Qualcomm Technologies, Inc.

QCA402x (CDB2x) Development Kit

User Guide

80-YA121-140 Rev. C

November 16, 2018

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Qualcomm Technologies, Inc. ("QTI") and its affiliates reserve the right to make any updates, corrections and any other modifications to its documentation. The information provided in this document represents QTI's knowledge and belief as of the date this document is provided. QTI makes no representation or warranty as to the accuracy of such information, and QTI assumes no liability for any use of the information in this documentation. You should obtain the latest information before placing orders for any hardware, and you should verify that such information is current and complete. Information published by QTI regarding any third-party products does not constitute a license to use such products or a warranty or endorsement thereof. Use of such information may require a license from a third party under the intellectual property rights of such third party, or a license from QTI or its affiliates under the intellectual property rights of QTI or its affiliates.

All hardware, equipment, components or products are sold subject to QTI's (or such other QTI affiliated company that is designated by QTI) standard terms and conditions of sale, as applicable. Notwithstanding anything to the contrary in this documentation or otherwise: (i) you do not receive any rights, licenses, or immunities from suit under any patents of Qualcomm Incorporated, QTI or their respective affiliates as a result of receiving this documentation (whether expressly, impliedly, by virtue of estoppel or exhaustion, or otherwise), (ii) without limitation, you shall not use or sell any wireless wide area network ("WWAN") baseband integrated circuit that you purchase or acquire from QTI or any product that incorporates any such WWAN baseband integrated circuit (whether alone or in combination with any other software or components) without a separate license or non-assertion covenant from Qualcomm® Incorporated in respect of or under all applicable patents, (iii) nothing in this document modifies or abrogates your obligations under any license or other agreement between you and Qualcomm Incorporated, including without limitation any obligation to pay any royalties, and (iv) you will not contend that you have obtained any right, license, or immunity from suit with respect to any patents of Qualcomm Incorporated, QTI or their respective affiliates under or as a result of receiving this documentation (whether expressly, impliedly, by virtue of estoppel or exhaustion, or otherwise).

Revision history

Revision	Date	Description
A	March 2018	Initial release
B	July 2018	Removed all references to Qmesh
C	November 2018	Updated the following chapters: ▪ Chapter 2 Onboard demo ▪ Chapter 3 Getting started ▪ Chapter 6 QCLI demo Added the following chapters: ▪ Chapter 8 HTC demo ▪ Chapter 9 Hosted mode demo

Contents

Figures

Tables

1 QCA402x development kit

1.1 Purpose

This document provides instructions for setting up the QCA402x development kit, building, and using the SDK with sample demo applications. The SDK supports the CDB20/24 reference design, which is based on the Qualcomm Technologies, Inc. QCA4020/QCA4024 Wireless SoC.

1.2 Conventions

- The function declarations, function names, type declarations, attributes, and code samples appear in a different font. **Example**: `#include`.
- Code variables appear in angle brackets. **Example**: `<number>`.
- Commands to be entered appear in a different font. **Example**: **`copy a:*.* b:.`**
- Button and key names appear in bold font. **Example**: Click **Save** or press **Enter**.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com.

2 Onboard demo

The onboard demo applications provide a mechanism to demonstrate end-to-end communication between CDB2x devices and the mobile app through cloud network or wireless technology. The onboard demo supports three kinds of modes as shown in

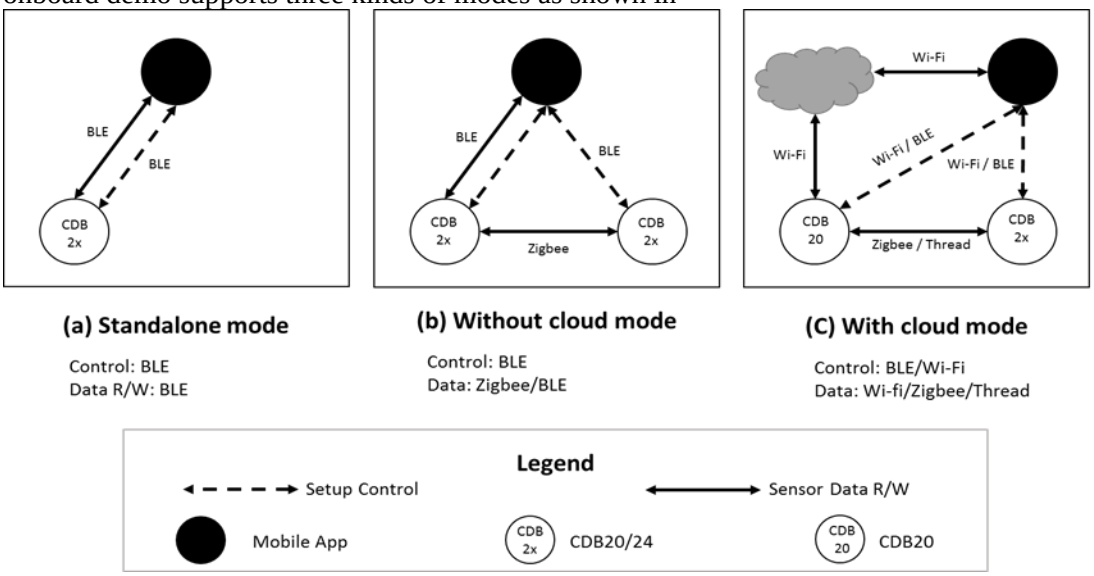


Figure 2-1.

- Standalone mode: In this mode, the mobile app directly reads/writes sensor data and controls setup on CDB2x device using a BLE connection.
- Without cloud mode: In this mode, the mobile app can read/write sensor data on CDB2x devices through Zigbee coordinator device.
- With cloud mode: In this mode, the mobile app can read/write sensor data on CDB2x devices through cloud network.

This section describes onboarding procedure and setup details for the standalone mode and without cloud mode. For details about the With cloud mode, see Chapter, Onboard demo with cloud mode.

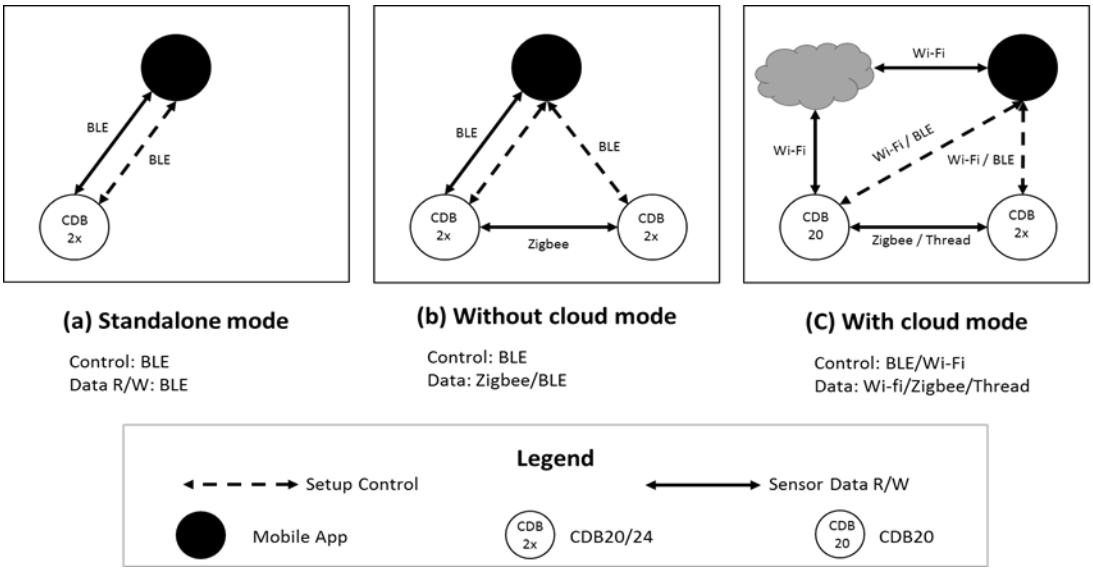


Figure 2-1 Topology of onboard demo

The consecutive sections demonstrate how to setup the onboard demo for CDB2x devices in offline mode, which does not, have cloud interface for both standalone and without cloud mode. The mobile application can read sensor data and control sensors on CDB2x devices. This communication is through BLE (between mobile application and CDB2x) and Zigbee (between CDB2x devices). Two applications are required to set up onboarding − first onboard demo application running on CDB2x device and second (QCA Onboarding and Sensor) application running on mobile device. For the mobile applications, see the Onboard mobile application section.

2.1 Standalone mode – onboard demo

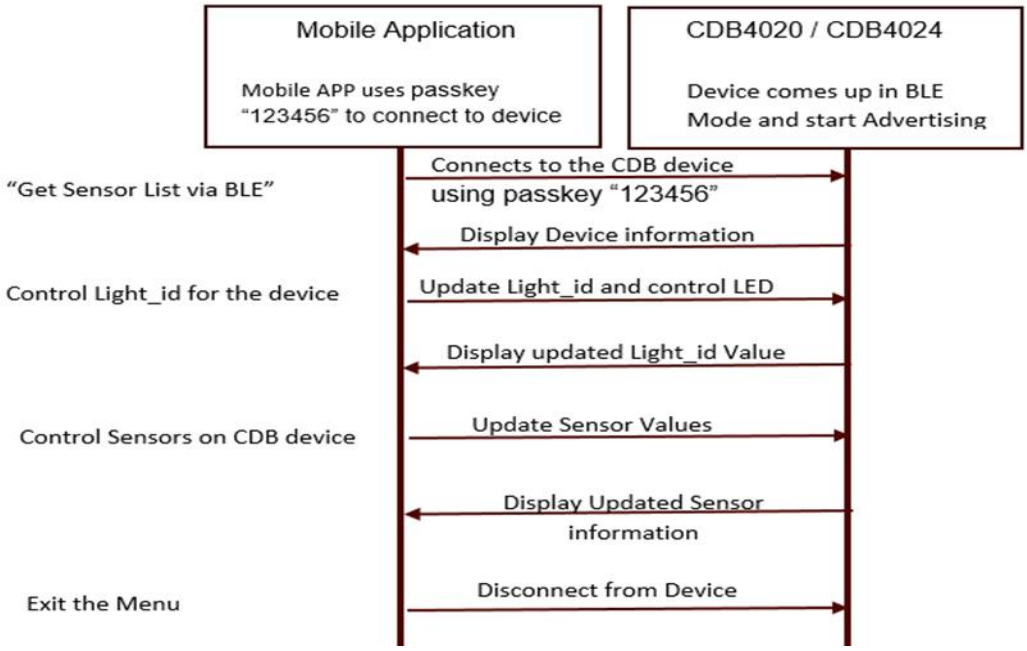


Figure 2-2 Standalone mode onboard demo

The CDB2x devices bootup on power cycle, come in BLE peripheral mode and start advertising.

Select "Get Sensor Data [BLE]" for standalone mode onboard demo and connect to the CDB2x device using qca_onboarding application from the android device. The mobile app uses BLE scan to connect to CDB2x devices with default password"123456".

Select device from the list to take user to the screen where it provides the details of the sensors as shown in Figure 2-3.

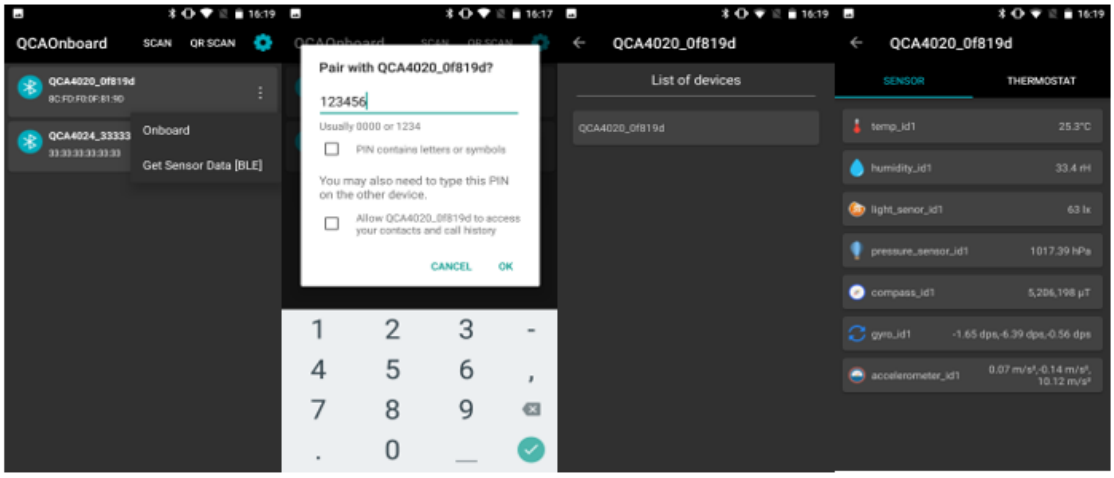


Figure 2-3 Mobile application in standalone mode

2.2 Without cloud mode – onboard demo

2.2.1 Prerequisites

1. The qca_onboarding mobile application has the provision to save list of previous onboarded ZigBee coordinators.
2. Before onboarding the device multiple times, it is required to delete the previous onboarded device, however it is not required while onboarding for the first time.
3. Click the "**Settings**" icon on top right corner of qca_onboarding mobile application. Select the Coordinator List, to find the list of saved Zigbee Coordinator devices.
4. Click the "**Delete**" icon and choose "**YES**" to remove the device from saved list of Zigbee Coordinators.

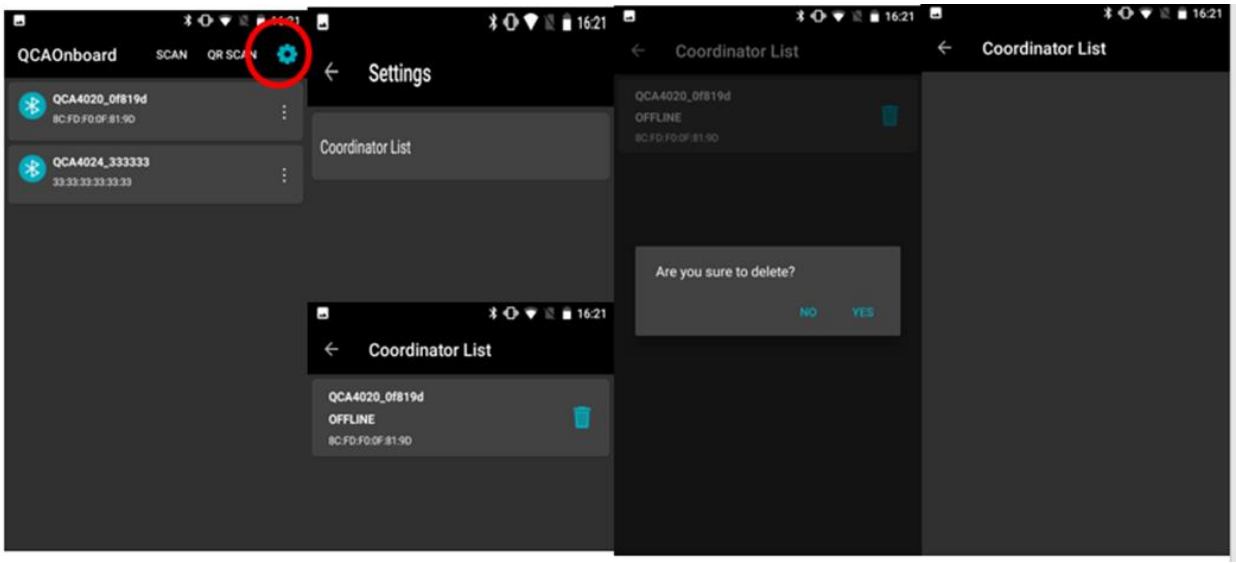


Figure 2-4 Delete saved Zigbee Coordinator Device With/Without cloud mode

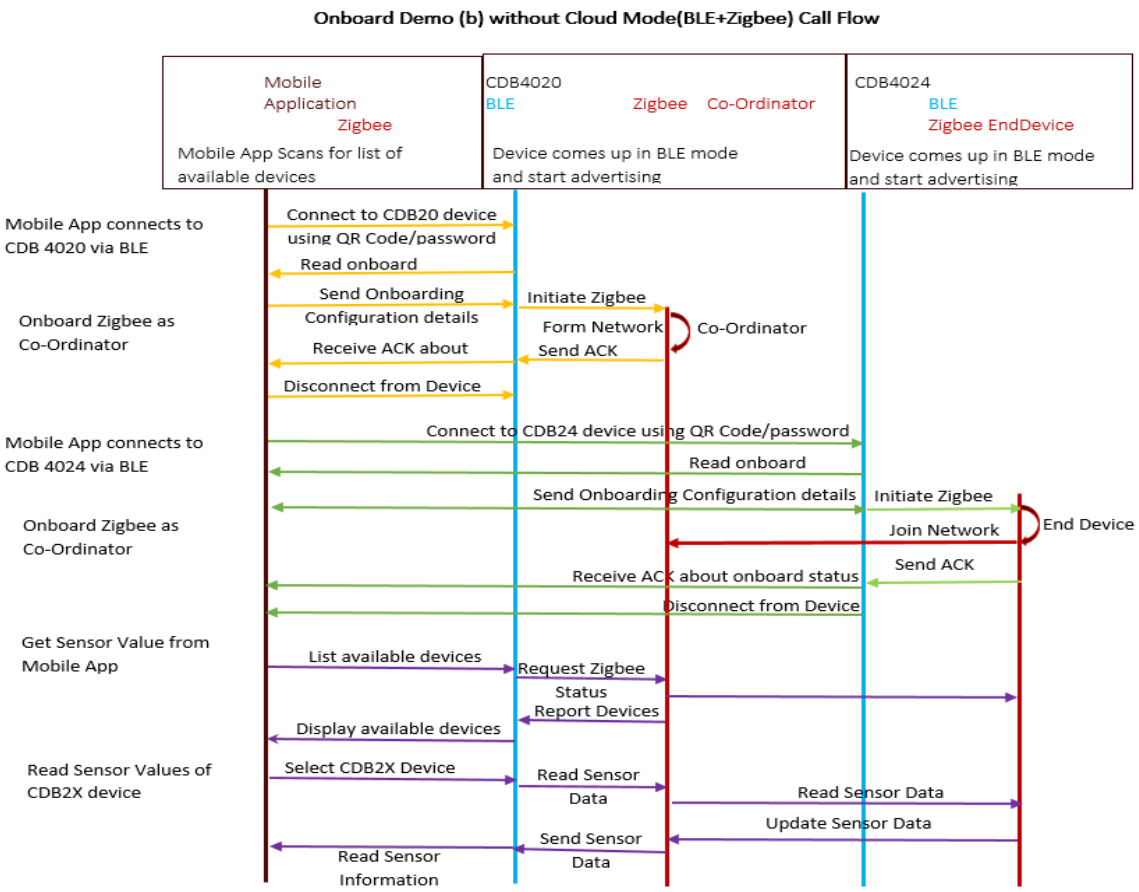


Figure 2-5 Without cloud mode onboard demo

CDB4020 and CDB4024 devices bootup on power cycle, come in BLE peripheral mode and start advertising.

The mobile application uses BLE scan to connect to CDB20/24 devices with default password "123456" for BLE onboarding.

After the CDB20 board receives the BLE passkey and proceeds onboarding from the mobile app, it validates the configuration, and saves it to the filesystem.

Onboard Zigbee coordinator on CDB20: On successful BLE connection of mobile app and CDB20 device, the mobile app reads the Zigbee service for onboard status through BLE connection. If the status is success and Zigbee mode is coordinator, the mobile app generates link key and sends the credentials packet to the CDB20 device running as Zigbee coordinator.

After the CDB20 device gets the operating mode and link key, the onboarding demo extracts the onboard details, validates the configuration, stores to the filesystem and, sends the ACK to the mobile app. The CDB20 device starts Zigbee service as a coordinator/router/end device based on the user-configured Zigbee operating mode.

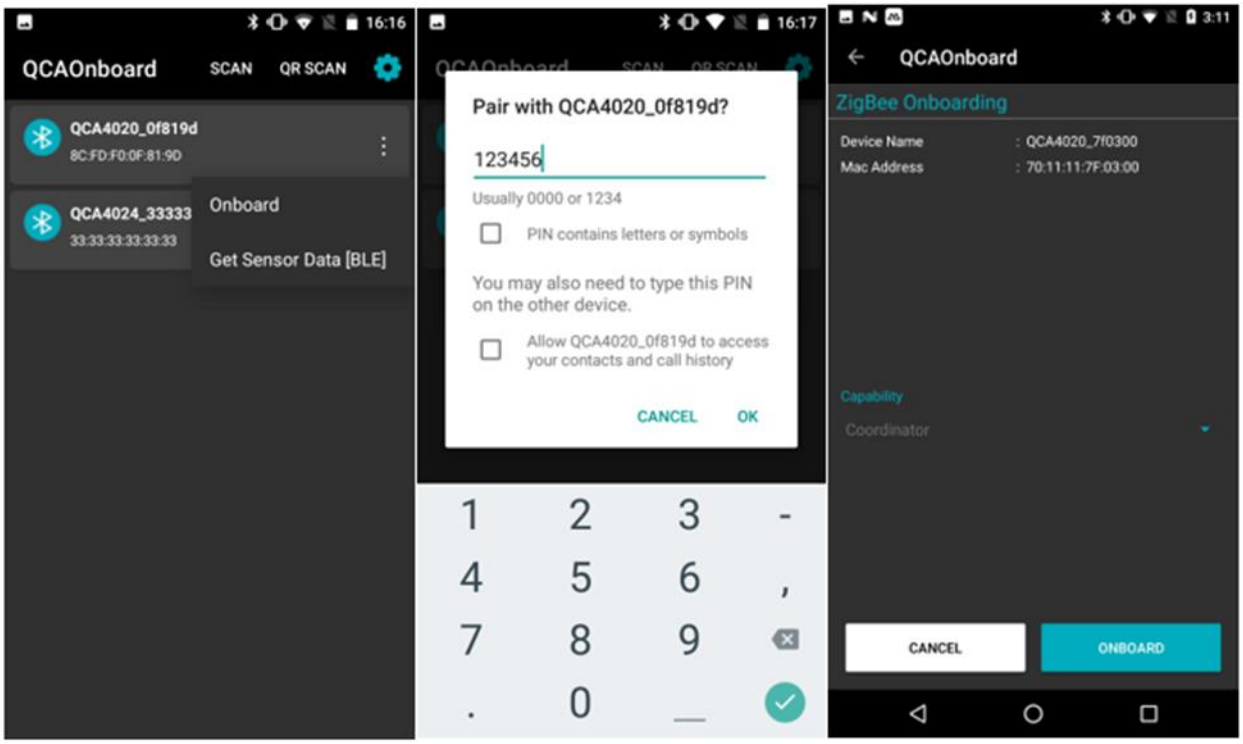


Figure 2-6 Onboard Zigbee coordinator on CDB20

After the mobile app receives the Zigbee onboard status notification, it displays the onboard status screen and status.

Onboard Zigbee end device on CDB24: Onboard the CDB24 device using BLE connection. The CDB24 device runs as the Zigbee end device connecting to the Zigbee coordinator CDB20 device.

On the mobile app main screen, select CDB24 device and onboard the device using the default password “123456” using BLE connection.

The mobile app checks for the CDB24 onboard status and displays the Zigbee connection status using BLE. The user must select the relevant Co-Ordinator CDB20 device name, which allows the CDB24 as end device to join the zigbee network using BLE connection.

1. If the CDB24 is onboarded successfully, the mobile app displays successful onboarding device status; else notifies with failure status and disconnects from the device.
2. **Get Sensor Data through Zigbee and BLE**: Once both CDB20/24 devices are onboarded successfully, the user is able to read sensor data and control sensors on the Zigbee end device CDB24 through Zigbee coordinator CDB20.
3. On the mobile app screen, connect to the CDB20 device QCA4020_XXXXXX, and choose “Get Sensor List [Zigbee+BLE]” button using BLE using default password “123456”.

