E:CC:F1:18 (random) (RSSI -92)
Device found: 55:78:0F:26:5E:2B (random) (RSSI -84)
Device found: CE:1D:86:26:FE:B7 (random) (RSSI -93)
Device found: A0:CD:F3:77:E6:15 (public) (RSSI -71)
Device found: C9:0F:39:26:C1:CC (random) (RSSI -92)
Device found: E1:C1:6E:2E:9D:E0 (random) (RSSI -94)
Device found: 62:BE:4B:FA:FE:07 (random) (RSSI -82)
Device found: C9:0F:39:26:C1:CC (random) (RSSI -83)
Device found: CE:1D:86:26:FE:B7 (random) (RSSI -96)
Device found: 7A:A1:AC:F1:DA:62 (random) (RSSI -87)
Device found: 73:44:01:AE:44:62 (random) (RSSI -95)
Device found: CB:6D:B6:97:27:B7 (random) (RSSI -92)
Device found: D0:17:69:EE:69:1B (public) (RSSI -37)
[AD]: 1 data_len 1
[AD]: 3 data_len 2
[AD]: 22 data_len 8
[AD]: 9 data_len 28
[device name]: unicast_media_receiver_right
Device found: 6D:A4:6E:CC:F1:18 (random) (RSSI -94)
Device found: 5C:54:63:FD:86:66 (random) (RSSI -95)
Device found: 63:AF:73:23:88:87 (random) (RSSI -96)
Device found: D0:17:69:EE:68:9D (public) (RSSI -40)
[AD]: 1 data_len 1
[AD]: 3 data_len 2
```

Once the application connects both the left and right profiles, it starts streaming the audio.

```
[AD]: 22 data_len 8
[AD]: 9 data_len 27
[device name]: unicast_media_receiver_left
Audio server found; connecting
MTU exchanged: 23/23
Connected: D0:17:69:EE:68:9D (public)
MTU exchanged: 65/65
Connected
Waiting for connection
Scan & connect right cis sink:
Scanning successfully started
Device found: CE:1D:86:26:FE:B7 (random) (RSSI -94)
Device found: 6D:A4:6E:CC:F1:18 (random) (RSSI -82)
Device found: 9C:19:C2:3E:D7:9B (public) (RSSI -72)
Device found: C9:0F:39:26:C1:CC (random) (RSSI -84)
Device found: E6:DD:1E:0C:A6:83 (random) (RSSI -89)
Device found: 57:1C:5A:5F:71:E3 (random) (RSSI -94)
Device found: F4:DB:3B:C2:D6:3F (random) (RSSI -100)
Device found: CE:1D:86:26:FE:B7 (random) (RSSI -96)
Device found: 6D:A4:6E:CC:F1:18 (random) (RSSI -92)
Device found: 7A:A1:AC:F1:DA:62 (random) (RSSI -87)
Device found: 57:1C:5A:5F:71:E3 (random) (RSSI -92)
Device found: CB:6D:B6:97:27:B7 (random) (RSSI -92)
Device found: 62:BE:4B:FA:FE:07 (random) (RSSI -84)
Device found: 55:78:0F:26:5E:2B (random) (RSSI -75)
Device found: 5C:B7:95:12:4A:D6 (random) (RSSI -92)
Device found: D0:B4:5D:59:C0:D1 (public) (RSSI -73)
Device found: C9:0F:39:26:C1:CC (random) (RSSI -93)
Device found: D0:17:69:EE:69:1B (public) (RSSI -36)
[AD]: 1 data_len 1
[AD]: 3 data_len 2
[AD]: 22 data_len 8
[AD]: 9 data_len 28
[device name]: unicast_media_receiver_right
Audio server found; connecting
MTU exchanged: 23/23
Connected: D0:17:69:EE:69:1B (public)
MTU exchanged: 65/65
Connected
Discover VCS

VCS discover finished
Discover VCS complete.
Discovering sinks
codec_capabilities 2000C4EC dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 5
data #0: type 0x01 len 2
f5
data #1: type 0x02 len 1

data #2: type 0x03 len 1

data #3: type 0x04 len 4
280078
data #4: type 0x05 len 1

meta #0: type 0x01 len 2
06
dir 1 loc 1
snk ctx 31 src ctx 0
Sink #0: ep 202D6DB0
Discover sinks complete: err 0
Sinks discovered
Configuring streams
Audio Stream 202EBC64 configured
Configured sink stream[0]
Stream configured
Setting stream QoS
QoS: waiting for 0 streams
Audio Stream 202EBC64 QoS set
Stream QoS Set
Enabling streams
Audio Stream 202EBC64 enabled
Streams enabled
Starting streams
Audio Stream 202EBC64 started
Streams started
Discover VCS

VCS discover finished
```

```
Discover VCS complete.
Discovering sinks
codec_capabilities 2000C4EC dir 0x01
codec 0x06 cid 0x0000 vid 0x0000 count 5
data #0: type 0x01 len 2
f5
data #1: type 0x02 len 1

data #2: type 0x03 len 1

data #3: type 0x04 len 4
280078
data #4: type 0x05 len 1

meta #0: type 0x01 len 2
06
dir 1 loc 2
snk ctx 31 src ctx 0
Sink #1: ep 202D72A0
Discover sinks complete: err 0
Sinks discovered
Configuring streams
Audio Stream 202EBC8C configured
Configured sink stream[1]
Stream configured
Setting stream QoS
QoS: waiting for 1 streams
Audio Stream 202EBC8C QoS set
Stream QoS Set
Enabling streams
Audio Stream 202EBC8C enabled
Streams enabled
Starting streams
Audio Stream 202EBC8C started
Streams started
```

Step 3: Input "pause" command to stop playing

```
BMR@left>> pause
pause

BMR@left>> Stream 2025554C stopped
```

Step 4: Input "play" command to start playing

```
BMR@left>> play
play

BMR@left>> Syncing to unicast
Stream 2025554C started
```

Step 5: Input "vol_set 100" to set the volume to level 100

```
BMR@left>> vol_set 100
vol_set 100

BMR@left>>
```

Step 6: Input "vol_up" command and increase the volume

```
BMR@left>> vol_up
vol_up

vol: 124
BMR@left>>
```

Step 7: Input "vol_down" command and decrease the volume

```
BMR@left>> vol_down
vol_down

vol: 99
BMR@left>>
```

Step 8: Input "vol_mute" command and mute the volume

```
BMR@left>> vol_mute

vol_mute

BMR@left>>
```

Step 9: Input "vol_unmute" command and unmute the volume

```
BMR@left>> vol_unmute
vol_unmute

vol: 99
BMR@left>>
```

5.23 Unicast media receiver

This section describes the application to demonstrate on how to use the unicast media receiver example of the LE audio feature.

The Unicast Media Receiver (UMR) role is defined for devices that receive media audio content from a source device in one or more Unicast Audio Streams. Typical devices implementing the UMR role include headphones, earbuds, and wireless speakers.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.23.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the example project

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Connect a speaker/headphone to the 3.5mm audio jack of i.MX RT EVK board

Step 3: Apply a power reset on i.MX RT EVK board

Step 4: Check the console on the connected computer screen to see the application start-up logs

5.23.2 Application execution

This section describes the steps for application execution.

Step 1: Press RESET button and restart the i.MX RT EVK board

When the demo starts, following message about the demo would appear on the console.

```
Copyright  2022  NXP

UMR>>
Unicast Media Receiver.

Please select sink role "left"|"right" use "init" command.
```

Step 2: Select the sink role

Once the sink role is selected, the application automatically start receiving the media samples and the following message would appear on the console.

```
UMR>> init left
```

```
UMR@left>> UMR@left>> Bluetooth initialized
Location successfully set
Supported contexts successfully set
Available contexts successfully set
Advertising successfully started
Connected: D0:17:69:EE:71:13 (public)
Security changed: D0:17:69:EE:71:13 (public) level 2 (error 0)
MCS server discover:
MCS server discovered.
ASE Codec Config: conn 202DD4EC ep 202D08C4 dir 1
codec 0x06 cid 0x0000 vid 0x0000 count 5
data #0: type 0x01 len 1

data #1: type 0x02 len 1

data #2: type 0x03 len 4
010000
data #3: type 0x04 len 2
28
data #4: type 0x05 len 1

  Frequency: 16000 Hz
  Frame Duration: 10000 us
  Channel allocation: 0x1
  Octets per frame: 40 (negative means value not pressent)
  Frames per SDU: 1
ASE Codec Config stream 2030A0B8
QoS: stream 2030A0B8 qos 202D0934
QoS: interval 10000 framing 0x00 phy 0x02 sdu 40 rtn 2 latency 10 pd 40000
Enable: stream 2030A0B8 meta_count 1
       Codec: freq 16000, channel count 1, duration 10000, channel alloc
0x00000001, frame len 40, frame blocks per sdu 1
Unicast stream started
```

Step 3: Input "pause" command to stop playing

```
BMR@left>> pause
pause

BMR@left>> Stream 2025554C stopped
Broadcast sink stoped!
```

Step 4: Input "play" command to start playing

```
BMR@left>> play
play

BMR@left>> Syncing to unicast
Stream 2025554C started
```

Step 5: Input "vol_set 100" to set the volume to level 100

```
BMR@left>> vol_set 100
vol_set 100

BMR@left>>
```

Step 6: Input "vol_up" command and increase the volume

```
BMR@left>> vol_up
vol_up

vol: 124
BMR@left>>
```

Step 7: Input "vol_down" command and decrease the volume

```
BMR@left>> vol_down
vol_down

vol: 99
BMR@left>>
```

Step 8: Input "vol_mute" command and mute the volume

```
BMR@left>> vol_mute
vol_mute

BMR@left>>
```

Step 9: Input "vol_unmute" command and unmute the volume

```
BMR@left>> vol_unmute
vol_unmute

vol: 99
BMR@left>>
```

5.24 Telephone call gateway Application

This section describes the application to demonstrate on how to use the telephone call gateway example of the LE audio feature.
The Call Gateway (CG) role is defined for telephony or VoIP applications. The CG device has the connection to the call network infrastructure. Typical devices implementing the CG role include smartphones, laptops, tablets, and PCs.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.24.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.

Step 1: Build and flash the example project

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.

Step 2: Apply a power reset on i.MX RT EVK board

Step 3: Check the console on the connected computer screen to see the application start-up logs

5.24.2 Application execution

NOTE: *This sample example works with “telephone call terminal” example. Refer the section “Telephone call terminal Application”.*

This section describes the steps for application execution.

Step 1: Press RESET button and restart the i.MX RT EVK board
When the demo starts, following message about the demo would appear on the console.

```
Copyright  2022  NXP

call_gateway>> Bluetooth initialized
Get required Source Capability from codec. Codec configurations:
    Frequency 16000
    Duration 10000
    Frame bytes 40
    Frame blocks per SDU 1
    Location 3, channel count 2.
Get required Sink Capability from codec. Codec configurations:
    Frequency 16000
    Duration 10000
    Frame bytes 40
    Frame blocks per SDU 1
    Location 3, channel count 2.
Scanning started
```

The scanning of the device is started automatically. It starts to scanning the telephony call terminal device.

After the connection is established, following logs would appear on the console.

```
Found device: A0:CD:F3:77:E6:15 (public)MTU exchanged: 23/23
Connected to peer: A0:CD:F3:77:E6:15 (public)
MTU exchanged: 65/65
Security changed: A0:CD:F3:77:E6:15 (public) level 2 (error 0)
codec capabilities on conn 202DE3C0 dir 1 codec 2000D054. Codec configurations:
    Frequency 8000, 16000, 24000, 32000, 44100, 48000,
    Duration 10000,
    Channel count 2.
    Frame length min 40, max 120
    Frame blocks per SDU 1
    Pref context 0x206
conn 202DE3C0 dir 1 loc 3
conn 202DE3C0 snk ctx 519 src ctx 3
conn 202DE3C0 dir 1 ep 202DAC30
Discover sinks complete: err 0
codec capabilities on conn 202DE3C0 dir 2 codec 2000D054. Codec configurations:
    Frequency 8000, 16000, 24000, 32000, 44100, 48000,
    Duration 10000,
    Channel count 2.
    Frame length min 40, max 120
    Frame blocks per SDU 1
    Pref context 0x206
conn 202DE3C0 dir 2 loc 3
conn 202DE3C0 snk ctx 519 src ctx 3
conn 202DE3C0 dir 2 ep 202DAE38
Discover sources complete: err 0
```

After the message "Discover sources complete: err 0” is printed on telephone call terminal side console, all features are ready for the TMAP.

Step 2: Input "help" command to get the available command list

```
call_gateway>> help

help
"help": List all the registered commands

"exit": Exit program
scanning <on>/<off>
vol_set [0-255]
vol_up
vol_down
vol_mute
vol_unmute
call_accept <callIndex>: Accept a incoming call
call_outgoing <telephone bearer index> <callee_URI>: Originate a call
call_hold <callIndex>: Hold a active call
call_retrieve <callIndex>: Retrieve a active call
call_term <callIndex>: Terminate a call
call_join <callIndex1> [<callIndex2> <callIndex3> ...]: Join the calls
remote_call_incoming <telephone bearer index> <callee_URI> <caller_URI>
<caller_name>: Simulate a call
remote_call_term <callIndex>: Terminate a call
```

```
remote_call_answer <callIndex>: Simulate the outgoing has been accepted by the
remote
remote_call_hold <callIndex>: Hold a active call
remote_call_retrieve <callIndex>: Retrieve a active call

call_gateway>>
```

Step 3: Initiate the local outgoing call

Input below command on the call gateway side to initiate the call.

call_outgoing 0 <XX>:<YY>

Following message would appear on the console.

```
call_gateway>> call_outgoing 0 tel:qq

call_outgoing 0 tel:qq

outgoing call: callee uri tel:qq

Audio Stream 2025A624 configured
Audio Stream 2025A5EC configured
Audio Stream 2025A624 QoS set
Audio Stream 2025A5EC QoS set
Audio Stream 2025A624 enabled
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16

Set default headphone volume 70

Audio Stream 2025A5EC enabled
Audio Stream 2025A5EC started
Audio Stream 2025A624 started
Return code 0, call index is 1

call_gateway>>
```

Step 4: Accept the call

Input below command on the call gateway side to accept the call

call_accept <call_index>

OR

Input below command on the call terminal side to accept the call

call_accept 0 <call_index>

Following message would appear on the console.

```
call_gateway>> Accept a call, call index 1

Audio Stream 2025A624 disabled
Audio Stream 2025A624 QoS set
Audio Stream 2025A5EC disabled
Fail to stop stream (err -77)
Audio Stream 2025A5EC QoS set
Audio Stream 2025A624 stopped with reason 0x13
Audio Stream 2025A5EC stopped with reason 0x13
Audio Stream 2025A624 enabled
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16

Set default headphone volume 70
```

```
Audio Stream 2025A5EC enabled
Audio Stream 2025A5EC started
Audio Stream 2025A624 started
```

Step 5: Reject/End the call

Input below command on the call gateway side to reject/end the call

call_term <call_index>

OR

Input below command on the call terminal side to reject/end the call the call

call_term 0 <call_index>

Following type of message would appear on the console.

```
call_gateway>> call_term 1

call_term 1

terminate the call: call index 1

Audio Stream 2025A624 disabled
Audio Stream 2025A624 QoS set
Audio Stream 2025A5EC disabled
Fail to stop stream (err -77)
Audio Stream 2025A5EC QoS set
Audio Stream 2025A624 stopped with reason 0x13
Audio Stream 2025A5EC stopped with reason 0x13
Audio Stream 2025A624 released
Audio Stream 2025A5EC released
Return code 0

call_gateway>>
```

Step 6: Initiate a call by remote.

Input below command to start the remote incoming call.

remote_call_incoming 0 <AA>:<BB> <CC>:<DD> <EE>

Following type of message would appear on the console.

```
remote_call_incoming 0 tel:qq tel:qq qq
incoming call: callee uri tel:qq, caller uri tel:qq
Audio Stream 202F0688 configured
Audio Stream 202F0650 configured
Audio Stream 202F0688 QoS set
Audio Stream 202F0650 QoS set
Audio Stream 202F0688 enabled
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16
Set default headphone volume 70
Audio Stream 202F0650 enabled
Audio Stream 202F0650 started
Audio Stream 202F0688 started
done, call index is 0
```

Step 7: Accept the call by remote

Input below command to accept the remote incoming call.

remote_call_answer <call_index>

Following type of message would appear on the console.

```
Remove answer the call: call index 1
Audio Stream 202F0688 disabled
Audio Stream 202F0688 QoS set
Audio Stream 202F0650 disabled
Fail to stop stream (err -77)
Audio Stream 202F0650 QoS set
Audio Stream 202F0688 stopped with reason 0x13
Audio Stream 202F0650 stopped with reason 0x13
Audio Stream 202F0688 enabled
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16
Set default headphone volume 70
Audio Stream 202F0650 enabled
Audio Stream 202F0650 started
Audio Stream 202F0688 started
Return code 0
```

5.25 Telephone call terminal Application

This section describes the application to demonstrate on how to use the telephone call terminal example of the LE audio feature.
The Call Terminal (CT) role is defined for headset type devices in telephony or VoIP applications. Typical devices implementing the CT role include wireless headsets, speakers, and microphones that participate in conversational audio.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.25.1 Prepare the setup for Application demo

This section describes the steps to prepare the setup for application demo execution.
Step 1: Build and flash the example project

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs.
Step 2: Apply a power reset on i.MX RT EVK board

Step 3: Check the console on the connected computer screen to see the application start-up logs.

5.25.2 Application execution

NOTE: *This sample example works with “telephone call gateway” example. Refer the section “Telephone call gateway Application”.*

This section describes the steps for application execution.
Step 1: Press RESET button and restart the i.MX RT EVK board
When the demo starts, following message about the demo would appear on the console.

```
call_terminal>> Bluetooth initialized
Advertising successfully started
MTU exchanged: 23/23
Starting TBS server discover
```

The scanning of the device is started automatically. It starts to scanning the telephony call gateway device. After the connection is established, following message would appear on the console.

```
Connected to peer: A0:CD:F3:77:E6:1D (public)
MTU exchanged: 65/65
Security changed: A0:CD:F3:77:E6:1D (public) level 2 (error 0)
Discover complete (err 0)! TBS count 1, GTBS found? Yes
```

After the message "Discover complete (err 0)! TBS count 1, GTBS found? Yes" is printed on telephone call terminal side console, all features are ready.
Step 2: Input "help" command to get the available command list

```
call_terminal>> help
help

"help": List all the registered commands

"exit": Exit program

vol_set [0-255]

vol_up

vol_down

vol_mute

vol_unmute
```

```
call_discover <subscribe flag>: Discover the TBS server features

call_accept <tbs index> <callIndex>: Accept a incoming call

call_outgoing <tbs index> <callee_URI>: Originate a call

call_hold <tbs index> <callIndex>: Hold a active call

call_retrieve <tbs index> <callIndex>: Retrieve a active call

call_term <tbs index> <callIndex>: Terminate a call

call_join <tbs index> <callIndex1> [<callIndex2> <callIndex3> ...]: Join the
calls

advertising <on>/<off>

call_terminal>>
```

Step 3: Initiate the local outgoing call

Input below command on the call terminal side to initiate the call.

call_outgoing 0 <XX>:<YY>

Following type of message would appear on the console.

```
call_terminal>> List current state of current calls (err 0). TBS Index 255,
call count 1, call state list,

call index 1, state 1, flags 1.

List current calls (err 0). TBS Index 255, call count 1, call list,

call index 1, state 1, flags 1, remote uri tel:qq

List current state of current calls (err 0). TBS Index 0, call count 1, call
state list,

call index 1, state 1, flags 1.

List current state of current calls (err 0). TBS Index 255, call count 1, call
state list,

call index 1, state 2, flags 1.

List current calls (err 0). TBS Index 255, call count 1, call list,

call index 1, state 2, flags 1, remote uri tel:qq

List current state of current calls (err 0). TBS Index 0, call count 1, call
state list,

call index 1, state 2, flags 1.

ASE Codec Config: conn 20270D1C ep 20275E68 dir 2
Codec configurations:
    Frequency 16000
    Duration 10000
    Frame bytes 40
    Frame blocks per SDU 1
    Location 3, channel count 2.
ASE Codec Config: conn 20270D1C ep 20275F30 dir 1
Codec configurations:
    Frequency 16000
    Duration 10000
    Frame bytes 40
    Frame blocks per SDU 1
    Location 3, channel count 2.
QoS: stream 20258CF8 qos 20275ED8
    interval 10000 framing 0x00 phy 0x02 sdu 80 rtn 2 latency 10 pd 40000
QoS: stream 2025F0C8 qos 20275FA0
    interval 10000 framing 0x00 phy 0x02 sdu 80 rtn 2 latency 10 pd 40000
Enable: stream 20258CF8 meta_count 1
Enable: stream 2025F0C8 meta_count 1
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16

Set default headphone volume 70

Start: stream 2025F0C8
Start: stream 20258CF8
```

Step 4: Accept the call

Input below command on the call gateway side to accept the call

call_accept <call_index>

OR

Input below command on the call terminal side to accept the call

call_accept 0 <call_index>

Following type of message would appear on the console.

```
call_terminal>> call_accept 0 3

call_accept 0 3

accept call: TBS index , call index 3

Return code 0

call_terminal>> Control Point status update. A call has been accepted (err 0).
TBS Index 0, call index 3

List current state of current calls (err 0). TBS Index 255, call count 2, call
state list,

call index 1, state 0, flags 0.

call index 3, state 3, flags 0.

List current calls (err 0). TBS Index 255, call count 2, call list,

call index 1, state 0, flags 0, remote uri tel:qq

call index 3, state 3, flags 0, remote uri tel:qq

List current state of current calls (err 0). TBS Index 0, call count 2, call
state list,

call index 1, state 0, flags 0.

call index 3, state 3, flags 0.
```

Step 5: Reject/End the call

Input below command on the call gateway side to reject/end the call
call_term <call_index>

OR

Input below command on the call terminal side to reject/end the call the call
call_term 0 <call_index>

Following message would appear on the console.

```
call_terminal>> call_term 0 3

call_term 0 3

Terminate call: TBS index 0, call index 3

Return code 0

call_terminal>> Call terminated(err 0). TBS Index 0, call index 3, reason 6.

Speaker mute

Call terminated(err 0). TBS Index 255, call index 3, reason 6.

Control Point status update. A call has been terminated (err 0). TBS Index 0,
call index 3

List current state of current calls (err 0). TBS Index 255, call count 1, call
state list,

call index 1, state 0, flags 0.

List current calls (err 0). TBS Index 255, call count 1, call list,

call index 1, state 0, flags 0, remote uri tel:qq

List current state of current calls (err 0). TBS Index 0, call count 1, call
state list,

call index 1, state 0, flags 0.
```

Step 6: Initiate a call by remote.
Input below command to start the remote incoming call.
remote_call_incoming 0 <AA>:<BB> <CC>:<DD> <EE>

Following type of message would appear on the console.

```
Read incoming call URI tel:qq (err 0). TBS Index 0.
incoming call inst_index 0, call_index = 1, uri tel:qq
Read Friendly name qq (err 0). TBS Index 0.
Read incoming call URI tel:qq (err 0). TBS Index 255.
incoming call inst_index 255, call_index = 1, uri tel:qq
Read Friendly name  (err 0). TBS Index 255.
List current state of current calls (err 0). TBS Index 255, call count 1, call
state list,
call index 1, state 0, flags 0.
List current calls (err 0). TBS Index 255, call count 1, call list,
call index 1, state 0, flags 0, remote uri tel:qq
List current state of current calls (err 0). TBS Index 0, call count 1, call
state list,
call index 1, state 0, flags 0.
ASE Codec Config: conn 202DE340 ep 202D9214 dir 2
Codec configurations:
    Frequency 16000
    Duration 10000
    Frame bytes 40
    Frame blocks per SDU 1
    Location 3, channel count 2.
ASE Codec Config: conn 202DE340 ep 202D92DC dir 1
Codec configurations:
    Frequency 16000
    Duration 10000
    Frame bytes 40
    Frame blocks per SDU 1
    Location 3, channel count 2.
QoS: stream 202EFF80 qos 202D9284
    interval 10000 framing 0x00 phy 0x02 sdu 80 rtn 2 latency 10 pd 40000
QoS: stream 202F6350 qos 202D934C
    interval 10000 framing 0x00 phy 0x02 sdu 80 rtn 2 latency 10 pd 40000
Enable: stream 202EFF80 meta_count 1
Enable: stream 202F6350 meta_count 1
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16
Set default headphone volume 70
Start: stream 202F6350
Start: stream 202EFF80
```

Step 7: Accept the call by remote

Input below command to accept the remote incoming call.

remote_call_answer <call_index>

Following type of message would appear on the console.

```
List current state of current calls (err 0). TBS Index 255, call count 1, call
state list,
call index 1, state 3, flags 1.
List current calls (err 0). TBS Index 255, call count 1, call list,
call index 1, state 3, flags 1, remote uri tel:qq
List current state of current calls (err 0). TBS Index 0, call count 1, call
state list,
call index 1, state 3, flags 1.
Disable: stream 202EFF80
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Fail to send stream (error -77)
Stop: stream 202EFF80
Disable: stream 202F6350
Audio Stream 202F6350 stopped with reason 0x13
Audio Stream 202EFF80 stopped with reason 0x13
Enable: stream 202EFF80 meta_count 1
Enable: stream 202F6350 meta_count 1
Init Audio SAI and CODEC, samplingRate :16000  bitWidth:16
Set default headphone volume 70
Start: stream 202F6350
Start: stream 202EFF80
```

5.26 Wireless UART Sample Application

The application implements a custom GATT based Wireless UART Profile that emulates UART over BLE. Central and peripheral role can be switched by user button (SW8). To test the service/profile the "IoT Toolbox" application can be used which is available for both Android and iOS. IoT Toolbox can be found on Apple App Store or Google Play Store.

5.26.1 wireless_uart Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.26.1.1 Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
BLE Wireless Uart demo start...
Bluetooth initialized
Advertising successfully started
```

The demo requires user interaction. The application will automatically start advertising the wireless uart Service after reset, the application can only accept 1 connection when configured as a peripheral.

The application will start scanning and connect to the wireless uart Service automatically.