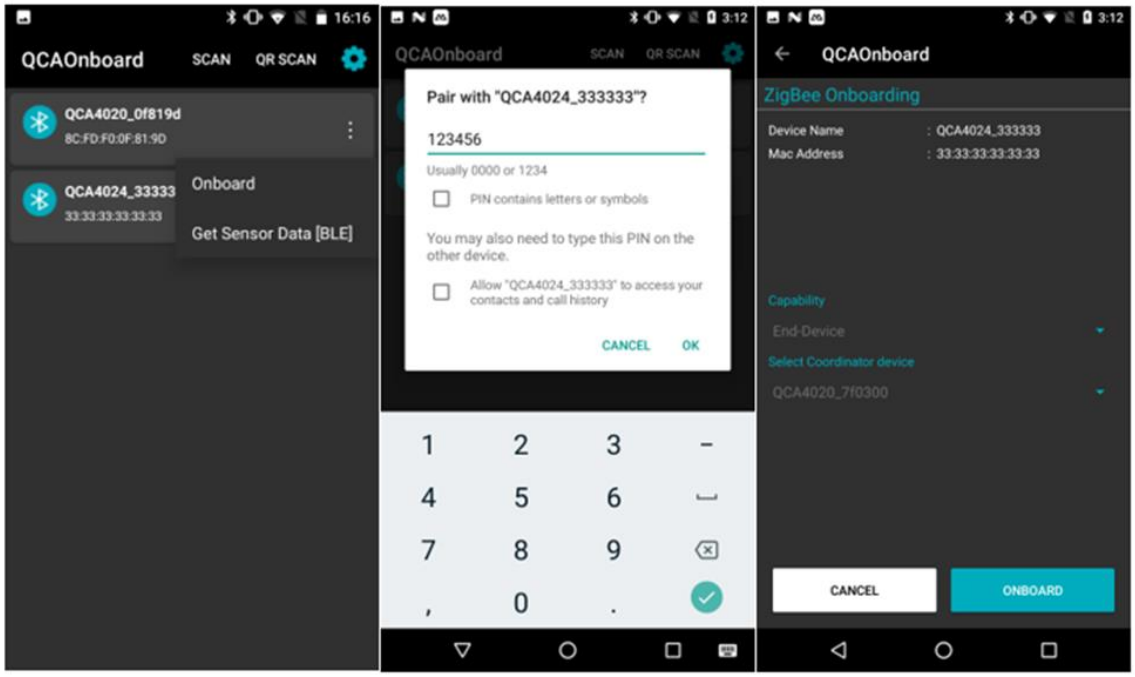
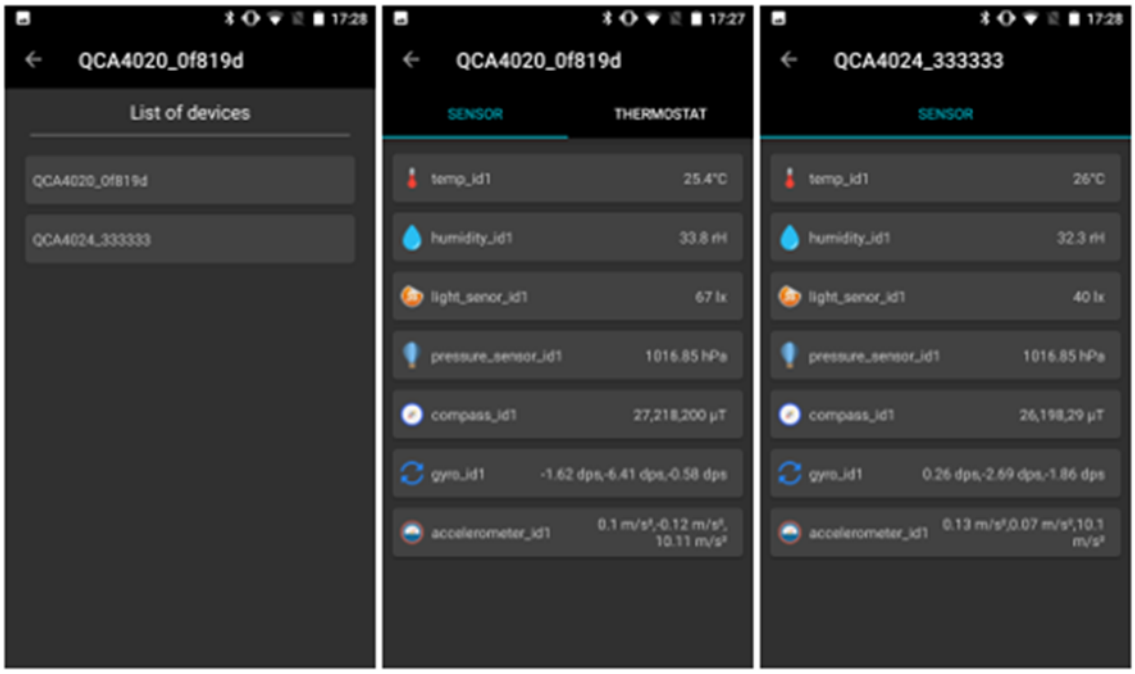


Figure 2-7 Zigbee onboarding status

4. The mobile app screen displays the list of devices under the Zigbee network.
5. The user can select the Zigbee-connected CDB20/24 device, read the sensor values, and control the light intensity using BLE connection.



6. The user can erase the stored credentials of the Zigbee network on CDB20/24 devices by issuing hard reset commands `<Onboard> <reset_onboard_info>` in the console.

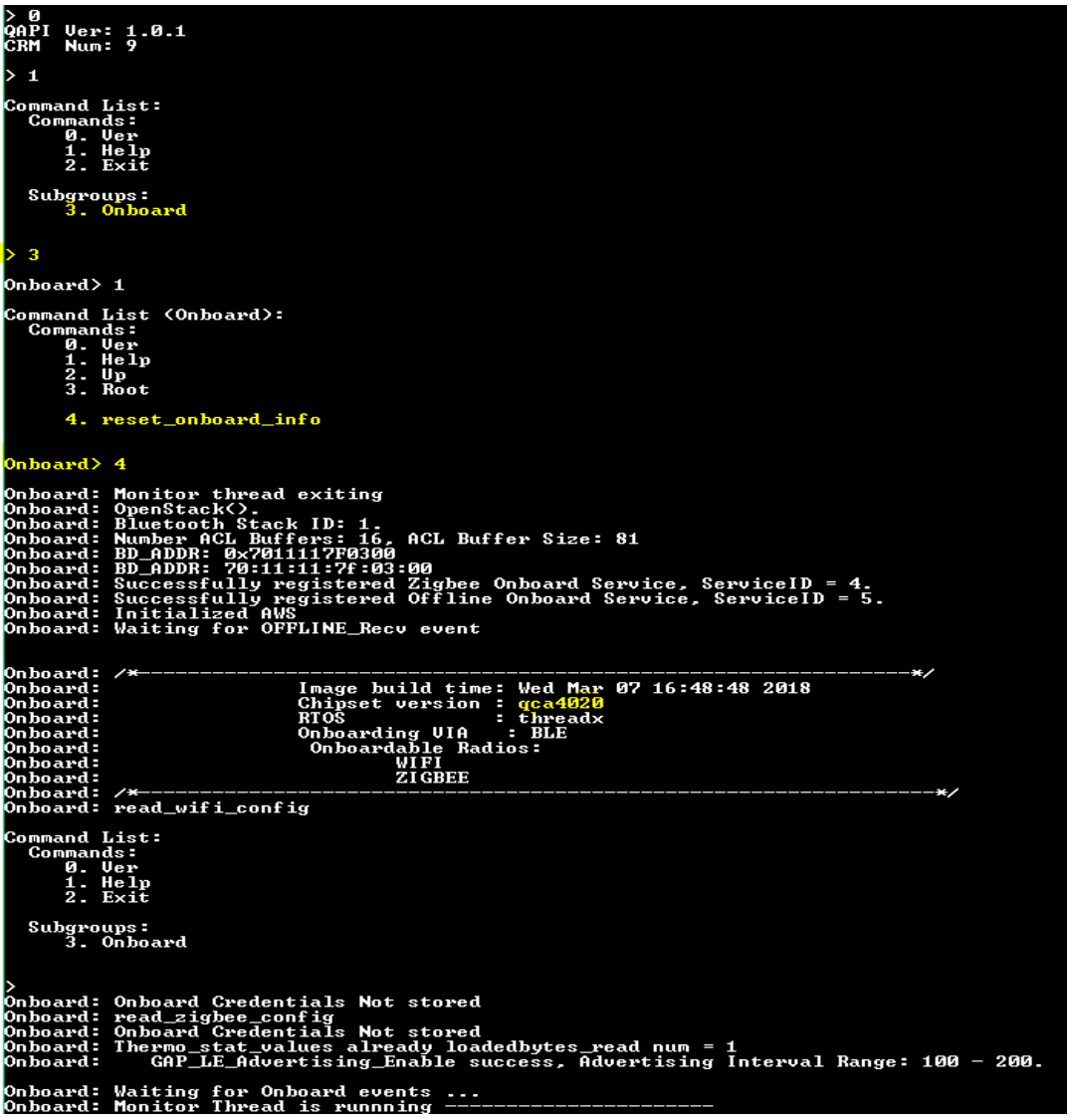


Figure 2-8 Reset onboard information on CDB20/24

2.3 Prerequisites to build in standalone/without cloud mode

Add the following jsmn files in CDB2x SDK under third party to parse the sensor data received from Thread border router/ Zigbee cooordinator. As AWS SDK contains jsmn, it is recommended to download AWS SDK for building both standalone/without cloud and with cloud mode.

1. Download AWS SDK from https://github.com/aws/aws-iot-device-sdk-embedded-C and extract it.
2. Create thirdparty/aws/awsiot/ folder under target directory.
3. Copy all files from aws-iot-device-sdk-embedded-C-3.0.1 folder to thirdparty/aws/awsiot/ folder.

2.4 Onboarding mobile application

There are two mobile applications available at **/target/mobileapp/android** – QCAOnboard and Sensor applications. The goal of these mobile applications is to onboard CDB devices to the user network as follows:

- Application scans for available devices; user should provide the credentials required to onboard the CDB.
- Application establishes connection to the board using Wi-Fi or BLE, based on the onboarding mode the device is configured.
- ADB tool can be used to install the application (on both Windows and Linux); Android device version 5.0 or higher is required.

The ADB tool (**adb install qca_onboarding.apk; adb install qca_sensors.apk**) can be used to install the application on both Windows and Linux.

3 Getting started

3.1 Overview of the QCA402x SDK

The QCA402x SDK contains sample demo applications that demonstrate usage of Qualcomm APIs (QAPI) to test chip features.

3.1.1 QCA402x SDK contents

```
target
|-- bin
|       |-- cortex-m0                //Cortex-M0 RAM Binary
|       |-- cortex-m4                //Cortex-M4 ROM symbol files
|       |-- wlan                     //WLAN firmware binary

|-- build
|       |-- scripts                  //Support scripts
|       |-- tools                    //Tools for image flashing, Device
                                       Config and so on.
|-- include                          //QAPI header files
|-- lib                              //Pre-compiled Qualcomm system
                                       Libraries and object files
|-- quartz
|       |-- demo
|       |      |-- Helloworld_demo    //Sample Helloworld app
|       |      |     |-- build/gcc    //GCC build scripts and Makefiles
|       |      |-- Onboard_demo       //Sample onboard AWS/Offline app
|       |      |-- QCLI_Demo          //Sample CLI-based app
|       |      |-- QCLI_power_demo    //Sample CLI-based app for
measuring power
|       |      |-- QCLI_uart_at_demo  //Sample UART AT command app
|       |      |-- Passthrough_demo   //External HCI bridge app
|       |-- platform                //Read-only version of platform
                                       Configuration files
|       |-- fwupgrade              //Source for firmware upgrade module
|       |-- ecosystem              //Source for ecosystem
|-- sectools                       //Qualcomm Image signing tools
```

3.1.2 Sample demo applications

The SDK contains sample demo applications with source code to demonstrate different features and technologies that QCA402x supports. Demos are in the **target\quart\demo*<name_of_demo>*** folder.

<name_of_demo>

Helloworld_demo: Demo application that can be used as reference to create new demo applications.

Onboard_demo: Onboard AWS/Offline demo applications provide a mechanism to demonstrate end to end communication between QCA402x and Mobile app through AWS cloud. QCA4020 connects to AWS through over Wi-Fi and MQTT connection. QCA4024 connects to AWS through Zigbee/Thread Mesh network bridged over by QCA4020.

QCLI_demo: CLI-based demo application that provides a mechanism to demonstrate different features and technologies that QCA402x supports. It also provides reference implementation and usage of customer facing QAPIs.

QCLI_power_demo: CLI-based demo application that provides a mechanism to demonstrate for measuring power.

QCLI_uart_at_demo: Demo application that provides the AT commands for exercising the functionality of the boards and provides the capability for any MCU with low memory foot print to use QCA402x over UART interface with AT commands defined.

- **Passthrough_demo:** This demo can be used to directly exchange HCI command and event between external test equipments and BLE firmware. This demo can be utilized for BLE Direct Test mode. For more details, see the BLE direct test mode section.

3.2 Supported toolchains

The SDK contains build scripts and makefiles for the GNU embedded toolchain for ARM-based processors.

- The toolchain supports Windows and Linux platform and can be downloaded from the ARM website at: https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads
- Supported version: 6.2
- Add the path to toolchain binaries to ‘PATH’ environment variable.

  Linux:

  ```
  export PATH=$PATH:/path/to/bin
  ```

  Example

  If ARM GNU toolchain is installed under `/home/linux/gcc-arm-none-eabi-6.2-2016-q4/bin`, set the path as:

  ```
  export PATH=$PATH:/home/linux/gcc-arm-none-eabi-6.2-2016-q4/bin
  ```

  Windows:

  ```
  set %PATH%=%PATH%:\path\to\bin
  ```

  Example

  If ARM GNU toolchain is installed under `C:\Program Files (x86)\GNU Tools ARM Embedded\6.2 2016q4\bin`, set the path as:

```
set %PATH%=%PATH%: C:\Program Files (x86)\GNU Tools ARM Embedded\6.2
2016q4\bin
```

3.3 Other requirements

- **Python:** Some of the support scripts are Python-based, Python 2.7.2, or higher.

  After installation, add the path to python.

  Example: If there is python.exe in `C:\CRMApps\Apps\Python276-64` folder, set path as follows: `set %PATH%=%PATH%:C:\CRMApps\Apps\Python276-64`

- **Eclipse IDE for C/C++:** GUI-based integrated development environment

  http://www.eclipse.org/downloads/eclipse-packages/

  Supported Version: Oxygen version - Release 4.7.2

- **Java:** Eclipse IDE has dependency on Java, JDK 8 or higher.

  After installation, add path to Java.

  Example: If Java.exe is present in C:\ProgramData\Oracle\Java\javapath, set path as follows:

  set %PATH%=%PATH%: C:\Program Files\Java\jdk1.8.0_161\bin

- **OpenOCD:** OpenOCD plug-in is required to establish the connection between Eclipse IDE and onboard FTDI JTAG debugger. The supported version is OpenOCD-20170609 available at http://gnutoolchains.com/arm-eabi/openocd/ .

  After extracting the folder, add the path to OpenOCD.

  Example: If there is openocd.exe in C:\ Program Files\OpenOCD-20170609\bin, set the path as follows:

  ```
  set %PATH%=%PATH%:C:\ Program Files\OpenOCD-20170609\bin
  ```

3.4 Third-party software

3.4.1 Real time operating system (RTOS)

The QCA402x mission mode firmware consists of system firmware and application firmware provided by Qualcomm. The system and application firmware depend on an underlying RTOS to schedule threads and manage memory, events, semaphores, timers, and more.

Qualcomm system firmware uses an RTOS abstraction layer called “QuRT™” which can be mapped on to various RTOS. The QuRT layer is tested with two popular RTOSes, ThreadX and FreeRTOS. The ThreadX RTOS is delivered in binary form as part of the SDK and is available for customers to use in QCA402x-based products. A QCA402x-compatible version of FreeRTOS can be downloaded from Linux Foundation Code Aurora Forum. The FreeRTOS binding for QuRT is also available for download from LF CAF.

Application programmers can choose to use *ThreadX* APIs (for a ThreadX-based QCA402x system) or *FreeRTOS* APIs (for a FreeRTOS-based QCA402x system) or *QuRT* APIs for RTOS support. If QuRT APIs are used, the application firmware remains *RTOS-independent*. It is also possible to use a different RTOS with QCA402x. One way to do this is to implement the mapping from QuRT APIs to the new RTOS, so that system firmware also uses the new RTOS.

3.4.2 Ecosystem

QCA402x supports these ecosystems:

- AWS: Amazon Web Services IoT provides secure, bidirectional communication between internet-connected things (such as sensors, actuators, embedded devices, or smart appliances) and the AWS cloud. The connection to the cloud is made using MQTT protocol over a secure TLS connection. See the Build with AWS IoT section to download AWS source code into thirdparty folder of the SDK release.

Azure: Microsoft Azure cloud platform provides a secure and scalable mechanism to connect IoT end nodes to the cloud. QCA402x SDK now supports Azure IoT device SDK. QCA402x device talks to the Azure cloud using secure MQTT protocol. For details, see the Build with Azure IoT SDK section.

3.5 Build sample applications

This section describes building sample demo applications with GNU GCC toolchain.

3.5.1 On Linux (or Cygwin)

- Configure build parameters.
  Configure the build parameters using the environment variables before building the demo.

Env. variable	Allowed values	Default value	Description
CHIPSET_VARIANT	qca4020 qca4024	qca4020	Chipset variant.
RTOS	threadx freertos	freertos	RTOS library.
BOARD_VARIANT	cdb carrier	carrier	Board variant. Select cdb option to build for CDB board
ECOSYSTEM	awsiot azure offline	none	Option to select ecosystem support

Example: The following commands configure the environment for QCA4024 ThreadX builds.

```
export CHIPSET_VARIANT=qca4024
export RTOS= threadx
export BOARD_VARIANT=cdb
export ECOSYSTEM=awsiot →Needed only for Onboard_demo with AWS support.
export ECOSYSTEM=offline →Needed for Onboard_demo with Offline support.
```

NOTE: A 64-bit Linux system is required to build Onboard_demo for AWS or Offline support. Before proceeding, follow the steps detailed in the Prerequisites to build onboard AWS demo section for the Onboard AWS demo.

1. Go to the Linux Makefile available under ***build/gcc*** directory in the demo directory.
   ```
   cd target\quartz\demo\<name_of_demo>\build\gcc
   ```

2. Install the device configuration files.

   `make prepare`

   The prepare command installs the device configuration files under the `src/export` directory. These device configuration files can now be edited to change system configuration. This command should only be run once unless the configuration files are deleted.

   For more details on device configuration feature, see Configuring an application section in the *QCA402x (CDB2x) Programmers Guide* (80-YA121-142).

3. Optionally, change the device configuration files.

4. Build the application.

   For Onboard_demo, run the command:

   `sh build.sh <Onboarding mode><Radio><Supported mode>`

   Example: `$ sh build.sh 1 1 1`

Parameter	Possible Values	Description
1 Onboarding Mode	1	WIFI Mode
	2	BLE Mode
2 Radio	1	THREAD
	2	ZIGBEE
3 Supported Mode Thread / Zigbee	1	Thread border router/Zigbee coordinator
	2	Thread joiner router/Zigbee end device

   For other demo applications, run "`make`"

5. The following operations are performed after running `make`:

   a. PropGen tool generates C files from device configuration xml files.

   b. GCC ARM compiler compiles the application source files and the device configuration source files.

   c. Linker script generation tool places all object files in specified memory regions. The output of this step is a linker script that contains memory placement details for all object files.

   d. Application object files are linked with precompiled Qualcomm system libraries. The output is the final executable in ELF format.

   e. If secure boot is enabled, the executable is signed by Qualcomm Image Signing Utility. If secure boot is not required, the image is postprocessed to add a hash section, which is required by the Primary Boot Loader.

      For more information on the steps, see Makefile.

   f. The final executable is now ready for flashing. The flashing procedure is described in the Flash the image section.

3.5.2 On Windows

Building the sample applications on Windows is similar to Linux. The Makefile is replaced by a window build script (`build.bat`).

1. Navigate to the location of build scripts.
   ```
   cd target\quartz\demo\<name_of_demo>\build\gcc
   ```
2. Install the device configuration files.
   ```
   build.bat prepare <chipset_variant> cdb
   ```
   The prepare command installs device configuration files under `src/export` directory. The device configuration files can now be edited to change system configuration. This command should only be run once unless the configuration files are deleted.

   For more details on device configuration feature, see "Configuring an application" section in the *QCA402x (CDB2x) Programmers Guide* (80-YA121-142).
3. Build the sample application. The configuration parameters can be passed as command-line parameters.

Parameter	Possible values	Description
1	F	FreeRTOS build
	T	ThreadX RTOS build
	prepare	Install Device Configuration files
	clobber	Delete Device Configuration files
2	4020	Chipset variant.
	4024	
3	c	Board Variant. Select cdb option to build for CDB board.
	cdb	
Optional parameters needed for Onboard_AWS_demo		
4	1	Wi-Fi Onboarding Mode
	2	BLE Onboarding Mode
5	1	THREAD
	2	ZIGBEE
6	1	Thread Border Router/ Zigbee Coordinator
	2	Thread Joiner Router/ Zigbee End Device

For the onboard demo, follow the procedure in the Prerequisites to build onboard AWS demo section, before continuing with the following steps:

```
build.bat <RTOS> <ChipSet> <Onboarding mode> <Radio> <Mode>
<Board_Variant>
```

Example

- For AWS on board demo: `build.bat t 4020 1 1 1` cdb
- For other demos: build.bat t 4020 cdb

This command builds the ThreadX version of the application for QCA4020 module.

4. The following operations are performed on running build.bat:
   a. PropGen tool generates C files from device configuration xml files.
   b. The GCC ARM compiler compiles application source files and device configuration source files.
   c. The linker script generation tool places all object files in specified memory regions. The output of this step is a linker script that contains memory placement details for all object files.
   d. Application object files are linked with precompiled Qualcomm system libraries. The output is the final executable in ELF format.
   e. If secure boot is enabled, the executable is signed by Qualcomm Image Signing Utility. If secure boot is not required, the image is postprocessed to add a hash section, which is required by the Primary Boot Loader.

   The Windows build scripts are available under *build/gcc* directory in under each demo directory.

3.5.3 Eclipse IDE

The QCA402x SDK contains a QCA402x project plug-in jar file to create applications and the Eclipse project files to build the sample demo applications.

3.5.3.1 Create a new application

1. Install the QCA plug-in jar file available at **/target/quartz/demo/EclipseSupportFiles.**
   - Copy the jar file (QCA402x_plugin.jar) to the dropin folder under the Eclipse IDE installed folder.
   - Restart the Eclipse IDE if running. To restart Eclipse, click on the **File** menu of Eclipse IDE and select the **Restart** menu item after a plug-in is installed.
2. Create a QCA402x project.
   a. Open the Eclipse IDE, go to the File menu, and select the options **File > New > Other.** The wizard opens.
   b. Select the QCA Chipset > QCA402x Project and click Next.
3. Configure the QCA402x project in the project wizard by providing the following details:
   - **Name:** Project name to be created
   - **Location:** Provide the SDK path, and navigate until SDK_source>/target/quartz/demo
   - **Selection of RTOS:** Choose either thread or FreeRTOS
   - **Chipset variant:** Choose the chipset variant either 4020 or 4024
   - **Board variant:** Choose the board variant as cdb
   - **Chipset revision:** Choose the chipset revision as 2p0
4. After providing the preceding information, click **Finish**.

3.5.3.2 Import sample demo applications

1. Install Eclipse project files for sample demo applications.

   To install Eclipse project files for sample demo applications, there is eclipseSupport.bat for Windows and eclipseSupport.sh for Linux in `<SDK_source>/target`.

2. Open the terminal and navigate to <SDK_source>/target, and run the “sh eclipseSupport.sh” or “eclipseSupport.bat” command.

   After executing the script, the Eclipse project files.cproject, project files, and the settings folder are updated in the respective folders of the demo applications.

3. Open the Eclipse application.

4. Go to File > Open Projects from File System and set import source for the demo application.

   Example: <SDK_source>/target/quartz/demo/QCLI_demo

   The user should be able to see the import source as Eclipse project.

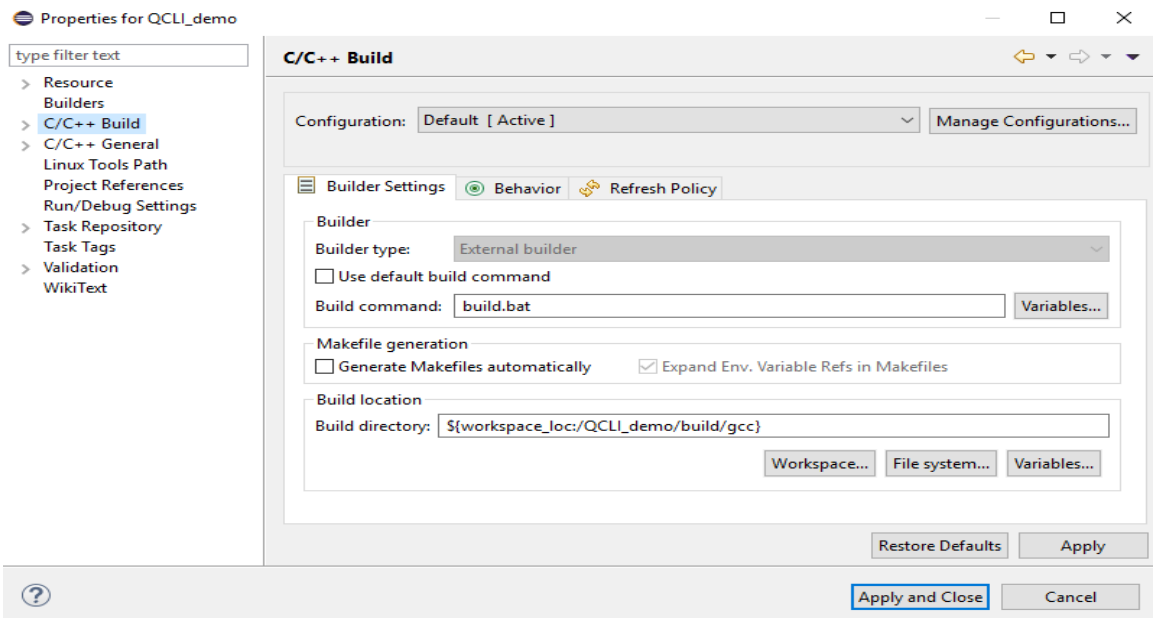
5. Click **Finish**.

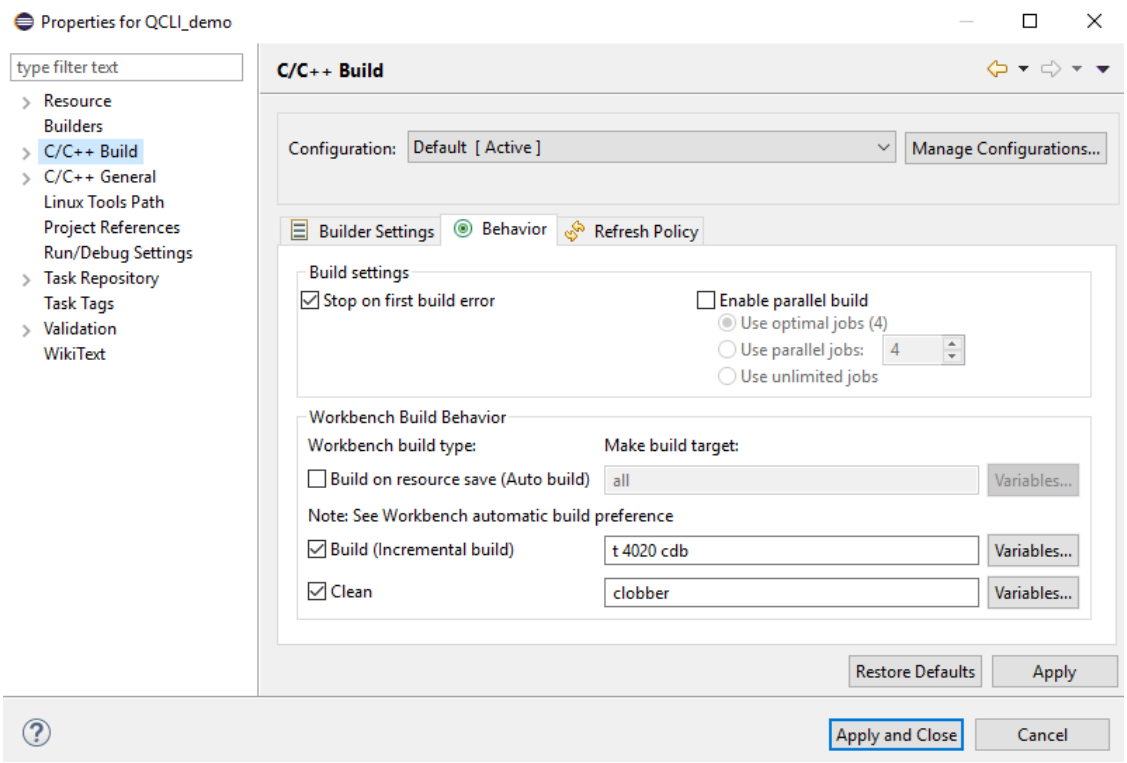
3.5.3.3 Build an application

1. Go to the Project properties > Configure build setting.
2. Right-click on the project name in project explorer and go to **Properties**.
3. Verify the build command and the build directory in the **Builder Settings** tab.

For Windows

1. Go to C/C++ build and set the build command to “build.bat”, and the build directory to “path to source”.
2. The following figure shows the build and clean command:

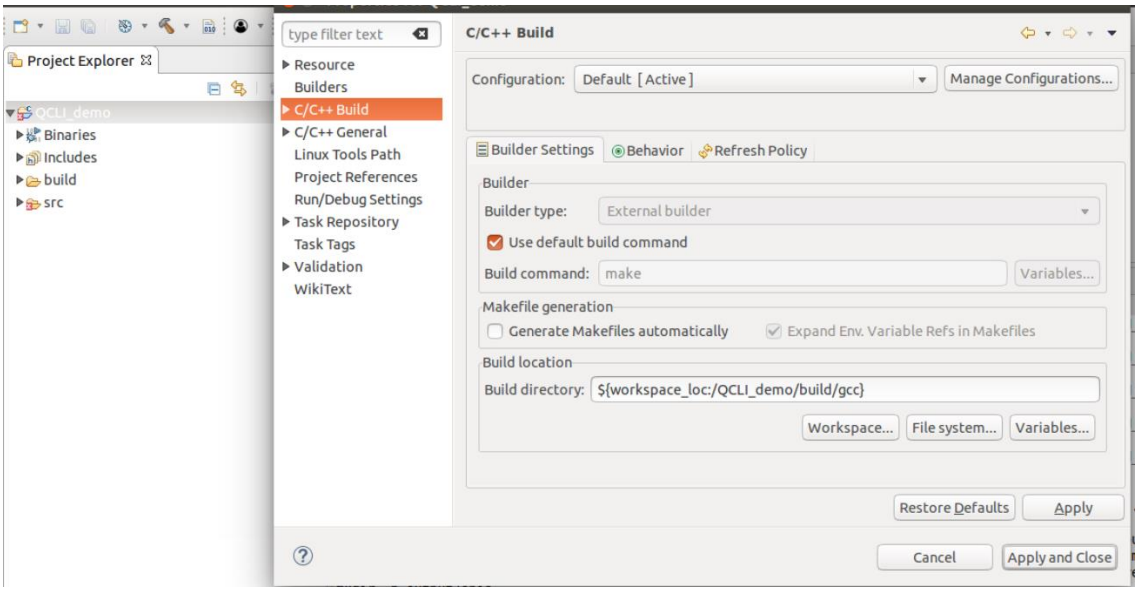


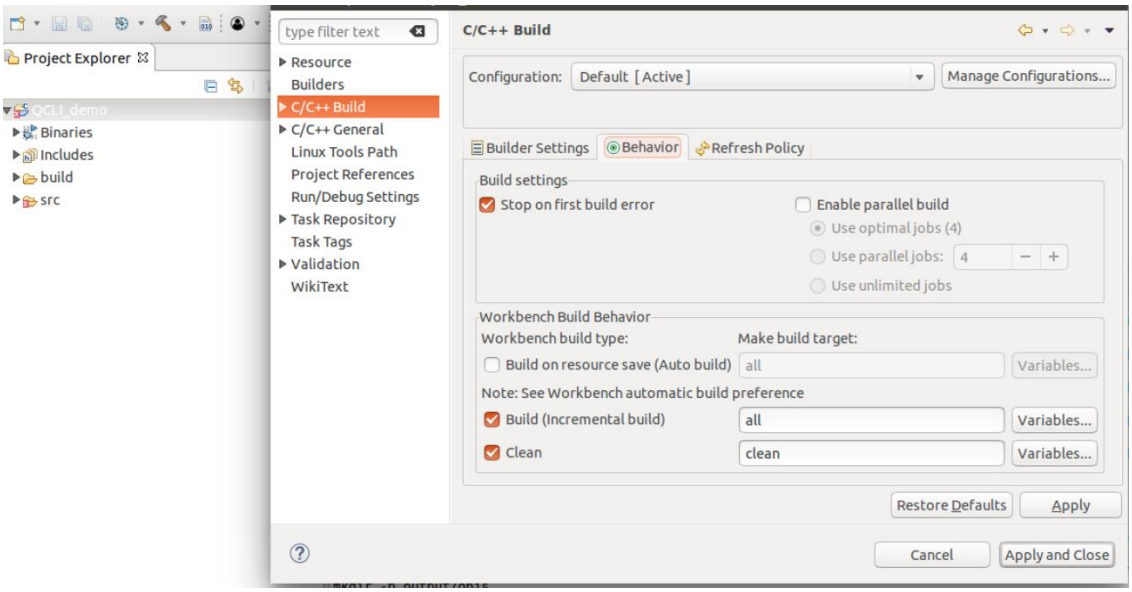


3. Go to C/C++ Build >Environment.
4. Set the **PATH** variable values to match the Windows “PATH environmental variables”.
5. Click Apply and Close.

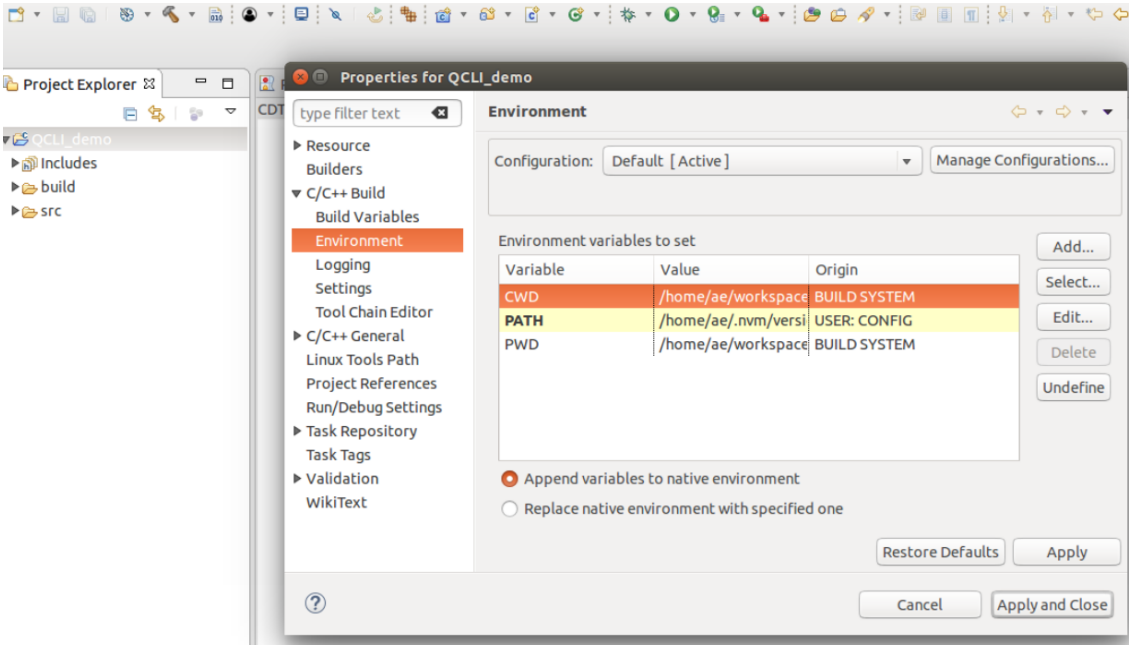
For Linux

1. Go to **C/C++ Build**, make sure the build command is set to **make**.

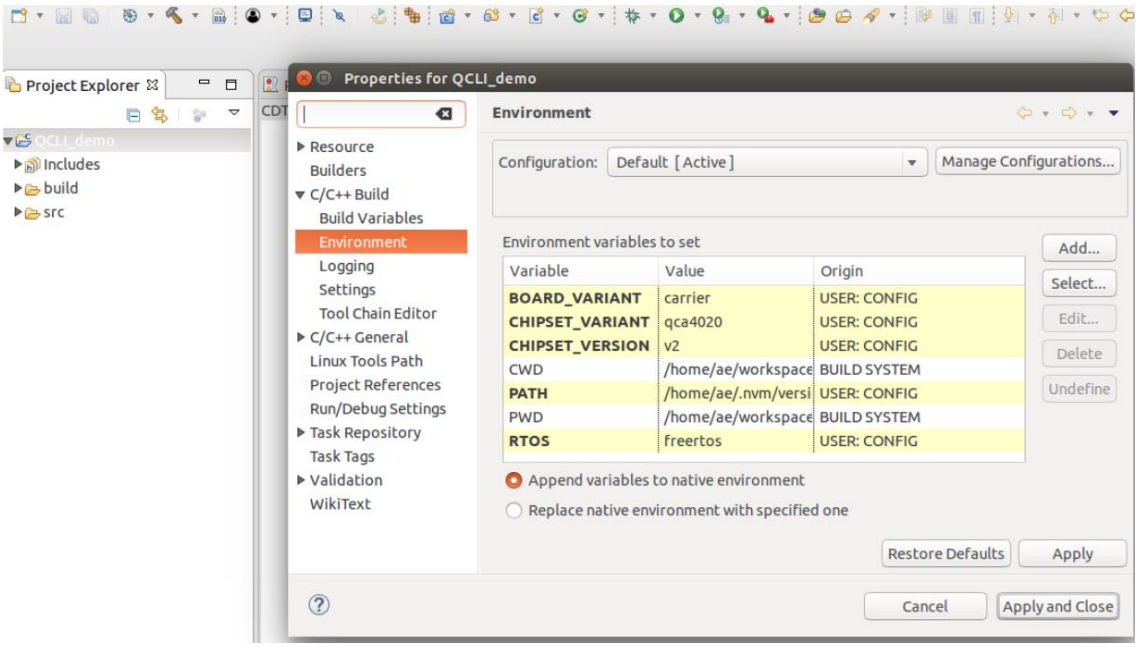




2. Go to **C/C++ Build >Environment**. Select ADD and enter the variable name and value.
3. Click **OK** to save the variable for project build.



These variables are used for building the project. For example, the variable RTOS is used to set thread or FreeRTOS as the target platform.



4. Click build button on the top menu or right-click on project name in Project Explorer window and click on **Build Project** to build the images.
5. To clean the project, right-click on the project name and select **Clean Project**.
6. To flash the images built, currently there is no support from Eclipse IDE. Use the command terminal to flash the device.

3.6 Flash the image

The SDK contains python-based tools and scripts that programs the images to flash. Flash the images using JTAG or using USB. The following images are flashed:

- Quartz_Hashed.elf : Sample application image for Cortex-M4. This can be replaced by OEM image.
- ioe_ram_m0_threadx_ipt.mbn: Cortex-M0 RAM image.
- wlan_fw_image.bin: Optional WLAN image (for QCA4020 only).

3.6.1 Flash the image using JTAG

The SDK contains scripts that use OpenOCD to interact with the Cortex-M4 core on the module.

Board setup

1. Set up the CDB2x board as described in Appendix, CDB2x board setup.
2. Put the jumper on J31 1&2 on the CDB2x board. After flashing through JTAG is done, remove the jumper.

3.6.1.1 Flashing on command line

Navigate to the location: `<SDK_source>/target/quartz/demo/QCLI_Demo/build/gcc`

On Windows

Run "`flash_openocd.bat`". This command opens an instance of OpenOCD, arm-gdb client, connect to the target CPU over JTAG and download a flash loader program, which will then write the firmware images to flash.

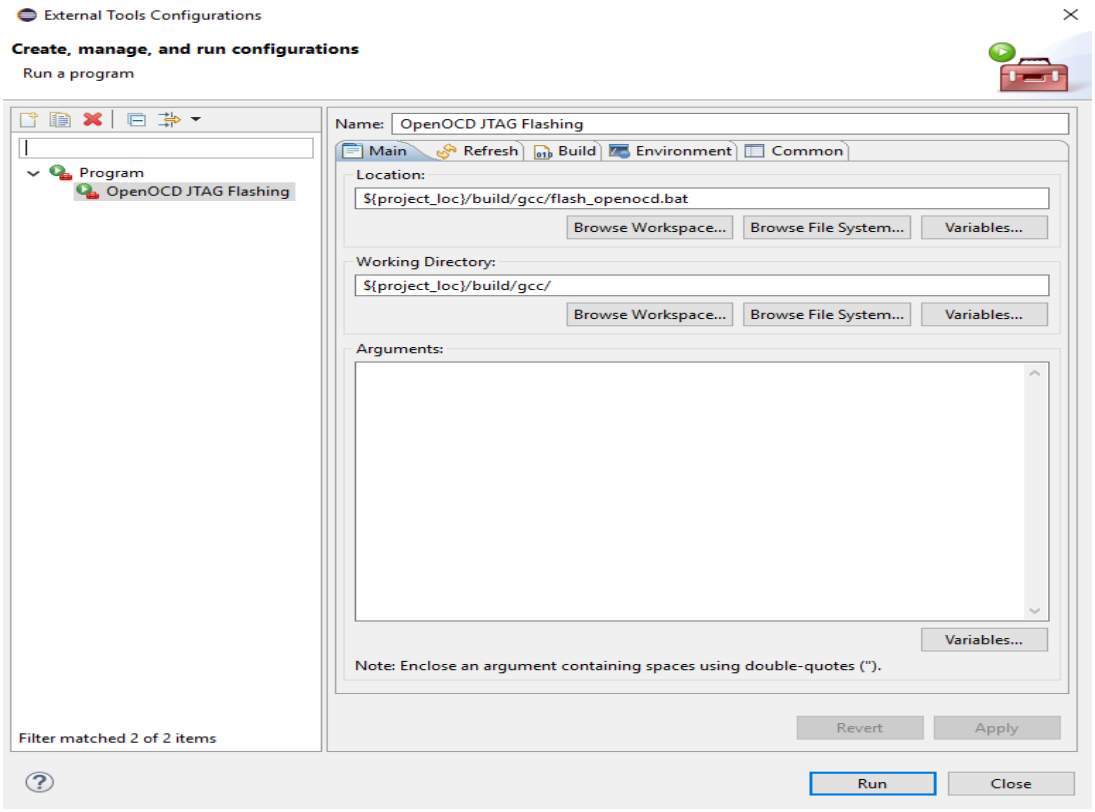
On Linux/Cygwin

Run "`flash_openocd.sh`". This command opens an instance of OpenOCD, arm-gdb client, connect to the target CPU over JTAG and download a flash loader program, which will then write the firmware images to flash.

3.6.1.2 Flashing with Eclipse IDE

1. Open Eclipse IDE and navigate to Run > External Tools > External Tools Configuration.
2. Click the program, create a new launch configuration, and update the paths as shown in the following **Main** tab.

   Provide the path for "`flash_openocd.bat`" or "`flash_openocd.sh`" in the location box.



3. To flash the program, click **Run > External Tools > OpenOCD JTAG Flashing**. The flashing logs can be seen on the Eclipse console.

3.6.2 Flash the image using USB

The QCA4020 SDK contains a python-based tool called qflash.py that allows flashing images over USB. The tool is available at- `target\build\tools\flash`. If the tool is invoked without any optional parameters, and does the following:

1. Generate a default firmware descriptor table.
2. Generate a default partition table.
3. Flash the default sample application elf files to the flash.

OEMs can use the optional parameters to tweak the descriptor tables based on their requirements.

```
qflash.py --help
usage: qflash.py [-h] [--comm COMM] [--app APP] [--nogen] [--nodev] [--
debug]
Use Emergency Download (EDL) Mode to program flash over USB.
optional arguments:
  -h, --help            show this help message and exit
  --comm COMM, --comm_port COMM
                        Specify QDLoader COM port number. Alternative:
set COMM_PORT environment variable.
  --app APP             Specify the path and/or name of an Application
Image to run on M4
  --nogen               Suppress generation of Partition and FWD tables
  --nodev               Do not access device; do not actually program
flash
  --debug, --verbose    Enable debug messages
```

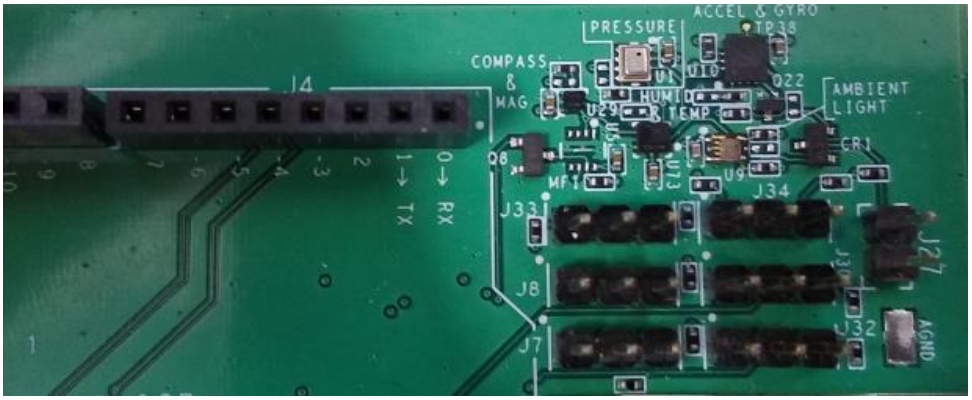
Prerequisites

- Connect a Micro USB cable between port J6 on the CDB2x and a USB port on a Windows PC.
- Install Qualcomm USB drivers and choose “Ethernet” option in the installation process.

3.6.2.1 Flashing default images over USB

Install J34 Jumper (Pins 1-2) on the front of CDB2x board and power cycle the system. This puts the chip in Emergency Download (EDL) mode. Look at the Windows Device Manager to confirm that the chip has entered EDL mode.

In EDL mode, look under "Port (COM & LPT)" and see an entry for "QDLoader".
Example: `Qualcomm HS-USB QDLoader 9008 (COM18)`. If "QDLoader" is not seen in Device Manager, it means that QCA402x is not in EDL mode. Ensure that the Micro USB cable is connected, and Qualcomm USB drivers are installed.



1. Navigate to the application build directory.

   Example: `cd target\quartz\demo\<name_of_demo>\build\gcc`

2. Run the following command to flash the image:
   python ..\..\..\..\..\build\tools\flash\qflash.py --comm XX

Flashing default images and file system image (optional)

1. Follow steps 1 and 2 from the preceding section, Flashing default images over USB.
2. Set following environment variables for qflash.py to create generated_partition_table.xml that contains primary filesystem(FS1) with a file system binary- ‘fs.bin’ and a secondary filesystem partition(FS2):

   ```
   Set FS1IMG=fs.bin
   Set FS2IMG=KEEP
   ```

3. Use `qflash.py` to flash the image:

   ```
   python ..\..\..\..\..\build\tools\flash\qflash.py --comm XX
   ```

NOTE: FS1IMG is a file system image used for the primary file system. The default value is ERASE.
"ERASE" is used to erase the current contents of the primary file system area.
"KEEP", is used to retain the current flash contents of the primary file system area.
FS1SZ is the size, in KB, used for the primary file system, the default size is 64 KB.
The size can be increased (must be multiple of 4 KB) by setting as “set FS1SZ=128”.

4. Remove jumper at J34. The board is now ready.

3.7 Run the application

3.7.1 Autoboot mode

The Autoboot mode refers to the mode where the system boots up on a power cycle and no JTAG connectivity is required. To run in autoboot mode, remove J34 and J31, and power cycle the system. For QCLI Demo, if the UART is connected and a serial console is open using a port setting of 115200, 8, n, 1, a CLI menu appears. User can now type commands to test the system features.

For more information on board setup, see Appendix, CDB2x board setup.

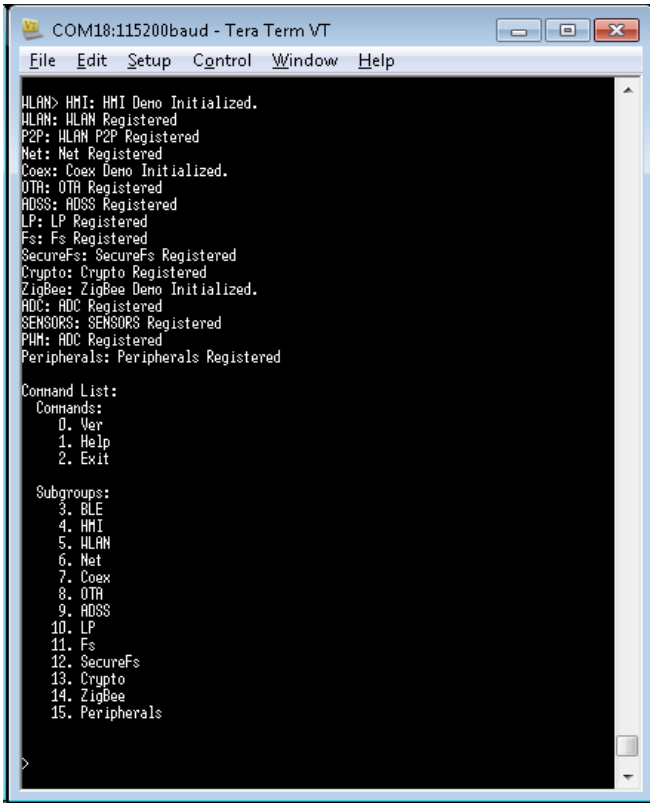


Figure 3-1 CLI menu in QCLI_Demo

3.7.2 JTAG debug mode

Optionally, an OEM can use JTAG to debug the application code. It is highly recommended to disable deep-sleep mode before using JTAG. After the processor enters deep-sleep, the JTAG is powered down and gdb connection will be terminated. The following steps can disable sleep and allow boot using JTAG.