Pressing the Button will switch from Peripheral mode to central mode and now it can connect to 8 devices. We can use "IoT Toolbox" or another wireless_uart example (use B to refer to) to test the current device.

peripheral role test:

Open "IoT Toolbox" application on an Android or iOS smartphone, select the "Wireless UART" option.
A device named "NXP_WU" should appear. Connect to "NXP_WU" by selecting the device from the scan list. The Android/iOS device should receive a prompt for a Bluetooth Pairing Request. Please complete the pairing process by entering the passkey that is displayed on the debug terminal. Once pairing is completed, we can now transmit and receive data over the emulated UART interface.

```
BLE Wireless Uart demo start...
Bluetooth initialized
Advertising successfully started
Connected to 4B:6B:F0:B6:7C:F8 (random)
GATT MTU exchanged: 65
[ATTRIBUTE] handle 40
[ATTRIBUTE] handle 41
Passkey for 4B:6B:F0:B6:7C:F8 (random): 994660
Security changed: 20:39:56:C6:6C:6C (public) level 4 (error 0)
Data received (length 5): hello
```

central role test:

let B work as default state after reset.
short press the user button(SW8), the example will work as central can automatically connect to any discovered wireless uart example. Each time short press, the example will scan and connect to wireless uart service if new device is found.

```
BLE Wireless Uart demo start...
Bluetooth initialized
Advertising successfully started
Scanning successfully started
[DEVICE]: 24:FC:E5:9F:EE:EB (public), AD evt type 3, AD data len 28, RSSI -92
[DEVICE]: 64:86:7F:5A:7C:7F (random), AD evt type 0, AD data len 23, RSSI -81
[DEVICE]: 64:86:7F:5A:7C:7F (random), AD evt type 4, AD data len 0, RSSI -80
[DEVICE]: 65:F2:7E:9A:AF:C7 (random), AD evt type 0, AD data len 19, RSSI -89
[DEVICE]: 65:F2:7E:9A:AF:C7 (random), AD evt type 4, AD data len 0, RSSI -89
[DEVICE]: 63:F2:B1:6A:FC:3D (random), AD evt type 0, AD data len 18, RSSI -80
[DEVICE]: 63:F2:B1:6A:FC:3D (random), AD evt type 4, AD data len 0, RSSI -80
[DEVICE]: 78:B3:AA:89:78:3B (random), AD evt type 0, AD data len 18, RSSI -80
[DEVICE]: 78:B3:AA:89:78:3B (random), AD evt type 4, AD data len 0, RSSI -79
[DEVICE]: 80:D2:1D:E8:2B:7E (public), AD evt type 0, AD data len 21, RSSI -43
Connected to 80:D2:1D:E8:2B:7E (public)
GATT MTU exchanged: 65
[ATTRIBUTE] handle 25
[ATTRIBUTE] handle 26
Security changed: 80:D2:1D:E8:2B:7E (public) level 2 (error 0)
```

NOTE*: The device address, AD event type data len, and RSSI are variable, it depends on all the Bluetooth device in test environment.*

Send data 12345 in B device's Serial port terminal, then current device will print the following log.

```
Data received (length 5): 12345
```

Send data 123 in current device's Serial port terminal, then B device will print the following log.

```
Data received (length 5): 123
```

5.27 Wi-Fi CLI over Wireless UART Sample Application

This section describes the application to demonstrate on how a wireless function based on "wifi_cli" demo and "wireless_uart" demo, enable users to use Wi-Fi command-line interface(CLI) over BLE wireless UART. The "IoT Toolbox" application can be used to test LE operations which is available for Android on Google Play Store and iOS on Apple App Store.

NOTE: *This sample application is only supported on IW612 with i.MX RT1170 EVKB board.*

5.27.1 Wi-Fi CLI over Wireless UART Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

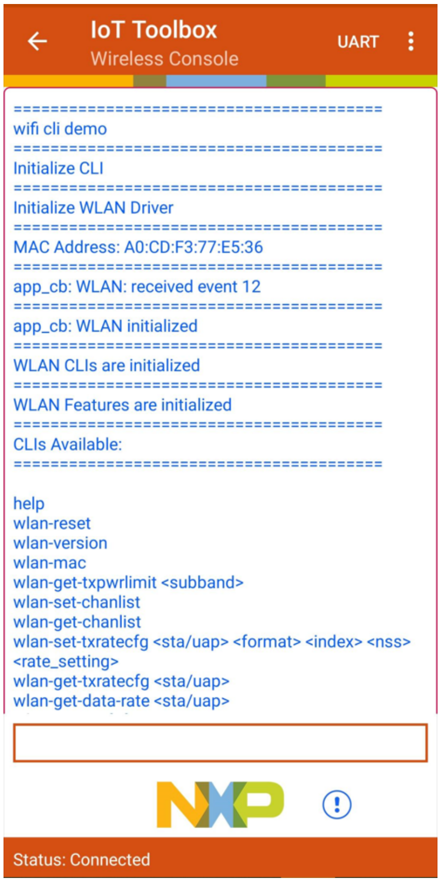
5.27.1.1 Run the application

Press the power reset button on i.MX RT1170 EVKB board to run the demo application downloaded on the board.

The application will automatically start advertising the wireless UART Service after reset. The demo require user interaction.

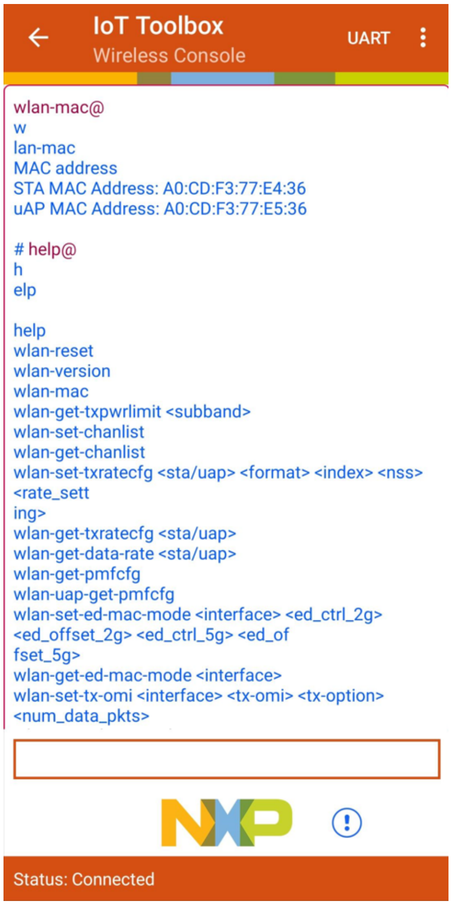
- **Open "IoT Toolbox" app on mobile and select the "Wireless UART" option.**
- **Search for the "NXP_WU" named device in the scan results of “IoT Toolbox” app.**
- **Click on the "NXP_WU" device to pair with i.MX RT EVK board.**

- **Accept the Pair request on "IoT Toolbox" app or else connection may fail.**
- **The following message of the demo would appear on the "IoT Toolbox" mobile app console.**



```
========================================
wifi cli demo
========================================
Initialize CLI
========================================
Initialize WLAN Driver
========================================
MAC Address: A0:CD:F3:77:E5:36
========================================
app_cb: WLAN: received event 12
========================================
app_cb: WLAN initialized
========================================
WLAN CLIs are initialized
========================================
WLAN Features are initialized
========================================
CLIs Available:
========================================

help
wlan-reset
wlan-version
wlan-mac
wlan-get-txpwrlimit <subband>
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss>
<rate_setting>
wlan-get-txratecfg <sta/uap>
wlan-get-data-rate <sta/uap>
```

- **After successful pairing, "IoT Toolbox" app can send/receive the data to the i.MX RT EVK board.**
- **"IoT Toolbox" app is not adding '\n' character on "Enter" key press. To avoid this limitation, change the default macro " #define END_CHAR '\n' " to any other uncommonly used character, such as " #define END_CHAR '@' " on the source code of the i.MX RT EVK board on this SDK path " <SDK_2_XX_X_MIMXRT1170-EVKB>\boards\evkbmimxrt1170\evkbmimxrt1170_wifi_cli_over_ble_wu\wifi\cli.c".**
- **Send the command ending with special character, example for "wlan-mac@" or "help@".**



```
wlan-mac@
w
lan-mac
MAC address
STA MAC Address: A0:CD:F3:77:E4:36
uAP MAC Address: A0:CD:F3:77:E5:36

# help@
h
elp

help
wlan-reset
wlan-version
wlan-mac
wlan-get-txpwrlimit <subband>
wlan-set-chanlist
wlan-get-chanlist
wlan-set-txratecfg <sta/uap> <format> <index> <nss>
<rate_sett
ing>
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_of
fset_5g>
wlan-get-ed-mac-mode <interface>
wlan-set-tx-omi <interface> <tx-omi> <tx-option>
<num_data_pkts>
```

Status: Connected

5.28 Shell Sample Application

Application Demonstrating the Interactive Shell Mode of Bluetooth Commands and APIs. It provides users full control over the Bluetooth Interface. User can control the basic Bluetooth operations such as advertising/scanning , device discovery , connection and pairing as well as direct access to the HCI command interface.

5.28.1 Shell Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.28.1.1 Shell Run the application

Press the power reset button on i.MX RT1060 EVK board to run the demo application downloaded on the board. When the demo starts, the following message about the demo would appear on the console.

```
Edgefast Bluetooth PAL shell demo start…

SHELL build: Aug  9 2022
Copyright  2020  NXP

@bt>
```

NOTE*: The shell information "SHELL build: Aug 10 2021" may be different, which depends on the compile date.*

The shell command list can be accessed by typing "help" in serial terminal. The demo can be configured to either "central" or "peripheral" by shell commands.

Here is an example of scan devices (the BLE 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]: 44:6D:F5:85:DC:5F (random), AD evt type 0, RSSI -64  C:1 S:1 D:0
SR:0 E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 44:6D:F5:85:DC:5F (random), AD evt type 4, RSSI -63  C:0 S:1 D:0 SR:1
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 6D:B3:D3:8E:ED:A2 (random), AD evt type 0, RSSI -77  C:1 S:1 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 6D:B3:D3:8E:ED:A2 (random), AD evt type 4, RSSI -76  C:0 S:1 D:0 SR:1
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 3F:FB:95:F7:F9:14 (random), AD evt type 3, RSSI -75  C:0 S:0 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 49:A3:4E:86:63:0C (random), AD evt type 0, RSSI -76  C:1 S:1 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 49:A3:4E:86:63:0C (random), AD evt type 4, RSSI -75  C:0 S:1 D:0 SR:1
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 5C:28:50:F9:DD:57 (random), AD evt type 0, RSSI -82  C:1 S:1 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 3B:95:00:4D:F3:EB (random), AD evt type 3, RSSI -82  C:0 S:0 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
[DEVICE]: 47:9D:D0:CB:5F:0D (random), AD evt type 0, RSSI -86  C:1 S:1 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
@bt> bt.scan off
Scan successfully stopped
```

```
@bt>
```

Here is an example of advertising (the BLE host must be initialized before):

```
@bt> bt.init
@bt> Bluetooth initialized

@bt> bt.advertise on
Advertising started
@bt> bt.advertise off
Advertising stopped
@bt>
```

RF Test Mode Operations

This section describes the commands to perform the RF test for Bluetooth Classic and Bluetooth Low Energy

NOTE : *The mentioned "command complete event" can be found in HCI log, U-DISK should be connected to usb port to get HCI log capture. CONFIG_BT_SNOOP macro is used to enable stack to capture the HCI log.*

Here is the log of rf_test_mode application:

```
>> help

@bt> help

+---"help": List all the registered commands
+---"exit": Exit program
+---"echo": Set echo(0 - disable, 1 - enable)
+---"bt": bt command entry
    +---"init": init [no-settings-load], [sync]
    +---"settings-load": settings-load [none]
    +---"id-create": id-create [addr]
    +---"id-reset": id-reset <id> [addr]
    +---"id-delete": id-delete <id>
    +---"id-show": id-show [none]
    +---"id-select": id-select <id>
    +---"name": name [name]
    +---"scan": scan <value: on, passive, off> [filter: dups, nodups] [fal]
    +---"scan-filter-set": scan-filter-set Scan filter set commands
        +---"name": name <name>
        +---"addr": addr <addr>
    +---"scan-filter-clear": scan-filter-clear Scan filter clear commands
        +---"all": all
        +---"name": name
        +---"addr": addr
    +---"advertise": advertise <type: off, on, scan, nconn> [mode: discov,
non_discov] [filter-accept-list: fal, fal-scan, fal-conn] [identity] [no-name]
[one-time] [name-ad][disable-37] [disable-38] [disable-39]
    +---"directed-adv": directed-adv <address: XX:XX:XX:XX:XX:XX> <type:
(public|random)> [mode: low] [identity] [dir-rpa]
    +---"connect": connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"disconnect": disconnect [none]
    +---"select": select <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"info": info <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"conn-update": conn-update <min> <max> <latency> <timeout>
    +---"data-len-update": data-len-update <tx_max_len> [tx_max_time]
    +---"phy-update": phy-update <tx_phy> [rx_phy] [s2] [s8]
    +---"channel-map": channel-map <channel-map: XXXXXXXXXX> (36-0)
    +---"oob": oob [none]
    +---"clear": clear <remote: addr, all>
    +---"security": security <security level BR/EDR: 0 - 3, LE: 1 - 4> [force-
pair]
    +---"bondable": bondable <bondable: on, off>
    +---"bonds": bonds [none]
    +---"connections": connections [none]
    +---"auth": auth <method: all, input, display, yesno, confirm, oob, status,
none>
    +---"auth-cancel": auth-cancel [none]
    +---"auth-passkey": auth-passkey <passkey>
    +---"auth-passkey-confirm": auth-passkey-confirm [none]
    +---"auth-pairing-confirm": auth-pairing-confirm [none]
    +---"auth-oob-tk": auth-oob-tk <tk>
    +---"oob-remote": oob-remote <address: XX:XX:XX:XX:XX:XX> <type:
(public|random)> <oob rand> <oob confirm>
    +---"oob-clear": oob-clear [none]
    +---"fal-add": fal-add <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"fal-rem": fal-rem <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
    +---"fal-clear": fal-clear [none]
    +---"fal-connect": fal-connect <on, off>
+---"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]
+---"l2cap": l2cap Bluetooth L2CAP shell commands
    +---"connect": connect <psm> [sec_level]
    +---"disconnect": disconnect [none]
    +---"metrics": metrics <value on, off>
    +---"recv": recv [delay (in miliseconds)
    +---"register": register <psm> [sec_level] [policy: allowlist, 16byte_key]
    +---"send": send <number of packets>
    +---"allowlist": allowlist [none]
        +---"add": add [none]
        +---"remove": remove [none]
+---"br": br Bluetooth BR/EDR shell commands
    +---"auth-pincode": auth-pincode <pincode>
    +---"connect": connect <address>
    +---"discovery": discovery <value: on, off> [length: 1-48] [mode: limited]
    +---"iscan": iscan <value: on, off>
    +---"l2cap-register": l2cap-register <psm>
    +---"l2cap-register-mode": l2cap-register-mode <psm> <mode:
                                    3. Enhanced Retransmission mode
                                    4. Streaming mode>
    +---"l2cap-connect": l2cap-connect <psm>
    +---"l2cap-disconnect": l2cap-disconnect [none]
    +---"l2cap-send": l2cap-send <number of packets>
    +---"oob": oob [none]
    +---"pscan": pscan <value: on, off>
    +---"sdp-find": sdp-find <HFPAG>
+---"rfcomm": rfcomm Bluetooth RFCOMM shell commands
    +---"register": register <channel>
    +---"connect": connect <channel>
    +---"disconnect": disconnect [none]
    +---"send": send <number of packets>




+---"a2dp": a2dp Bluetooth A2DP shell commands
    +---"register_sink_ep": register_sink_ep <select codec.
                        1:SBC
                        2:MPEG-1,2
                        3:MPEG-2,4
                        4:vendor
                        5:sbc with delay report and content protection services
                        6:sbc with all other services(don't support data
transfer yet)>
    +---"register_source_ep": register_source_ep <select codec.
                        1:SBC
                        2:MPEG-1,2
                        3:MPEG-2,4
                        4:vendor
                        5:sbc with delay report and content protection services
                        6:sbc with all other services(don't support data
transfer yet)>
    +---"connect": connect [none]
    +---"disconnect": disconnect [none]
    +---"configure": configure [none]
    +---"discover_peer_eps": discover_peer_eps [none]
    +---"get_registered_eps": get_registered_eps [none]
    +---"set_default_ep": set_default_ep <select endpoint>
    +---"configure_ep": configure_ep "configure the default selected ep"
    +---"deconfigure": deconfigure "de-configure the default selected ep"
    +---"start": start "start the default selected ep"
    +---"stop": stop "stop the default selected ep"
    +---"send_media": send_media <second> "send media data to the default
selected ep"
    +---"send_delay_report": send_delay_report <delay> "a2dp sink send delay
report to default selected ep"



+---"avrcp": avrcp Bluetooth AVRCP shell commands
    +---"init_ct": init_ct [none]
    +---"init_tg": init_tg [none]
    +---"ctl_connect": ctl_connect "create control connection"
    +---"brow_connect": brow_connect "create browsing connection"
    +---"ct_list_all_cases": ct_list_all_cases "display all the test cases"
    +---"ct_test_case": ct_test_case <select one case to test>
    +---"ct_test_all": ct_test_all "test all cases"
    +---"ct_reg_ntf": ct_reg_ntf <Register Notification. select event:
                                     1. EVENT_PLAYBACK_STATUS_CHANGED
                                     2. EVENT_TRACK_CHANGED
                                     3. EVENT_TRACK_REACHED_END
                                     4. EVENT_TRACK_REACHED_START
                                     5. EVENT_PLAYBACK_POS_CHANGED
                                     6. EVENT_BATT_STATUS_CHANGED
                                     7. EVENT_SYSTEM_STATUS_CHANGED
                                     8. EVENT_PLAYER_APPLICATION_SETTING_CHANGED
                                     9. EVENT_NOW_PLAYING_CONTENT_CHANGED
                                     a. EVENT_AVAILABLE_PLAYERS_CHANGED
                                     b. EVENT_ADDRESSED_PLAYER_CHANGED
                                     c. EVENT_UIDS_CHANGED
                                     d. EVENT_VOLUME_CHANGED>
    +---"tg_notify": tg_notify <Notify event. select event:
                                     1. EVENT_PLAYBACK_STATUS_CHANGED
                                     2. EVENT_TRACK_CHANGED
                                     3. EVENT_TRACK_REACHED_END
                                     4. EVENT_TRACK_REACHED_START
                                     5. EVENT_PLAYBACK_POS_CHANGED
                                     6. EVENT_BATT_STATUS_CHANGED
                                     7. EVENT_SYSTEM_STATUS_CHANGED
                                     8. EVENT_PLAYER_APPLICATION_SETTING_CHANGED
                                     9. EVENT_NOW_PLAYING_CONTENT_CHANGED
                                     a. EVENT_AVAILABLE_PLAYERS_CHANGED
                                     b. EVENT_ADDRESSED_PLAYER_CHANGED
                                     c. EVENT_UIDS_CHANGED
                                     d. EVENT_VOLUME_CHANGED>
    +---"ca_init_i": ca_init_i "Init cover art initiator"
    +---"ca_init_r": ca_init_r "Init cover art responder"
    +---"ca_connect": ca_connect "create cover art connection"
    +---"ca_test": ca_test "cover art test all cases"





+---"bt_test": bt_test Bluetooth BR/EDR test mode commands
    +---"enter_test_mode": enter_test_mode Enable device under test mode
    +---"tx_test": tx_test test_scenario[1] hopping_mode[1] tx_channel[1]
rx_channel[1] tx_test_interval[1] pkt_type[1] data_length[2] whitening[1]
num_pkt[4] tx_pwr[1]
    +---"rx_test": rx_test test_scenario[1] tx_channel[1] rx_channel[1]
pkt_type[1] num_pkt[4] data_length[2] tx_addr[6] report_err_pkt[1]
    +---"reset": reset  Reset the HCI interface
+---"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....

@bt>
>>
>> bt.init
bt.init
```

```
>>
>> Bluetooth initialized
Settings Loaded

>>
```

Enable the device under test mode

This command performs HCI reset

```
@bt> hci.generic_command 0x03 0x0003
```

Command output:

```
hci.generic_command 0x03 0x0003
HCI Command Response : @bt> 00
```

This command enables perform HCI reset

```
@bt> hci.generic_command 0x03 0x001a 0x3
```

Command output:

```
hci.generic_command 0x03 0x001a 0x3
HCI Command Response : @bt> 00
```

This command sets event filter

```
@bt> hci.generic_command 0x03 0x0005 0x02 0x00 0x02
```

Command output:

```
hci.generic_command 0x03 0x0005 0x02 0x00 0x02
HCI Command Response : @bt> 00
```

This command puts the controller into the test mode

```
@bt> bt_test.enter_test_mode
```

```
Enable device under test mode
@bt>  HCI Command Response : 00
```

Set the transmit test parameters for Bluetooth Classic

This command sets the transmit test parameters. An HCI reset command is required after this test to resume normal Bluetooth operation.

```
@bt> bt_test.tx_test 01 00 01 01 0D 03 0F 00 00 00 00 00 00 04
```

Command output example:

```
rx_on_start default set to=80

synt_on_start default set to=80

tx_on_start default set to=80

phd_off_start default set to=80

test_scenario= 1

hopping_mode= 0

tx_channel= 1
```

```
rx_channel= 1

tx_test_interval= d

pkt_type= 3

data_length= f 0

whitening= 0

num_pkt= 0 0 0 0

tx_pwr= 4

@bt> HCI Command Response : 00
```

Command Parameters :

Name	**Length**	**Description**
RxOnStart	1	These 4 parameters should be set to 0x80. ***NOTE :*** *bt_test.tx_test command includes these 4 parameters with the default value set to 0x80*
SyntOnStart	1	
TxOnStart	1	
PhdOffStart	1	
TestScenario	1	0x01 = PATTERN_00 (data pattern: 0x00) 0x02 = PATTERN_FF (data pattern: 0xFF) 0x03 = PATTERN_55 (data pattern: 0x55) 0x04 = PATTERN_PRBS (data pattern: 0xFE) 0x09 = PATTERN_0F (data pattern: 0x0F) 0xFF = exit test
HoppingMode	1	0x00 = fix frequency 0x01 = hopping set
TxChannel	1	Transmit Frequency = (2402+k) MHz, where k is the value of TxChannel
RxChannel	1	Receive Frequency = (2402+k) MHz, where k is the value of RxChannel
TxTestInterval	1	Poll interval in frames for the link (units, 1.25 ms)
PacketType	1	Transmit Packet Type 0x03 = DM1 0x04 = DH1 0x0A = DM3 0x0B = DH3 0x0E = DM5 0x0F = DH5 0x14 = 2-DH1 0x18 = 3-DH1 0x1A = 2-DH3 0x1B = 3-DH3 0x1E = 2-DH5 0x1F = 3-DH5
Length	2	Length of Test Data
Whitening	1	0x00 = disabled 0x01 = enabled

Number of Test Packets	4	0 = infinite (default)
Tx Power	1	Signed value of Tx power (dBm) Range = -20 dBm to 12 dBm (default = 4 dBm)

End transmitter test for Bluetooth Classic:

```
@bt>  bt_test.tx_test FF 00 01 01 0D 03 0F 00 00 00 00 00 00 04
```

Observe the packet count in vendor-specific command complete event in HCI logs (Refer to the table below for Event details).

<table>
<tr><th>Event Name</th><th>Event Code</th><th>Event ID</th><th>Parameters</th></tr>
<tr><td>Tx Test Result</td><td>0xFF</td><td>0x19</td><td>
<table>
<tr><th>Name</th><th>Length</th><th>Value</th></tr>
<tr><td>Status</td><td>1</td><td>0x00 = completed
0x01 = aborted</td></tr>
<tr><td>Total Packets</td><td>4</td><td>(in hexadecimal)</td></tr>
</table>
</td></tr>
</table>

Set the Receive Test parameters for Bluetooth Classic

This command sets the receive test parameters. An HCI reset command is required after this test to resume normal Bluetooth operation.

```
@bt> bt_test.rx_test 01 01 01 03 10 00 00 00 0F 00 20 4E F6 EC 1F 26 00
```

Command output example :

```
test_scenario= 1

tx_channel= 1

rx_channel= 1

pkt_type= 3

num_pkt= 10 0 0 0

data_length= f 0

tx_am_addr default set to= 1

tx_addr:
20
4e
f6
ec
1f
26

report_err_pkt= 0

@bt> HCI Command Response : 00
```

Command Parameters :

Name	Length	Description
TestScenario	1	Test Scenario 0x01 = receiver test, 0–pattern 0x02 = receiver test, 1–pattern 0x03 = receiver test, 1010–pattern 0x04 = receiver test, PRBS–pattern 0x09 = receiver test, 1111 0000–pattern 0xFF = abort test mode
TxFrequency	1	Transmit Frequency f = (2402+k) MHz
RxFrequency	1	Receive Frequency f = (2402+k) MHz
TestPacketType	1	Test Packet Type 0x03 = DM1 0x04 = DH1 0x0A = DM3 0x0B = DH3 0x0E = DM5 0x0F = DH5 0x14 = 2-DH1 0x18 = 3-DH1 0x1A = 2-DH3 0x1B = 3-DH3 0x1E = 2-DH5 0x1F = 3-DH5
Expected Number of Packets	4	--
Length of Test Data	2	Should not be bigger than the maximum size of the specified test packet type
Tx AM Address	1	Default = 0x01
Transmitter BD Address	6	This is used to derive the access code
Report Error Packets	1	Report Error Packets 0x00 = none (default) 0x01 to 0xFE = number of packets to report

End receiving test for Bluetooth Classic:

```
@bt> bt_test.rx_test FF 01 01 03 10 00 00 00 0F 00 20 4E F6 EC 1F 26 00
```

Observe the packet count in vendor-specific command complete event in HCI logs (Refer to the table below for Event details).

Event Name	EventCode	Event ID	Parameters

			Name	Length	Value
Receive Test Result	0xFF	0x01	Status	1	0x00 = completed 0x01 = aborted
			Total Packets (Expected)	4	(in hexadecimal)
			No Rx Count	4	(in hexadecimal)
			Successful Correlation Count	4	(in hexadecimal)
			HEC Match Count	4	(in hexadecimal)
			HEC Match CRC Packets Count	4	(in hexadecimal)
			Payload Hdr Error Count	4	(in hexadecimal)
			CRC Error Count	4	(in hexadecimal)
			Total Packet Received	4	(in hexadecimal)
			Packet OK Count	4	(in hexadecimal)
			Drop Packet Count	4	(in hexadecimal)
			Packet Error Rate (%)	4	(in hexadecimal)
			Total Number of Bits (Expected)	4	(in hexadecimal)
			Total Number of Bit Errors (Lost+Drop)	4	(in hexadecimal)
			Bit Error Rate	4	(in hexadecimal)
			Total Number of Bytes (Received)	4	(in hexadecimal)
			Total Number of Bit Errors (Received)	4	(in hexadecimal)
			Average RSSI	4	(in decimal)

Perform HCI reset

An HCI Reset command is required after this test to resume normal Bluetooth operations.

```
@bt> bt_test.reset
API returned success...
```

Bluetooth LE Set TX Power

This command sets the Bluetooth LE transmit power level.