1. Edit device configuration file to disable sleep:

   target/quartz/demo/XXX_Demo/src/export/DevCfg_master_devcfg_out_cdb.xml

2. Change the Sleep Driver setting as follows:

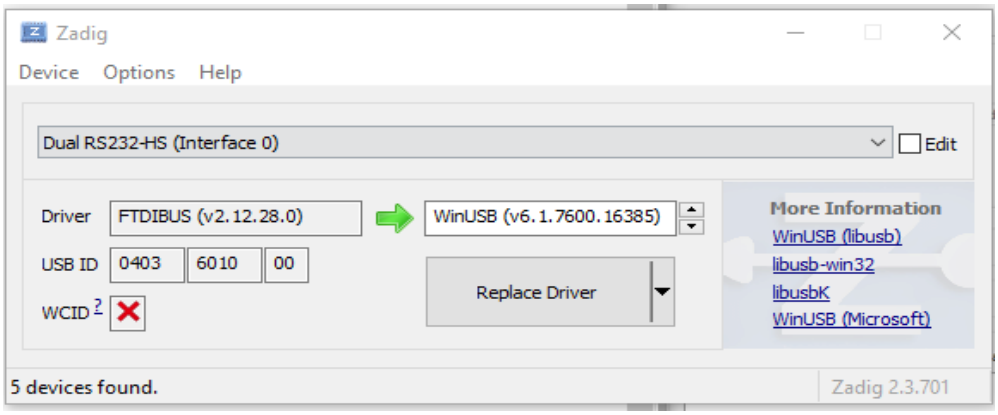
```
<driver name="Sleep">
  <global_def>
    <var_seq name="devcfgSleepData" type="0x00000003">
        0, 0, 0,
        180, 166, 200,
        end
    </var_seq>
  </global_def>
  <device id="0x02000018">
    <props id="0x1" oem_configurable="false" type="0x00000014">
devcfgSleepData </props>
    <props id="0x2" oem_configurable="false" type="0x00000002"> 0
</props>
    <props id="0x3" oem_configurable="false" type="0x00000002"> 632
</props>
    <props id="0x4" oem_configurable="false" type="0x00000002"> 96
</props>
  </device>
</driver>
```

3. Build the application. Follow the procedure from sections Build sample applications (gcc) and Eclipse IDE (Eclipse).
4. Flash the newly built application (Follow the procedure from section Flash the image).
5. Reset the board after flashing the image.

3.7.2.1 Install FTDI driver for JTAG

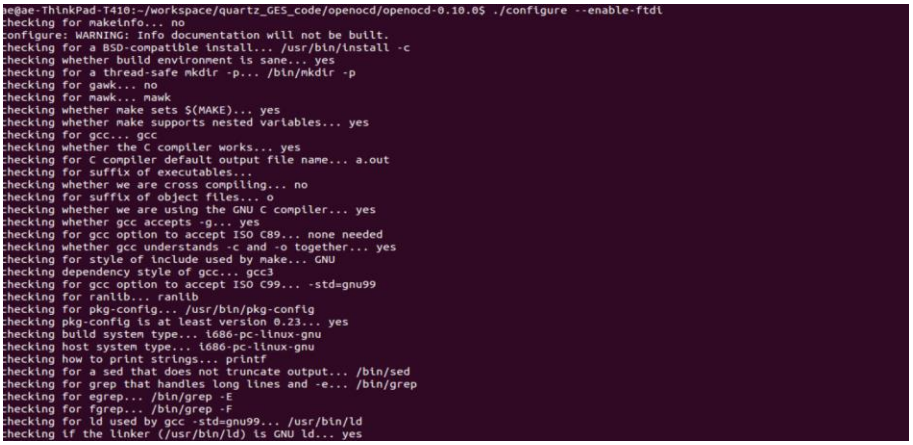
For Windows

- Download zadig application from http://zadig.akeo.ie/. Connect the board (CDB2x) and the two COM ports (that is, COM 15 and COM 16) are shown in the Device Manager. The lower port number (COM 15) is for JTAG and the upper port number (COM 16) is for serial connection.
- Run zadig.exe file and go to **Options > List All devices** and select the device in the drop down. Dual RS232-HS (interface 0) represents lower port number (COM 15) and dual RS232-HS (interface 1) represents lower port number (COM 16). The following example shows Select Dual RS232-HS (interface 0) to install FTDI on the WinUSB driver for JTAG interface.



For Linux

To enable FTDI driver support on openocd, open a terminal, navigate to the openocd-0.10.0 folder, and issue the command. . `/configure -enable-ftdi`



1. Connect the CDB2x device using USB and two serial ports are available:
   - **/dev/ttyUSB0**: Lower port is for JTAG
   - **/dev/ttyUSB1**: Upper port is for serial connection.
2. Open a serial console terminal on the port /dev/ttyUSB1.

3.7.2.2 Debugging through GDB

1. For JTAG debug mode, put the jumper on J31 1&2 on the CDB2x board and reset the board.
2. Run OpenOCD server and check that JTAG tap detected.
3. Run the `openocd -f qca402x_openocd.cfg` command in the first command prompt.

```
GNU ARM Eclipse 64-bits Open On-Chip Debugger 0.10.0-dev-00287-g85cec24-dirty (2016-01-10-10:13)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
adapter speed: 5000 kHz
Info : clock speed 5000 kHz
Info : JTAG tap: QM4.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
Info : JTAG tap: auto0.tap tap/device found: 0x300220e1 (mfg: 0x070, part: 0x0022, ver: 0x3)
Warn : AUTO auto0.tap - use "jtag newtap auto0 tap -irlen 11 -expected-id 0x300220e1"
Info : QM4.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: QM4.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
Info : JTAG tap: auto0.tap tap/device found: 0x300220e1 (mfg: 0x070, part: 0x0022, ver: 0x3)
Warn : Only resetting the Cortex-M core, use a reset-init event handler to reset any peripherals or configu
re hardware srst support.
QM4.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x000001c0 msp: 0x10000154
    TargetName         Type       Endian TapName            State
--  ------------------ ---------- ------ ------------------ ------------
 0* QM4.cpu            cortex_m   little QM4.cpu            halted
```

4. Run `arm-none-eabi-gdb -x v2\quartzcdb.gdbinit` command in the second command prompt. This command loads the symbols from the image and sets appropriate breakpoints. The execution stops at system software entry point. User can optionally set breakpoints in application code.

The following example shows user sets breakpoints at `app_init`, `app_start`, and `QCLI_Thread` in QCLI demo application code.

```
warning: Loadable section "ROM_MOM_MISSIONROM_RW_REGION__rw-data" outside of ELF segments
warning: Loadable section "ROM_SOM_MISSIONROM_RW_REGION__rw-data" outside of ELF segments
warning: Loadable section "ROM_STDLIBS_RW_REGION" outside of ELF segments
warning: Loadable section "ROM_FOM_MISSIONROM_RW_REGION__rw-data" outside of ELF segments
Breakpoint 2 at 0x100a05e4: file /local/mnt/workspace/CRMBuilds/CNSS_W.QZ.1.1-00019-QZHW-1_20180323_125856/b/ioesw_proc/core/boot/sb
l//sbl1_mc.c, line 329.
Breakpoint 3 at 0x10fbc: file /local/mnt/workspace/CRMBuilds/CNSS_W.QZ.1.0.r4-00013-QZFPGA-1_20170509_214818/b/ioesw_proc/core/v2/ro
m/drivers/debugtools/err/src/apps_proc/arm//err_jettison_core_m4.s, line 83.
(gdb) b app_init
Breakpoint 4 at 0x1005a3d0: file ..\..\src\qcli\pal.c, line 384.
(gdb) b app_start
Breakpoint 5 at 0x1005a478: file ..\..\src\qcli\pal.c, line 423.
(gdb) b QCLI_Thread
Breakpoint 6 at 0x1005a1ac: file ..\..\src\qcli\pal.c, line 323.
(gdb) c
Continuing.

Breakpoint 4, app_init (ColdBoot=0x1) at ..\..\src\qcli\pal.c:384
384     {
(gdb)
Continuing.

Breakpoint 5, app_start (ColdBoot=0x1) at ..\..\src\qcli\pal.c:423
423     {
(gdb)
Continuing.

Breakpoint 6, QCLI_Thread (Param=0x0) at ..\..\src\qcli\pal.c:323
323     {
(gdb) list
318
319        It will finish initialization of the sample and then function as a
320        receive thread for the console.
321     */
322     static void QCLI_Thread(void *Param)
323     {
324        uint32_t CurrentIndex;
325
326
327        /* Display the initialize command list. */
```

Useful gdb commands:

- backtrace: shows a stack backtrace of the current thread
- x/50wa $sp: decodes the stack
- x/10xw 0x10000000: dump 10 words starting at given address, say, 0x10000000
- info all-registers: show all register contents
- x/10i $pc: displays a sequence of 10 instructions at an address

The QCA402x SDK provides gdb support macros for thread management with ThreadX at `target/quartz/gdb/gdb.threadx.thread`

- source <sdk_root>/target/quartz/gdb/gdb.threadx.thread: import the macros
- bpat: brief print ALL threads information like address, name, and priority
- btthread <thread_address>: backtrace for the specified thread

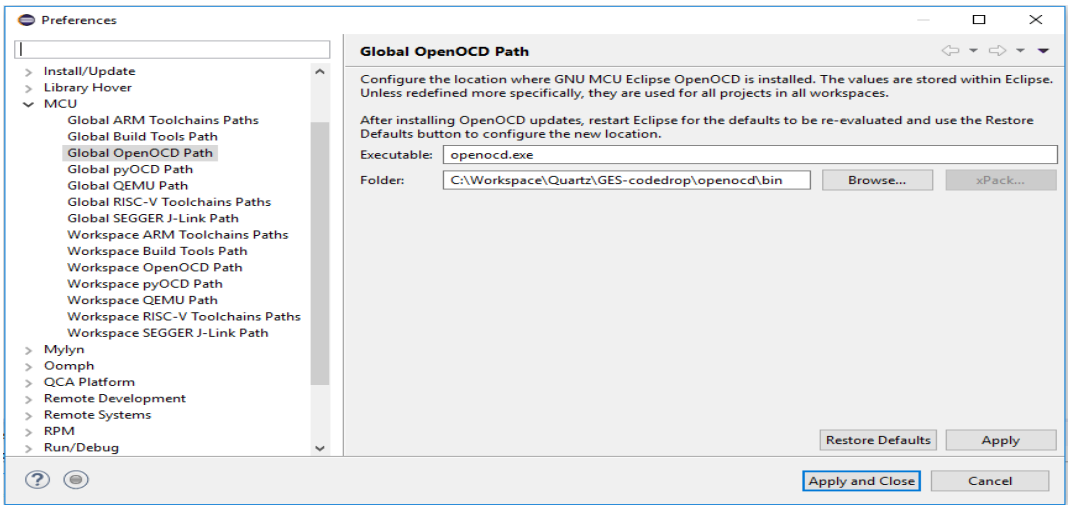
```
Breakpoint 4, QCLI_Thread (Param=0x0) at ..\..\src\qcli\pal.c:323
323     {
(gdb) bpat
Ready Threads:
Priority 10 Ready Threads:
thread = 0x10096f58    name = QCLI Thre    prio = 10 *
Priority 16 Ready Threads:
thread = 0x10087ec8    name = DSR          prio = 16
Priority 31 Ready Threads:
thread = 0x10087af8    name = IDLE         prio = 31


Stopped threads:
thread = 0x10084208    name = root         prio = 7
thread = 0x10088758    name = ATS_TIMER    prio = 12
thread = 0x1008a188    name = SIGNAL       prio = 16
thread = 0x1008ae30    name = BULK         prio = 16
thread = 0x1008b4a0    name = PLAT_CAL_    prio = 29
thread = 0x1008cf80    name = rfs_serve    prio = 10
thread = 0x1008d5a8    name = QIPC         prio = 20
thread = 0x10092928    name = netmain      prio = 18
thread = 0x10093218    name = nettick      prio = 3
thread = 0x10095288    name = post_ssl_    prio = 12
thread = 0x10095638    name = ssl_task     prio = 3
```

3.7.2.3 Debugging through Eclipse

This section briefs about the OpenOCD plug-in usage with Eclipse.

For more details, see https://gnu-mcu-eclipse.github.io/debug/openocd/

1. Install GNU MCU plug-in for Eclipse.
2. Go to **Help > Install new Software** and in the work with window
3. Select/type “GNU MCU Eclipse Plug-ins - http://gnu-mcu-eclipse.netlify.com/v4-neon-updates“ and click **Enter**.
4. Select the **GNU ARM** tools and click **Finish** to install.
5. Set openOCD path:
6. Go to Window > Preferences > MCU > Global OpenOCD path and set the openOCD path
7. Click Apply and Close.



Debugging a project:

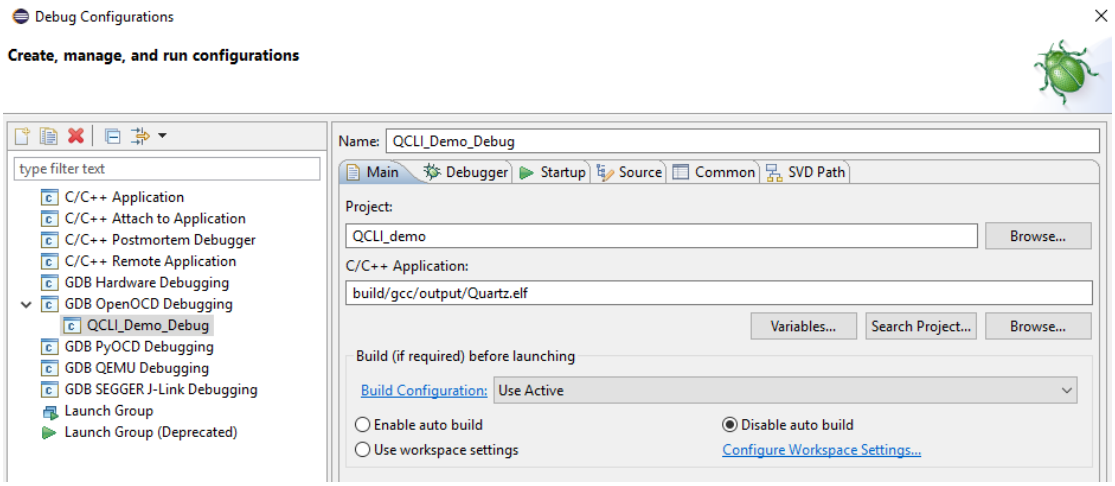
a. Go to Run >Debug Configurations > GDB OpenOCD Debugging and set the Application, OpenOCD options as show in the following images.

b. Select the Project in the project window.

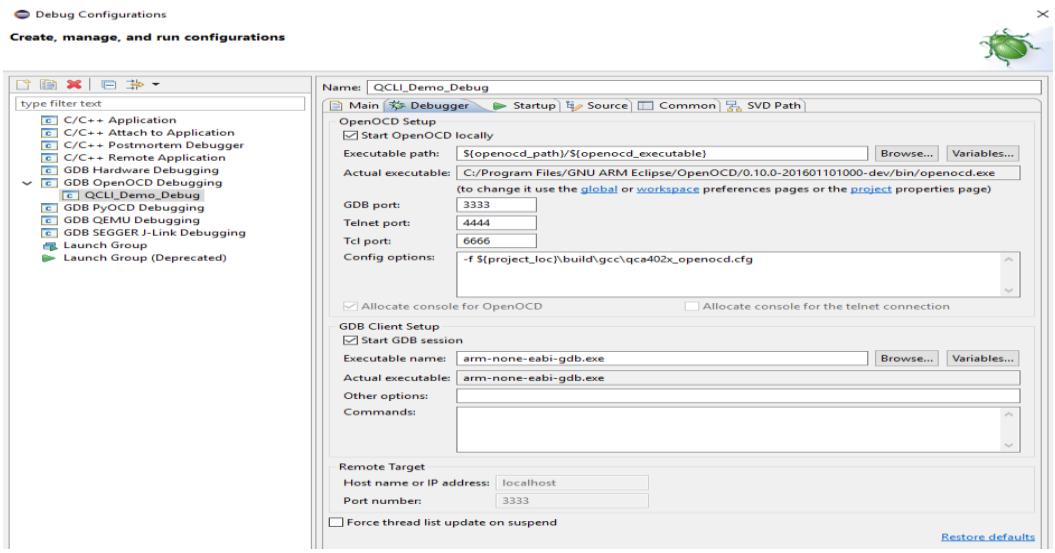
c. Select the Quartz.elf as the **C/C++** application.

   i If the the image is already built, select **Disable auto build**.

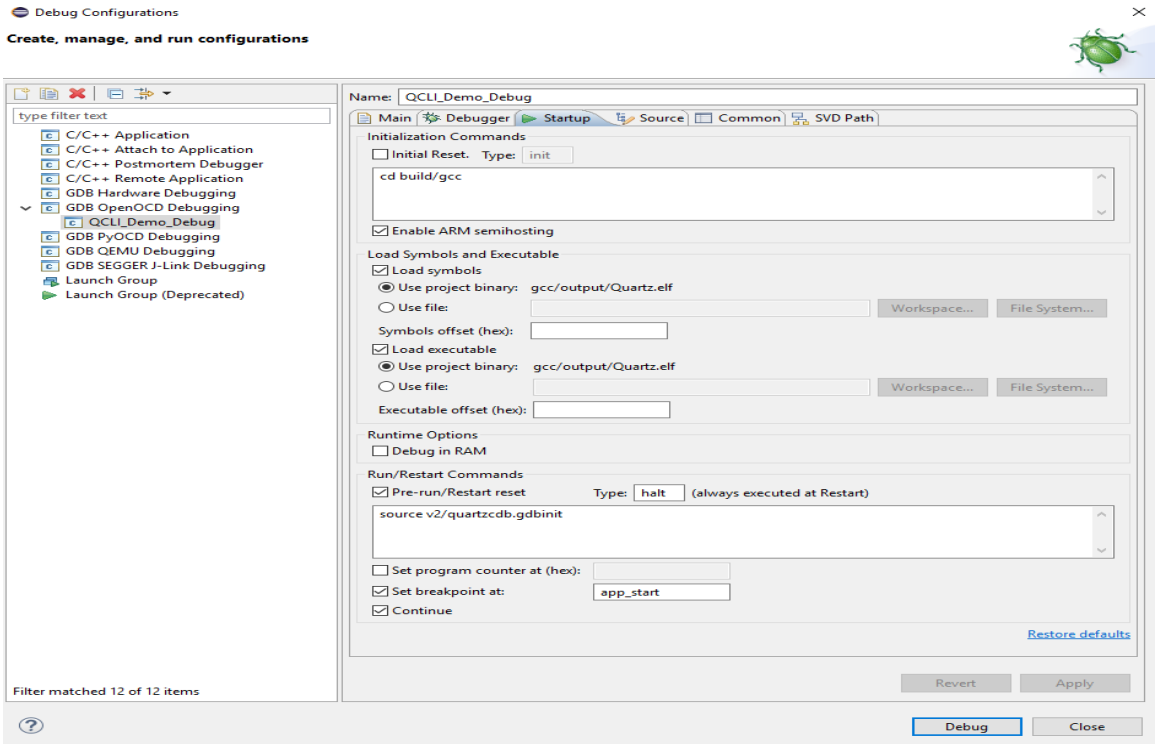
   ii If image is not built, select **Use workspace settings**.



d. In the Debugger window, provide the openocd executable path with config option, **-f ${project_loc}\build\gcc\qca402x_openocd.cfg**". Provide the **arm-none-eabi-dgb** executable path for the GDB client.



e. In the Startup window, set the **Initialization Commands** as “cd build/gcc” and set **Run/Restart Commands** as “source v2/quartzcdb.gdbinit”. **Set breakpoint at** “app_start” and click **Apply**.



f. Go to **Run >Debug Configurations** and click **Debug**. The first breakpoint hits when the debug starts. In the debugger console window, type **c** and press Enter. Next break point will be at app_start. Enter **c** in the debugger window to continue. With this the application must be able to run.

3.8 FreeRTOS and QuRT

3.8.1 Download FreeRTOS and QuRT source

This release includes:

- AWS FreeRTOS derived from https://github.com/aws/amazon-freertos.
- QuRT
- Some special files that need extra get from amazon website like PKCS. The files are available at https://github.com/aws/amazon-freertos/tree/master/lib/third_party/pkcs11.

  These PKCS header files should be placed at lib/third_party/pkcs11/.
- A script (demos/qcom/qca402x/common/application_code/install_pkcs_headerfiles.sh) that can be run in Linux shell or Cygwin (windows).

3.8.2 Build FreeRTOS and QuRT libraries

Makefiles and build.bat are provided to build the libraries for FreeRTOS and QuRT. Build.bat is for Windows environment and makefile is for Linux environment. The GNU tools for ARM embedded processors are required while building libraries.

3.8.2.1 Build QuRT library

1. Go to the "thirdparty/QuRT/2.0" directory.
2. For Linux, run the Makefile using the `make all` command.
   For Windows, run build.bat.

   It builds the **QuRT.lib** and **QuRT_shared.lib** in the output folder, **./QuRT/FreeRTOS/2.0/output**.
3. Copy the newly generated libraries to **target/lib/cortex-m4IPT/freertos**.

3.8.2.2 Build AWS FreeRTOS library

1. Go to the "thirdparty/aws_freertos/demos/qcom/qca402x/common/application_code" directory.
2. For Linux, run the Makefile using the `make all` command.
   For Windows, run build.bat.

   It builds the **freertos.lib** library in the output folder:
   /thirdparty/aws_freertos/demos/qcom/qca402x/common/application_code /output
3. Copy the newly generated library to **target/lib/cortex-m4IPT/freertos.**

3.9 Build ecosystem

3.9.1 Build with AWS IoT SDK

The Amazon Web Services IoT provides secure, bidirectional communication between internet-connected things (such as sensors, actuators, embedded devices, or smart appliances) and the AWS cloud. The connection to the cloud is made using MQTT protocol over a secure TLS connection.

This section describes how to build AWS demo. GNU Embedded toolchain 6.2 or higher is required.

1. Download AWS SDK from https://github.com/aws/aws-iot-device-sdk-embedded-C and extract it.
2. Create thirdparty/aws/awsiot/ folder under target directory.
3. Copy all files from aws-iot-device-sdk-embedded-C-3.0.1 folder to thirdparty/aws/awsiot/ folder.
4. On Windows, add the following environment variables to set path to libraries.

   ```
   set TOOLLIBPATH=path\to\lib\gcc
   set NEWLIBPATH=path\to\arm-none-eabi\lib
   ```

   Example: If ARM GNU toolchain is installed under `C:\Program Files (x86)\GNU Tools ARM Embedded\6.2 2016q4\`, set path as follows.

   TOOLLIBPATH=`C:\Program Files (x86)\GNU Tools ARM Embedded\6.2 2016q4\lib\gcc\arm-none-eabi\6.2.1\thumb\v7e-m`

   NEWLIBPATH=`C:\Program Files (x86)\GNU Tools ARM Embedded\6.2 2016q4\arm-none-eabi\lib\thumb\v7e-m`

   On Linux, add the following environment variables to set path to libraries:

   ```
   export TOOLLIBPATH=path/to/lib/gcc
   export NEWLIBPATH=path/to/arm-none-eabi/lib
   ```

   Example: If ARM GNU toolchain is installed under `/home/linux/gcc-arm-none-eabi-6.2-2016-q4/`, set path as follows.

   export TOOLLIBPATH=`/home/linux/gcc-arm-none-eabi-6.2-2016-q4/lib/gcc/arm-none-eabi/6.2.1/thumb/v7e-m`

   export NEWLIBPATH=`/home/linux/gcc-arm-none-eabi-6.2-2016-q4/arm-none-eabi/lib/thumb/v7e-m`

3.9.1.1 Create IAM users (Console)

To create one or more IAM users from the AWS management console:

- Sign in to the AWS management console and open the IAM console at https://console.aws.amazon.com/iam/.

CAUTION: In the navigation pane, choose **Users** and then choose **Add user**.

CAUTION: Type the user name for the new user. This is the sign-in name for AWS. To add more than one user at the same time, choose **Add another user** for each additional user and type their user names. Up to 10 users can be added at one time.

NOTE: User names can be a combination of up to 64 letters, digits, and these characters: plus (+), equal (=), comma (,), period (.), at sign (@), and hyphen (-). Names must be unique within an account. They are not distinguished by case.
For example, create two users named *TESTUSER* and *testuser*.

CAUTION: Select the type of access this set of users will have. Select programmatic access, access to the AWS Management Console, or both.

- Select **Programmatic access** if the users require access to the API, AWS CLI, or Tools for Windows PowerShell. This creates an access key for each new user. View or download the access keys when the **Final** pageis received.

CAUTION: Choose Next: Permissions.

CAUTION: On the **Set permissions** page, specify how to assign permissions to this set of new users. Choose one of the following three options:

- **Create a New Group:** Choose this option to create a new group as "**qca_testing_group**" and attached the following policies and click on review to view policies attached.

NOTE: The policies required for the AWS IoT dashboard are:

- AWSIoTFullAccess
- AmazonCognitoPowerUser

- **Add user to group**. Choose this option to assign the appropriate permission policies created to the users. IAM displays a list of all currently defined groups, along with their attached policies. You can select one or more existing groups. Choose **"qca_testing_group".**

CAUTION: Choose **Next: Review** to see all the choices made up to this point. When you are ready to proceed, choose **Create user**.

CAUTION: To view the users' access keys (access key IDs and secret access keys), choose **Show** next to each password and secret access key to see. To save the access keys, choose **Download .csv** (for example, credentials.csv) and then save the file to a safe location.

CAUTION: This is the only opportunity to view or download the secret access keys and provide this information to the users before they can use the AWS API. Save the user's new access key ID and secret access key in a safe and secure place. **Access to the secret key(s) is not provided again after this step**.

CAUTION: Provide each user with their credentials. On the final page choose **Send email** next to each user. The local mail client opens with a draft that can be customized and send. The email template includes the following details to each user:

- User name
- URL to the account sign-in webpage. Use the following example, substituting the correct account ID number or account alias:

  `https://AWS-account-D(or)alias.signin.aws.amazon.com/console`

For more information, see How IAM Users Sign in to AWS.

CAUTION: The user's password is not included in the generated email. Provide them to the customer in a way that complies with the organization's security guidelines.

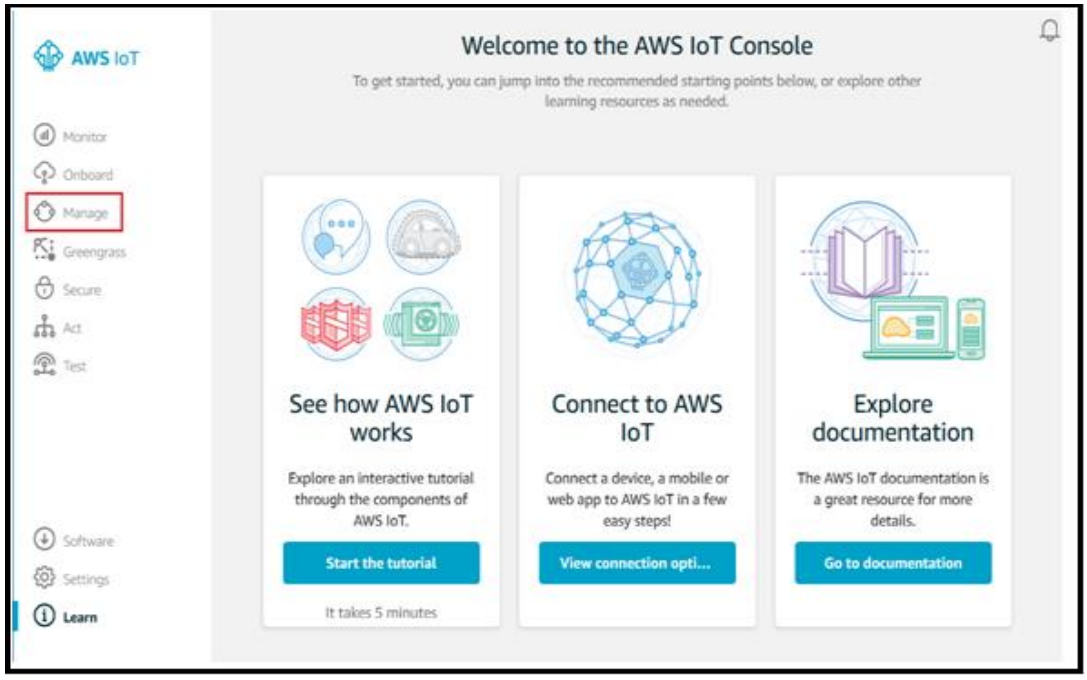
CAUTION: (Optional) Grant the user's permission to manage their own security credentials. For more information, see Allow Users to Manage Their Own feature Passwords, Access Keys and SSH Keys.

3.9.1.2 Register a device in Thing registry

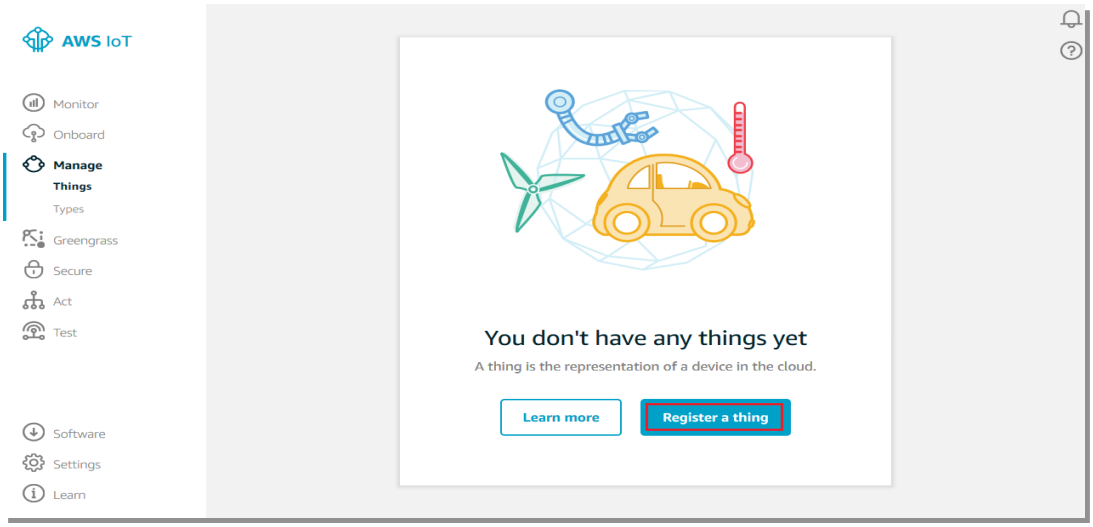
Devices connected to AWS IoT are represented by things in the thing registry. The Thing registry can be used to keep a record of all the devices that are connected to the AWS IoT account.

To register the device in the Thing registry:

1. On the **Welcome to the AWS IoT Console** page, in the left navigation pane, choose **Manage** to expand the choices, and then choose **Things** (as necessary).

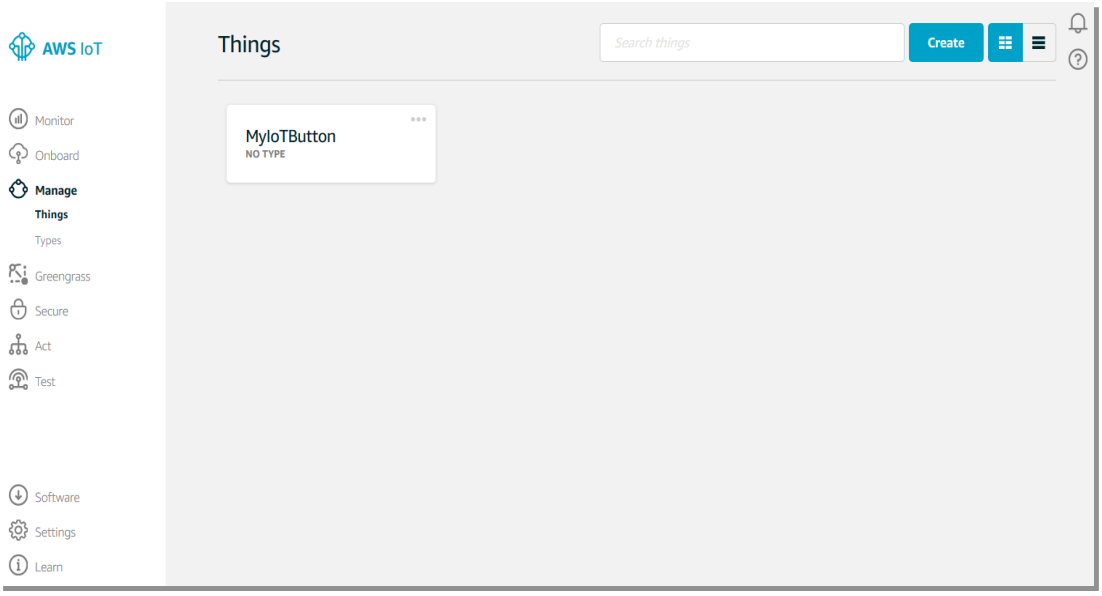


2. On the page that says you do not have any things yet, choose **Register a thing**.



CAUTION: On the **Register a thing** page, in the **Name** field, type a name for the device, such as `MyIoTButton`. Choose **Create thing** to add the device to the thing registry.

The results are shown in the following figure.



3.9.1.3 Register a thing type in Thing registry

To register the thing type in the Thing registry:

- On the **Welcome to the AWS IoT Console** page, in the left navigation pane, choose **Manage** to expand the choices, and then choose **Types** (as necessary).

CAUTION: On the page that says, “You don’t have anything types yet”, choose **Create thing type**.

CAUTION: On the **Create a thing type** page, in the **Name** field, type a name for the ThingType, as either **`MyIoTThingType`** `or` **`qcathings`**. Choose **Create thing type** to add the thing type to the thing registry.

3.9.1.4 Register a thing type to thing

To register the thing type to thing in the Thing registry:

- On the **Welcome to the AWS IoT Console** page, in the left navigation pane, choose **Manage** to expand the choices, and then choose **Things** (as necessary).

CAUTION: Select the thing where you want to attach the thing type. On the thing page, click **Edit**.

CAUTION: On the **Edit** page, Choose **Thing type** to add the thing from the drop-down menu and click **Update**.

3.9.1.5 Create and activate a device certificate

The communication between the device and AWS IoT is protected by using X.509 certificates. AWS IoT can generate a certificate for you or you can use your own X.509 certificate. Certificates must be activated prior to use.

To create and activate a device certificate:

- In the left navigation pane, choose **Secure**, **Certificates** (as necessary), and then click **Create a certificate**.

CAUTION: On the Create a certificate page, choose **Create certificate**.

CAUTION: On the **Certificate created!** page, choose **Download** for the certificate, private key, and the root CA for AWS IoT (the public key need not be downloaded). Save each of them to the computer, and then choose **Activate** to continue.

The downloaded filenames may appear differently than those listed on the **Certificate created!** page.

For example:

- 2a540e2346-certificate.pem.crt.txt
- 2a540e2346-private.pem.key
- 2a540e2346-public.pem.key

CAUTION: Although it is unlikely, root CA certificates are subject to expiration and/or revocation. If this occurs, copy new a root CA certificate to the device.

CAUTION: Click **Done**.

3.9.1.6 Create an AWS IoT policy

The X.509 certificates are used to authenticate the device with AWS IoT AWS IoT policies are used to authorize the device to perform AWS IoT operations, such as subscribing or publishing to MQTT topics. The device presents its certificate when sending messages to AWS IoT. To allow the device to perform AWS IoT operations, create an AWS IoT policy and attach it to the device certificate.

To create an AWS IoT policy:

- In the left navigation pane, choose **Secure**, and then **Policies**. On the **You don't have a policy yet** page, choose **Create a policy**.

CAUTION: On the **Create a policy** page, in the **Name** field, type a name for the policy (for example; `MyIoTButtonPolicy`). In the **Action** field, type **iot:Connect**. In the **Resource ARN** field, type `*`. Check the **Allow** check box. This allows all clients to connect to AWS IoT.

CAUTION: After entering the information for the policy, choose **Create**.

CAUTION: IoT.* gives access for every action on thing to the policy, where * in resource ARN indicate that it can allow action for any things which are present in console.

3.9.1.7 Attach an AWS IoT policy to a device certificate

After the policy is created, attach it to the device certificate. Attaching an AWS IoT policy to a certificate gives the device the permissions specified in the policy.

To attach an AWS IoT policy to a device certificate:

- In the left navigation pane, choose **Secure**, and then **Certificates**.

CAUTION: In the box created for the certificate, choose **...** to open a drop-down menu, and then choose **Attach policy**.

CAUTION: In the **Attach policies to certificate(s)** dialog box, select the check box next to the policy created in the previous step, and then choose **Attach**.

3.9.1.8 Attach a certificate to a thing

A device must have a certificate, private key and root CA certificate to authenticate with AWS IoT.

It is recommended to attach the device certificate to the thing that represents the device in AWS IoT. This allows to create AWS IoT policies that grants permissions based on certificates attached to the things.

To attach a certificate to the thing representing the device in the thing registry:

- In the box created for the certificate, choose **...** to open a drop-down menu, and then choose **Attach thing**.

CAUTION: In the **Attach things to certificate(s)** dialog box, select the check box next to the thing registered, and then choose **Attach**.

CAUTION: To verify the thing is attached, select the box representing the certificate.

CAUTION: On the **Details** page for the certificate, in the left navigation pane, choose **Things**.

CAUTION: To verify the policy is attached, on the **Details** page for the certificate, in the left navigation pane, choose **Policies**.

3.9.2 Build with Azure IoT SDK

Microsoft Azure cloud platform provides a secure and scalable mechanism to connect IoT end nodes to the cloud. QCA402x SDK now supports Azure IoT device SDK. QCA402x device talks to the Azure cloud using secure MQTT protocol.

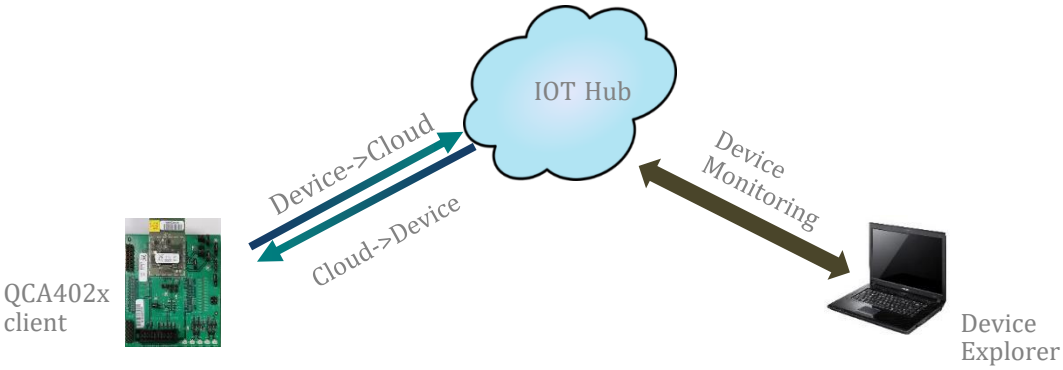


Figure 3-2 Microsoft Azure IoT SDK

The Azure SDK in Quartz is tested against branch 2017-07-14. Git client is required to clone the repository.

1. Navigate to the <SDK_source>/target/quartz/thirdparty directory.

CAUTION: Run the following command to clone:

```
git clone --recursive --branch 2017-07-14
https://github.com/Azure/azure-iot-sdk-c.git azure
```

To build the QCLI demo with Azure support *for the first time*:

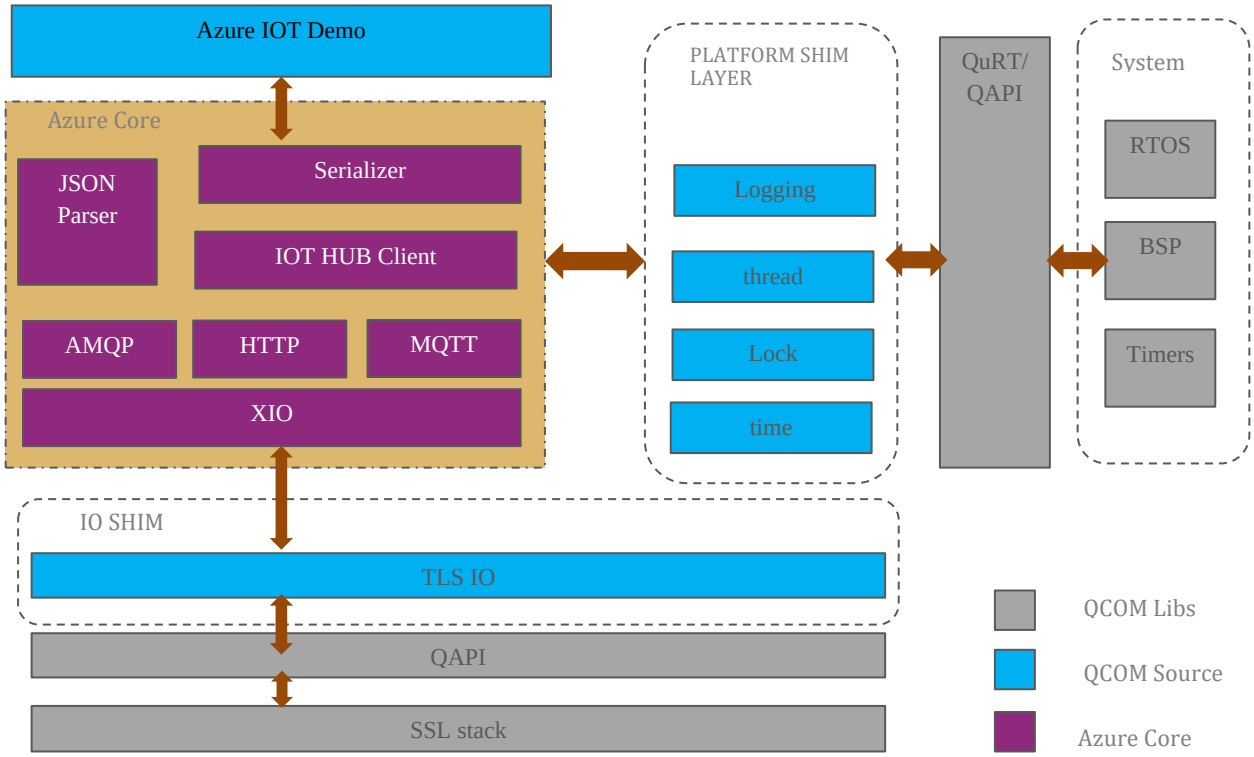
2. Azure SDK requires some utility functions from arm-none-eabi-gcc libs. To link with the required libraries, set the following environment variables-

```
set NEWLIBPATH=C:\Program Files (x86)\GNU Tools ARM Embedded\6.2
2016q4\arm-none-eabi\lib\thumb\v7e-m
set TOOLLIBPATH=C:\Program Files (x86)\GNU Tools ARM Embedded\6.2
2016q4\lib\gcc\arm-none-eabi\6.2.1\thumb\v7e-m
set Ecosystem=azure
```

3. Build QCLI demo (see Build sample applications section).

3.9.2.1 QCA402x Azure software architecture

The different software modules of QCA402x SDK and their interaction with the core Azure stack are shown in the following image. The Azure stack invokes QCA402x system APIs and networking services to interact with the IoT hub. The source code for the platform and the IO shim layers is provided in the SDK.



3.9.2.2 Set up IoT Hub

An Azure IoT account is required to create an IoT Hub. There is a Qualcomm account that can be used for testing. See the following link on how to create the IoT Hub-

https://github.com/Azure/azure-iot-sdk-csharp/blob/master/doc/setup_iothub.md

A unique connection string is associated with each hub instance. This string is needed to communicate with the hub.

3.9.2.3 Create a device

To communicate with the Azure IoT cloud a device instance must be created on the IoT Hub. The device can be created by a PC-based tool called Device Explorer. The Device Explorer binary can be downloaded from the following link-

https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer

3.10 BLE direct test mode

The BLE direct test mode is for automated conformance testing, which allows the tester to control the implementation through the PHY interface and test interface. Test commands and events over HCI are exchanged through the test interface.

The QCA402x SDK contains passthrough demo application which can exchange the HCI test commands and events through UART interface. Passthrough demo application is available at:

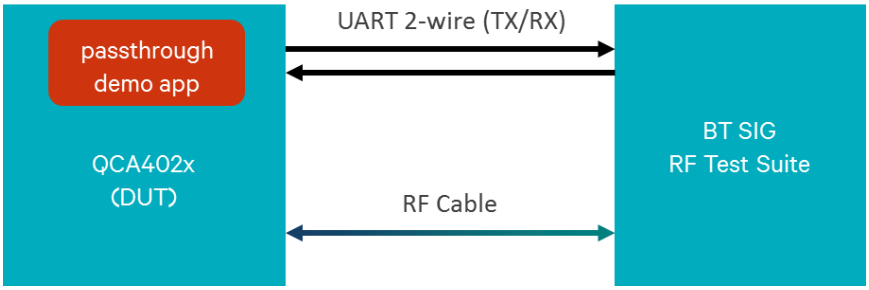
`target\quartz\demo\Passthrough_demo`

Configure the UART baud rate according to the UART configuration in RF test suite.

The following example shows how Passthrough demo application uses Debug UART interface and UART baud rate sets 115.2kbps at target\quartz\demo\Passthrough_demo\src\app_main.c

```
#define UART_BAUD_RATE    (115200) //(1000000)
// InitializationData.Flags = PT_UART_INITIALIZATION_DATA_FLAG_USE_HS_UART |
PT_UART_INITIALIZATION_DATA_FLAG_USE_HW_FLOW;
```

The BLE direct test mode setup using a 2-wire UART interface is shown in the following figure:



4 Helloworld demo

The helloworld demo is a simple demo application that can be used as reference to create new demo applications. The necessary files for this demo are in the **<sdk_source>/target/quartz/demo/Helloworld_demo** directory.

```
|-- Helloworld_demo                    //Sample Helloworld application
|       |-- build
|       |      |-- gcc
|       |      |     |-- v2                 //folder for gdb debug script
|       |      |     |-- Makefile           //build script for Linux
|       |      |     |-- build.bat          //build script for Windows
|       |      |     |-- 4020_secimage.xml //Image signing config file
|       |      |     |-- app.config         //linker script config file
|       |      |     |-- debug.bat          //debug script for Windows
|       |-- src
|       |      |-- pal
|       |      |      |-- pal.c             //App init/start/logging
|       |      |      |-- pal.h             //App header file
|       |      |-- hello_world.c            //App main loop
```

- The “Helloworld_demo” folder contains the **build** and **src** directories. The “build/gcc” compiler builds this demo using “Makefile” or “build.bat”. The **src** folder contains the **hello_world.c** file and a **pal** folder for the platform initialization of this demo.
- The hello_world program creates a “HelloWorld_Thread” thread which prints the “Hello World” messages to the console periodically. The pal folder is the platform abstraction layer which contains the **pal.c** and **pal.h** files responsible for the initialization and the start of the application.
- If the user wants to initialize any prerequisites for the demo, it can be done in app_init ().
- The Helloworld_demo is initialized as follows:

  `Platform > app_init() > Initialize_Sample() > Initialize_demo()> PAL_Initialize() > PAL_Uart_Init()`
- The HelloWorld_Thread() thread is created as part of the App_start(). The main loop starts printing the hello world messages to the console periodically.

  `Platform > app_start() > App_start() > QuRT_thread_create() > HelloWorld_Thread()`

5 Onboard demo with cloud mode

This chapter explains the steps to onboard the CDB2x to the AWS cloud network. This demo requires a qca_sensors mobile application to read sensor data and control sensors on CDB2x devices. The communication happens through BLE or Wi-Fi (between mobile application and CDB20) and Zigbee/Thread (between CDB2x devices). The CDB20 device is connected to the Home AP to send the sensor information to the AWS cloud network. Two applications are required to set up onboarding − first onboard demo application running on CDB2x device, and the second (QCAOnboard and Sensor) application running on the mobile device. For the mobile applications, see Onboard mobile application section.

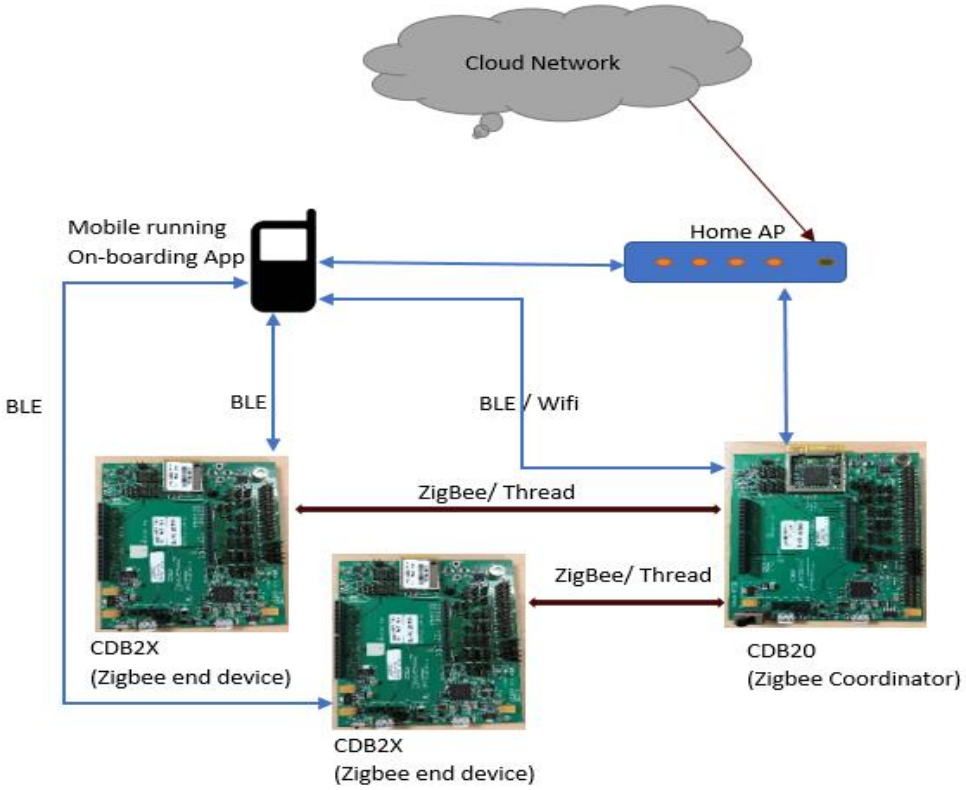


Figure 5-1 Cloud mode onboarding setup

5.1 Prerequisites to build onboard AWS demo

Download and modify the code as suggested in the Build with AWS IoT section.

Edit the following files to include the contents of private key, certificate and root ca downloaded in the Create and Activate a Device Certificate section for a AWS thing in the following arrays under a single line statement.