```
bt> le_test.set_tx_power 4
```

```
 tx_power= 4
@bt> HCI Command Response : 00
```

Parameter	Length	Definition
TX_POWER	1	Min value : 0xE2 (-30 dBm) Max value : 0x14 (20 dBm) Default value = 0x00

Bluetooth LE Transmitter test

To start a test where the DUT generates test reference packets at a fixed interval, use LE Transmitter Test[V2] command. For more details on the command please 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
```

Command output example :

```
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.

Bluetooth LE receiver test :
To start a test where the DUT receives test reference packets at a fixed interval, use LE Receiver Test[V2] command. For more details on the command please refer to Section 7.8.28 Bluetooth Core Specification v5.3 Vol 0 Part A.

```
@bt> le_test.rx_test 01 01 00
```

Command output example :

```
rx_channel= 1

phy= 1

modulation_index= 0

@bt> HCI Command Response : 00
```

End Test for Bluetooth LE:

To end any test for Bluetooth LE use the below command

```
@bt> le_test.end_test
API returned success...
>>
```

Running a2dp

The commands are as follows:

```
+---"a2dp": a2dp Bluetooth A2DP shell commands
    +---"register_sink_ep": register_sink_ep <select codec.
1:SBC
2:MPEG-1,2
3:MPEG-2,4
4:vendor
5:sbc with delay report and content protection services
6:sbc with all other services(don't support data transfer yet)>
    +---"register_source_ep": register_source_ep <select codec.
1:SBC
2:MPEG-1,2
3:MPEG-2,4
4:vendor
5:sbc with delay report and content protection services
6:sbc with all other services(don't support data transfer yet)>
    +---"connect": connect [none]
    +---"disconnect": disconnect [none]
    +---"configure": configure [none]
    +---"discover_peer_eps": discover_peer_eps [none]
    +---"get_registered_eps": get_registered_eps [none]
    +---"set_default_ep": set_default_ep <select endpoint>
    +---"configure_ep": configure_ep "configure the default selected ep"
    +---"deconfigure": deconfigure "de-configure the default selected ep"
    +---"start": start "start the default selected ep"
    +---"stop": stop "stop the default selected ep"
    +---"send_media": send_media <second> "send media data to the default 
selected ep"
```

Test flow:

1. **Create ACL connection between two devices (A and B).**
2. **In device B, input "a2dp.register_sink_ep x" to initialize sink endpoint.**
3. **In device A, input "a2dp.register_source_ep x" to initialize source endpoint.**
4. **In device A, input "a2dp.connect" to create a2dp connection with the default ACL connection.**
5. **In device A, input "a2dp.configure" to configure the a2dp connection.**
6. **In device A, input "a2dp.start" to start the a2dp media.**
7. **In device A, input "a2dp.send_media x" to send media data for x seconds.**
8. **For other commands:**
    i. **"a2dp.disconnect" is used to disconnect the a2dp.**
    ii. **"a2dp.discover_peer_eps" is used to discover peer device's endpoints.**
    iii. **"a2dp.get_registered_eps" is used to get the local registered endpoints.**
    iv. **"a2dp.set_default_ep" is used to set the default selected endpoint.**
    v. **"a2dp.deconfigure" de-configure the endpoint, then it can be configured again.**
    vi. **"a2dp.stop" stops media.**
    vii. **"a2dp.send_delay_report" send delay report.**

Running avrcp

The commands are as follows:

```
+---"avrcp": avrcp Bluetooth AVRCP shell commands
    +---"init_ct": init_ct [none]
    +---"init_tg": init_tg [none]
    +---"ctl_connect": ctl_connect "create control connection"
    +---"brow_connect": brow_connect "create browsing connection"
    +---"ct_list_all_cases": ct_list_all_cases "display all the test cases"
    +---"ct_test_case": ct_test_case <select one case to test>
    +---"ct_test_all": ct_test_all "test all cases"
    +---"ct_reg_ntf": ct_reg_ntf <Register Notification. select event:
                                    1. EVENT_PLAYBACK_STATUS_CHANGED
                                    2. EVENT_TRACK_CHANGED
                                    3. EVENT_TRACK_REACHED_END
                                    4. EVENT_TRACK_REACHED_START
                                    5. EVENT_PLAYBACK_POS_CHANGED
                                    6. EVENT_BATT_STATUS_CHANGED
                                    7. EVENT_SYSTEM_STATUS_CHANGED
                                    8. EVENT_PLAYER_APPLICATION_SETTING_CHANGED
                                    9. EVENT_NOW_PLAYING_CONTENT_CHANGED
                                    a. EVENT_AVAILABLE_PLAYERS_CHANGED
                                    b. EVENT_ADDRESSED_PLAYER_CHANGED
                                    c. EVENT_UIDS_CHANGED
                                    d. EVENT_VOLUME_CHANGED>
    +---"tg_notify": tg_notify <Notify event. select event:
                                    1. EVENT_PLAYBACK_STATUS_CHANGED
                                    2. EVENT_TRACK_CHANGED
                                    3. EVENT_TRACK_REACHED_END
                                    4. EVENT_TRACK_REACHED_START
                                    5. EVENT_PLAYBACK_POS_CHANGED
                                    6. EVENT_BATT_STATUS_CHANGED
                                    7. EVENT_SYSTEM_STATUS_CHANGED
                                    8. EVENT_PLAYER_APPLICATION_SETTING_CHANGED
                                    9. EVENT_NOW_PLAYING_CONTENT_CHANGED
                                    a. EVENT_AVAILABLE_PLAYERS_CHANGED
                                    b. EVENT_ADDRESSED_PLAYER_CHANGED
                                    c. EVENT_UIDS_CHANGED
                                    d. EVENT_VOLUME_CHANGED>
    +---"ca_init_i": ca_init_i "Init cover art initiator"
    +---"ca_init_r": ca_init_r "Init cover art responder"
    +---"ca_connect": ca_connect "create cover art connection"
    +---"ca_test": ca_test "cover art test all cases"
```

Test flow:

1. **Create ACL connection between two devices (A and B).**
2. **In device B, input "avrcp.init_tg" to initialize Target.**
3. **In device A, input "avrcp.init_ct" to initialize Controller.**
4. **In device B, input "avrcp.ca_init_r" to initialize Cover Art responder.**
5. **In device A, input "avrcp.ca_init_i" to initialize Cover Art Initiator.**
6. **In device A, input "avrcp.ctl_connect" to create AVRCP Control connection.**
7. **In device A, input "avrcp.brow_connect" to create AVRCP Browsing connection.**
8. **In device A, input "avrcp.ct_test_all" to test all the cases.**
9. **In device A, input "avrcp.ct_reg_ntf" to register notification.**
10. **In device B, input "avrcp.tg_notify" to notify.**
11. **In device A, input "avrcp.ca_test" to test all the cover art commands.**
12. **For other commands:**
    i. **In device A, input "avrcp.ct_list_all_cases" to list all the test cases.**
    ii. **In device A, input "avrcp.ct_test_case x" to test one selected case.**

Running BR/EDR L2CAP

1. **Create ACL connection between two devices (A and B).**
2. **In device A and B, input "br.l2cap-register <psm>" to register one psm (for example: br.l2cap-register 1001).**
3. **In device A, input "br.l2cap-connect <psm>" to create l2cap connection (for example: br.l2cap-connect 1001).**
4. **In device A, input "br.l2cap-send x" to send data.**
5. **In device A, input "br.l2cap-disconnect" to disconnect the l2cap connection.**
6. **In device A and B, input "br.l2cap-register-mode <psm>" to register one psm (for example: br.l2cap-register 1003).**
7. **In device A, input "br.l2cap-connect <psm>" to create l2cap connection (for example: br.l2cap-connect 1003).**
8. **In device A, input "br.l2cap-send x" to send data.**
9. **In device A, input "br.l2cap-disconnect" to disconnect the l2cap connection.**

Example of BLE pairing and bonding

GATT peripheral role side

Initialize the Host.

```
@bt> bt.init
```

Start Advertising

```
@bt> bt.advertise on
```

After the connection is established, perform the pairing sequence, it could be started from peripheral side by *“bt.security <level>”,* such as

```
@bt> bt.security 2
```

If the bondable is unsupported by peripheral role then enter the command below and repeat step 3.

```
@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 few seconds

```
@bt> bt.scan off
```

Select the target board and create a new connection. If the target is not listed, repeat “scan on” and “scan off” then enter *“bt.connect <remote address: XX:XX:XX:XX:XX:XX> <type: (public|random)>” ,*

```
@bt> bt.connect 11:22:33:44:55:66 public
```

After the connection is established, perform the pairing sequence, it could be started from central side by “bt.security <level>", such as

```
@bt> bt.security 2
```

If the bondable is unsupported by central role then enter the command below and repeat the previous step

```
@bt> bt.bondable off
```

After all the operations are performed , we can initiate a disconnection from the central device

```
@bt> bt.disconnect
```

Example of GATT data signing

GATT peripheral role side

Initialize the Host

```
@bt> bt.init
```

Enable Advertising

```
@bt> bt.advertise on
```

After the connection is established, perform the pairing sequence, it could be started from peripheral side by “bt.security <level>”

```
@bt> bt.security 2
```

After the authentication is successfully, disconnect the connection, it could be started from peripheral side by

```
@bt> bt.disconnect
```

Reinitiate the advertising and wait for new connection. After the connection is established (LL encryption should be disabled), add new service ""

```
@bt> gatt.register
```

GATT central role side

Initialize the Host

```
@bt> bt.init
```

Scanning advertising packets

```
@bt> bt.scan on
```

A few seconds later, stop the scanning

```
@bt> bt.scan off
```

Select the target board and create a new connection. If the target is not listed, then scan for the devices

```
@bt> "bt.connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>"
```

After the connection is established, perform the pairing sequence, it could be started from central side by "*bt.security <level>*

```
@bt> bt.security 2
```

After the authentication is successfully, disconnect the connection, it could be started from central side by

```
@bt> bt.disconnect
```

Repeat the previous steps to start and stop scanning for few more seconds and Reinitiate the connection. After the connection is established (LL encryption should be disabled), perform the GATT data signing sequence , i.e., *"gatt.signed-write <handle> <data> [length] [repeat]"*

```
@bt> gatt.signed-write 22 AA 1
```

After all the operations are performed , we can initiate a disconnection from the central device

```
@bt> bt.disconnect
```

Example of GATT Service Changed Indication,

GATT peripheral role side,

Initialize the Host

```
@bt> bt.init
```

Advertising

```
@bt> bt.advertise on
```

After the connection is established. and waiting for the service changed indication is subscribed

Add new service

```
@bt> "gatt.register"
```

Wait for the Central device to finish performing the operations. After that Remove the added service

```
@bt> "gatt.unregister".
```

GATT central role side,

Initialize the Host

```
@bt> "bt.init"
```

Scanning advertising packets

```
@bt> "bt.scan on"
```

A few seconds later, stop the scanning

```
@bt> "bt.scan off"
```

Select the target board and create a new connection. If the target is not listed, repeat the previous steps to start and stop scanning for few more seconds

```
@bt> "bt.connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>"
```

After the connection is established, subscribe the GATT service changed indicator.

i.e. : *"bt.subscribe <CCC handle> <value handle> [ind]"*

```
@bt> gatt.subscribe f e ind
```

After all the operations are performed , we can initiate a disconnection from the central device

```
@bt> bt.disconnect
```

Example of GATT Service Dynamic Database Hash

GATT peripheral role side,

Initialize the Host

```
@bt> bt.init
```

Advertising

```
@bt> bt.advertise on
```

After the connection is established. and waiting for the service changed indication is subscribed,

Add new service

```
@bt> gatt.register
```

Wait for the Central device to perform the operations and then remove the added service

```
@bt> gatt.unregister
```

GATT central role side,

Initialize the Host

```
@bt> bt.init
```

Scanning advertising packets

```
@bt> bt.scan on
```

A few seconds later, stop the scanning

```
@bt> bt.scan off
```

Select the target board and create a new connection. If the target is not listed, repeat the previous steps to start and stop scanning for few more seconds

```
@bt> bt.connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
```

After the connection is established, subscribe the GATT service changed indicator

```
@bt> bt.subscribe <CCC handle> <value handle> [ind]
i.e : gatt.subscribe f e ind
```

If the indication is indicated, read DB hash, i.e. : *“gatt.read <handle> [offset]”* or *“gatt.read-uuid <UUID> [start handle] [end handle]”*

```
@bt> gatt.read 13
Or
@bt> gatt.read-uuid 2b2a
```

After all the operations are performed , we can initiate a disconnection from the central device

```
@bt> bt.disconnect
```

Example of PHY 1M/2M update.

GATT peripheral role side,

Initialize the Host

```
@bt> bt.init
```

Enable Advertising

```
@bt> bt.advertise on
```

After the connection is established. Send PHY update command such as
“bt.phy-update <tx_phy> <rx_phy> ”. tx_phy/rx_phy could be 1(1M) or 2(2M).

```
@bt> bt.phy-update 2 2
```

The message "LE PHY updated: TX PHY LE 2M, RX PHY LE 2M" would be printed if the PHY is updated.

NOTE: *If peer don't support PHY update, then this message will not be printed.*

GATT central role side,

Initialize the Host

```
@bt> bt.init
```

start scan

```
@bt> bt.scan on
```

Bluetooth devices around your current Bluetooth will be list, for example,

stop scan

```
@bt> bt.scan off
[DEVICE]: 72:78:C1:B5:0F:DA (random), AD evt type 4, RSSI -32 BLE Peripheral
C:0 S:1 D:0 SR:1 E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms),
SID: 0xff

[DEVICE]: C4:0D:02:55:5E:AD (random), AD evt type 0, RSSI -83  C:1 S:1 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff

[DEVICE]: 66:8F:26:27:1F:52 (random), AD evt type 0, RSSI -82  C:1 S:1 D:0 SR:0
E:0 Prim: LE 1M, Secn: No packets, Interval: 0x0000 (0 ms), SID: 0xff
```

connect target device

```
bt.connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
@bt> bt.connect 72:78:C1:B5:0F:DA random
```

Send PHY update command

```
bt.phy-update <tx_phy> [rx_phy] [s2] [s8]", tx_phy/rx_phy could be 1(1M) or
2(2M).
such as "bt.phy-update 2 2/bt.phy-update 1 2". NOTE, the "s2" and "s8" are
unsupported.
@bt>
```

After all the operations are performed , we can initiate a disconnection from the central device

```
@bt> bt.disconnect
```

NOTE :*The message "LE PHY updated: TX PHY LE 2M, RX PHY LE 2M" would be printed if the phy is updated. NOTE, if peer don't support phy update, then this message will not be printed*.

Example of Filter Accept List.

GATT peripheral role side

Initialize the Host

```
@bt> bt.init
```

Adding device to Filter Accept List

```
bt.fal-add <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>
@bt> bt.fal-add 11:22:33:44:55:66 public
```

Enable Advertising

```
@bt> bt.advertise on fal-conn
```

Only the device in Filter Accept List can connect to the current device or else no log will be printed.

***NOTE:** if device address is added after command bt.advertise on, then Filter Accept List will take effect after re-star advertise. the bt.advertise off and bt.advertise on can be used to re-start the advertise.*

GATT central role side

Initialize the Host

```
@bt> bt.init
```

Adding device to Filter Accept List

```
"bt.fal-add <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>"
@bt> bt.fal-add 80:D2:1D:E8:2B:7E public
```

Initiate connection with the Filter Accept Listed device with the command *"bt.fal-connect on"*. The device will be connected with the following log.

```
@bt> Connected: 80:D2:1D:E8:2B:7E (public)
```

Initiate disconnection with the Filter Accept Listed "bt.disconnect". device will be disconnect.

```
@bt> Disconnected: 80:D2:1D:E8:2B:7E (public) (reason 0x16)
```

Remove the device from the Filter Accept List

```
@bt> bt.fal-rem 80:D2:1D:E8:2B:7E public
```

Running BR/EDR RFCOMM

***NOTE:** Only 1 rfcomm connection is supported in shell project.*

RFCOMM Server Side

Initialize Bluetooth

```
@bt> bt.init
```

Turn on pscan

```
@bt> br.pscan on
```

Turn on iscan

```
@bt> br.iscan on
```

Register rfcomm server channel 5

```
@bt> rfcomm.register 5
```

After rfcomm connection is created, To send data

```
@bt> rfcomm.send <count of sending>
```

After rfcomm connection is created, To disconnect with peer device

```
@bt> rfcomm.disconnect
```

RFCOMM Client Side

Initialize Bluetooth

```
@bt> bt.init
```

Enable Discovery

```
@bt> br.discovery on
```

Create Connection , i.e "br.connect <remote device address>"

```
@bt> br.connect 80:D2:1D:E8:2B:7E
```

Create RFCOMM connection on channel 5

```
@bt> rfcomm.connect 5
```

After connection, Send Data , i.e : "rfcomm.send <count of sending>"

```
@bt> rfcomm.send 3
```

After finishing the test , disconnect the RFCOMM connection

```
@bt> rfcomm.disconnect
```

Running Generic HCI Commands

This functionality allows execution of arbitrary command to the wireless controller.

The command format is as given below

hci.generic_command <ogf> <ocf> <n parameters>..

i.e. : Checking the firmware version with the vendor specific command

```
@bt> hci.generic_command 3f 0f
```

We get the command response

```
HCI Command Response :
@bt> 00 15 5B 10 40 00 00 02 04
```

Independent reset

The independent reset feature allows the host to reset the Bluetooth controller and re download the Bluetooth only firmware through UART without powering OFF the Bluetooth controller. Where the host resets the controller and re downloads the firmware:

- **To initialize new operations**
- **When the host detects unresponsiveness of the Bluetooth controller**

In addition, IR feature allows the Wi-Fi driver or the Bluetooth driver to reset and re download their own firmware without depending on each other. For example if the Wi-Fi and Bluetooth combo firmware has been downloaded initially, and the Bluetooth firmware is not responding to host, the host uses the independent reset feature to reset and re download the Bluetooth only firmware without affecting the Wi-Fi operations.

In-band independent reset

This command is given as below:

```
@bt> bt.ind_reset inband
```

Command output response:

```
IR configured successfully for mode 2, ir_state = 3
EtherMind: Bluetooth OFF ...
Sending Inband IR Trigger
download starts(384072)
.....................
download success!
IR exit with state = 0
```

Out-of-band independent reset

This command is given as below:

```
@bt> bt.ind_reset oob
```

Command output:

```
IR configured successfully for mode 1, ir_state = 3
EtherMind: Bluetooth OFF ...
Sending Out of Band IR Trigger
download starts(384072)
.....................
download success!
IR exit with state = 0
```

5.29 peripheral_beacon Sample Application

Application demonstrating the BLE Peripheral role, This application implements types of beacon applications.
Beacon: A simple application demonstrating the BLE Broadcaster role functionality by advertising Company Identifier, Beacon Identifier, UUID, A, B, C, RSSI.
Eddystone : The Eddystone Configuration Service runs as a GATT service on the beacon while it is connectable and allows configuration of the advertised data, the broadcast power levels, and the advertising intervals.

iBeacon: This simple application demonstrates the BLE Broadcaster role functionality by advertising an Apple iBeacon.

5.29.1 peripheral_beacon Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project, building an application, running an application in debug mode, and flashing an application program for a few IDEs. Please refer to section 2.1 for serial console tool setup.

5.29.1.1 peripheral_beacon Run the application

This application contains 3 Different type of Beacons configurations.

Beacon : A simple Application demonstrating the BLE Broadcaster role functionality

To configure the sample application, go to app_bluetooth_config.h and do the following changes to the macros.

Here we are enabling the Beacon app and disabling the other beacon configurations.

```
/* Select witch beacon application to start */
#define BEACON_APP  1
#define IBEACON_APP 0
#define EDDYSTONE   0
```

After changing the macros , recompile the example and flash the application onto the board.

After the example is flashed successfully you will be able to see the following initialization logs on the terminal

```
Starting Beacon Demo
Bluetooth initialized
Beacon started, advertising as 00:E9:3A:B9:E0:24 (public)
```

On the BLE Scanner side our device should be visible as an advertiser.

Eddystone : The Eddystone Configuration Service runs as a GATT service on the beacon while it is connectable and allows configuration of the advertised data, the broadcast power levels, and the advertising intervals. It also forms part of the definition of how Eddystone-EID beacons are configured and registered with a trusted resolver.

To configure the sample application, go to app_bluetooth_config.h and do the following changes to the macros.

Here we are enabling the Eddystone and disabling the other beacon configurations.

```
/* Select witch beacon application to start */
#define BEACON_APP  0
```

```
#define IBEACON_APP 0
#define EDDYSTONE   1
```

After changing the macros , recompile the example and flash the application onto the board.

After the example is flashed successfully you will be able to see the following initialization logs on the terminal

```
Bluetooth initialized
Initial advertising as 00:E9:3A:B9:E0:24 (public)
Configuration mode: waiting connections...
```

On the BLE Scanner side our device should be visible as an advertiser.

iBeacon : This is a simple application demonstrates the BLE Broadcaster role functionality by advertising an Apple iBeacon. The calibrated RSSI @ 1 meter distance can be set using an IBEACON_RSSI build variable (e.g., IBEACON_RSSI=0xc8 for -56 dBm RSSI), or by manually editing the default value in the ibeacon.c file.

To configure the sample application, go to app_bluetooth_config.h and do the following changes to the macros.

Here we are enabling the Eddystone and disabling the other beacon configurations.

```
/* Select witch beacon application to start */
#define BEACON_APP  0
#define IBEACON_APP 1
#define EDDYSTONE   0
```

After changing the macros , recompile the example and flash the application onto the board.

After the example is flashed successfully you will be able to see the following initialization logs on the terminal.

```
Starting iBeacon Demo
Bluetooth initialized
iBeacon started
```

On the BLE Scanner side our device should be visible as an advertiser

5.30 audio_profile Sample Application

There are five parts working in the demo: AWS cloud, Android app, audio demo (running on i.MX RT1060 EVK board), U-disk and Bluetooth headset.

- **With an app running on the smart phone (Android phone), the end users can connect to AWS cloud and control the audio demo running on the i.MX RT1060 EVK board through AWS cloud. Some operations like play, play next, pause, etc. can be used to control the media play functionalities.**
- **Audio demo running on the RT1060 EVK board connects to the AWS through Wi-Fi, also a connection can be established between the i.MX RT1060 EVK board and a Bluetooth headset.**
- **To get the media resource (mp3 files) from the U-disk, an HS USB host is enabled, and a U-disk with mp3 files should be connected to i.MX RT1060 EVK board via the USB port.**
- **After that, the audio demo will search the root directory of U-disk for the audio files and upload the audio file list to AWS, then the list would be shown in the app running on the smart phone.**
- **Finally, the music can be played out via the Bluetooth headset once end user controls the app to play the mp3 file.**

Prerequisites and Important details about this Demo :

- **This demo can NOT function with the default setting provided in SDK package**
- **AWS Account is mandatory to run this demo. User must create their own AWS account and configure the IoT Thing.**
- **WiFi SSID , WiFi Password etc. must be updated.**
- **The music files names in U-disk need to be English.**
- **The volume of audio adjustment is not supported.**

5.30.1 User Configurations

Some of the AWS Client Credentials related macros that user need to configure based on requirement are listed in below table along with source file name.

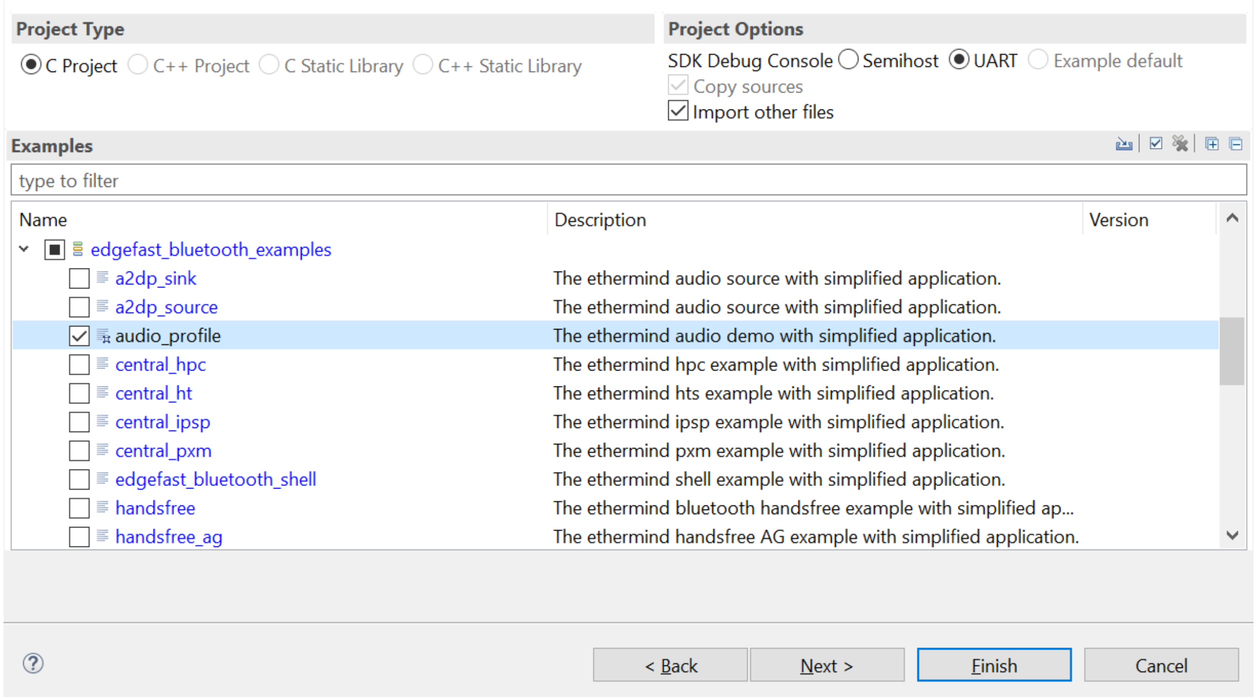
The *aws_clientcredential.h* file is available in the SDK source; path is given in section 1.3 “References”.

Table 24: audio_profile Application Configurations

Feature	Macro definition	Value set for Example	File name	Details
AWS Client Credentials	clientcredentialMQTT_BROKER_ENDPOINT	“a2qkq65ssjggf7-ats.iot.us-east-1.amazonaws.com”	aws_clientcredential.h	These credentials are required to connect the correct end point of AWS IoT Thing.
	clientcredentialIOT_THING_NAME	“MusicPlayer”		
	clientcredentialWIFI_SSID	“NXP_Demo”		
	clientcredentialWIFI_PASSWORD	“123456789”		
	clientcredentialMQTT_BROKER_PORT	“8883”		

5.30.2 audio_profile Application Execution

Please refer to the previous section 3.1.1 to run the demo using MCUXpresso IDE. Refer below figures for importing Bluetooth example and selection of Bluetooth module.