- /target/quartz/demo/Onboard_AWS_demo/src/include/cert_buf.h

```
aws_thing_privkey[] = {};
aws_thig_cert[] = {};
aws_calist[]= {};
```

Example

```
aws_thing_privkey[] =
{
"-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA……..……..1lmZ+g==-----END RSA PRIVATE KEY-----"
};
```

- /target/quartz/demo/Onboard_AWS_demo/src/ecosystem/aws/aws_util.h

  Include the actual AWS server details and the Thing name in the following macros:

```
//AWS Server Details
#define HOST_ID    "Abcdefxxxx.xxxx.xxx. amazonaws.com "
#define THING_NAME    "ABCD_XXX "
```

Example

```
#define HOST_ID    "a3eoxxxxxxxz0.iot.us-west-2.amazonaws.com"
#define THING_NAME   "IOTDEMO"
```

- Configuration changes for Onboard_demo (CDB20 & CDB24) for **Threadx**
  - /target/quartz/demo/Onboard_AWS_demo/src/export/DevCfg_master_devcfg_out_cdb.xml

  By default, the Data Execution Prevention (DEP) is enabled on QCA402x. To make changes in the code and data memory regions, adjusted the DEP configuration. Modified the DevCfg_master_devcfg_out.xml file to adjust DEP configuration region.

  In the following line, change 0XF1 to 0XF9 to increase the heap size to 128 K:

```
<!-- FOM Code and Data region = --> 0x00, 0x00, 0x00, 0x10, 0x00,0x00,
0x10, 0x00, 0x02, 0xF9, 0x06, 0x00,
```

  - /target/bin/cortex-m4/threadx/DefaultTemplateLinkerScript.ld

  To increase data memory (RAM_FOM_APPS_DATA_MEMORY) by 128 KB, decrease code memory (RAM_FOM_APPS_RO_MEMORY) by the same amount. Make the following changes in the DefaultTemplateLinkerScript.ld script:

  Change:

  RAM_FOM_APPS_RO_MEMORY (Rx) : ORIGIN = 0x10046000, LENGTH = **0x3a000**

  RAM_FOM_APPS_DATA_MEMORY (W) : ORIGIN = **0x10080000**, LENGTH = **0x10000**

  To:

  RAM_FOM_APPS_RO_MEMORY (Rx) : ORIGIN = 0x10046000, LENGTH = **0x1a000**

RAM_FOM_APPS_DATA_MEMORY (W) : ORIGIN = **0x10060000**, LENGTH = **0x30000**

- Configuration changes for Onboard_demo (CDB20 and CDB24) for **FreeRTOS**
  - Building the freertos library

Re-sizing FreeRTOS Heap FreeRTOS requires a dedicated memory pool used for allocating RTOS-specific elements such as task stack. OEM may adjust this allocation based on their needs by following the steps.

- cd to "./FreeRTOS/1.0/FreeRTOS/Demo/QUARTZ" directory.
  Edit FreeRTOSConfig.h, change configTOTAL_HEAP_SIZE as following:
- Set the configTOTAL_HEAP_SIZE to 0x1C800
- Rebuild FreeRTOS library (See section 3.8)
  - /target/bin/cortex-m4/freertos/DefaultTemplateLinkerScript.ld

Change:
RAM_FOM_APPS_RO_MEMORY (Rx): ORIGIN = 0x10046000, LENGTH = **0x3a000**

RAM_FOM_APPS_DATA_MEMORY (W): ORIGIN = **0x10080000**, LENGTH = **0x10000**

/*FOM HEAP begins at the end of FOM_APP_DATA region that is used-up by the application, an increase in data usage decreases HEAP and vice-versa*/
RTOS_HEAP_SIZE = 0x1C800

- target/target/thirdparty/aws/awsiot/include/aws_iot_config.h

```
#define AWS_IOT_MQTT_TX_BUF_LEN 1024
#define AWS_IOT_MQTT_RX_BUF_LEN 1024
```

CAUTION: Adjust the preceding macros based on the message size.

5.2 LED status indication

This section explains the LED indication during onboarding and functional phase/stages. The jumper settings required to verify the functional phases and light sensor toggle are listed in the CDB2x board setup section.

Red LED

- Glows constantly if the board is initialized successfully and is ready for onboarding.
- Blinks if initialization fails during board powerup or on reset.
- Off when other LED indications are fine.

Green LED

- Blinks if a subset of the radios is onboarded.
  For example, if the device allows Wi-Fi and Zigbee to be onboarded, and the user has onboarded only Wi-Fi or Zigbee, the Green LED blinks. Glows constantly if all the allowed radios are onboarded.

Blue LED

- Qualcomm sensors app controls this LED through the AWS server.
- LED will turn on/off based on the input provided from the AWS server.

WLAN LED

- Blinks while connecting to an AP.
- Glows constantly when connected to AP.
- Off when AP is not connected.

5.3 Run onboard AWS demo

- Before proceeding, follow the prerequisites listed for the Zigbee network in the Without cloud mode onboard demo section, for the onboard AWS demo.
- CDB4020 and CDB4024 devices bootup on power cycle, come in BLE/Wi-Fi peripheral mode (based on how the device is configured while building) and start advertising.
- Mobile app connects to CDB20/24 devices with passkey ”123456” for BLE onboarding and passphrase “123456789” for Wi-Fi onboarding.
- Once CDB20 board receives the BLE/Wi-Fi passkey and onboards from the mobile app, it validates the configuration, and saves it to the filesystem.
- Mobile app reads the CDB20 board status and displays the Wi-Fi onboarding screen. Enter the Home AP credentials to connect to the network.
  Onboard Zigbee coordinator/Thread joiner router on CDB20
- On successful BLE/Wi-Fi connection of the mobile app and CDB20 device, the mobile app reads the Zigbee/Thread service for onboard status through BLE connection. If the status is success, and either Zigbee is in coordinator mode or Thread is in joiner router mode, the mobile app generates link key and sends the credentials packet to the CDB20 device running as Zigbee coordinator or Thread joiner router.
- After the CDB20 device gets the operating mode and link key, onboarding demo extracts the onboard details, validates the configuration, stores to the filesystem and sends the ACK to the mobile app. The CDB20 device starts Zigbee/Thread service as a coordinator/joiner router based on user-configured Zigbee/Thread operating mode.
- After the mobile app receives the Zigbee onboard status notification it displays the onboard status screen and status.

Onboard Zigbee end device/Thread joiner on CDB2x

- User can “onboard” the CDB2x device using BLE connection.
  - In case of Zigbee onboarding, the CDB2x device runs as Zigbee end device connecting to the Zigbee coordinator CDB20 device.
  - In case of Thread onboarding, the CDB2x device runs as Thread joiner connecting to the Thread joiner router CDB20 device. On the mobile app main screen, select the CDB2x device and onboard the device by using default password “123456” using BLE connection or passphrase “123456789” using Wi-Fi connection.
- The mobile app checks for the CDB2x onboard status and displays the Zigbee/Thread connection status using BLE/Wi-Fi, selects the relevant coordinator CDB20 device name or joiner router CDB20 device name which lets the CDB2x as end device to join the Zigbee network, or joiner to join the Thread network using BLE/Wi-Fi connection.
- If the CDB24 is onboarded successfully, the mobile app displays successful onboarding device status; else notifies with failure status and disconnects from the device.

Get sensor data [Zigbee+BLE] using qca_sensors application

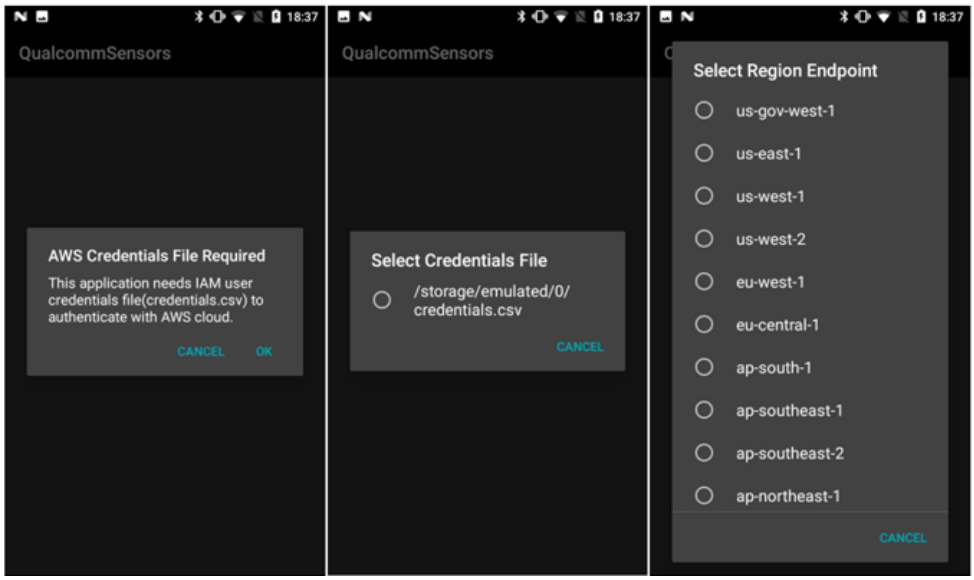
After successfully onboarding CDB2x devices, the Zigbee end device/Thread joiner sends the sensor information to Zigbee coordinator/thread joiner router. Use the AWS Dashboard Desktop Application to observe sensor data in live graph format or use the qca_sensors app to view the sensor information.

- To view sensor data using qca_sensor mobile application, store **credentials.csv** file in the Android device downloaded from Creating IAM Users (Console).

CAUTION: After the application is launched, the user will be prompted a dialog to have the IAM user credentials.csv file saved in the Phone.

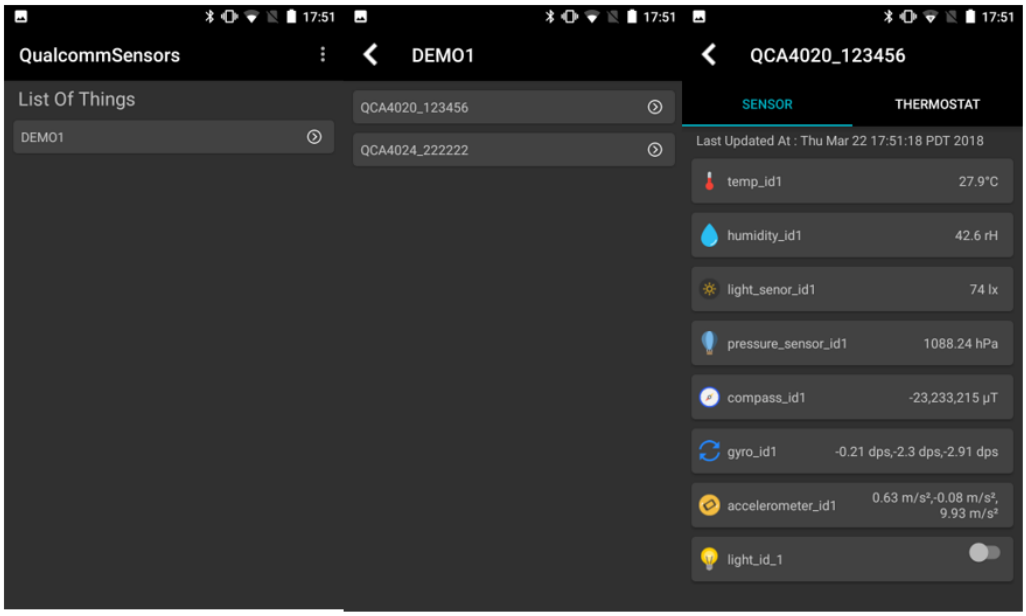
CAUTION: The user selects the file and AWS region from the dialog displayed with the list of available regions.

CAUTION: The previous two steps are required only once from the user to connect to AWS IoT services while communicating sensors and thermostat data:



CAUTION: After successful login, the application displays the list of things from the AWS IoT which has the details of sensors and thermostat of one or more boards connected to each thing.

CAUTION: Choose a device under the list of devices which are associated with it to view the sensors and thermostat data.



5.4 Console for the onboarding demo

If the high-speed UART is connected and a serial console is open, a CLI menu appears. Use the reset command to reset the onboarding information save on the CDB to allow it to be onboarded again from the mobile app.

CAUTION: QCLI command interface is provided only for debugging purpose. Ue the reset command to reset the onboarding information and save it on CDB to allow it to be onboarded again from the mobile app.

1. Banner logs for the onboarding demo using Wi-Fi is shown here.

```
Onboard: Enable WLAN numVDEV:2
Onboard: Waiting for Credentials ...

Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit

  Subgroups:
     3. Onboard


>
Onboard: Operate mode set sucsess
Onboard: CONNECTING to SSID:QCA4020, pwd:123456789
Onboard: setting to ap mode
Onboard:
Onboard: Setting SSID to QCA4020
Onboard:
Onboard: WLANCB: dID:0, cbID:0
Onboard: Connect event on devId:0 val:1
Onboard: devid - 0 1 CONNECTED MAC addr 00:03:7f:20:01:42
Onboard: UDP Server started.........
Onboard: WLANCB: dID:0, cbID:0
Onboard: Connect event on devId:0 val:1
Onboard: devid - 0 0 CONNECTED MAC addr 60:8e:08:31:a6:5a
Onboard: DHCPv4s: Client IP=192.168.0.11  Client MAC=60:8e:08:31:a6:5a
Onboard: Received bytes are: 5
Onboard: sendig data is {"fwVersion":"0.1", "chipset":4020, "mode":1, "onboard":0}
```

CAUTION: Onboard CDB20 to a home AP which is in the open mode.

```
Onboard: sendig data is {"fwVersion":"0.1", "chipset":4020, "mode":1, "onboard":0}
Onboard: sent bytes58
Onboard: Received bytes are: 15
Onboard: SSID = net
Onboard: WLANCB: dID:1, cbID:0
Onboard: PASSWDLEN = 0
Onboard: Connect event on devId:1 val:0
Onboard: sending data is: {"fwVersion":"0.1", "chipset":4020, "mode":1, "onboard":1}
Onboard: devId 1 Disconnected MAC addr 00:00:00:00:00:00
Onboard: sent bytes are: 58
Onboard: sent bytes are: 58
Onboard: sent bytes are: 58
Onboard: WLANCB: dID:0, cbID:0
Onboard: Connect event on devId:0 val:0
Onboard: REF_STA Disconnected MAC addr 60:8e:08:31:a6:5a devId 0
Onboard: Operate mode set sucsess
Onboard: CONNECTING to SSID:net, pwd:----
Onboard:
Onboard: Setting SSID to net
Onboard:
Onboard: Connect to network sucsess

Onboard: Waiting for Credentials ...
Onboard: WLANCB: dID:1, cbID:0
Onboard: Connect event on devId:1 val:1
Onboard: devid - 1 1 CONNECTED MAC addr 78:02:f8:95:02:17
Onboard: Get DHCP addr
Onboard: signal event handled
```

CAUTION: Onboard CDB20 to a home AP using BLE.

```
Onboard: BD_ADDR: 0x00001D123456
Onboard: BD_ADDR: 56:34:12:1d:00:00
Onboard: Enable WLAN numVDEV:2Wlan enable_success
Onboard: Successfully registered Wi-Fi Service, ServiceID = 4.
Onboard: Successfully registered Zigbee Onboard Service, ServiceID = 5.
Onboard: read_wifi_config

Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit

  Subgroups:
     3. Onboard


>
Onboard: Onboard Credentials Not stored
```

CAUTION: Connect to the AWS server.

```
Onboard: Waiting for Onboard events ...
Onboard: DHCPv4c: IP=192.168.43.222  Subnet Mask=255.255.255.0  Gateway=192.168.43.1
Onboard: START AWS: running(0)
Onboard: Shadow Init

Onboard: Shadow Connect

Onboard: Shadow Connection successful

Onboard: shadow reconnect status done
Onboard: In while RC Value:0
```

CAUTION: Update the sensor data to the AWS shadow server.

```
Onboard: ============================================================
Onboard: BD_ADDR: 56:34:12:1d:00:00
Onboard:    ------  Temperature ------
Onboard: DegCx8 T0:0xa6  T1:0x10  T0_T_MSB:c4
Onboard: DegCx8 with MSB T0:0xa6  T1:0x110
Onboard: register 2's comp T0:1 T1:730 T:576
Onboard: DegCx8 T0:166 T1:272 Tx10:2496
Onboard: Current Temperature:31.2

Onboard: ------  Relative Humidity ------
Onboard: rHx2 H0:0x3c  H1:0x8d
Onboard: 2's comp H0_T0:-2 H1_T0:-10209 H:-3169
Onboard: rHx2 Hx10:851
Onboard: Current rH:42.5% rH
Onboard:    ------  Temperature ------
Onboard: DegCx8 T0:0xa6  T1:0x10  T0_T_MSB:c4
Onboard: DegCx8 with MSB T0:0xa6  T1:0x110
Onboard: register 2's comp T0:1 T1:730 T:577
Onboard: DegCx8 T0:166 T1:272 Tx10:2497
Onboard: Current Temperature:31.2
```

CAUTION: **Zigbee** coordinator initializes.

```
Onboard: read_zigbee_config
Onboard: Onboard Credentials Not stored
Onboard:    GAP_LE_Advertising_Enable success, Advertising Interval Range: 100 - 200.

Onboard: Waiting for Onboard events ...
Onboard: Monitor Thread is runnning ----------------------
Onboard: waiting on Monitor thread
Onboard: OpenStack().
Onboard: Bluetooth Stack ID: 1.
Onboard: Number ACL Buffers: 16, ACL Buffer Size: 81
Onboard: BD_ADDR: 0x5A1CED0A1B20
Onboard: BD_ADDR: 20:1b:0a:ed:1c:5a
Onboard: Enable WLAN numVDEV:2Wlan enable_success
Onboard: Successfully registered Wi-Fi Service, ServiceID = 4.
Onboard: Successfully registered Zigbee Onboard Service, ServiceID = 5.
Onboard: read_wifi_config

Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit

  Subgroups:
     3. Onboard
```

CAUTION: Zigbee end device is connected to the coordinator screen.

```
Onboard: ZigBee stack initialized.
Onboard: Extended Address: 0000000000000000
Onboard: Successfully set Extended Address
Onboard: Extended Address: 000000001D123456
Onboard: Short Address:    0xFFFF
Onboard: In Join Link Key  :ZigBeeAlliance09
Onboard: Suuccessfully joined Zigbee network
Onboard:  Extended Address = 0000000000000000
Onboard: Device List:
Onboard:  ID | Type | Address          | Endpoint
Onboard: ----+------+------------------+----------
Onboard:  Failed to get Device Id
Onboard: waiting on Monitor thread
Onboard: Join confirm:
Onboard:   Status:        0
Onboard:   NwkAddress:    0xB4ED
Onboard:   ExtendedPanId: FFFFFFFFED0A1B20
Onboard:   Channel:       25
```

CAUTION: Zigbee coordinator light switch cluster sends on/off command.

```
Onboard:  Extended Address = 000000001D123456
Onboard: Device List:
Onboard:  ID | Type | Address          | Endpoint
Onboard: ----+------+------------------+----------
Onboard:   1 | Ext  | 000000001D123456 | 111
Onboard:  Index= 1
Onboard:  Endpoint= 111
Onboard: Device is registered with ExtAddr = 000000001D123456
Onboard:  Device Id = 1
Onboard:  Cluster Endpoint 111
Onboard: SUCCESS: qapi_ZB_CL_OnOff_Send_Toggle
Onboard: waiting on Monitor thread
Onboard: OnOff Client Default Response:
Onboard:   Status:        0
Onboard:   CommandID:     0x02
Onboard:   CommandStatus: 0
```

CAUTION: Zigbee end device light cluster receives on/off command.

```
Onboard: In ZCL_OnOff_Demo_Create_ServerEnd Device Successfully Joined network
Onboard: Zigbee is configured successfully

Onboard: waiting on Monitor thread
Onboard: OnOff Server State Change: ON
Onboard: led_config called to switch ON LED

Onboard: waiting on Monitor thread
Onboard: OnOff Server State Change: OFF
Onboard: led_config called to switch OFF LED
```

CAUTION: **Thread** joiner initializes.

```
> Onboard: OpenStack().
Onboard: Bluetooth Stack ID: 1.
Onboard: Number ACL Buffers: 16, ACL Buffer Size: 81
Onboard: BD_ADDR: 0x5A1CED0A1B20
Onboard: BD_ADDR: 5a:1c:ed:0a:1b:20
Onboard: Successfully registered Zigbee Onboard Service, ServiceID = 4.
Onboard: Successfully registered thread Onboard Service, ServiceID = 5.


Onboard: /*-----------------------------------------------------------------*/
Onboard:                Image build time : Tue Jan  2 15:58:41 2018
Onboard:                Chipset version : qca4024
Onboard:                RTOS : threadx
Onboard:                Onboarding VIA   : BLE
Onboard:                 Onboardble Radios:
Onboard:                        ZIGBEE
Onboard:                        THREAD
Onboard: /*-----------------------------------------------------------------*/
```

CAUTION: Thread joiner connects to the border router.

```
Onboard: Calling Start_Thread_Joiner
Onboard: Start_Thread_Joiner
Onboard:  In Device_Config == 1
Onboard: Thread Initialized Successfully:
Onboard: Network Configuration:
Onboard:    Channel:           16
Onboard:    PAN_ID:            8DA8
Onboard:    Extended_PAN_ID:   0001020304050607
Onboard:    NetworkName:       Test Network
Onboard:    MasterKey:         F32B7B515AD61BFAF32B7B515AD61BFA
Onboard: SUCCESS: qapi_TWN_Set_Max_Poll_Period
Onboard: Max_Poll_Period set is: 1


Onboard:  In Joiner_Confirm_Status = 0

Onboard: waiting on Monitor thread
Onboard: SUCCESS: qapi_TWN_Joiner_Start


Onboard:  In Joiner_Confirm_Status = 0

Onboard: waiting on Monitor thread
Onboard: Joiner Result: Success
Onboard: Network Configuration:
Onboard:    Channel:           24
Onboard:    PAN_ID:            8DA8
Onboard:    Extended_PAN_ID:   0001020304050607
Onboard:    NetworkName:       Test Network
Onboard:    MasterKey:         F32B7B515AD61BFAF32B7B515AD61BFA
```

CAUTION: Thread joiner sends data to the border router.

```
Onboard: ------  Relative Humidity ------
Onboard: rHx2 H0:0x3e  H1:0x8c
Onboard: 2's comp H0_T0:10 H1_T0:-12712 H:-2150
Onboard: rHx2 Hx10:752
Onboard: Current rH:37.6% rH
Onboard:   ------  Temperature ------
Onboard: DegCx8 T0:0xa5  T1:0x10  T0_T_MSB:c4
Onboard: DegCx8 with MSB T0:0xa5  T1:0x110
Onboard: register 2's comp T0:1 T1:703 T:410
Onboard: DegCx8 T0:165 T1:272 Tx10:2273
Onboard: Current Temperature:28.4

Onboard: ------  Relative Humidity ------
Onboard: rHx2 H0:0x3e  H1:0x8c
Onboard: 2's comp H0_T0:10 H1_T0:-12712 H:-2152
Onboard: rHx2 Hx10:752
Onboard: Current rH:37.6% rH
Onboard: CH 1:0   CH 0:0
Onboard: Light ch0 Lux=0


Onboard:  Context Data:{"state":{"reported":{"QCA4024_0a1b20":{"temperature"
```

CAUTION: Thread border router initializes.

```
Onboard: Calling Start_Thread_Boarder_Router
Onboard: Start_Thread_Boarder_Router
Onboard: Thread_Initialize
Onboard:  In Device_Config == 0
Onboard: Thread Initialized Successfully:
Onboard: Network Configuration:
Onboard:    Channel:           11
Onboard:    PAN_ID:            FFFF
Onboard:    Extended_PAN_ID:   DEAD00BEEF00CAFE
Onboard:    NetworkName:       OpenThread
Onboard:    MasterKey:         00112233445566778899AABBCCDDEEFF After Thread_Initialize
Onboard: Thread_UseDefaultInfo
Onboard: Network Configuration:
Onboard:    Channel:           16
Onboard:    PAN_ID:            8DA8
Onboard:    Extended_PAN_ID:   0001020304050607
Onboard:    NetworkName:       Test Network
Onboard:    MasterKey:         F32B7B515AD61BFAF32B7B515AD61BFA After Thread_Initialize
Onboard: Thread_Interface_Start
Onboard: SUCCESS: qapi_TWN_StartNetwork State Changed: Detached
Onboard:  After Thread_Interface_Start

>
Onboard: Thread_MeshCoP_CommissionerStart
Onboard: SUCCESS: qapi_TWN_Commissioner_Start After Thread_MeshCoP_CommissionerStart


>
Onboard: Network State Changed: Leader
```

CAUTION: Thread border router receives data from the joiner.

```
Onboard: Device List:
Onboard:  ID | Board Name     | IP Address
Onboard: ----+----------------+--------------------
Onboard:  1 | QCA4024_0a1b21 | FD00:102:304::87A:146:4DBA:A3C4
Onboard:  Receiving Data from Joiner
Onboard: Received bytes are: 167        {"state":{"reported":{"QCA4024_0a1b21"
Onboard: Board_Name = QCA4024_0a1b21
Onboard: After Extracting Board_Name = QCA4024_0a1b21
Onboard:            total       used       free
Onboard: Heap:     308480     139888     168592
```

The user can erase the stored credentials of the Zigbee network on CDB20/24 devices by issuing hard reset commands `<Onboard> <reset_onboard_info>` in the console.

```
> 0
QAPI Ver: 1.0.1
CRM  Num: 9

> 1

Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit

  Subgroups:
     3. Onboard

> 3

Onboard> 1

Command List (Onboard):
  Commands:
     0. Ver
     1. Help
     2. Up
     3. Root

     4. reset_onboard_info

Onboard> 4

Onboard: Monitor thread exiting
Onboard: OpenStack().
Onboard: Bluetooth Stack ID: 1.
Onboard: Number ACL Buffers: 16, ACL Buffer Size: 81
Onboard: BD_ADDR: 0x7011117F0300
Onboard: BD_ADDR: 70:11:11:7f:03:00
Onboard: Successfully registered Zigbee Onboard Service, ServiceID = 4.
Onboard: Successfully registered Offline Onboard Service, ServiceID = 5.
Onboard: Initialized AWS
Onboard: Waiting for OFFLINE_Recv event


Onboard: /*-------------------------------------------------------------------*/
Onboard:                Image build time: Wed Mar 07 16:48:48 2018
Onboard:                Chipset version : qca4020
Onboard:                RTOS            : threadx
Onboard:                Onboarding VIA   : BLE
Onboard:                 Onboardable Radios:
Onboard:                        WIFI
Onboard:                        ZIGBEE
Onboard: /*-------------------------------------------------------------------*/
Onboard: read_wifi_config

Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit

  Subgroups:
     3. Onboard


>
Onboard: Onboard Credentials Not stored
Onboard: read_zigbee_config
Onboard: Onboard Credentials Not stored
Onboard: Thermo_stat_values already loadedbytes_read num = 1
Onboard:    GAP_LE_Advertising_Enable success, Advertising Interval Range: 100 - 200.

Onboard: Waiting for Onboard events ...
Onboard: Monitor Thread is runnning ----------------------
```

5.5 AWS dashboard application

This section explains how to setup the AWS dashboard application (a web application) on Linux and Windows machines.