Project Type

● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options

SDK Debug Console ○ Semihost ● UART ○ Example default

☑ Copy sources

☑ Import other files

Examples

type to filter

Name	Description	Version
■ edgefast_bluetooth_examples		
□ a2dp_sink	The ethermind audio source with simplified application.	
□ a2dp_source	The ethermind audio source with simplified application.	
☑ audio_profile	The ethermind audio demo with simplified application.	
□ central_hpc	The ethermind hpc example with simplified application.	
□ central_ht	The ethermind hts example with simplified application.	
□ central_ipsp	The ethermind ipsp example with simplified application.	
□ central_pxm	The ethermind pxm example with simplified application.	
□ edgefast_bluetooth_shell	The ethermind shell example with simplified application.	
□ handsfree	The ethermind bluetooth handsfree example with simplified ap...	
□ handsfree_ag	The ethermind handsfree AG example with simplified application.	

Figure 42 : Selection of audio_profile application in MCUXpresso IDE

Please refer to the previous sections 3.1.2-3.1.4 for instructions on importing a project, building an application, running an application in debug mode and flashing an application program for a few IDEs. Please refer to section 2.1 for information about the serial console setup.

5.30.2.1 Create and configure AWS Account

Follow the link to create a new AWS account:
https://console.aws.amazon.com/console/home

Follow the instructions to create a new AWS account:
https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/

5.30.2.2 Create an AWS IoT Policy

This section describes the steps to create a policy for AWS IoT.

Browse to the AWS IoT console
https://console.aws.amazon.com/iotv2/

Click “Policies” inside the “Secure” tab:

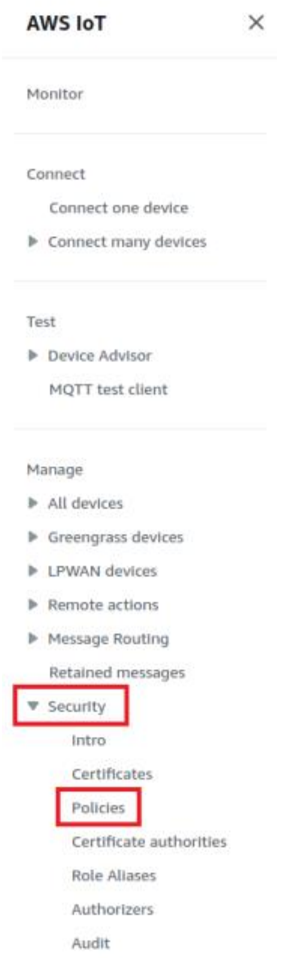


Figure 42: Creating a new policy

Create a new policy: Enter a name to identify a policy. For example, the policy name is "**MusicPlayerPolicy**".

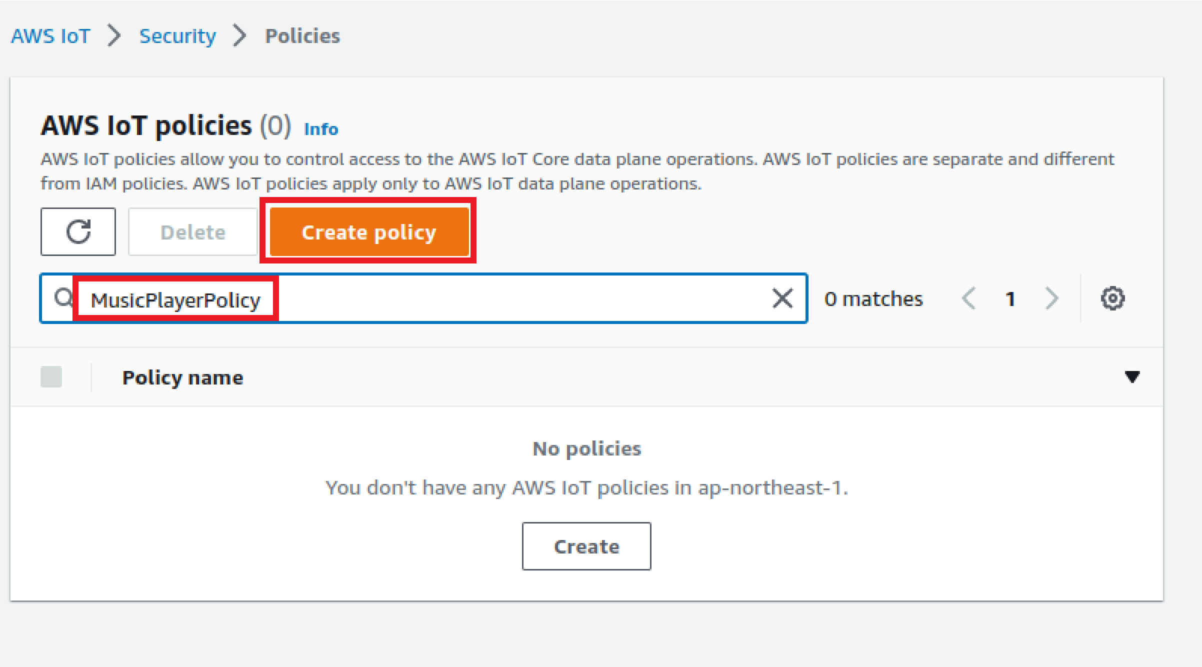

AWS IoT > Security > Policies
AWS IoT policies (0) Info
AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.
Delete
Create policy
MusicPlayerPolicy
0 matches
1
Policy name
No policies
You don't have any AWS IoT policies in ap-northeast-1.
Create

Figure 43: Policy Name

In the "Policy statements" section, click “JSON”.

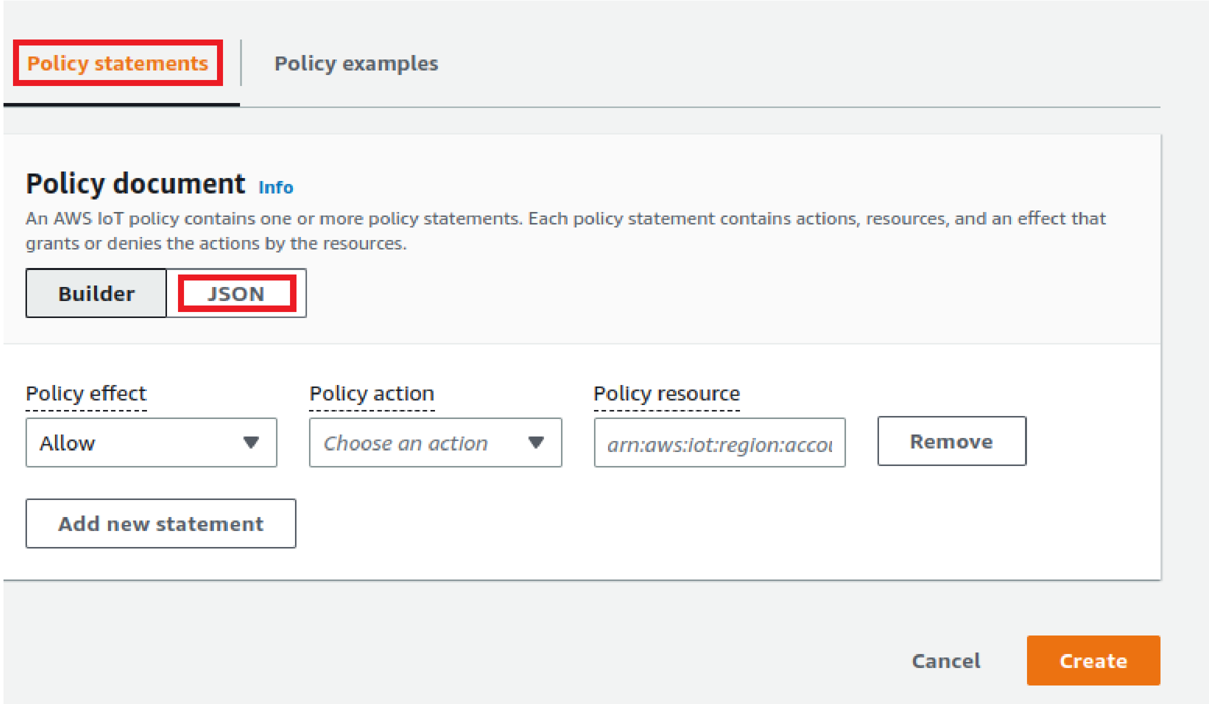
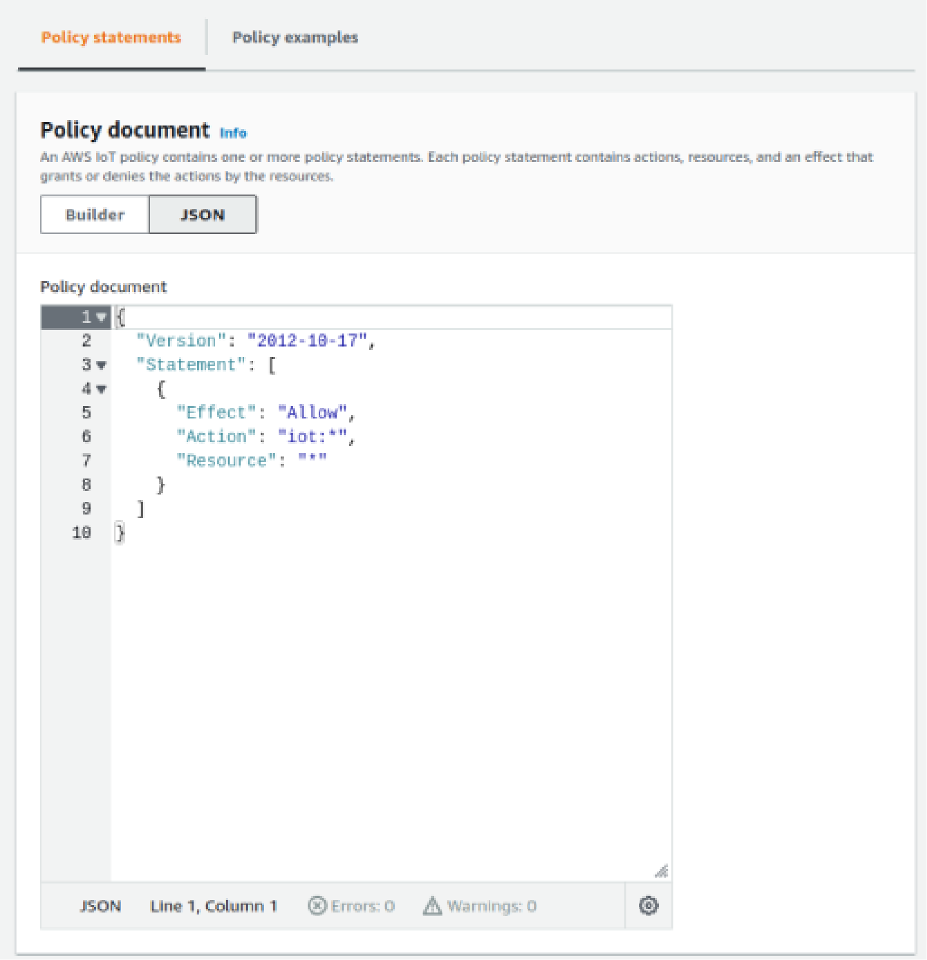


Figure 44: Policy statements

Add the JSON into the Policy editor window and create a policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```



Policy document Info

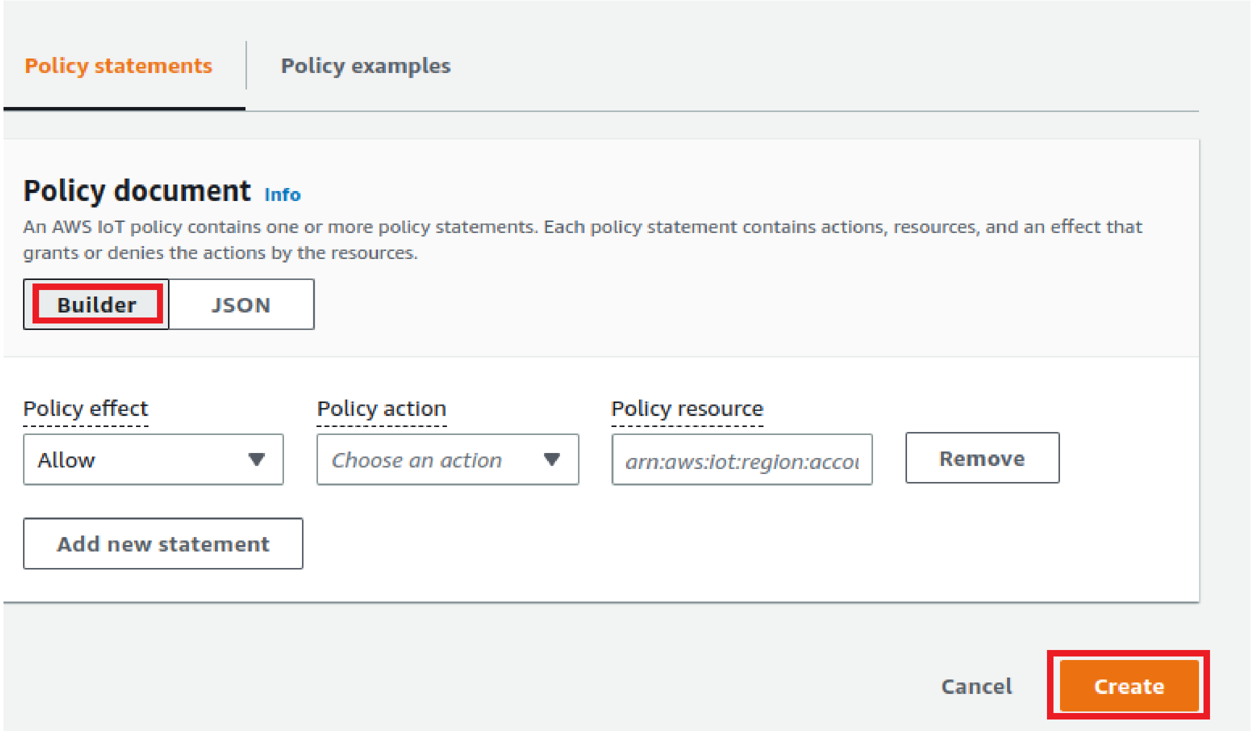
An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

Figure 45: Adding the required JSON into the policy editor window

Go back to the "Builder" and click on "Create"

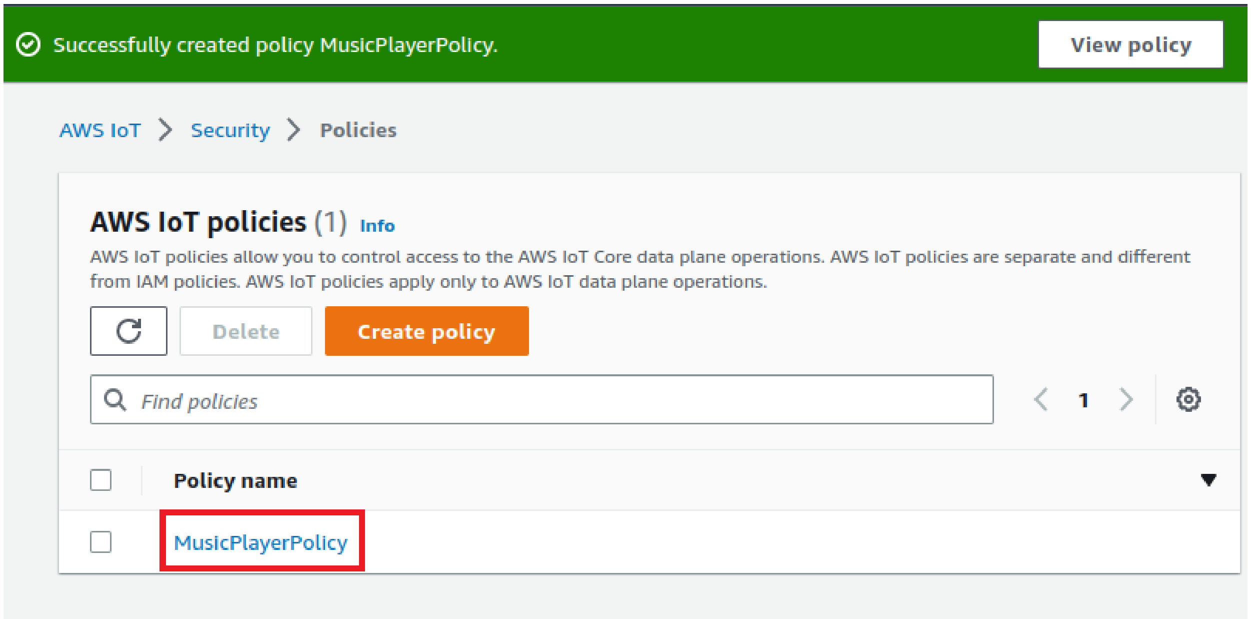


Policy document Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Figure 43: Create a policy with required JSON

Upon successful creation of Policy following screen will appear:



Successfully created policy MusicPlayerPolicy.

AWS IoT policies (1) Info

AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

Figure 44: Showing the success of policy creation

5.30.2.3 Create IoT thing, private key, and certificate for device

Open the "AWS IoT console"

```
https://console.aws.amazon.com/iot/
```

From the navigation pane, click "Things" inside "All devices" tab.

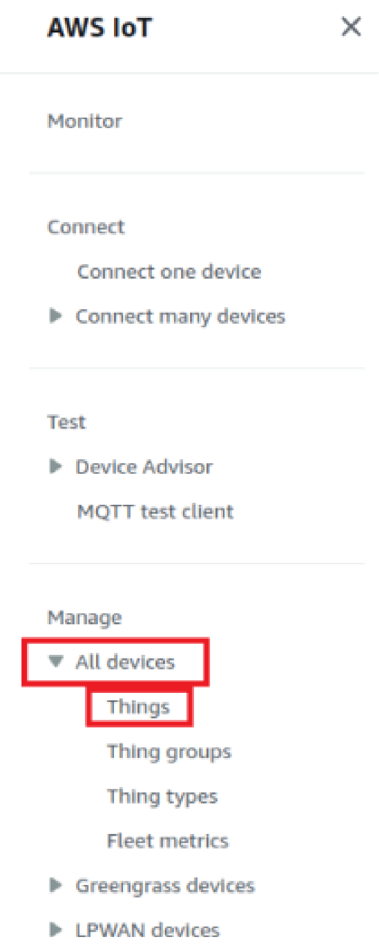
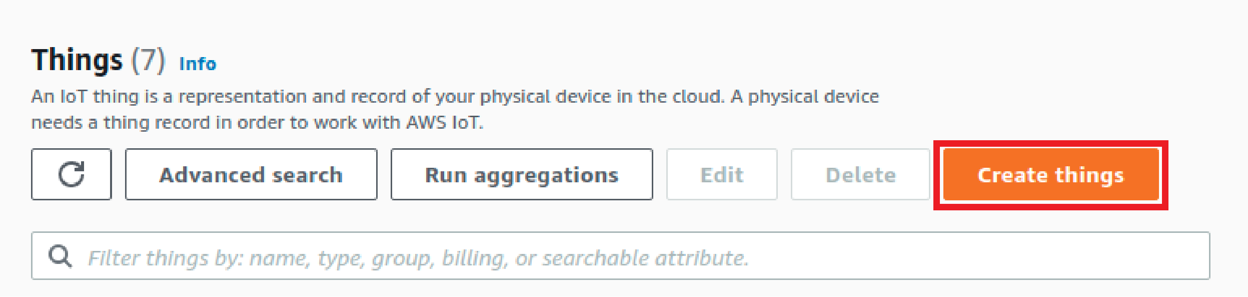


Figure 45: Selection of Things from AWS IoT tab

Click on "**Create**".



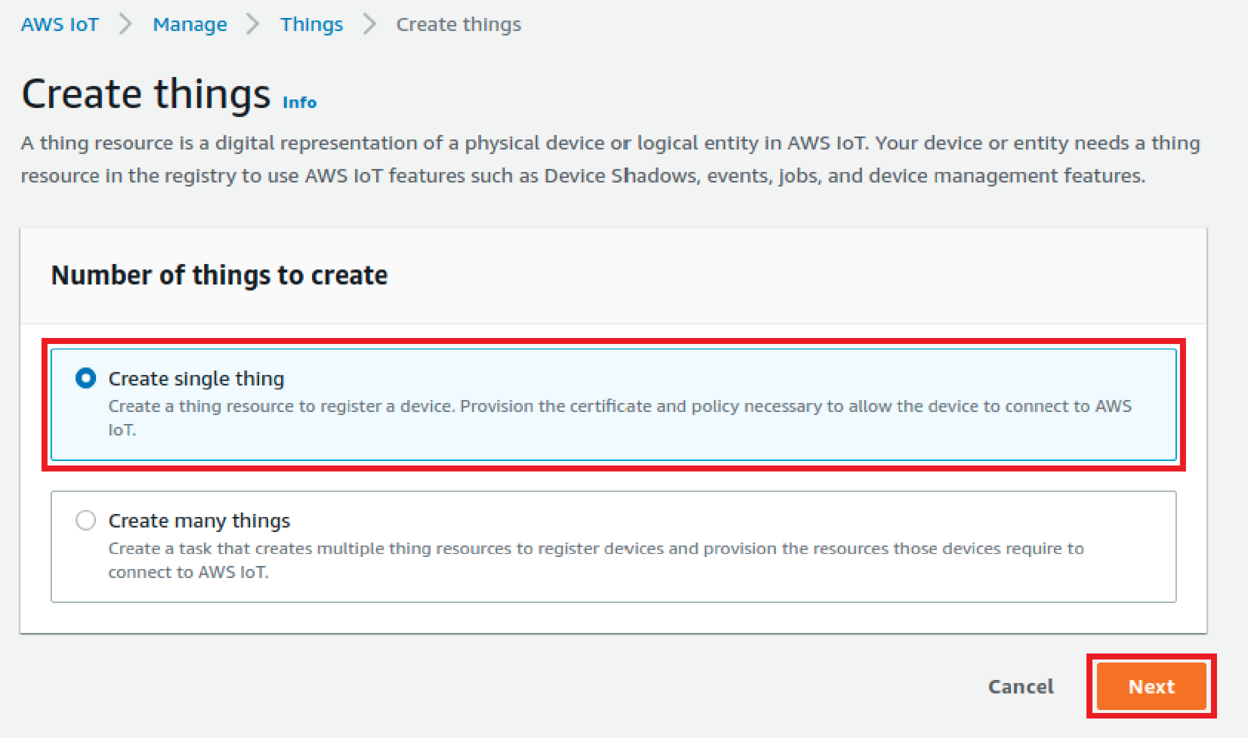
Things (7) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Advanced search
Run aggregations
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.

Figure 46: Creating a new Thing

Click "**Create a single thing**"



AWS IoT > Manage > Things > Create things

Create things Info

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

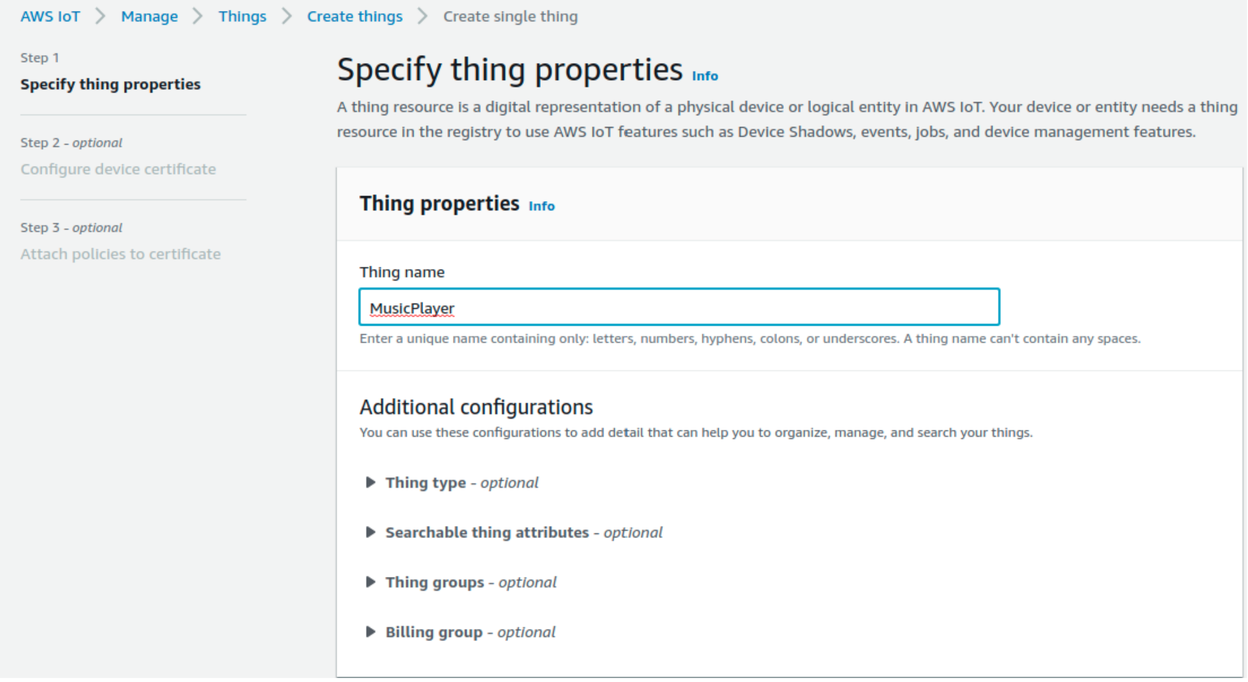
Number of things to create

● Create single thing
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

○ Create many things
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Figure 47: Creating a new Thing

Enter a name for your device, and then choose "**Next**". For example, the thing name is "**MusicPlayer**".



AWS IoT > Manage > Things > Create things > Create single thing

Step 1
Specify thing properties

Step 2 - *optional*
Configure device certificate

Step 3 - *optional*
Attach policies to certificate

Specify thing properties Info

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties Info

Thing name: MusicPlayer

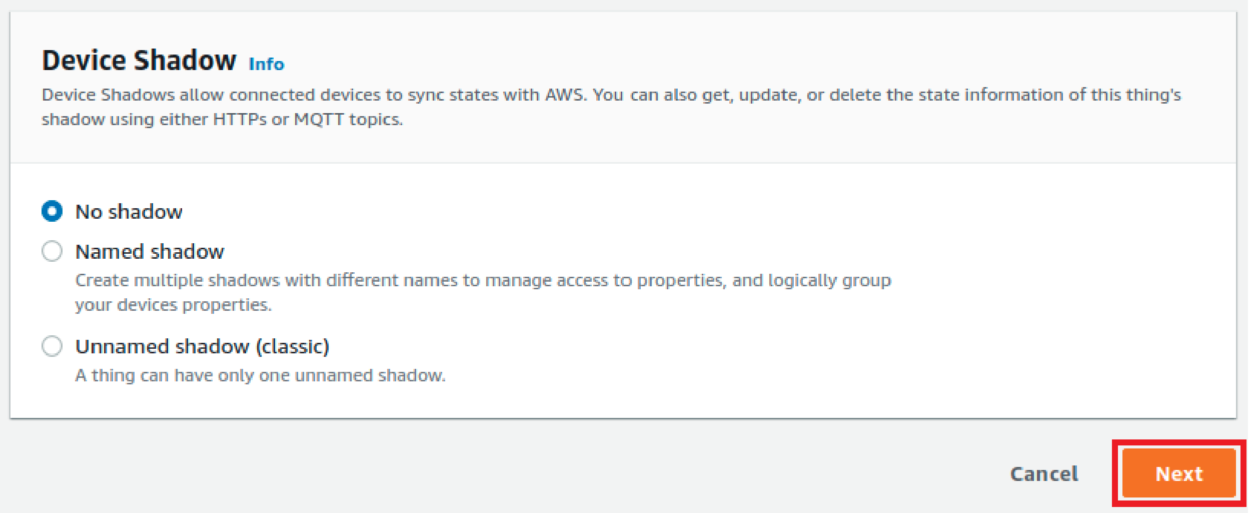
Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ **Thing type** - *optional*
- ▶ **Searchable thing attributes** - *optional*
- ▶ **Thing groups** - *optional*
- ▶ **Billing group** - *optional*

Figure 48: Giving name to a new thing



Device Shadow Info

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

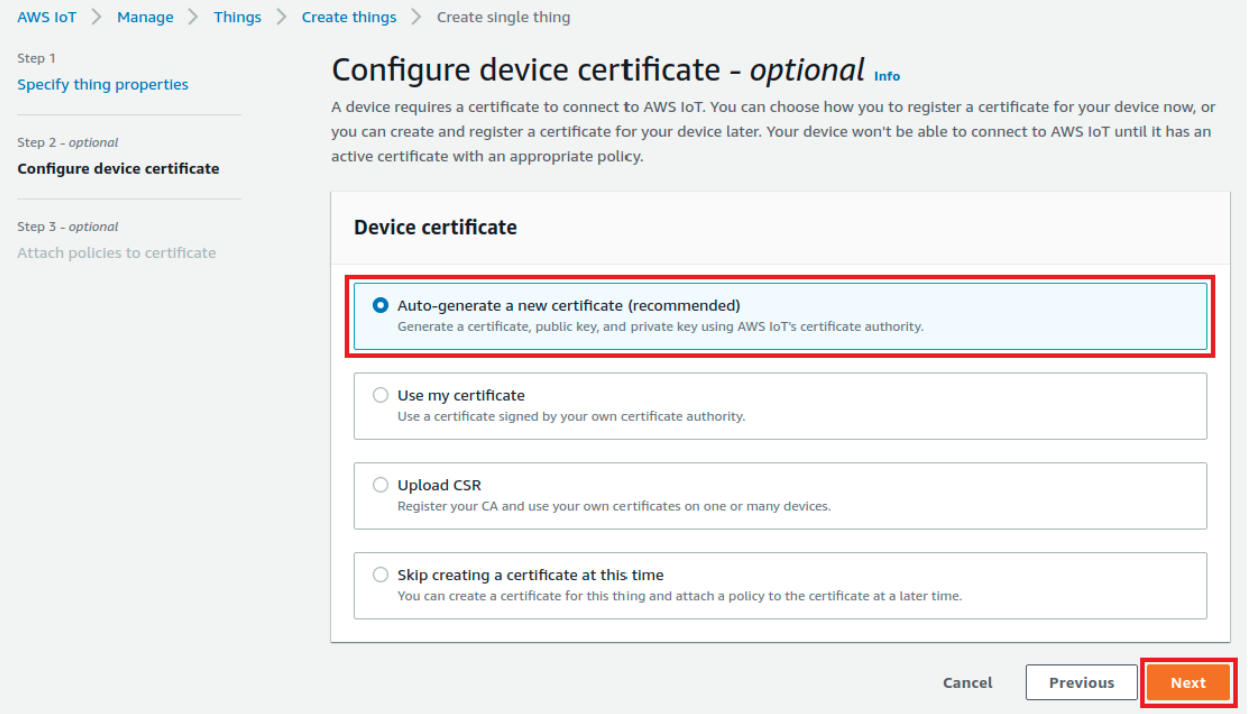
● No shadow

○ Named shadow
Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

○ Unnamed shadow (classic)
A thing can have only one unnamed shadow.

Figure 49: Click next to proceed for creating a new thing

Click "**Create certificate**"



AWS IoT > Manage > Things > Create things > Create single thing
Step 1
Specify thing properties
Step 2 - optional
Configure device certificate
Step 3 - optional
Attach policies to certificate

Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how you to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

● Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

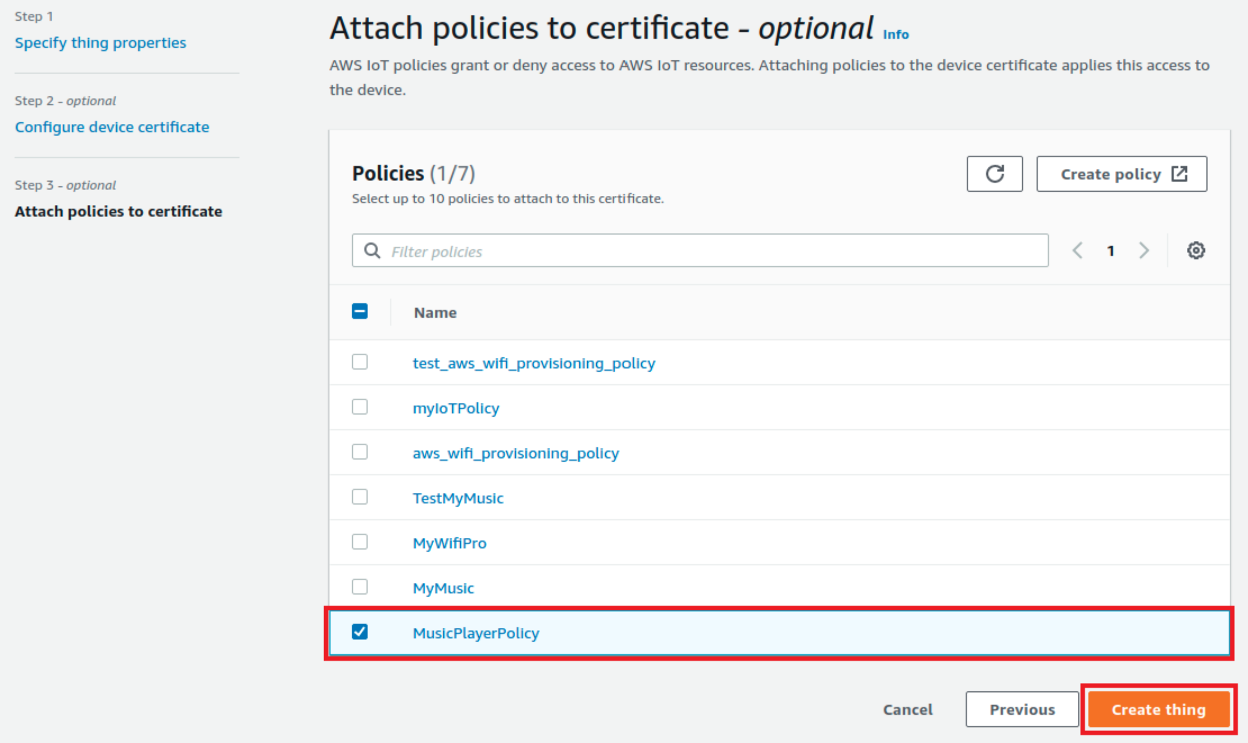
○ Use my certificate
Use a certificate signed by your own certificate authority.

○ Upload CSR
Register your CA and use your own certificates on one or many devices.

○ Skip creating a certificate at this time
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Figure 50: Selecting Device Certificate configuration for a new Thing

Select a policy to attach to your certificate that grants your device access to AWS IoT operations and click "**Create Thing**".



Step 1

Specify thing properties

Step 2 - *optional*

Configure device certificate

Step 3 - *optional*

Attach policies to certificate

Attach policies to certificate - *optional* Info

AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

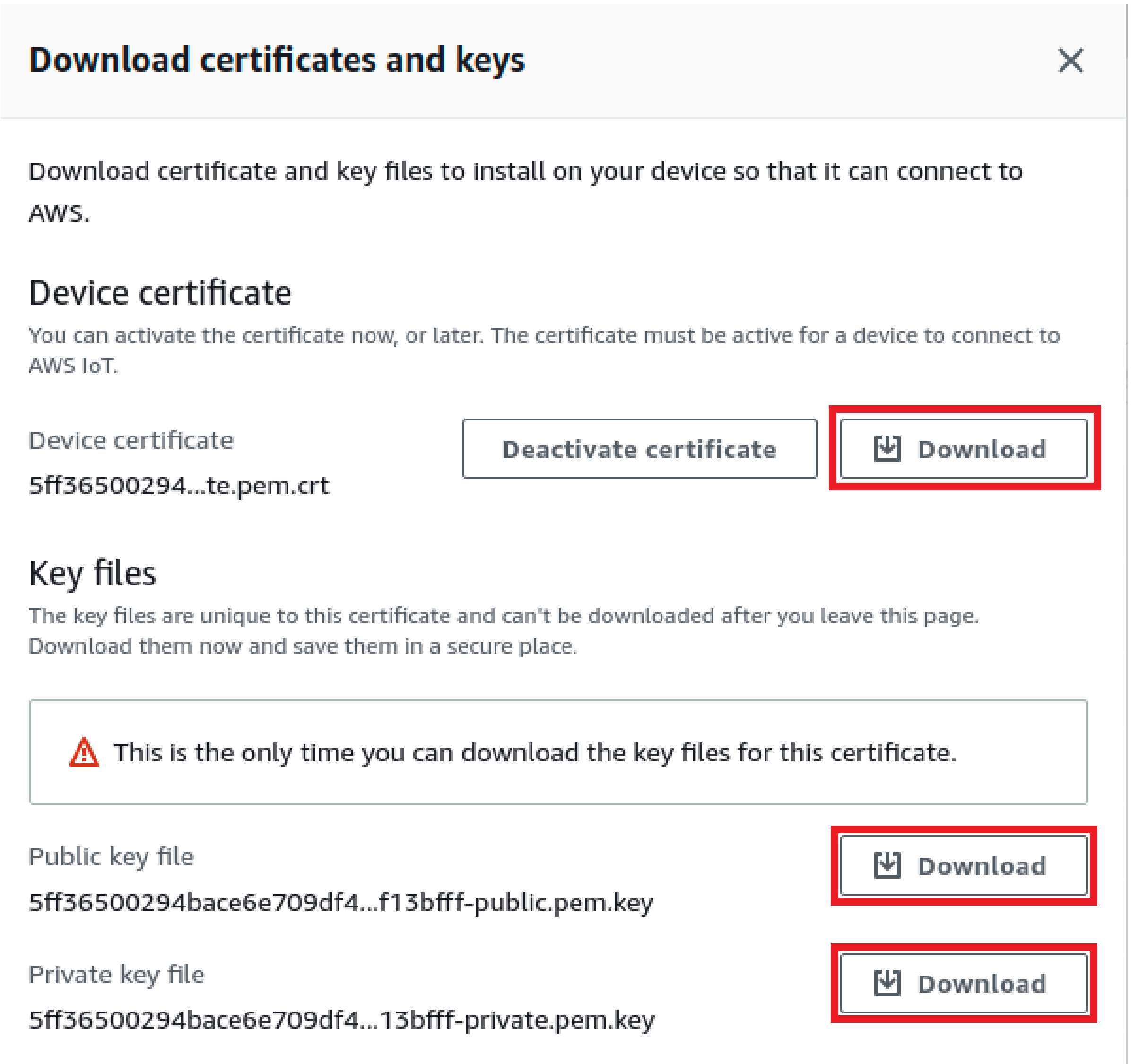
Policies (1/7)

Select up to 10 policies to attach to this certificate.

☐	Name
☐	test_aws_wifi_provisioning_policy
☐	myIoTPolicy
☐	aws_wifi_provisioning_policy
☐	TestMyMusic
☐	MyWifiPro
☐	MyMusic
☑	MusicPlayerPolicy

Figure 51: Attach a policy and create a Thing

Download the thing's certificate, public key, and private key.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
5ff36500294...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

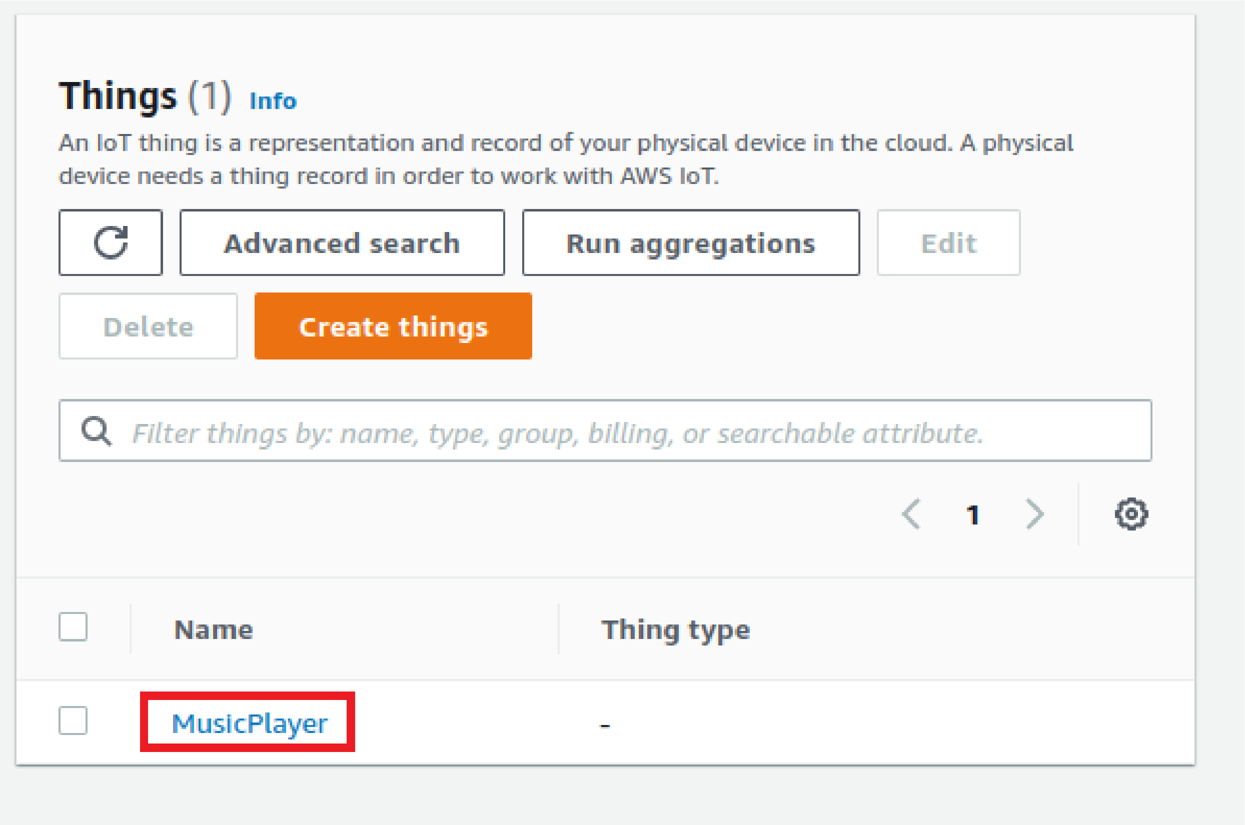
⚠ This is the only time you can download the key files for this certificate.

Public key file
5ff36500294bace6e709df4...f13bfff-public.pem.key

Private key file
5ff36500294bace6e709df4...13bfff-private.pem.key

Figure 52: Downloading the certificate, public and private keys

Click the thing that you just created from the list.



Things (1) Info

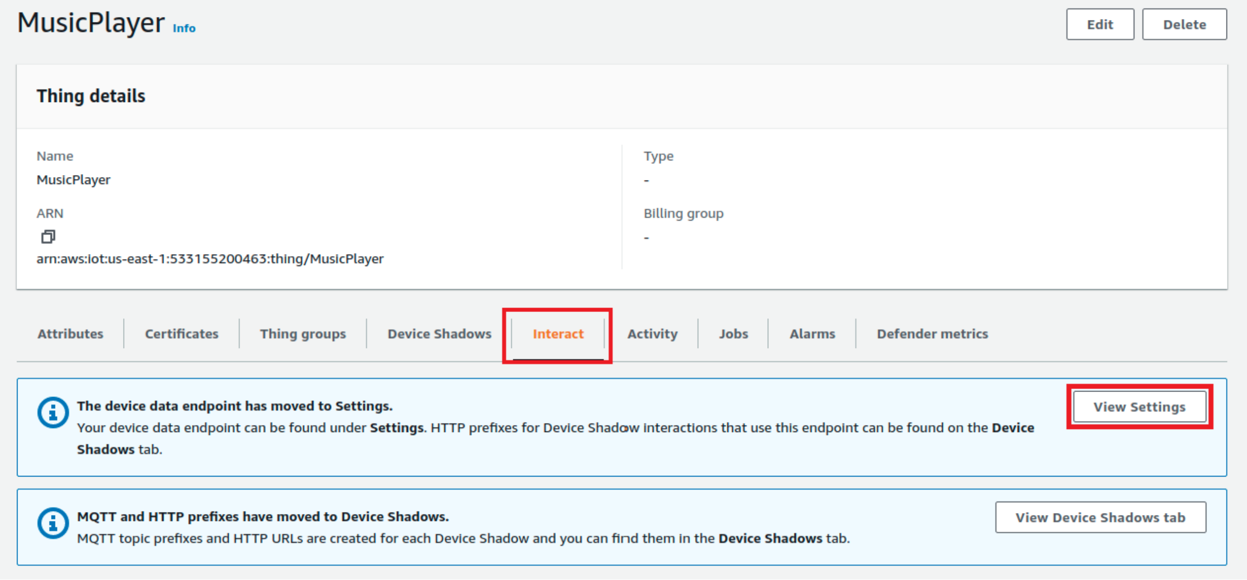
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Advanced search
Run aggregations
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.
1

□	Name	Thing type
□	MusicPlayer	-

Figure 53: Selecting the policy and Register Thing

Click "**Interact**" from your thing's page and open “**View Settings**”.



MusicPlayer Info

Thing details

Name: MusicPlayer

ARN: arn:aws:iot:us-east-1:533155200463:thing/MusicPlayer

Type: -

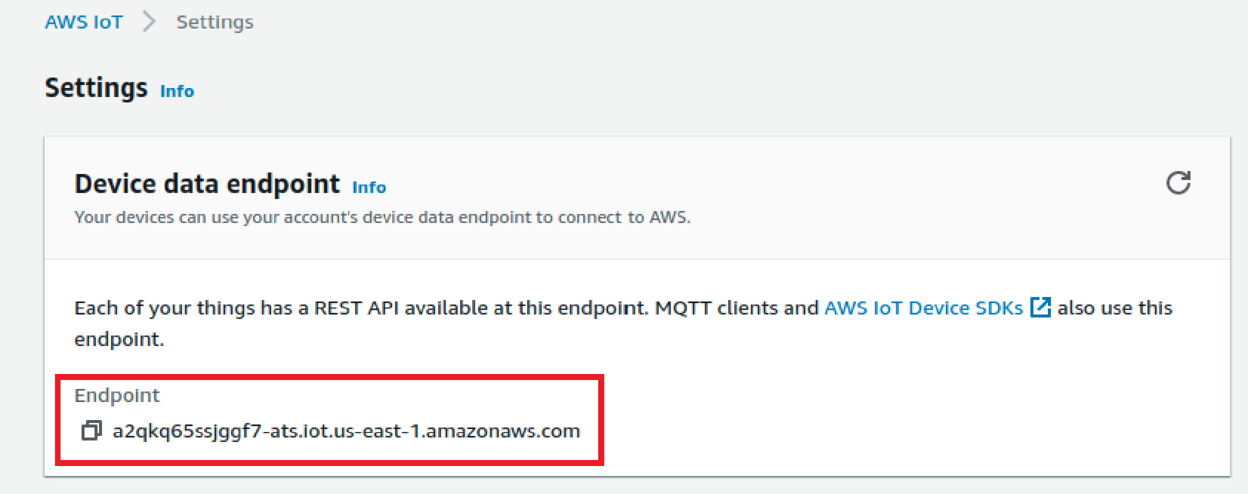
Billing group: -

The device data endpoint has moved to Settings.
Your device data endpoint can be found under **Settings**. HTTP prefixes for Device Shadow interactions that use this endpoint can be found on the **Device Shadows** tab.

MQTT and HTTP prefixes have moved to Device Shadows.
MQTT topic prefixes and HTTP URLs are created for each Device Shadow and you can find them in the **Device Shadows** tab.

Figure 54: Selecting Interact and opening View Settings to get Endpoint

Make a NOTE of the AWS IoT REST API endpoint to use it for next sections.



Settings Info

Device data endpoint Info

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint
a2qkq65ssjggf7-ats.iot.us-east-1.amazonaws.com

Figure 55: Copy the AWS Iot REST API endpoint

5.30.2.4 Configure the AWS IoT Certificate and Private Keys

FreeRTOS needs the AWS IoT certificate and private keys associated with your registered thing and its permissions policies to successfully communicate with AWS IoT on behalf of your device.

FreeRTOS is a C language project, and the certificate and private key must be specially formatted to be added to the project.

Get the PEM-to-C-stringp.py file from SDK (<MCUXpresso SDK>\middleware\aws_iot\amazon-freertos\tools\certificate_configuration)

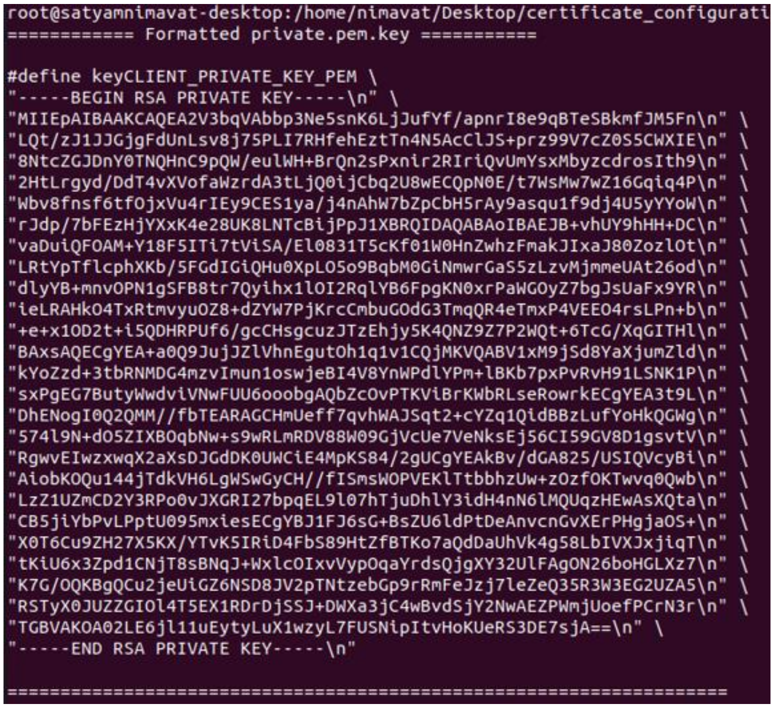
Usage:

PEM-to-C-string.py [-h] [--private-key-file PRIVATE_KEY_FILE] [--cert-file CERT_FILE] [--root-ca-file ROOT_CA_FILE] [--pem-file PEM_FILE]

Execute this script with private-key.pem and certificate.pem file. This Python script will arrange the key in a format which is required for the project and print it on console.

For example:

```
PEM-to-C-string.py -h --private-key-file PRIVATE_KEY_FILE.key.pem --cert-file CERT_FILE.pem
```

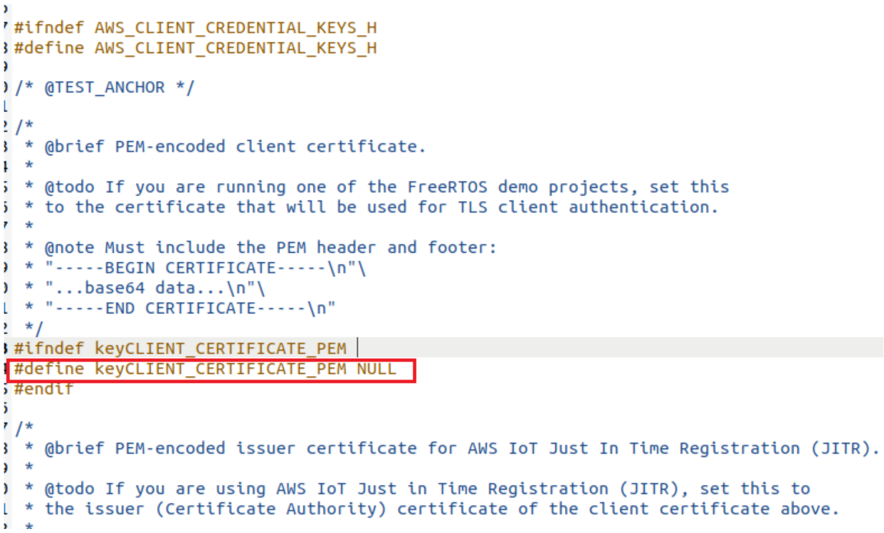


```
root@satyamnimavat-desktop:/home/nimavat/Desktop/certificate_configurati
============ Formatted private.pem.key ===========

#define keyCLIENT_PRIVATE_KEY_PEM \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEpAIBAAKCAQEA2V3bqVAbbp3Ne5snK6LjJufYf/apnrI8e9qBTeSBkmfJM5Fn\n" \
"LQt/zJ1JJGjgFdUnLsv8j75PLI7RHfehEztTn4N5AcClJS+prz99V7cZ0S5CWXIE\n" \
"8NtcZGJDnY0TNQHnC9pQW/eulWH+BrQn2sPxnir2RIriQvUmYsxMbyzcdrosIth9\n" \
"2HtLrgyd/DdT4vXVofaWzrdA3tLjQ0ijCbq2U8wECQpN0E/t7WsMw7wZ16Gqiq4P\n" \
"Wbv8fnsf6tfOjxVu4rIEy9CES1ya/j4nAhW7bZpCbH5rAy9asqu1f9dj4U5yYYoW\n" \
"rJdp/7bFEzHjYXxK4e28UK8LNTcBijPpJ1XBRQIDAQABAoIBAEJB+vhUY9hHH+DC\n" \
"vaDuiQFOAM+Y18F5ITi7tViSA/El0831T5cKf01W0HnZwhzFmakJIxaJ80Zozl0t\n" \
"LRtYpTflcphXKb/5FGdIGiQHu0XpLO5o9BqbM0GiNmwrGaS5zLzvMjmmeUAt26od\n" \
"dlyYB+mnvOPN1gSFB8tr7Qyihx1lOI2RqlYB6FpgKN0xrPaWGOyZ7bgJsUaFx9YR\n" \
"ieLRAHkO4TxRtmvyuOZ8+dZYW7PjKrcCmbuGOdG3TmqQR4eTmxP4VEEO4rsLPn+b\n" \
"+e+x1OD2t+i5QDHRPUf6/gcCHsgcuzJTzEhjy5K4QNZ9Z7P2WQt+6TcG/XqGITHl\n" \
"BAxsAQECgYEA+a0Q9JujJZlVhnEgutOh1q1v1CQjMKVQABV1xM9jSd8YaXjumZld\n" \
"kYoZzd+3tbRNMDG4mzvImun1oswjeBI4V8YnWPdlYPm+lBKb7pxPvRvH91LSNK1P\n" \
"sxPgEG7ButyWwdviVNwFUU6ooobgAQbZcOvPTKViBrKWbRLseRowrkECgYEA3t9L\n" \
"DhENogI0Q2QMM//fbTEARAGCHmUeff7qvhWAJSqt2+cYZq1QidBBzLufYoHkQGWg\n" \
"574l9N+dO5ZIXBOqbNw+s9wRLmRDV88W09GjVcUe7VeNksEj56CI59GV8D1gsvtV\n" \
"RgwvEIwzxwqX2aXsDJGdDK0UWCiE4MpKS84/2gUCgYEAkBv/dGA825/USIQVcyBi\n" \
"AiobKOQu144jTdkVH6LgWSwGyCH//fISmsWOPVEKlTtbbhzUw+zOzfOKTwvq0Qwb\n" \
"LzZ1UZmCD2Y3RPo0vJXGRI27bpqEL9l07hTjuDhlY3idH4nN6lMQUqzHEwAsXQta\n" \
"CB5jiYbPvLPptU095mxiesECgYBJ1FJ6sG+BsZU6ldPtDeAnvcnGvXErPHgjaOS+\n" \
"X0T6Cu9ZH27X5KX/YTvK5IRiD4FbS89HtZfBTKo7aQdDaUhVk4g58LbIVXJxjiqT\n" \
"tKiU6x3Zpd1CNjT8sBNqJ+WxlcOIxvVypOqaYrdsQjgXY32UlFAgON26boHGLXz7\n" \
"K7G/OQKBgQCu2jeUiGZ6NSD8JV2pTNtzebGp9rRmFeJzj7leZeQ35R3W3EG2UZA5\n" \
"RSTyX0JUZZGIOl4T5EX1RDrDjSSJ+DWXa3jC4wBvdSjY2NwAEZPWmjUoefPCrN3r\n" \
"TGBVAKOA02LE6jl11uEytyLuX1wzyL7FUSNipItvHoKUeRS3DE7sjA==\n" \
"-----END RSA PRIVATE KEY-----\n"

===================================================================
```

Figure 56: Certificates printed on console logs

Open aws_clientcredential_keys.h from the below mentioned SDK path.
SDK/boards/<board>/edgefast_bluetooth_examples/audio_profile.