5.5.1 Install the AWS application

5.5.1.1 Installation on Linux with Ubuntu version 16.04

Prerequisites:

- Download and install Node.J, Node Package Manager (NPM), and MongoDB.

```
$ sudo apt-get update
$ sudo apt-get install build-essential libssl-dev curl
```

- Use the latest nvm version in the place of v0.33.0.

```
$ curl -sL
https://raw.githubusercontent.com/creationix/nvm/v0.33.0/insta
ll.sh -o install_nvm.sh
$ bash install_nvm.sh
```

CAUTION: To apply the changes, log off, and log-on again.

CAUTION: Install and use the version required by the commands shown here.

```
$ nvm install 8.7.0
$ nvm use 8.7.0
$ nvm -v
v8.7.0
$ npm -v
v5.4.2
```

Node.JS and NPM are installed successfully.

Install MongoDB

- To install MongoDB, execute the following commands.

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --
recv EA312927
$ echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-
org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-
org-3.4.list
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
```

- To use MongoDB as a service, create a unit file.

```
$ sudo vim /etc/systemd/system/mongodb.service
```

CAUTION: Paste the following code to the mongodb.service file.

```
[Unit]
Description=High-performance, schema-free document-oriented
database
After=network.target

[Service]
User=mongodb
ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf

[Install]
WantedBy=multi-user.target
```

CAUTION: MongoDB is ready to be used as an init service.

```
$ sudo systemctl start mongodb
$ sudo systemctl enable mongodb
$ sudo systemctl status mongodb
$ sudo systemctl stop mongodb
```

5.5.1.2 Installation on Windows version 7 or higher

Prerequisites:

- Download and install Node.JS, Node Package Manager (NPM), and MongoDB.

CAUTION: Download the 64-bit or 32-bit **.msi** file from https://nodejs.org/en/download/.

CAUTION: Go to the downloads directory and double-click on the installer and follow the installation wizard.

Node.J and NPM are installed. The node and NPM commands can be used in the command prompt.

Installing MongoDB

- Create an account and download MongoDB 64-bit or 32-bit **.msi** from https://www.mongodb.com/download-center#atlas.

CAUTION: Go to downloads directory, double-click the installer, and follow the installation wizard to install MongoDB.

By default, MongoDB uses the **\data\db** directory for data storage.

CAUTION: Create the **\data\db** directory using the command, `md \data\db`.

CAUTION: To start mongoDB, run this command in the command prompt:

```
C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe
```

5.5.2 Set up AWS application

To setup the web application on both Windows and Linux systems, make the following changes to the source code of the AWS dashboard application available at **target\ browser\QCOM-AWS-Dashboard-v1.x.x.**

- Change to projects root directory and copy **config/global.sample.js** file to **config/global.js**.

```
$ cd QCOM-AWS-Dashboard-v 1.x.x
$ cp config/global.sample.js config/global.js
```

CAUTION: Open **config/global.js** file and modify as following (there are three nested objects):

- **MongoDB** object: Has the MongoDB configuration.
  - **hostname**: Provide the IP address of the machine where MongoDB is running.
  - **port**: Provide the port number where MongoDB is running.
  - **db**: Provide database name to store the data inside MongoDB.

  (To know the port number of MongoDB, execute the command `$ sudo netstat -nlp | grep "mongod".`)

  Example:

```
mongodb : {
      hostname :  "localhost", //localhost
      port : 27017 , //27017
      db : "QCA4020" //ruby-quartz
```

- **aws_iot** object: The configuration in this object is used to access AWS IoT.
  - **host**: Provide the AWS host URL.
  - **thing_type_name**: Provide the thing type name to filter from AWS IoT.
- **app** object: The configuration in this object is used to start the application.
  - **port**: Provide the port number for the application to start listening.
  - **polling_interval**: Provide the time interval for polling in seconds.

  Example:

```
aws_iot : {
host: "axxxxxxxxx.iot.us-west-2.amazonaws.com",
  thing_type_name: "qcathings",
  breach_topic_name: "DEMO1"
```

CAUTION: Copy config/aws.sample.json file to config/aws.json

```
$ cp config/aws.sample.json config/aws.json
```

CAUTION: Open **config/aws.json** and provide the following details available in credentials.csv file downloaded from the AWS IAM console.

- **accessKeyId**: Provide AWS IoT access key.
- **secretAccessKey**: Provide AWS IoT secret access key.

- **region**: Provide AWS region name in which the things are created.

This configuration is used to authorize access AWS IoT data.

5.5.2.1 Changing region for the dashboard application

To change/update the region in future, change one or more configurations manually. **AWS Javascript sdk** does not support dynamic changes in credentials. The javascript cannot update credentials and fetch other from AWS, including **accessKeyId, secretAccessKey,** and **region**. To do so, edit the **region** field in **config/aws.json,** provide respective endpoint in the **host** field in **config/global.js,** and rerun the application.

5.5.2.2 Installing the node modules/packages

Change to the project's root directory and install NPM in the terminal/command prompt. This command installs all dependencies listed in the **package.json** file.

```
$ cd QCOM-AWS-Dashboard-v 1.x.x
$ npm install
```

5.5.3 Run AWS Application

CAUTION: The web application will start only if all the above-mentioned components are installed.

To run the AWS dashboard web application, change to the project's root directory and run the following command:

```
$ cd QCOM-AWS-Dashboard-v 1.x.x
$ node app.js
```

This command starts the web application at the port specified in configuration file. Access the dashboard by typing **http://localhost:9000** in the browser.

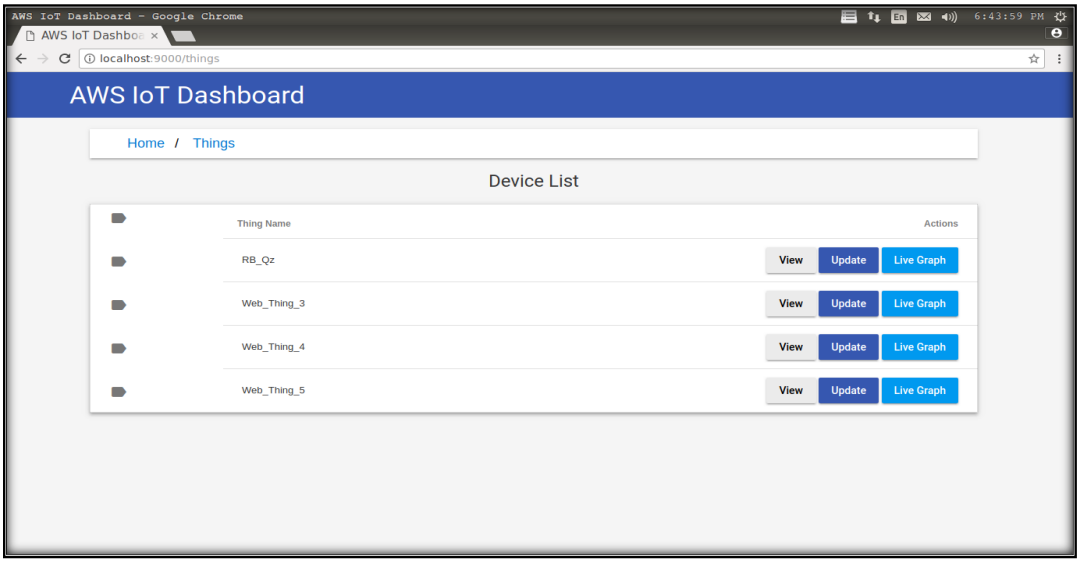


Figure 5-2 AWS IoT dashboard screen

The dashboard screen displays the list of available things under the AWS account. The things list is fetched from AWS IoT by using the filter key as thing type name. Each thing contains the data of one or more boards/devices connected to it. Each thing has three options − to view devices, update thermostat, and view live graph.

5.5.3.1 View

Clicking the **View** button displays data over a period (of time) and the status of thing.

In the **View** screen, the status of all connected devices and sensors inside a device for the selected thing is displayed. Click on each tab/device to see the status of all sensors connected to that device.

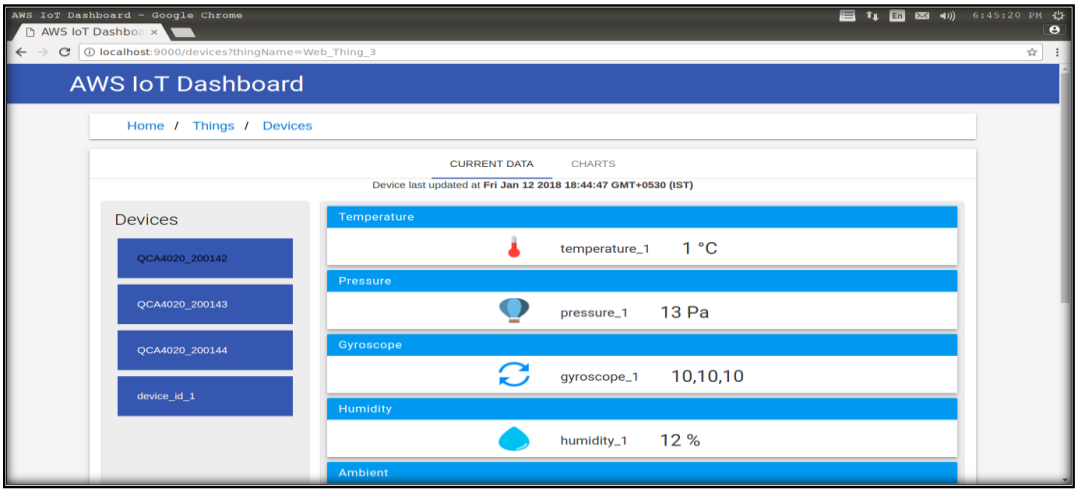


Figure 5-3 View option in the dashboard

5.5.3.2 Update

Clicking the **Update** button displays the update page where the thermostat can be updated.

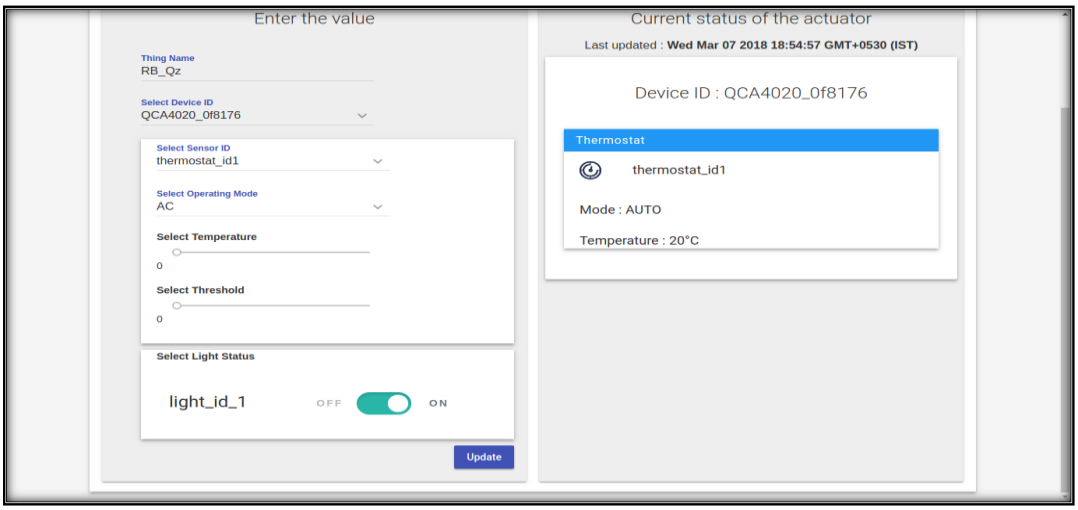


Figure 5-4 Update option in the dashboard

In this page, update the values of the devices connected to the selected thing as follows:

- Update the values for thermostat, dimmer value, and light.

CAUTION: Select the **deviceID** and update with new values.

CAUTION: After seeing the status of the device, update the **sensorid**.

CAUTION: Click **update**. An update request is sent to AWS IoT.

At this stage update request is sent, but the sensor is not updated. The sensor is updated when it accepts the user-given value from AWS IoT. This is reported by the device waiting for the update request.

5.5.3.3 Live graph

This section provides the live updates from the paired device sensor values. In case the data is not accurate, manually read the page to check the updated sensor values from the paired devices.

The live section shows the data for respective sensor after fetching the values from the AWS shadow. The sensor data, that is, updates from shadow for each of the onboard sensors is directly streamed into the visualization library. Specify the device name and type of sensor to view the live chart graphs.

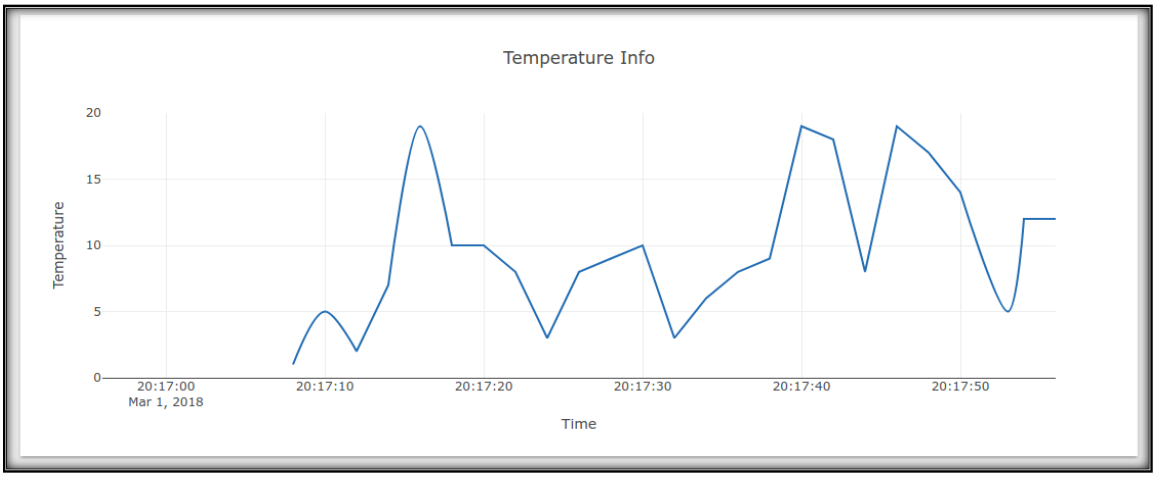
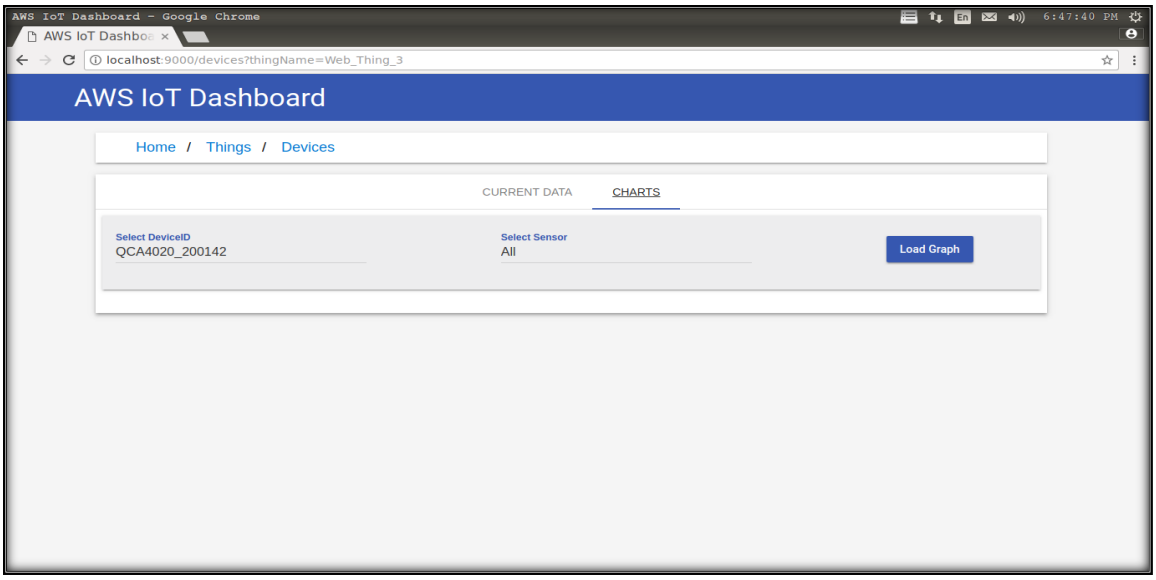


Figure 5-5 Temperature sensor live graph from a selected paired device

5.5.3.4 Charts

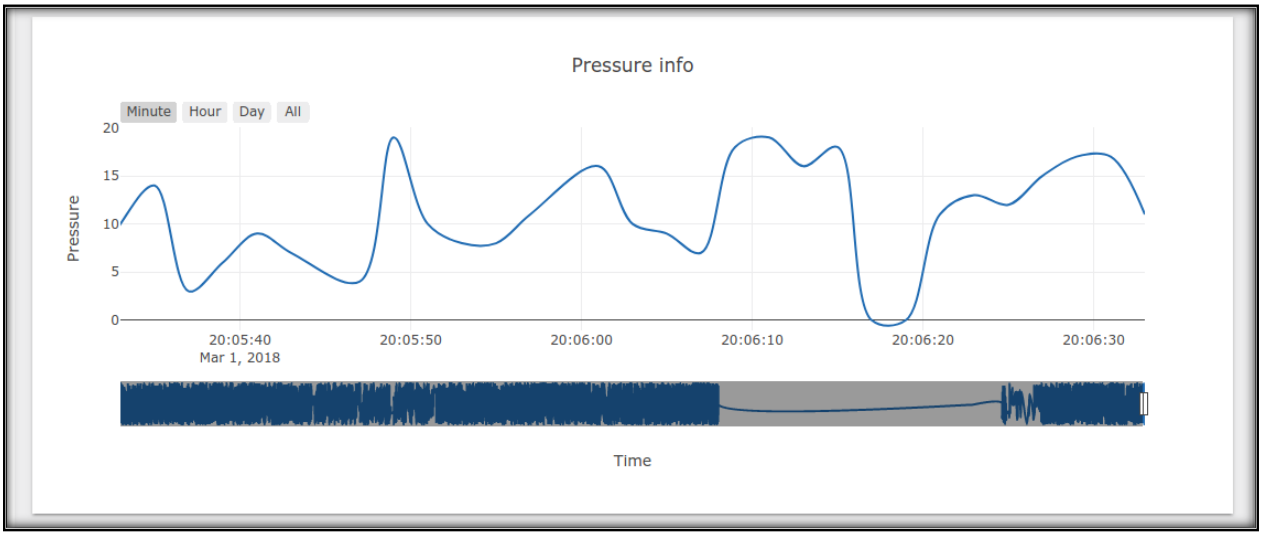
This section is found under the **View** tab for a device. The graphical view of data over a period for a device, and a sensor type or for all sensor types in a device can be seen here. Two filters are provided, one for selecting device, and another for selecting the sensor type. First select the device and then the sensor type. If **All** is selected in the sensor type, temperature, pressure, humidity, and ambient graphs are loaded for the selected device.



CAUTION: The data shown in the following graphs is loaded from MongoDB stored over a period.

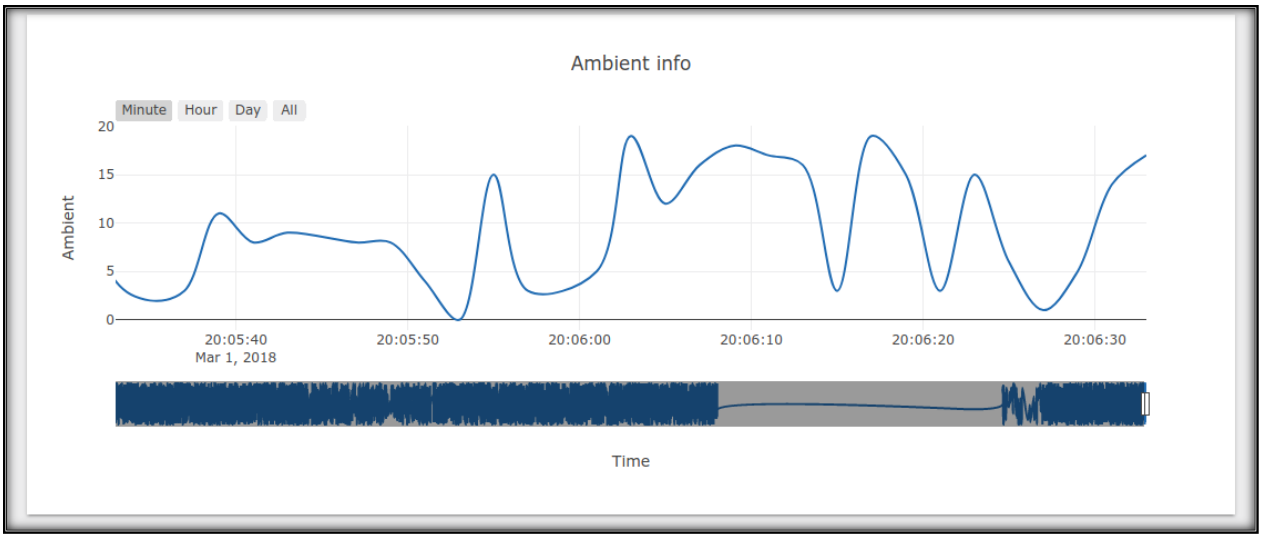
Pressure sensor graph

The pressure sensor graph is plotted with pressure vs. time for multiple pressure sensors in a device. The x-axis represents time and y-axis represents pressure in Pascal (Pa).



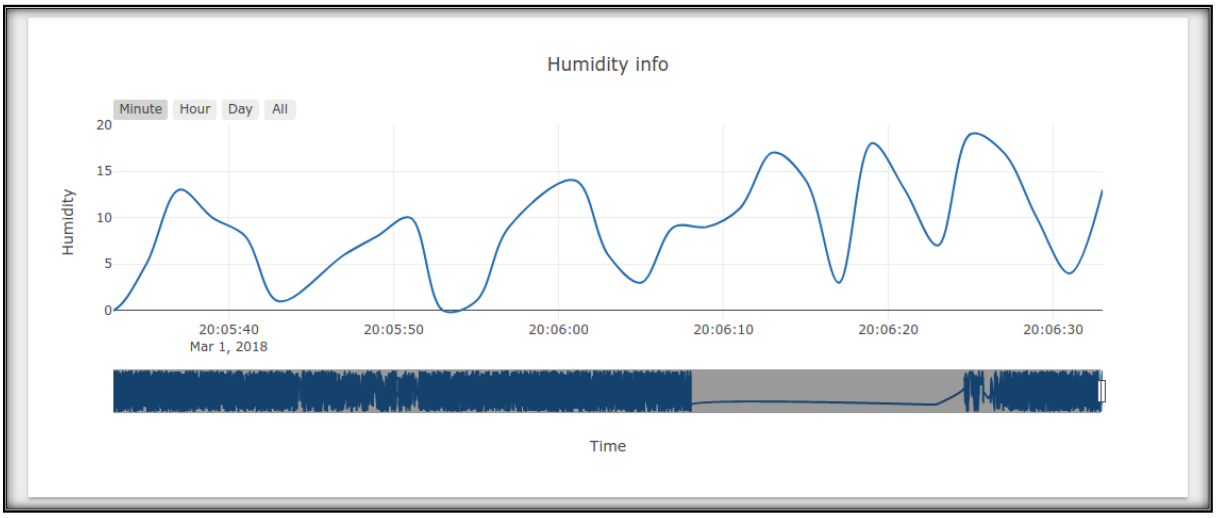
Ambient sensor graph

The ambient sensor graph is plotted with luminosity vs. time flux for multiple ambient sensors in a device. The x-axis represents time and y-axis represents luminosity flux in Lux (lx).



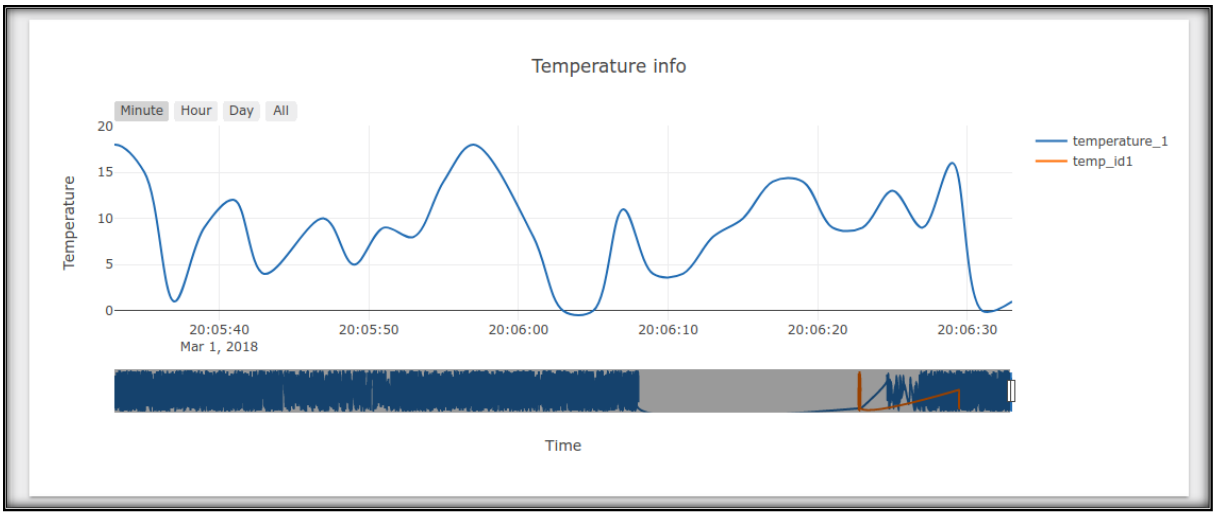
Humidity graph

The humidity graph is plotted with humidity vs. time for multiple humidity sensors in a device. The x-axis represents time and y-axis represents humidity in percentage (**%**).



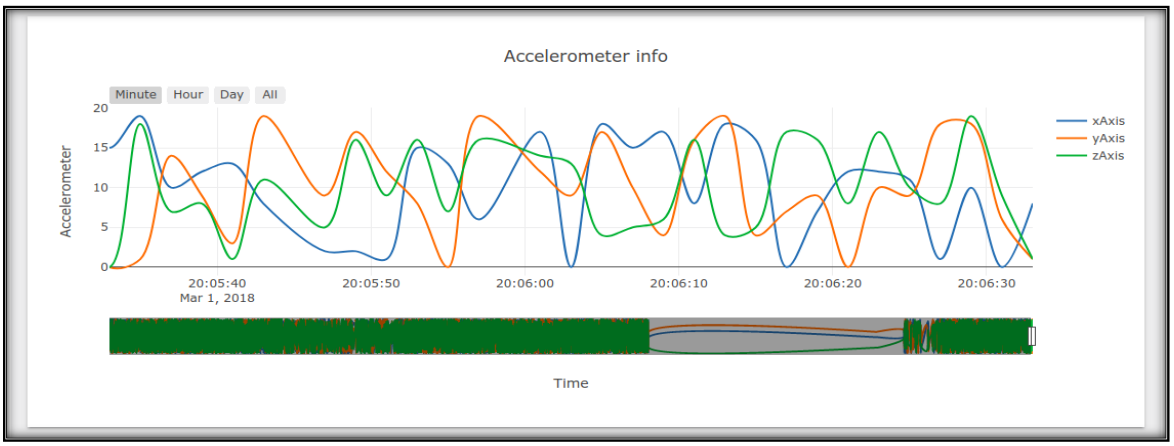
Temperature graph

The temperature graph is plotted with temperature vs. time for multiple temperature sensors in a device. The x-axis represents time and y-axis represents temperature in degree celsius (°C).



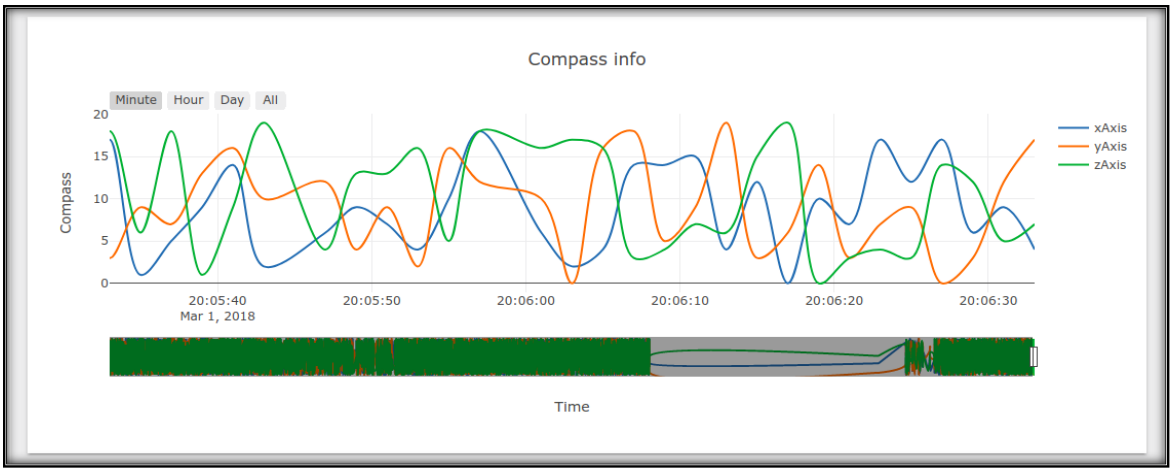
Accelerometer graph

The accelerometer graph is plotted with accelerometer vs. time for accelerometer sensors in a device.



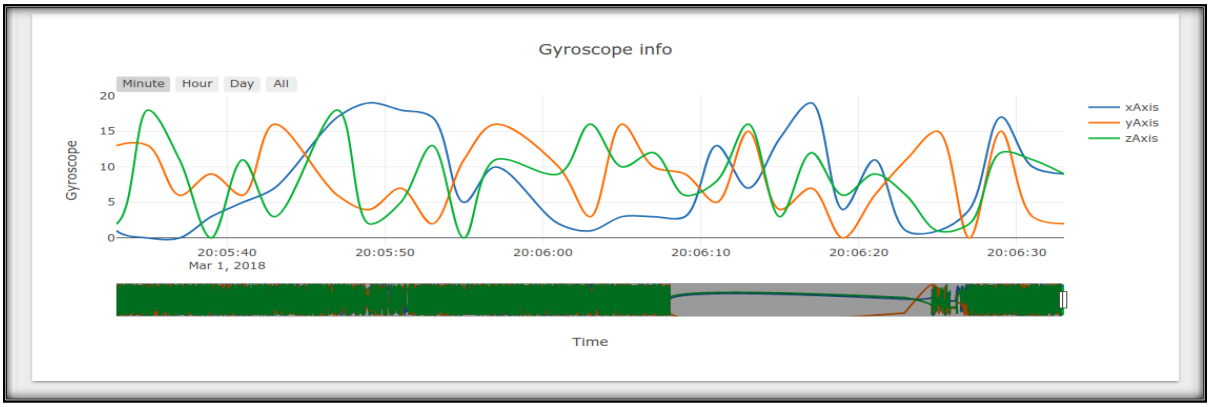
Compass Graph

The compass graph is plotted with compass vs. time for compass sensors in a device.



Gyroscope Graph

The gyroscope graph is plotted with gyroscope vs. time for gyroscope sensors in a device.



6 QCLI demo

QCLI is a CLI-based demo application that demonstrates different features and technologies supported by QCA402x platform. It also provides reference implementation and usage of customer facing QAPIs.

The QCLI command framework is implemented as a set of nested menus. Each menu provides a list of sub-commands that correspond to a feature or technology.

Two example command lists are given here − the first being at the top-level and the second after navigating into the WLAN group.

- Top-level command list