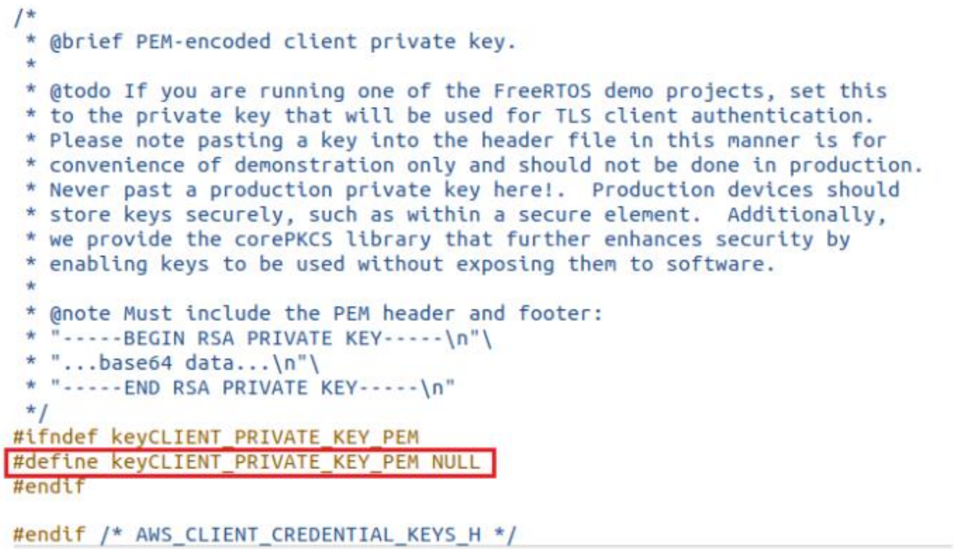
Find keyCLIENT_CERTIFICATE_PEM and paste formatted certificate as a value.



```
#ifndef AWS_CLIENT_CREDENTIAL_KEYS_H
#define AWS_CLIENT_CREDENTIAL_KEYS_H

/* @TEST_ANCHOR */

/*
 * @brief PEM-encoded client certificate.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the certificate that will be used for TLS client authentication.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"\
 * "...base64 data...\n"\
 * "-----END CERTIFICATE-----\n"
 */
#ifndef keyCLIENT_CERTIFICATE_PEM
#define keyCLIENT_CERTIFICATE_PEM NULL
#endif

/*
 * @brief PEM-encoded issuer certificate for AWS IoT Just In Time Registration (JITR).
 *
 * @todo If you are using AWS IoT Just in Time Registration (JITR), set this to
 * the issuer (Certificate Authority) certificate of the client certificate above.
 *
```

Figure 57: Adding client certificate

Then find keyCLIENT_PRIVATE_KEY_PEM and paste formatted private-key as value.



```
/*
 * @brief PEM-encoded client private key.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the private key that will be used for TLS client authentication.
 * Please note pasting a key into the header file in this manner is for
 * convenience of demonstration only and should not be done in production.
 * Never past a production private key here!.  Production devices should
 * store keys securely, such as within a secure element.  Additionally,
 * we provide the corePKCS library that further enhances security by
 * enabling keys to be used without exposing them to software.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN RSA PRIVATE KEY-----\n"\
 * "...base64 data...\n"\
 * "-----END RSA PRIVATE KEY-----\n"
 */
#ifndef keyCLIENT_PRIVATE_KEY_PEM
#define keyCLIENT_PRIVATE_KEY_PEM NULL
#endif

#endif /* AWS_CLIENT_CREDENTIAL_KEYS_H */
```

Figure 58: Adding private key

NOTE: The certificate and private key are hard-coded for demonstration purposes only. Production-level applications should store these files in a secure location.

5.30.2.5 Configure the AWS IoT endpoint

User need to update FreeRTOS with your AWS IoT endpoint so the application running on the board can send requests to the correct endpoint.

Open “*aws_clientcredential.h*” file.

Set the “clientcredentialMQTT_BROKER_ENDPOINT” as per the Rest API Endpoint.

```
#define clientcredentialMQTT_BROKER_ENDPOINT          "a2qkq65ssjggf7-
ats.iot.us-east-1.amazonaws.com"
```

Set the “clientcredentialIOT_THING_NAME” as per the name of IoT Thing

```
#define clientcredentialIOT_THING_NAME                "MusicPlayer"
```

Set the “clientcredentialWIFI_SSID” as the connected Wi-Fi SSID

```
#define clientcredentialWIFI_SSID                     "NXP_Demo"
```

Set the “clientcredentialWIFI_PASSWORD” as the connected Wi-Fi Password.

```
#define clientcredentialWIFI_PASSWORD                 "123456789"
```

Set the “clientcredentialMQTT_BROKER_PORT” as 443

```
#define clientcredentialMQTT_BROKER_PORT              8883
```

Rebuild the application and flash it on the target board.

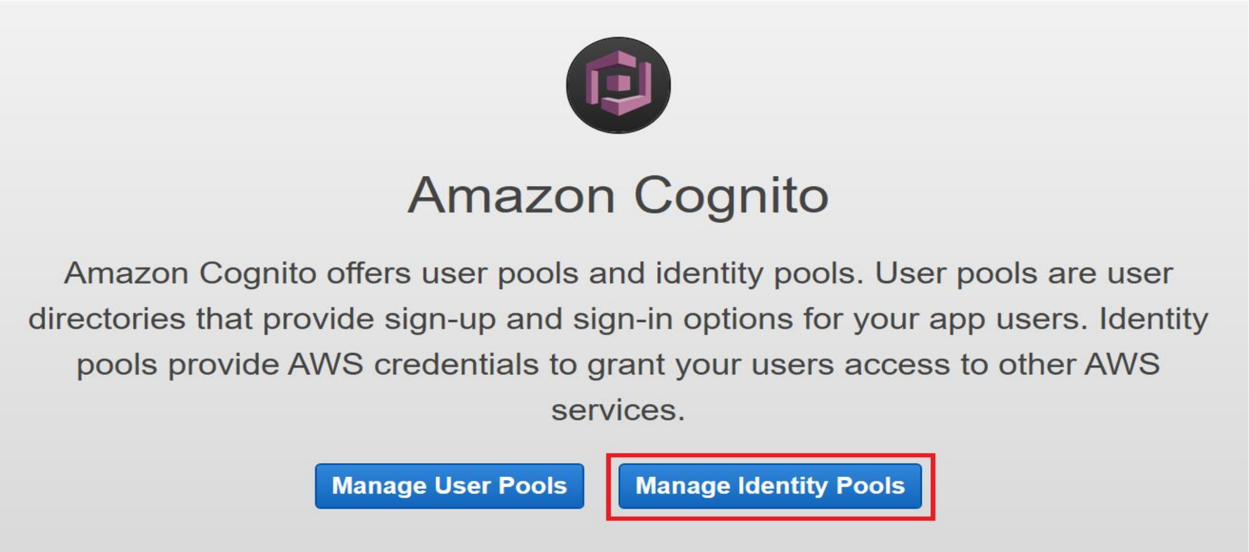
Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

Prepare the Android application

Open the Amazon Cognito console,

```
https://console.aws.amazon.com/cognito/home
```

Choose "Manage Identity Pools"



Amazon Cognito

Amazon Cognito offers user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools provide AWS credentials to grant your users access to other AWS services.

Figure 59: Manage Identity Pools

Click “**Create new identity pool**”.

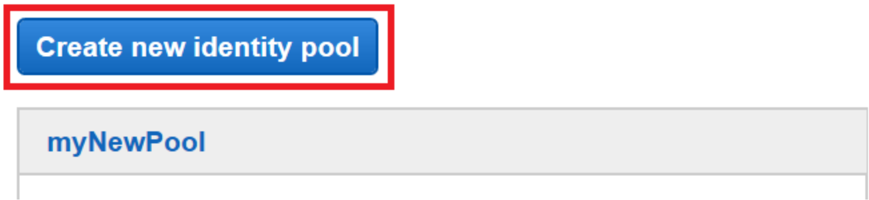
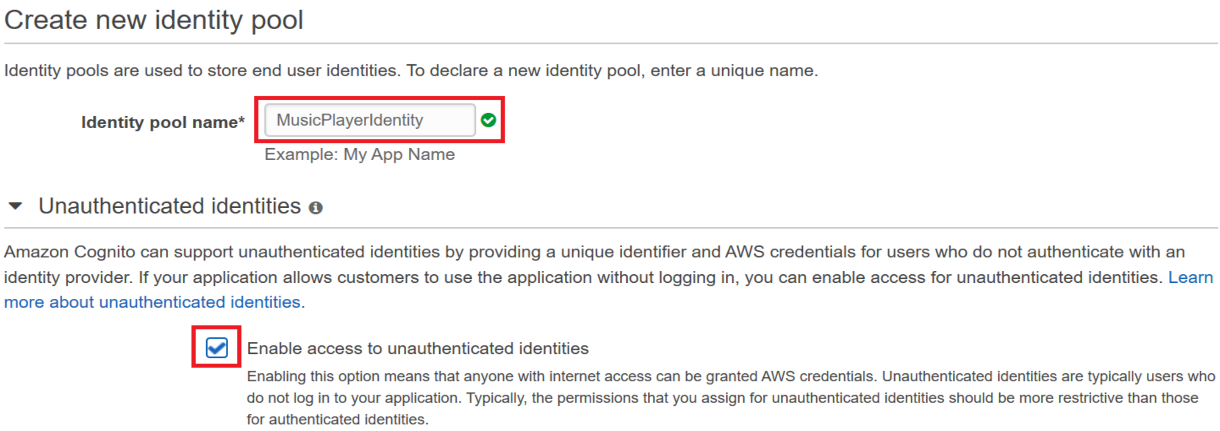


Figure 60: Create new identity pool

Enter a name for your identity pool. Such as the pool name is "MusicPlayerIdentity", enable unauthenticated access.



Create new identity pool

Identity pools are used to store end user identities. To declare a new identity pool, enter a unique name.

Identity pool name* MusicPlayerIdentity ✓

Example: My App Name

▾ Unauthenticated identities

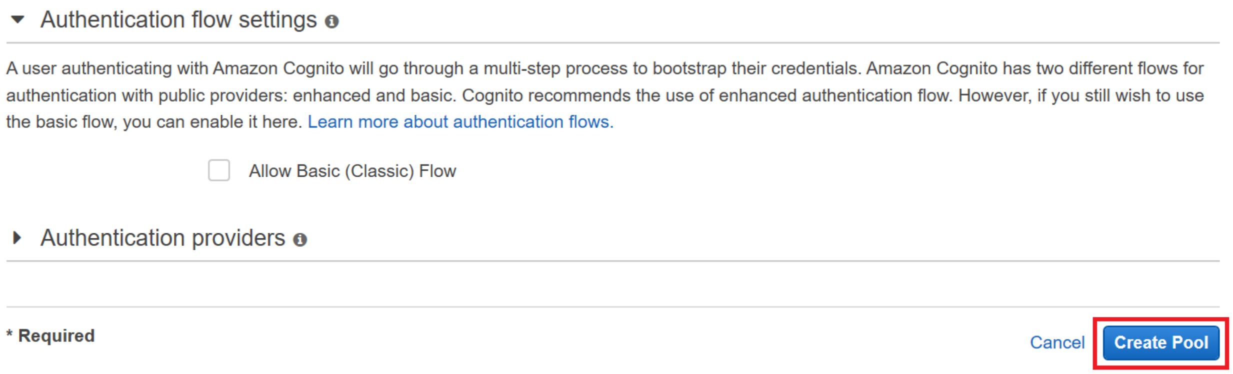
Amazon Cognito can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider. If your application allows customers to use the application without logging in, you can enable access for unauthenticated identities. Learn more about unauthenticated identities.

☑ Enable access to unauthenticated identities

Enabling this option means that anyone with internet access can be granted AWS credentials. Unauthenticated identities are typically users who do not log in to your application. Typically, the permissions that you assign for unauthenticated identities should be more restrictive than those for authenticated identities.

Figure 61: Identity pool name

Click "**Create Pool**".



Authentication flow settings

A user authenticating with Amazon Cognito will go through a multi-step process to bootstrap their credentials. Amazon Cognito has two different flows for authentication with public providers: enhanced and basic. Cognito recommends the use of enhanced authentication flow. However, if you still wish to use the basic flow, you can enable it here. Learn more about authentication flows.

☐ Allow Basic (Classic) Flow

Authentication providers

* Required

Figure 62: Create pool

Click “**Allow**” to create a pool

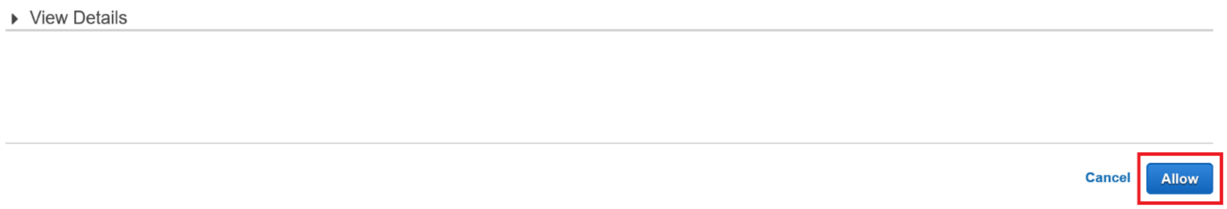
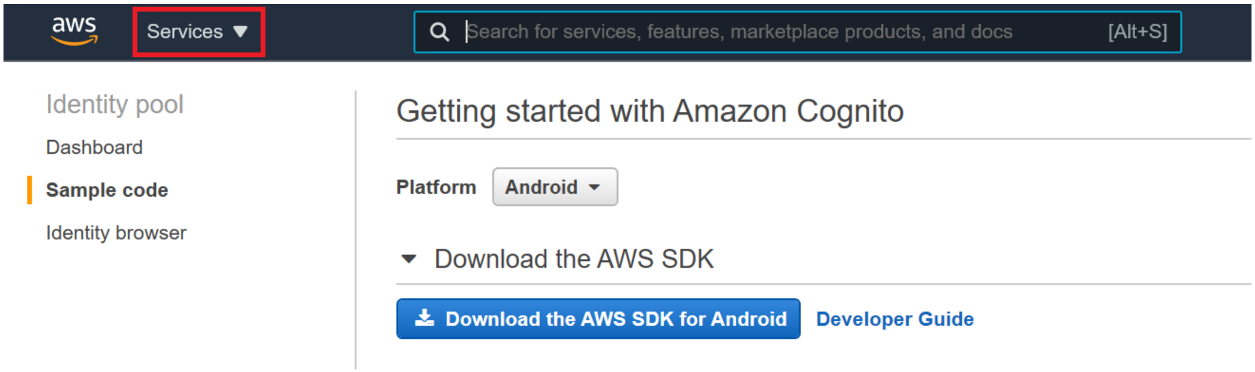


Figure 63: Allow to create a pool

Click “**Services**”



aws
Services
Search for services, features, marketplace products, and docs
[Alt+S]
Identity pool
Dashboard
Sample code
Identity browser
Getting started with Amazon Cognito
Platform
Android
Download the AWS SDK
Download the AWS SDK for Android
Developer Guide

Figure 64: Open services menu

Click “**IAM**” inside “**All Services**”

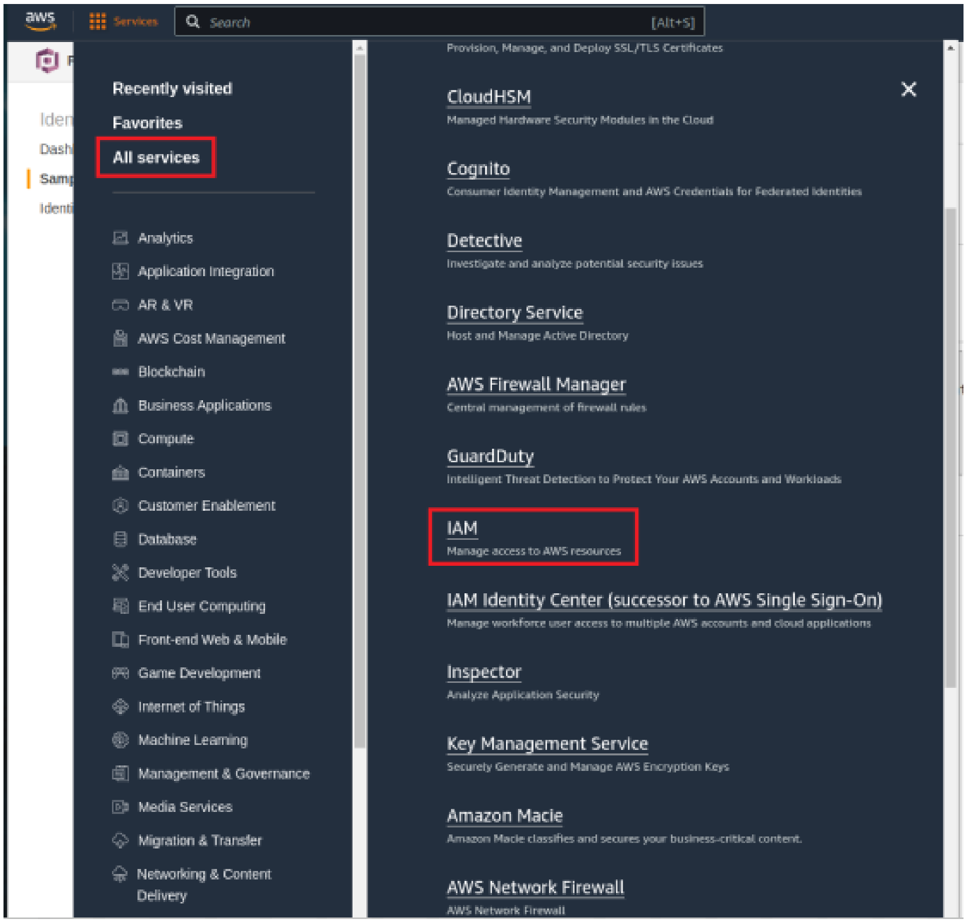
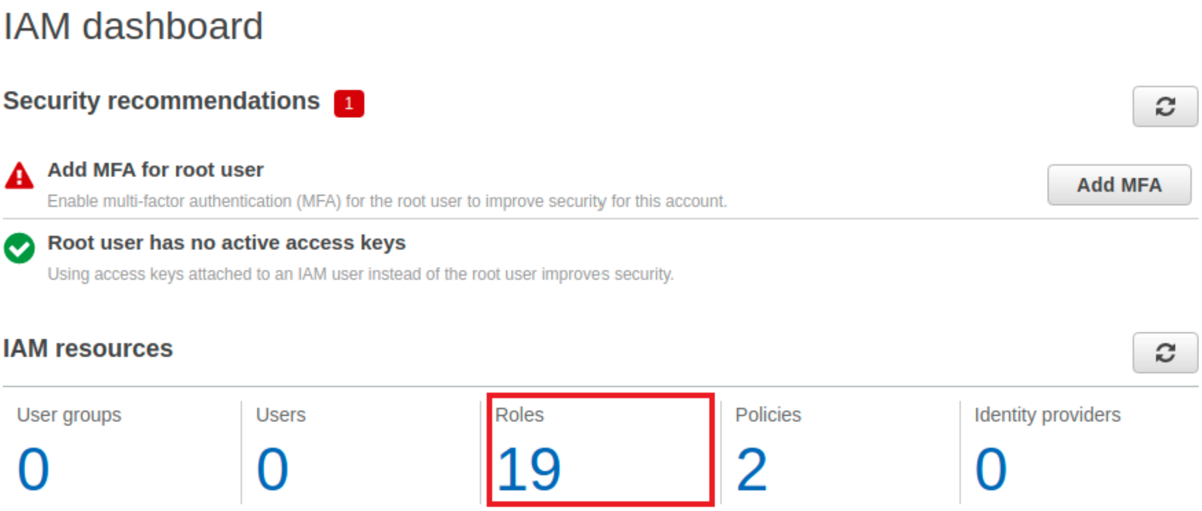

aws
Services
Search
[Alt+S]
Recently visited
Favorites
All services
Analytics
Application Integration
AR & VR
AWS Cost Management
Blockchain
Business Applications
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Management & Governance
Media Services
Migration & Transfer
Networking & Content Delivery
Provision, Manage, and Deploy SSL/TLS Certificates
CloudHSM
Managed Hardware Security Modules in the Cloud
Cognito
Consumer Identity Management and AWS Credentials for Federated Identities
Detective
Investigate and analyze potential security issues
Directory Service
Host and Manage Active Directory
AWS Firewall Manager
Central management of firewall rules
GuardDuty
Intelligent Threat Detection to Protect Your AWS Accounts and Workloads
IAM
Manage access to AWS resources
IAM Identity Center (successor to AWS Single Sign-On)
Manage workforce user access to multiple AWS accounts and cloud applications
Inspector
Analyze Application Security
Key Management Service
Securely Generate and Manage AWS Encryption Keys
Amazon Macie
Amazon Macie classifies and secures your business-critical content.
AWS Network Firewall
AWS Network Firewall

Figure 65: Open IAM

Click “**Roles**” from “**IAM dashboard**”.



IAM dashboard

Security recommendations 1

Add MFA for root user

Enable multi-factor authentication (MFA) for the root user to improve security for this account.

Add MFA

Root user has no active access keys

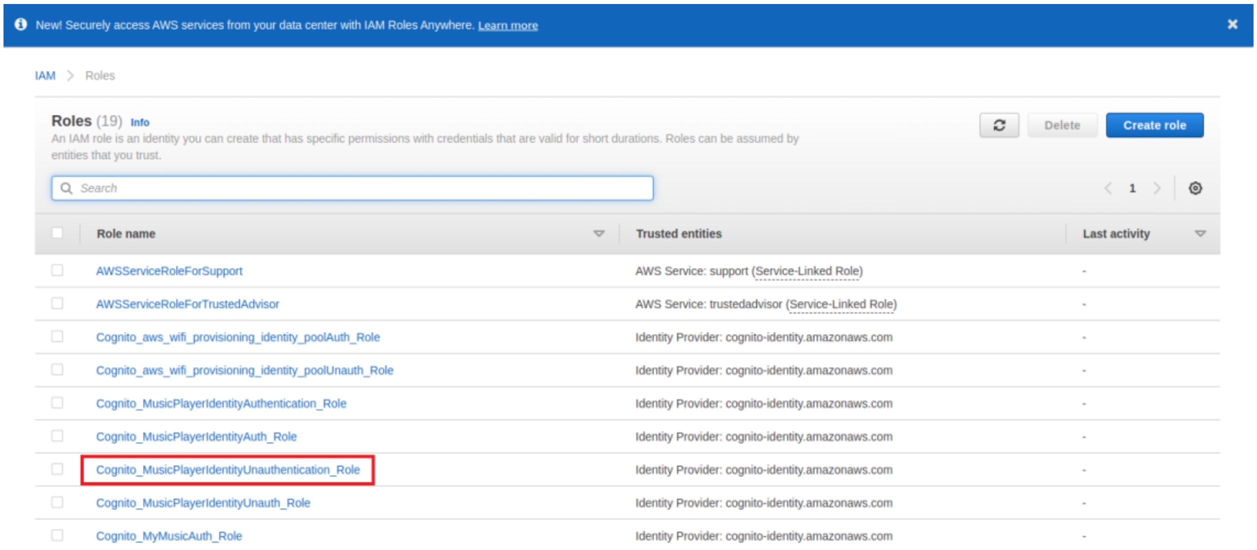
Using access keys attached to an IAM user instead of the root user improves security.

IAM resources

User groups	Users	Roles	Policies	Identity providers
0	0	19	2	0

Figure 66: Open available roles

Click “**Cognito_MusicPlayerIdentityUnauth_Role**”.



New! Securely access AWS services from your data center with IAM Roles Anywhere. Learn more

IAM > Roles

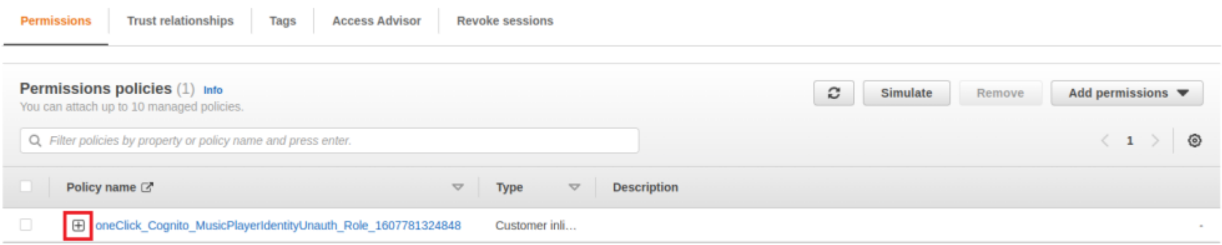
Roles (19) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

	Role name	Trusted entities	Last activity
□	AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
□	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
□	Cognito_aws_wifi_provisioning_identity_poolAuth_Role	Identity Provider: cognito-identity.amazonaws.com	-
□	Cognito_aws_wifi_provisioning_identity_poolUnauth_Role	Identity Provider: cognito-identity.amazonaws.com	-
□	Cognito_MusicPlayerIdentityAuthentication_Role	Identity Provider: cognito-identity.amazonaws.com	-
□	Cognito_MusicPlayerIdentityAuth_Role	Identity Provider: cognito-identity.amazonaws.com	-
□	Cognito_MusicPlayerIdentityUnauthentication_Role	Identity Provider: cognito-identity.amazonaws.com	-
□	Cognito_MusicPlayerIdentityUnauth_Role	Identity Provider: cognito-identity.amazonaws.com	-
□	Cognito_MyMusicAuth_Role	Identity Provider: cognito-identity.amazonaws.com	-

Figure 67: Selecting un-authentication role

Click the arrow as shown to edit the policy.



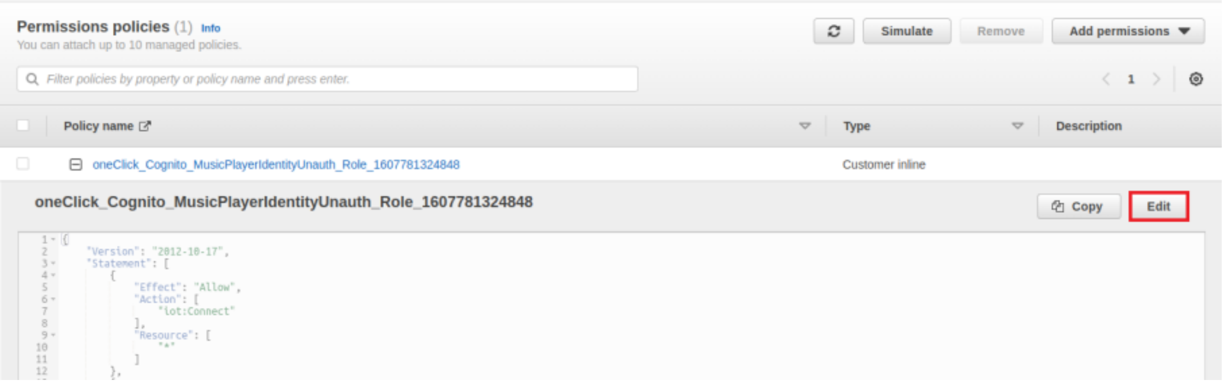
Permissions policies (1) Info

You can attach up to 10 managed policies.

	Policy name	Type	Description
□	⊞ oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848	Customer inli...	-

Figure 68: Open policy content

Click “**Edit**”.



Permissions policies (1) Info

You can attach up to 10 managed policies.

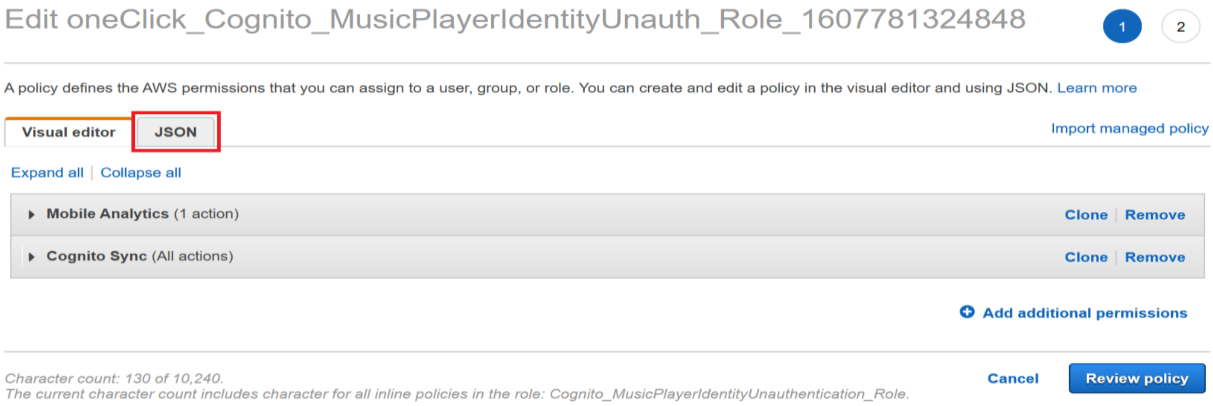
	Policy name	Type	Description
□	⊟ oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848	Customer inline	

oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
```

Figure 69: Edit the policy

Click “**JSON**”



Edit oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Mobile Analytics (1 action)

Cognito Sync (All actions)

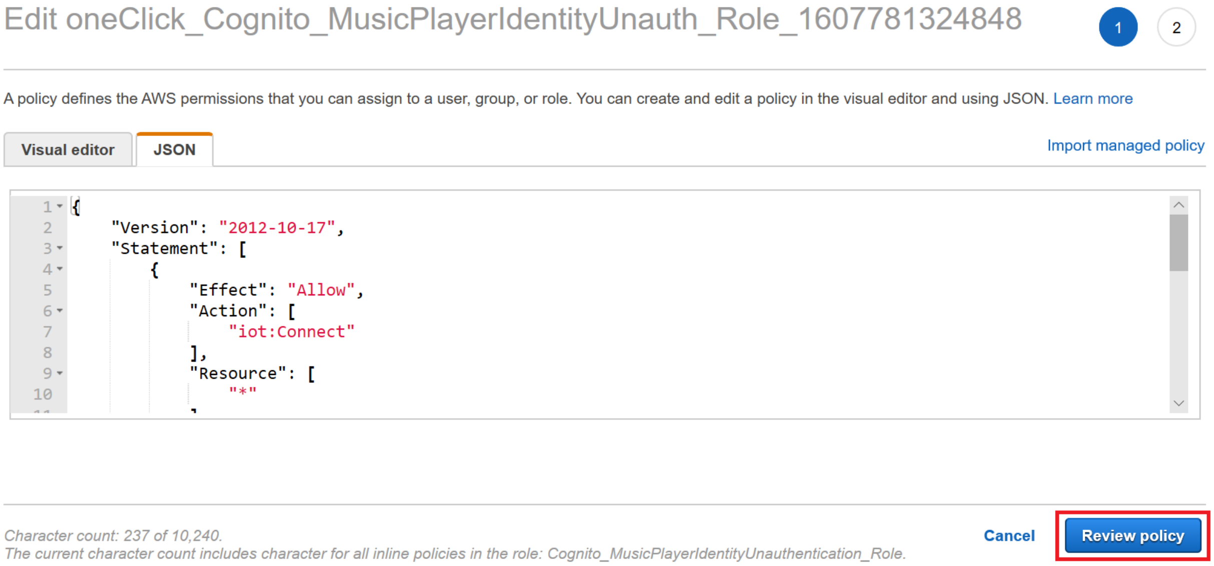
Character count: 130 of 10,240.
The current character count includes character for all inline policies in the role: Cognito_MusicPlayerIdentityUnauthentication_Role.

Figure 70: Open JSON tab

Fill the below content to the policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Click "**Review Policy**"



Edit oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848

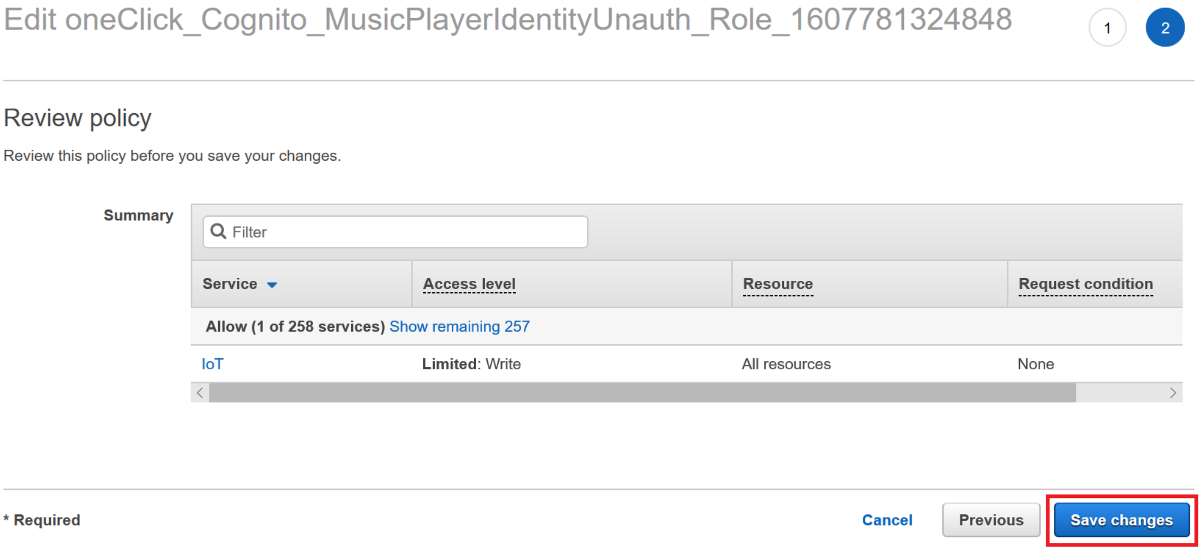
A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
```

Character count: 237 of 10,240.
The current character count includes character for all inline policies in the role: Cognito_MusicPlayerIdentityUnauthentication_Role.

Figure 71: Review policy

Click "**Save changes**"



Edit oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848

Review policy

Review this policy before you save your changes.

Summary

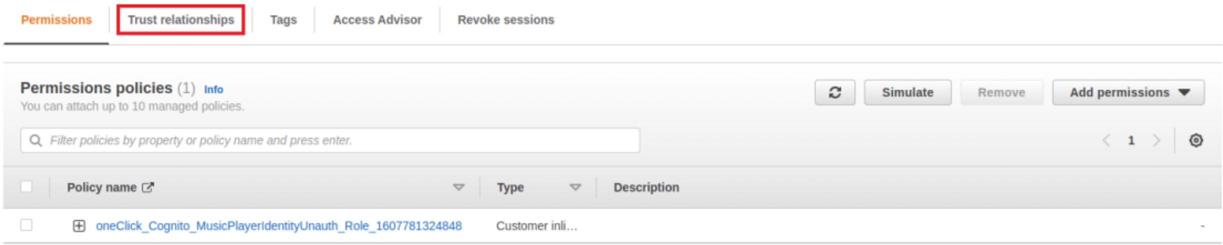
Service	Access level	Resource	Request condition
Allow (1 of 258 services) Show remaining 257			
IoT	Limited: Write	All resources	None

* Required

Figure 72: Save changes for the role selected

The pool is created successfully.

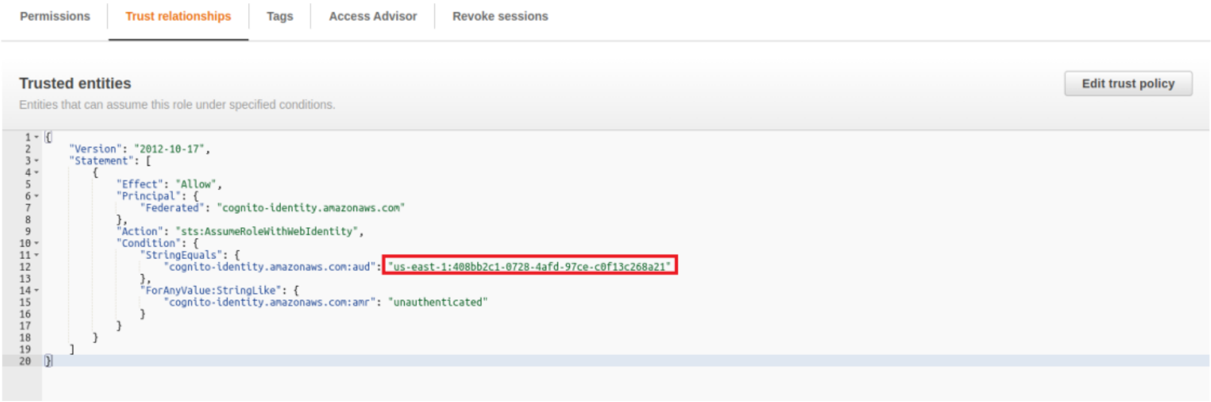
Click “**Trust relationships**”


Permissions
Trust relationships
Tags
Access Advisor
Revoke sessions
Permissions policies (1) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter policies by property or policy name and press enter.
1

Policy name	Type	Description
oneClick_Cognito_MusicPlayerIdentityUnauth_Role_1607781324848	Customer inli…	-

Figure 73: Open Trust relationships tab

Make a copy of the Identity pool ID to use for next section.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": "us-east-1:408bb2c1-0728-4afd-97ce-c0f13c268a21"
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com:amr": "unauthenticated"
                }
            }
        }
    ]
}
```

Figure 74: copy Identity pool ID

5.30.2.6 Prepare Configuration File for the Android Application

Prepare "*AwsMusicControlPreferences.properties*" file with yours AWS credentials.

Its structure looks like this:

```
customer_specific_endpoint=<REST API ENDPOINT>
cognito_pool_id=<COGNITO POOL ID>
thing_name=<THING NAME>
region=<REGION>
```

Where:

customer_specific_endpoint is the endpoint that is configured in *aws_clientcredential.h*

cognito_pool_id is the copied pool id in above step.

thing_name is the created Thing name.

region is the front part of the cognito pool id.

For Example:

```
customer_specific_endpoint=a2qkq65ssjggf7-ats.iot.us-east-1.amazonaws.com
cognito_pool_id=us-east-1:408bb2c1-0728-4afd-97ce-c0f13c268a21
thing_name=MusicPlayer
region=us-east-1
```

To run Android application,

Install and run pre-build *AwsRemoteControl.apk* on Android device, file path is referenced in section 1.3 “References”.

Then in both cases when asked to select *AwsMusicControlPreferences.properties* file with AWS IoT preferences. Then control the music.

NOTE: *Application requires at least Android version 5.1 (Android SDK 22).*

5.30.2.7 Run the application

The log below shows the output of the demo in the console window. The log can be different based on your Wi-Fi network configuration and based on the actions, which you have done in the Android application.

After the log "**Use mobile application to control the remote device**.", the shell command can be used to connect to Bluetooth headset.

```
usb host init done
mass storage device attached:pid=0x6387vid=0x58f address=1
usb msd device is ready
Available audio files:
  demo-1-109869.mp3
  demo-4-109870.mp3
  ruling-planet-biab-demo-song-remix-117443.mp3
  trance-eye-biab-demo-song-remix-117445.mp3
  vlog-hip-hop-18447.mp3
0 197 [main_task] Warning: could not clean-up old crypto objects. 6

1 197 [main_task] Initializing Wi-Fi...

MAC Address: 00:E9:3A:B9:E0:35
2 3372 [main_task] Wi-Fi initialized successfully.

3 3374 [main_task] Connecting to: NXP_Demo

4 12982 [main_task] Wi-Fi connected

5 12983 [main_task] IP Address acquired: 192.168.131.241

6 12984 [MQTT] [INFO] Creating a TLS connection to a2nxzv2h17k05v.ats.iot.cn-
north-1.amazonaws.com.cn:8883.
7 23840 [MQTT] [INFO] (Network connection 0x20257550) TLS handshake successful.
8 23841 [MQTT] [INFO] (Network connection 0x20257550) Connection to
a2nxzv2h17k05v.ats.iot.cn-north-1.amazonaws.com.cn established.
9 23841 [MQTT] [INFO] Creating an MQTT connection to the broker.
10 24744 [MQTT] [INFO] MQTT connection established with the broker.
11 24744 [MQTT] [INFO] Successfully connected to MQTT broker.
12 24745 [SHADOW_DEV] [INFO] MQTT Agent is connected. Initializing shadow
device task.
13 24745 [SHADOW_DEV] [INFO] Sending subscribe request to agent for shadow
topics.
14 24761 [SHADOW_APP] [INFO] MQTT Agent is connected. Initializing shadow
update task.
15 24761 [SHADOW_APP] [INFO] Sending subscribe request to agent for shadow
topics.
16 25620 [SHADOW_APP] [INFO] Received subscribe ack for shadow update topics.
17 25620 [SHADOW_DEV] [INFO] Successfully subscribed to shadow update topics.
18 25621 [SHADOW_DEV] [INFO] Publishing to /get message using client token
25621.
19 25621 [MQTT] [INFO] Publishing message to
$aws/things/MarkWangMusicPlayer/shadow/get.

20 25627 [SHADOW_APP] [INFO] Publishing to /update with following client token
25626.
```

```
21 25627 [MQTT] [INFO] Publishing message to
$aws/things/MarkWangMusicPlayer/shadow/update.

22 25634 [SHADOW_DEV] [INFO] Successfully sent a publish message to /get topic.
Bluetooth initialized

Copyright  2022  NXP

>> 23 26418 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
24 26419 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
25 26420 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
26 26420 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
27 26478 [MQTT] [INFO] De-serialized incoming PUBLISH packet:
DeserializerResult=MQTTSuccess.
28 26478 [MQTT] [INFO] State record updated. New state=MQTTPubAckSend.
29 26488 [MQTT] [INFO] De-serialized incoming PUBLISH packet:
DeserializerResult=MQTTSuccess.
30 26489 [MQTT] [INFO] State record updated. New state=MQTTPubAckSend.
31 26490 [MQTT] [INFO] Received accepted response for update with token 25626.
32 26494 [SHADOW_DEV] [INFO] Received an accepted response for shadow GET
request.
```

Use command “**help**” to list the available options:

```
>> help

"help": List all the registered commands

"exit": Exit program

"bt": BT related function
  USAGE: bt [connectaddress|finddevice|connectdevice|disconnect|deletedevice]
    connectaddress connect to the device of the address parameter,for example:
bt connectaddress xx:xx:xx:xx:xx:xx. Address format(LSB-MSB): xx:xx:xx:xx:xx:xx
    finddevice     start to find BT devices
    connectdevice  connect to the device that is found, for example: bt
connectdevice n (from 1)
    disconnect     current connection
    deletedevice   delete all devices. Ensure to disconnect the HCI link
connection with the peer device before attempting to delete the bonding
information.
>>
```

Use command “**bt finddevice**” to scan nearby Bluetooth devices

```
>> bt finddevice
>> Discovery started. Please wait ...
BR/EDR discovery complete
[1]: 70:F0:87:C0:FC:0E, RSSI -65 iPhone
[2]: BC:17:B8:74:2C:9F, RSSI -52 Galaxy
[3]: 50:82:D5:78:31:DA, RSSI -78 iPhone 6
[4]: 00:00:AB:CD:87:D6, RSSI -38 Airdopes 441
[5]: 04:C8:07:25:29:73, RSSI -69 Mi A3
```

Use command “**bt connectdevice <number>**” to connect to remote Bluetooth device

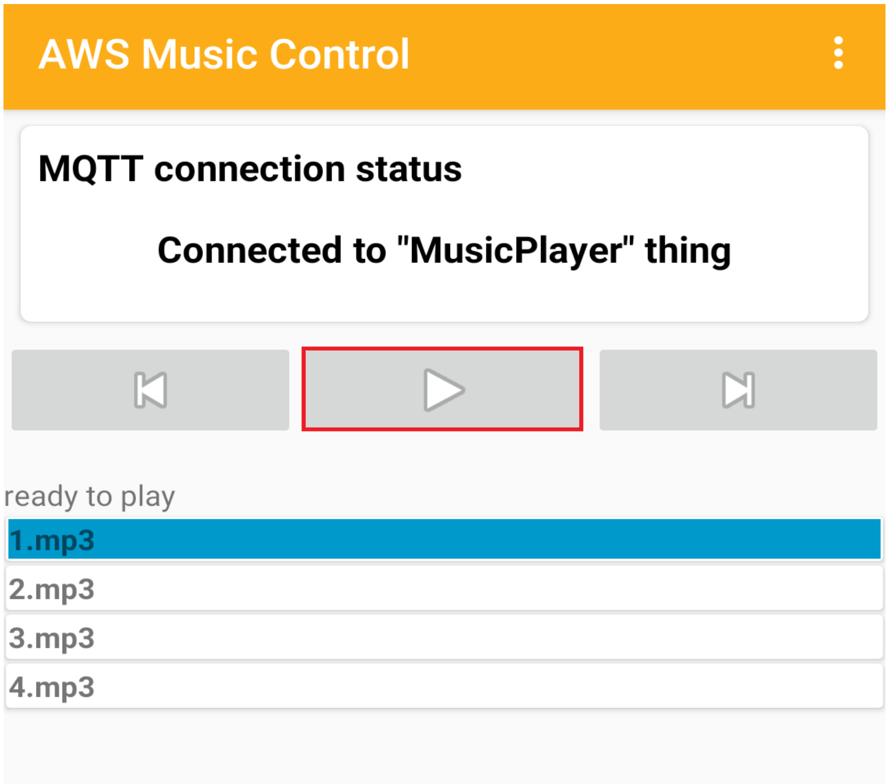
Here, “number” value can be found from the logs of “**bt finddevice**” command used above.

```
>> bt connectdevice 4
>> Connection pending
SDP discovery started
Connected
sdp success callback
A2DP Service found. Connecting ...
Security changed: 7E:5E:2B:2E:9A:C3 (0xed) level 2
```

```
33 114978 [SHADOW_APP] [INFO] Publishing to /update with following client token
114977.
34 114978 [MQTT] [INFO] Publishing message to
$aws/things/MarkWangMusicPlayer/shadow/update.
35 115685 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
36 115685 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
37 115742 [MQTT] [INFO] De-serialized incoming PUBLISH packet:
DeserializerResult=MQTTSuccess.
38 115743 [MQTT] [INFO] State record updated. New state=MQTTPubAckSend.
39 115745 [MQTT] [INFO] Received accepted response for update with token
114977.
```

Open the android app and load the *AwsRemoteControlPreferences.properties*, wait for connection to get complete.

Use the smartphone application to play the music.



AWS Music Control

MQTT connection status

Connected to "MusicPlayer" thing

ready to play

1.mp3

2.mp3

3.mp3

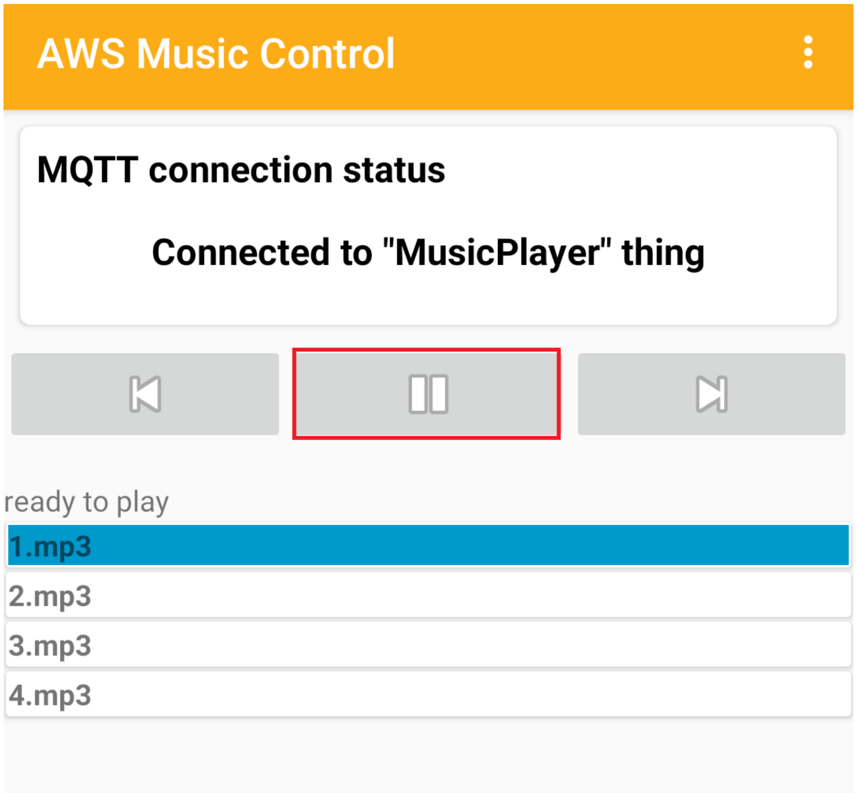
4.mp3

Figure 75: Play music using Android application

Following logs will be appear on the i.MX RT1060 EVK board console:

```
>> start play
[STREAMER] Message Task started
[STREAMER] start playback
Starting playback 0
47 118031 [AWS-RemoteCtrl] [INFO ][MQTT][lu] (MQTT connection 0x20230c48) MQTT
PUBLISH operation queued.
48 118045 [iot_thread] [WARN ][Shadow][lu] Received a Shadow UPDATE response
with no client token. This is possibly a response to a bad JSON document:
{"state":{"desired":{"playIndex":0,"playState":true}},"metadata":{"desired":{"p
layIndex":{"timestamp":1649 118045 [iot_thread] [WARN ][Shadow][lu] Shadow
UPDATE callback received an unknown operation.
50 118695 [iot_thread] [INFO ][Shadow][lu] Shadow UPDATE of MusicPlayer was
ACCEPTED.
51 118695 [AWS-RemoteCtrl] Successfully performed update.
STREAM_MSG_UPDATE_POSITION
  position: 1005 ms
STREAM_MSG_UPDATE_POSITION
  position: 2011 ms
STREAM_MSG_UPDATE_POSITION
  position: 3004 ms
```

Use the smartphone application to pause the music.



MQTT connection status

Connected to "MusicPlayer" thing

Figure 76: Pause music using Android application

Following logs will be appear on the i.MX RT1060 EVK board console:

```
stop play
52 214884 [AWS-RemoteCtrl] [INFO ][MQTT][lu] (MQTT connection 0x2022fdb0) MQTT
PUBLISH operation queued.
53 214885 [iot_thread] [WARN ][Shadow][lu] Received a Shadow UPDATE response
with no client token. This is possibly a response to a bad JSON document:
{"state":{"desired":{"playIndex":0,"playState":false}},"metadata":{"desired":{"
playIndex":{"timestamp":154 214885 [iot_thread] [WARN ][Shadow][lu] Shadow
UPDATE callback received an unknown operation.
55 215504 [iot_thread] [INFO ][Shadow][lu] Shadow UPDATE of MusicPlayer was
ACCEPTED.
56 215504 [AWS-RemoteCtrl] Successfully performed update.
```

Use command “**bt disconnect**” to release the connection with Headset.

```
>> bt disconnect
>> 46 689546 [iot_thread] [ERROR][NET][lu] Error -27648 while sending data.
47 689547 [AWS-RemoteCtrl] [INFO ][MQTT][lu] (MQTT connection 0x20230c48) MQTT
PUBLISH operation queued.
Disconnected (reason 0x16)
58 249328 [iot_thread] [INFO ][Shadow][lu] Shadow UPDATE of MusicPlayer was
ACCEPTED.
59 249329 [AWS-RemoteCtrl] Successfully performed update.
```

Use command “**bt deletedevice**” to remove bound and authentication information of all the connected devices.

```
>> bt deletedevice
>>
```

5.31 Bluetooth Only firmware Download Test Procedure

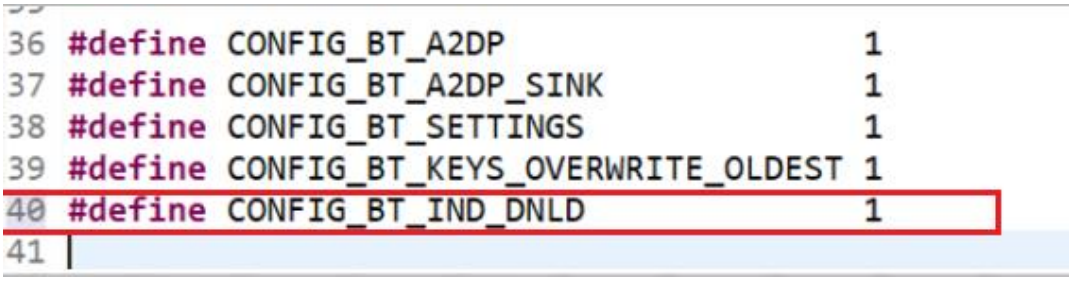
This section describes the steps to configure any Bluetooth / Bluetooth LE sample application for Bluetooth only firmware download mode.

5.31.1 Bluetooth only firmware download Application Execution

Please refer to the previous sections 3.1.1-3.1.4 and 5.1.1 for instructions on importing a project. Select any of the Bluetooth / Bluetooth LE application from the list and import it as described.

5.31.1.1 Build the Application

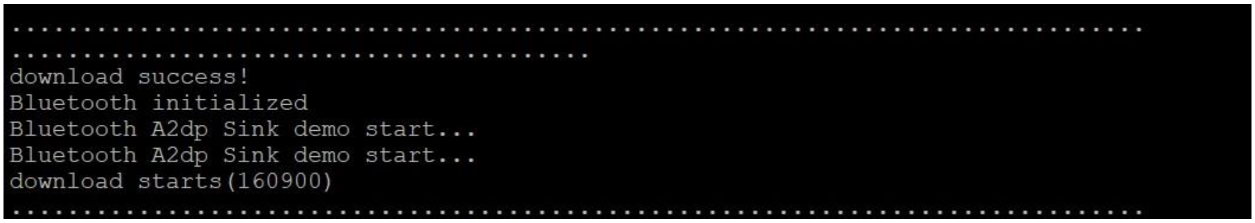
Enable the macro **CONFIG_BT_IND_DNLD** in the application config file and refer the previous sections 3.1.1.3 and 3.1.1.5 to build and flashing the project.



```
#define CONFIG_BT_A2DP                   1
#define CONFIG_BT_A2DP_SINK              1
#define CONFIG_BT_SETTINGS               1
#define CONFIG_BT_KEYS_OVERWRITE_OLDEST  1
#define CONFIG_BT_IND_DNLD               1

```

5.31.1.2 Run the Application

Press the power reset button on i.MX RT board to run the demo application downloaded on the board. When the demo starts, it will download the Bluetooth only Firmware. The following message about the demo would appear on the console.



```
................................................................................
.........................................
download success!
Bluetooth initialized
Bluetooth A2dp Sink demo start...
Bluetooth A2dp Sink demo start...
download starts(160900)
................................................................................
```

6 802.15.4 Sample Application

This chapter describes the Thread example application that is available in the OpenThread repository, and the steps to configure, debug and execute this example.

Thread is an IPv6-based networking protocol designed for low-power Internet of Things devices in an IEEE802.15.4-2006 wireless mesh network. OpenThread released by Google is an open-source implementation of Thread.

The communication between the Open-thread RCP stack and the Link Layer (LL) is implemented via the SPI interface as shown on Figure 76.

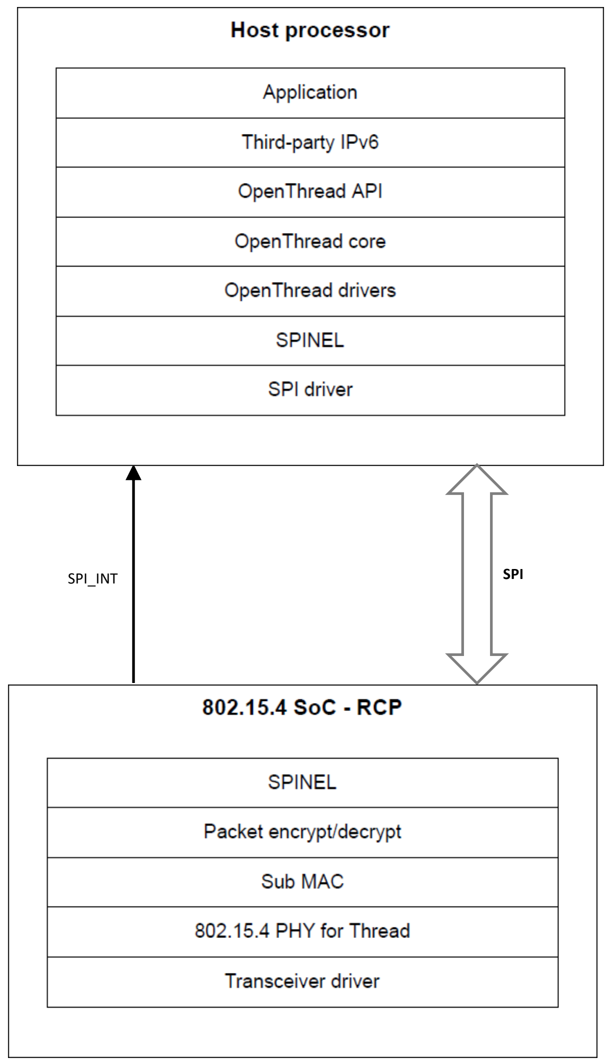


Figure 76. 802.15.4 software architecture

The setup is done between an i.MX RT+ IW612 NXP-based wireless module. The instructions in this guide use an i.MXRT1170 EVK board. Please check the UM11823 - Getting Started with IW612 Evaluation Board and i.MX RT1170 Running RTOS for details of how to build the OpenThread CLI demo.

6.1 CLI Sample Application

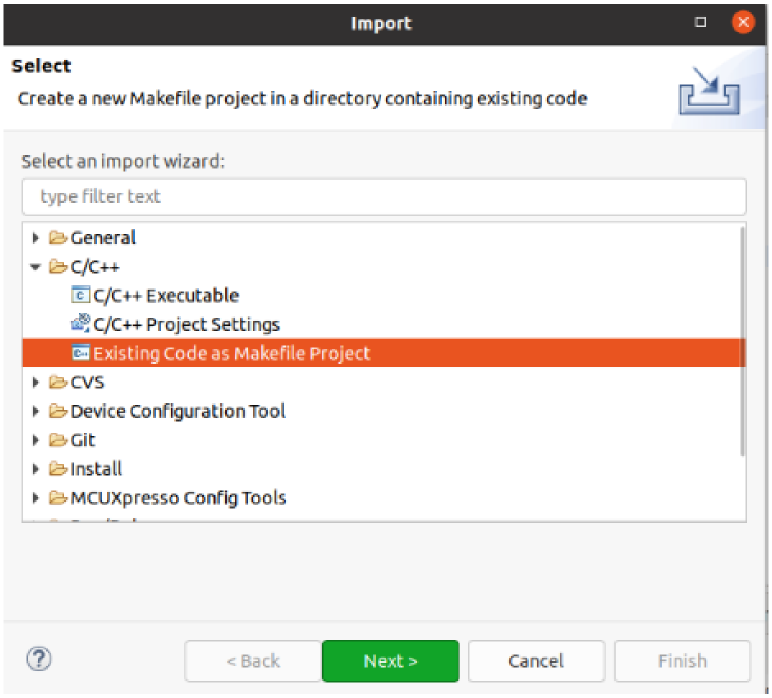
This sample application section describes the steps to configure the i.MX RT1170 EVK board and IW612 wireless module to create/join a thread network and other useful commands using the command line.

It is recommended to use a Linux PC to compile and debug this example.

6.1.1 Create debug session

After following the steps from the Getting Started guide UM11441 section “Run a 802.15.4 demo application”, a folder “build_rt1170” with the executable would be created.

To create the debug session is needed to import to whole ot-nxp folder in MCUXpresso IDE as a “Makefile Project”:



Import

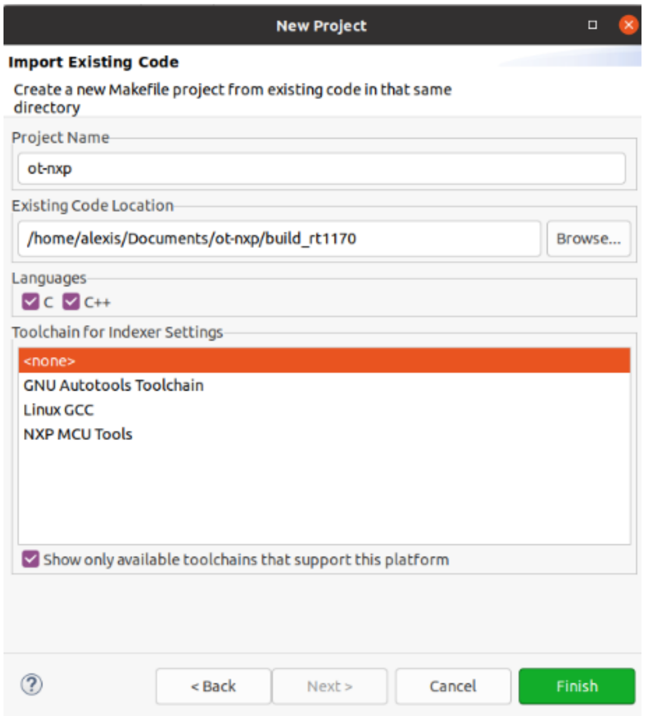
Select

Create a new Makefile project in a directory containing existing code

Select an import wizard:

type filter text

Use none as Toolchain for Indexer Settings:



New Project

Import Existing Code

Create a new Makefile project from existing code in that same directory

Project Name: ot-nxp

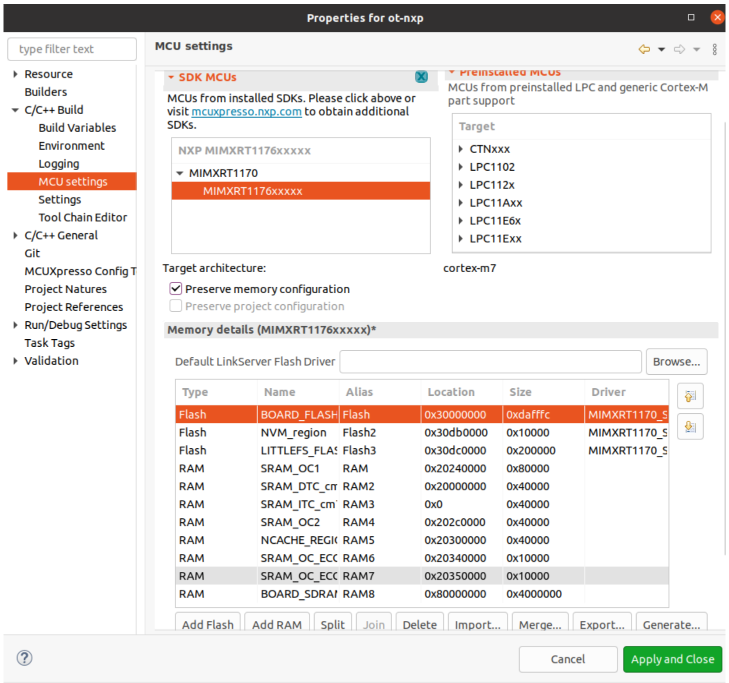
Existing Code Location: /home/alexis/Documents/ot-nxp/build_rt1170

Languages: ☑ C ☑ C++

Toolchain for Indexer Settings:
- <none>
- GNU Autotools Toolchain
- Linux GCC
- NXP MCU Tools

☑ Show only available toolchains that support this platform

Configure memory map to match the following picture. To open this window right click on the Project -> Properties -> C/C++ Build -> MCU Settings -> Select MIMXRT1170 -> Apply & Close:



Properties for ot-nxp

MCU settings

SDK MCUs

MCUs from installed SDKs. Please click above or visit mcuxpresso.nxp.com to obtain additional SDKs.

NXP MIMXRT1176xxxxx
- MIMXRT1170
  - MIMXRT1176xxxxx

Preinstalled MCUs

MCUs from preinstalled LPC and generic Cortex-M part support

Target: CTNxxx, LPC1102, LPC112x, LPC11Axx, LPC11E6x, LPC11Exx

Target architecture: cortex-m7

☑ Preserve memory configuration

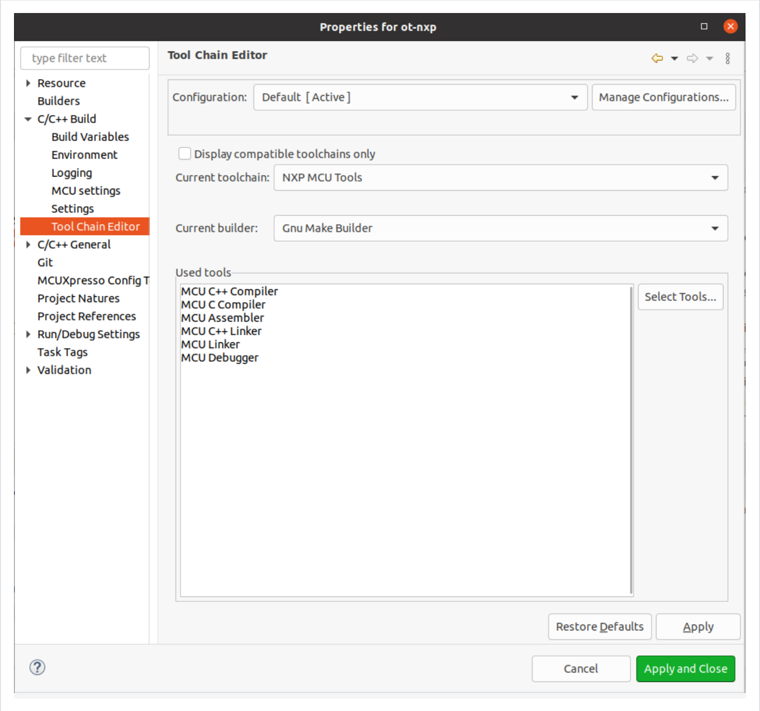
☐ Preserve project configuration

Memory details (MIMXRT1176xxxxx)*

Default LinkServer Flash Driver:

Type	Name	Alias	Location	Size	Driver
Flash	BOARD_FLASH	Flash	0x30000000	0xdafffc	MIMXRT1170_S
Flash	NVM_region	Flash2	0x30db0000	0x10000	MIMXRT1170_S
Flash	LITTLEFS_FLAS	Flash3	0x30dc0000	0x200000	MIMXRT1170_S
RAM	SRAM_OC1	RAM	0x20240000	0x80000	
RAM	SRAM_DTC_cm	RAM2	0x20000000	0x40000	
RAM	SRAM_ITC_cm	RAM3	0x0	0x40000	
RAM	SRAM_OC2	RAM4	0x202c0000	0x40000	
RAM	NCACHE_REGI	RAM5	0x20300000	0x40000	
RAM	SRAM_OC_ECC	RAM6	0x20340000	0x10000	
RAM	SRAM_OC_ECC	RAM7	0x20350000	0x10000	
RAM	BOARD_SDRA	RAM8	0x80000000	0x4000000	

Configure the toolchain editor to NXP MCU Tools. To open this window right click on the Project -> C/C++ Build-> Tool Chain Editor -> NXP MCU Tools -> Apply & Close



Properties for ot-nxp

Tool Chain Editor

Configuration: Default [Active]

☐ Display compatible toolchains only

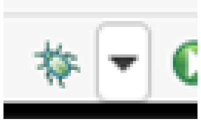
Current toolchain: NXP MCU Tools

Current builder: Gnu Make Builder

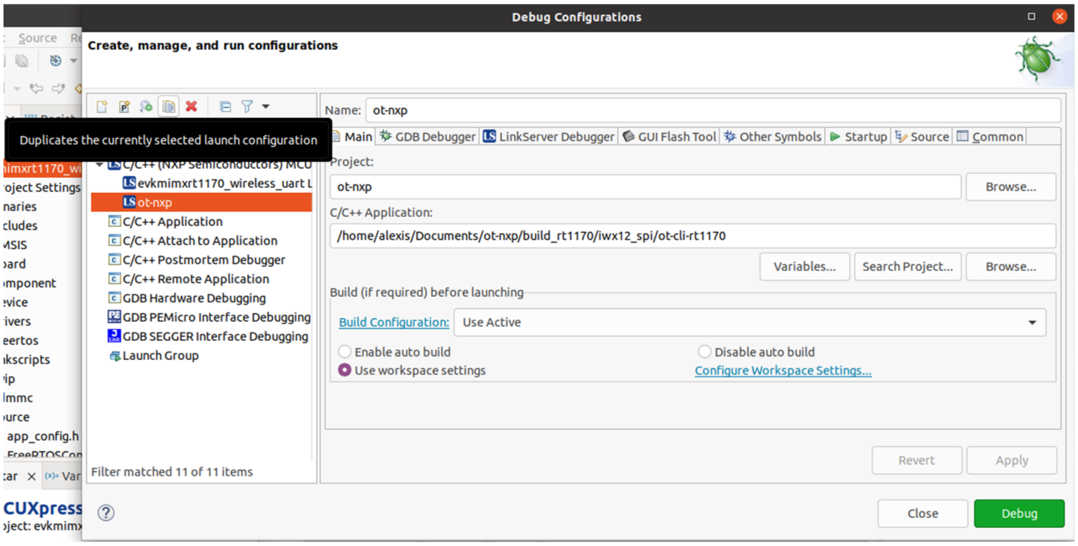
Used tools

MCU C++ Compiler
MCU C Compiler
MCU Assembler
MCU C++ Linker
MCU Linker
MCU Debugger

To create the debug session, it’s needed to duplicate a current debug session for the same chipset, in this case, it is used a Wi-Fi CLI debug session. In the drop-down menu on the “green bug” select the Debug configurations:



Click on the duplicate icon and update the project to point to the ot-nxp workspace project. Update C/C++ Applications to point to the ot-nxp executable generated on path ot-nxp/build_rt1170/iwx12_spi/ot-cli-rt1170:



Debug Configurations

Create, manage, and run configurations

Duplicates the currently selected launch configuration

Name: ot-nxp

Project:

ot-nxp

C/C++ Application:

/home/alexis/Documents/ot-nxp/build_rt1170/iwx12_spi/ot-cli-rt1170

Build (if required) before launching

Build Configuration: Use Active

○ Enable auto build ○ Disable auto build

● Use workspace settings

Filter matched 11 of 11 items

After that, a debug session for the OpenThread project should be generated correctly.

6.1.2 ot-nxp Application Execution

6.1.2.1 Start-up logs

The following character can be observed on the console once the devices (i.MX RT1170 EVK board and IW612 module) are up and running and it shows that 802.15.4 module is ready for the operations.

```
>
```

6.1.2.2 Help Command

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

```
> help
bbr
bufferinfo
ccathreshold
ccm
channel
child
childip
childmax
childsupervision
childtimeout
coap
commissioner
contextreusedelay
counters
dataset
delaytimermin
discover
dns
domainname
eidcache
eui64
extaddr
extpanid
factoryreset
fake
fem
ifconfig
ipaddr
ipmaddr
joiner
joinerport
keysequence
leaderdata
leaderweight
log
mac
mliid
mlr
mode
multiradio
neighbor
netdata
netstat
networkdiagnostic
networkidtimeout
networkkey
networkname
panid
parent
parentpriority
partitionid
```

```
ping
pollperiod
preferrouterid
prefix
promiscuous
pskc
rcp
region
releaserouterid
reset
rloc16
route
router
routerdowngradethreshold
routereligible
routeridrange
routerselectionjitter
routerupgradethreshold
scan
service
singleton
state
tcp
thread
tvcheck
txpower
udp
unsecureport
version
Done
```

6.1.2.3 Factory Reset 15.4 module

The factoryreset command is used to reset any change made on the current network and reset the device.

```
> factoryreset
Done
```

6.1.2.4 Scan command

The scan command is used to scan the visible thread devices.

```
> scan
| PAN  | MAC Address      | Ch | dBm | LQI |
+------+------------------+----+-----+-----+
| 2233 | 2a702820dd853ea7 | 11 | -44 | 136 |
Done
```

6.1.2.5 Add leader network data

Before creating a network, we need to define certain parameters as the network key, network channel, PAN ID and network name.

```
> dataset init new
Done
> dataset channel 11
Done
> dataset networkkey 00112233445566778899AABBCCDDEE00
Done
> dataset panid 0x0123
Done
> dataset networkname ot-example
Done
> dataset commit active
Done
```

6.1.2.6 Start network

Using the data previously set, a network would be created.

```
> ifconfig up
Done
> thread start
Done
```

6.1.2.7 Enable commissioner

To enable other devices to join the network, it's needed to enable the commissioner role, this is used to authenticate a device onto the network.

```
> commissioner start
Commissioner: petitioning
Done
Commissioner: active
> commissioner joiner add * NXP123(pskd/pskc)
Done
> ~ Discovery Request from 2a702820dd853ea7: version=2,joiner=1
Commissioner: Joiner start 2a702820dd853ea7
Commissioner: Joiner connect 2a702820dd853ea7
Commissioner: Joiner finalize 2a702820dd853ea7
Commissioner: Joiner end 2a702820dd853ea7
```

6.1.2.8 Join network

For a device to be able to join it need to have the same network key and pskd/pskc. Also the device need to change to a joiner role to be able to send and receive the information to connect to the current network.

```
> dataset networkkey 00112233445566778899AABBCCDDEE00
Done
> dataset commit active
Done
> ifconfig up
Done
> joiner start NXP123 (pskd)
Done
Join success
> thread start
Done
```

6.1.2.9 Ping Devices

For a device to be able to ping another, it's needed the IPv6 of the target device. Use the ipaddr command on the target device to obtain the IPv6 on the target device.

```
> ipaddr
fd70:e262:f738:8d2e:0:ff:fe00:9001
fd70:e262:f738:8d2e:6c0d:de9c:7602:20ab
fe80:0:0:0:2cea:23a4:654:8c28
Done
```

After obtaining the IPv6 of the target device. The following command can be used to ping other devices.

```
> ping fe80:0:0:0:f4e7:f954:e813:7e4a
16 bytes from fe80:0:0:0:f4e7:f954:e813:7e4a: icmp_seq=1 hlim=64 time=35ms
1 packets transmitted, 1 packets received. Packet loss = 0.0%. Round-trip
min/avg/max = 35/35.0/35 ms.
Done
```

6.1.2.10 UDP Server/Client

The sample application implements a UDP protocol communication

To open a socket on the server:

```
> udp open
Done
> udp bind :: 1234
Done
```

To connect to the socket on the client side:

```
> udp open
Done
> udp connect fe80:0:0:0:f4e7:f954:e813:7e4a 1234
Done
```

Send a command from client to server:

```
> udp send hello
```

Output from server:

```
> 5 bytes from fe80:0:0:0:2cea:23a4:654:8c28 49154 hello
```

6.1.2.11 Other useful commands

Router

Print all the routers on the network, depending of the parameter is the information

Table

```
> router table
| ID | RLOC16 | Next Hop | Path Cost | LQ In | LQ Out | Age | Extended MAC
| Link |
+----+--------+----------+-----------+-------+--------+-----+------------------
+------+
| 24 | 0x6000 |       63 |         0 |     0 |      0 |   0 | 2eea23a406548c28
|    0 |
| 36 | 0x9000 |       63 |         0 |     3 |      3 |  18 | f6e7f954e8137e4a
|    1 |

Done
```

List:

```
> router list
24 36
Done
```

Router ID:

```
> router 36
Alloc: 1
Router ID: 36
Rloc: 9000
Next Hop: fc00
Link: 0
Done
```

EUI64

Get the factory-assigned IEEE EUI-64

```
> eui64
ffffffffffffffff
Done
```

Router eligible

By default, the example sets the devices as a REED (Router Eligible Device). In case the devices needs to join the network only as an endpoint you can change this using the following command.

Disable router role:

```
> routereligible disable
Done
```

Enable router role:

```
> routereligible enable
Done
```

Get router role:

```
> routereligible
Enabled
Done
```

7 Acronyms and abbreviations

Table 25: Acronyms and Abbreviations

Terms	Definition
ACS	Auto Channel Selection
AP	Access Point
API	Application Program Interface
CLI	Command Line Interface
CMSIS	Cortex® Microcontroller Software Interface Standard
DFP	Device Family Pack
DHCP	Dynamic Host Configuration Protocol
DHCPD	DHCP daemon
ED	Energy Detection
EU	European Union
EVK	Evaluation Kit
Ext AP	External Access Point
Ext STA	External Station
FW	Firmware
IDE	Integrated Development Environment
IP	Internet Protocol
lwIP	Lightweight IP
NAT	Network Address Translation
PS	Power Save
Rx	Receive
SD	Secure Digital
SDK	Software Development Kit
SSID	Service Set Identifier
STA	Station/client
SW	Software
TCP	Transmission Control Protocol
TRPC	Transmit Rate-based Power Control
Tx	Transmit
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
MFP	Management Frame Protection
OTP	One Time Programmable
ETSI	European Telecommunications Standards Institute
A2DP	Advanced Audio Distribution Profile
HFP	Hands-Free Profile
SPP	Serial Port Profile
BT	Bluetooth

BLE	Bluetooth Low Energy
PXR	Proximity Reporter
PXM	Proximity Monitor
HTS	Health Thermometer Service
IPSP	Internet Protocol Support Profile
HTTP	Hypertext Transfer Protocol
ACL	Asynchronous Connection-Less Link
AWS	Amazon Web Services
HCI	Host Controller Interface
UART	Universal Asynchronous Receiver Transmitter
PCM	Pulse-code Modulation
HS	High Speed
USB	Universal Serial Bus
EIP	Event in progress
PRI	Priority
PTA	Packet Traffic Arbiter

8 Contact Us

Please refer following links for more product details, queries and support.

- **Home Page: nxp.com**
- **Web Support: nxp.com/support**
  NXP Community: **https://community.nxp.com/**

9 Legal Information

9.1 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.

9.2 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 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.

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.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a 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 or documented vulnerabilities. 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

9.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Figures

Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10 Jan 2025
Document identifier: UM11442