```
Commands:
   1. Help
   2. Exit
Subgroups:
   3. HMI
   4. WLAN
   5. Net
```

- WLAN group command list

```
Commands:
   1. Help
   2. Up
   3. Root
   4. Enable
   5. Connect
   6. Disconnect
Subgroups:
   7. SupPart1
   8. SupPart2
```

The command list will start with the list of commands available at the current level. The top of this list will be the QCLI internal commands. After the commands, the available groups will be listed. If there are no subgroups available, this section will not exist.

6.1 Configuring QCLI demo

The QCLI demo provides a CLI menu for all the features and technologies supported on QCA402x. A user may choose to disable certain features (for example, due to lack of flash space). For this purpose, a configuration file is provided in the SDK that allows configuring each feature to "true" or "false".

For example:

- To disable the Zigbee feature, edit the **env.config** file:

  Change: *'CFG_FEATURE_ZIGBEE=true'*
  To: '*CFG_FEATURE_ZIGBEE=false'*

CAUTION: To apply the new configuration-

- On Windows, run *'setenv.bat'*.
- On Linux, run '*setenv.sh'*.

CAUTION: Rebuild the demo with the new configuration.

6.2 QCLI internal commands

6.2.1 Help

This command can be used to display a command list or usage for a command. It will be the first command in the list for every menu level.

6.2.2 Exit

This command is used to quit the demo, but its operation is platform-specific. For platforms which the demo is running as an app (such as Windows or Linux), this command will typically exit the application and for embedded platforms, it will typically reset the processor. This command is only available from the root command group (when the prompt is only '>').

6.2.3 Up

This command is used to navigate up one group level. For instance, if the command is executed from within the "MyDemo\SubPart1" command group, QCLI will navigate to the "MyDemo" command group. This command is only available while within a group or subgroup (not at the root level).

6.2.4 Root

This command is used to immediately navigate to the root group level. It is only available within a group or subgroup.

6.3 QCLI subgroups

6.3.1 BLE

The BLE demo is intended to demonstrate the operation of the Qualcomm® Bluetopia™ (Bluetooth Protocol Stack) and how to use its APIs.

6.3.1.1 InitializeBluetooth

This command initializes the Bluetooth Protocol Stack and must be called before any other commands within the BLE demo.

This command will initialize the GATT Profile and register/configure the following GATT-based services: Device Information Service (DIS), Generic Access Profile Service (GAPS), and Transmit Power Service (TPS).

If initialization is successful, the command will print out the initialization information of the Bluetooth Protocol Stack. This includes information like the Bluetooth Stack ID of the Bluetooth Protocol Stack that has been initialized, the local Bluetooth address, and the Bluetooth controller version.

6.3.1.2 ShutdownBluetooth

This command shuts down the Bluetooth Protocol Stack.No command can be called within the BLE demo after this command is used.

6.3.1.3 QueryVersion

This command queries the version of the Bluetooth Controller.

6.3.1.4 SetBLERadio

This command selects the BLE radio used by the Bluetooth Protocol Stack.

Parameter Name	Type/Range	Description
Radio	1-2	Selects the BLE radio.

6.3.1.5 SetDiscoverabilityMode

This command sets the GAP LE Discoverability Mode that is used when the local device is advertising. The default GAP LE Discoverability Mode is set to ‘General Discoverable’ by default when the Bluetooth Protocol Stack is initialized.

Parameter Name	Type/Range	Description
Mode	(0 = Non-Discoverable, 1 = Limited Discoverable, 2 = General Discoverable)	Sets the GAP LE Discoverability Mode.

6.3.1.6 SetConnectabilityMode

This command sets the GAP LE Connectability Mode that is used when the local device is advertising. The default GAP LE Connectability Mode is set to ‘Connectable’ by default when the Bluetooth Protocol Stack is initialized.

Parameter Name	Type/Range	Description
Mode	(0 = Non-Connectable, 1 = Connectable, 2 = Direct Connectable)	Sets the GAP LE Connectability Mode.

6.3.1.7 SetPairabilityMode

This command sets the GAP LE Pairability Mode that will be used when the local device is advertising. The default GAP LE Pairability Mode is set to ‘Pairable w/Secure Connections’ by default when the Bluetooth Protocol Stack is initialized.

‘Pairable w/Secure Connections’ is also referred to as ‘Pairable w/Extended Events’

Parameter Name	Type/Range	Description
Mode	(0 = Non-Pairabile, 1 = Pairable, 2 = Pairable w/Secure Connections)	Sets the GAP LE Pairablility Mode.

6.3.1.8 ChangePairingParameters

This command sets GAP LE Pairing Parameters that are used during pairing.

Parameter Name	Type/Range	Description
I/O Capability	(0 = DisplayOnly, 1 = Display Yes/No, 2 = Keyboard Only, 3 = No Input/Output, 4 = Keyboard/Display)	Sets the GAP LE Input/Output Capability of the local device.
MITM Required	(0 = No, 1 = Yes)	Sets if Man in the Middle (MITM) is required during pairing.
Secure Connections Required	(0 = No, 1 = Yes)	Sets if Secure Connection is required during pairing.

The default GAP LE I/O Capability is set to ‘No Input/Output’ by default when the Bluetooth Protocol Stack is initialized.

The default GAP LE MITM requirement is set to ‘Yes’ by default when the Bluetooth Protocol Stack is initialized.

The default GAP LE Secure Connections requirement is set to ‘Yes’ by default when the Bluetooth Protocol Stack is initialized. For pairing of secure connections to take place, both the local and remote devices must support Bluetooth 4.2 and the GAP LE Pairability Mode must be set to ‘Pairably w/Secure Connections’.

6.3.1.9 LEPasskeyResponse

This command clears/sets the fixed GAP LE Passkey that is used during pairing if a passkey response is required.

Parameter Name	Type/Range	Description
Clear/Set	(0 = Clear, 1 = Set)	Clears the current GAP LE Passkey or sets the GAP LE Passkey.
6 Digit Passkey	(000000-999999)	Sets the GAP LE passkey that will be used during pairing. This is required if the 'Clear/Set' parameter is set to '1'.

6.3.1.10 GetLocalAddress

This command queries the Bluetooth address of the local Bluetooth controller.

6.3.1.11 AdvertiseLE

This command enables/disables GAP LE Advertising. If advertising is enabled, this command is used to enable un-directed or directed advertising.

Parameter Name	Type/Range	Description
Enable/Disable	(0 = Disable, 1 = Enable)	Enables or disables un-directed or directed GAP LE Advertising.
Direct BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the target remote device that will receive directed advertisements. This is optional (see the following).

The GAP LE Connectability Mode must be set to 'Direct Connectable' to use direct advertising. If this is not the case, then this command reverts to the undirect advertising.

If the optional Direct BD_ADDR is specified and the preceding requirements for directed advertising are met, then this command enables the directed advertising.

6.3.1.12 ScanLE

This command enables/disables GAP LE Scanning.

Parameter Name	Type/Range	Description
Enable/Disable	(0 = Disable, 1 = Enable)	Enables or disables GAP LE Scanning.
Filter Policy	(0=No Filter, 1=White List, 2=No White List Directed RPA, 3=White List Directed RPA)	The Filter Policy that will be used to filter advertisement. This is optional. RPA – Resolvable Private Address.

This command uses the Filter Policy of 'No Filter' if the Filter Policy is not optionally specified by the Filter Policy parameter.

If the Filter Policy parameter is specified, then it can be used to filter advertisements based on remote devices that have been added to the White List and/or Resolving List, in the Bluetooth controller.

6.3.1.13 ConnectLE

This command sends a GAP LE connection request to a remote device.

Parameter Name	Type/Range	Description
Use White List	(0 = Disable, 1 = Enable)	Enables or disables using the White List in the Bluetooth Controller for connecting to a remote device.
BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device that will receive the connection request. This is optional.
ADDR Type	(0 = Public, 1 = Random, 2 = Public Identity, 3 = Random Identity)	The Bluetooth address type of the remote device that will receive the connection request. This is optional.

If the Use White List parameter is enabled, then the BD_ADDR and ADDR Type parameters will not be used. The first Bluetooth address of a remote device, found in the White List, in the Bluetooth controller, will receive the connection request.

If the Use White List parameter is disabled, the BD_ADDR and ADDR Type parameters must be used to specify the remote device that will receive the connection request.

The following GATT-based services: AIOS, BAS, HOGP, SCPS, and SPPLE cannot be registered/un-registered after a connection has been established with a remote device.

If a connection request is successful, then the global remote device connection count will be incremented, and the selected remote device will be updated to the remote device that just connected. This behavior has been implemented since the user will want to perform commands for the remote device that just connected. An example would be the 'PairLE' command to pair with the remote device after connecting (if security is not re-established) or the 'DiscoverServices' command to discover GATT-based services supported by the remote device.

The selected remote device may be explicitly updated with the 'SelectRemoteDevice' command.

6.3.1.14 DisconnectLE

This command disconnects the selected remote device.

If a remote device is connected, then the selected remote device will always be set to the address of the remote device that is currently selected.

If a disconnection request is successful, then the global remote device connection count will be decremented. If there are still remote devices that are connected, then the selected remote device will be updated to the first connected remote device that is found. The selected remote device may be explicitly updated with the 'SelectRemoteDevice' command.

6.3.1.15 CancelConnectLE

This command cancels an outstanding connection request that has been sent to a remote device.

6.3.1.16 DiscoverServices

This command starts a process to discover all GATT services on the selected remote device. A GATT service must be discovered before read/write requests can be issued for the Characteristics and Descriptors of a service.

6.3.1.17 PairLE

This command starts pairing with the selected remote device.

By default, the Long-Term Key (LTK) and Identity Resolving Key (IRK) will be exchanged during pairing.

If the IRK has been exchanged, then the identity information has also been exchanged. This includes the Identity Address and Identity Address type. These are stored in the remote device information and are required to add a remote device to the Resolving List in the Bluetooth Controller.

Another pairing process may not be started until the current pairing process has successfully completed or timed out.

6.3.1.18 SelectRemoteDevice

This command explicitly sets a connected remote device as the selected remote device. This device is used by many BLE commands as the target remote device since multiple remote devices may be connected.

Parameter Name	Type/Range	Description
GATT Connection ID	(Positive, non-zero)	The GATT Connection ID for the connection with the remote device that will be set as the selected remote device.

To determine the GATT Connection ID of a connected remote device, the 'DisplayRemoteDevices' command is used. If the GATT Connection ID displayed is zero, then the remote device is not connected (GATT Connection IDs are unique, positive, and non-zero). The GATT Connection ID is received with the GATT Connection event.

If a connection request is successful, then the global remote device connection count will be incremented, and the selected remote device is updated to the remote device that connected. This behavior is implemented since the user wants to perform commands for the remote device that connected. An example would be the 'PairLE' command to pair with the remote device after connecting (if security is not re-established) or the 'DiscoverServices' command to discover GATT-based services supported by the remote device.

If a disconnection request is successful, then the global remote device connection count is decremented. If there are still remote devices that are connected, then the selected remote device is updated to the first connected remote device that is found.

6.3.1.19 DisplayRemoteDevices

This command displays information about each remote device that is stored in the remote device information.

6.3.1.20 AddDeviceToWhiteList

This command adds a remote device to the White List in the Bluetooth controller.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device that is added to the White List in the Bluetooth controller.
ADDR Type	(0 = Public, 1 = Random, 2 = Public Identity, 3 = Random Identity)	The Bluetooth address type of the Bluetooth address that is added to the White List in the Bluetooth controller.

6.3.1.21 RemoteDeviceFromWhiteList

This command removes a remote device from the White List in the Bluetooth controller.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device that was previously added to the White List in the Bluetooth controller.

6.3.1.22 AddDeviceToResolvingList

This command adds a remote device to the Resolving List in the Bluetooth controller.

The remote device's identity information and Identity Resolving Key (IRK) must have been exchanged during pairing before this command can be used.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device that is added to the Resolving List in the Bluetooth controller.

6.3.1.23 RemoteDeviceFromWhiteList

This command removes a remote device from the Resolving List in the Bluetooth controller.

The remote device's identity information and Identity Resolving Key (IRK) must have been exchanged during pairing before this command can be used.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device that was previously added to the Resolving List in the Bluetooth controller.

6.3.1.24 SetAuthPayloadTimeout

This command sets the authentication payload timeout for a specified remote device.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device.
Timeout	Positive, nonzero	The specified timeout in milliseconds.

6.3.1.25 GetGATTMTU

This command queries the maximum GATT MTU.

6.3.1.26 SetGATTMTU

This command sets the maximum GATT MTU.

Parameter Name	Type/Range	Description
MTU	(23-517)	The MTU that will be set as the maximum GATT MTU.

6.3.1.27 GetScanParameters

This command queries the current GAP LE scan parameters.

6.3.1.28 SetScanParameters

This command sets the GAP LE scan parameters.

Parameter Name	Type/Range	Description
Scan Interval	Positive, nonzero	The scan interval in milliseconds.
Scan Windows	Positive, nonzero	The scan window in milliseconds.

6.3.1.29 GetAdvertisingParameters

This command queries the current GAP LE advertising parameters.

6.3.1.30 SetAdvertisingParameters

This command sets the GAP LE advertising parameters.

Parameter Name	Type/Range	Description
Minimum Advertising Interval	Positive, nonzero	The minimum advertising interval in milliseconds.
Maximum Advertising Interval	Positive, nonzero	The maximum advertising interval in milliseconds.

6.3.1.31 GetConnectionParameters

This command queries the current GAP LE connection parameters.

6.3.1.32 SetConnectionParameters

This command sets the GAP LE connection parameters.

Parameter Name	Type/Range	Description
Minimum Advertising Interval	Positive, nonzero	The minimum advertising interval in milliseconds.
Maximum Advertising Interval	Positive, nonzero	The maximum advertising interval in milliseconds.
Slave Latency	Positive, nonzero	The number of connection events

6.3.1.33 AIOS

The following commands are for the GATT-based Automation Input/Output Service (AIOS). The demo may be configured as an AIOS Server by using the 'RegisterAIOS' command. Otherwise, the demo functions as an AIOS Client.

6.3.1.33.1 RegisterAIOS

This command registers an AIOS instance. This command cannot be used while connected to a remote device.

6.3.1.33.2 UnregisterAIOS

This command un-registers an AIOS instance. This command cannot be used while connected to a remote device.

6.3.1.33.3 ConfigureRemoteAIOS

This command configures notifications for an Analog or Digital Characteristic instance on an AIOS Server.An AIOS client must configure a characteristic instance for notifications before it can receive notifications from an AIOS Server.

Parameter Name	Type/Range	Description
Characteristic Type	(0= Digital, 1 = Analog)	The Characteristic type.
Characteristic ID	(0-1)	The Characteristic instance ID.
Configure Notifications	(0= Disable, 1 = Enable)	Enables or disables notifications.

6.3.1.33.4 ReadCharacteristic

This command reads an Analog or Digital Characteristic instance on an AIOS Server.

Parameter Name	Type/Range	Description
Characteristic Type	(0= Digital, 1 = Analog)	The Characteristic type.
Characteristic ID	(0-1)	The Characteristic instance ID.

6.3.1.33.5 ReadPresentationFormat

This command reads an Analog or Digital Characteristic instance's Presentation Format Descriptor on an AIOS Server.

Parameter Name	Type/Range	Description
Characteristic Type	(0= Digital, 1 = Analog)	The Characteristic type.
Characteristic ID	(0-1)	The Characteristic instance ID.

6.3.1.33.6 ReadNumberOfDigitals

This command reads a Digital Characteristic instance's Number of Digitals Descriptor on an AIOS Server.

Parameter Name	Type/Range	Description
Characteristic ID	(0-1)	The Characteristic instance ID.

6.3.1.33.7 WriteDigitalOutput

This command writes a Digital (Output) Characteristic instance's value on an AIOS Server.

Parameter Name	Type/Range	Description
Characteristic ID	(0-1)	The Characteristic instance ID.
Digital Octet 1	UINT8	The first four digital signals. Each digital signal is a 2-bit value.
Digital Octet 2	UINT8	The last four digital signals. Each digital signal is a 2-bit value.

6.3.1.33.8 WriteAnalogOutput

This command writes an Analog (Output) Characteristic instance's value on the AIOS Server.

Parameter Name	Type/Range	Description
Characteristic ID	(0-1)	The Characteristic instance ID.
Analog Value	UINT16	The analog value.

6.3.1.33.9 SetDigitalInput

This command allows the AIOS server to set the value for a Digital (Input) Characteristic instance.

Parameter Name	Type/Range	Description
Characteristic ID	(0-1)	The Characteristic instance ID.
Digital Octet 1	UINT8	The first four digital signals. Each digital signal is a 2-bit value.
Digital Octet 2	UINT8	The last four digital signals. Each digital signal is a 2-bit value.

6.3.1.33.10 SetAnalogInput

This command allows an AIOS server to set the value for an Analog (Input) Characteristic instance.

Parameter Name	Type/Range	Description
Characteristic ID	(0-1)	The Characteristic instance ID.
Analog Value	UINT16	The analog value.

6.3.1.33.11 NotifyAIOSCharacteristic

This command allows the AIOS Server to send a notification for a Digital or Analog Characteristic instance's value to a remote AIOS Client. An AIOS client must configure a characteristic instance for notifications before it can receive notifications from an AIOS Server.

Parameter Name	Type/Range	Description
Characteristic Type	(0= Digital, 1 = Analog)	The Characteristic type.
Characteristic ID	(0-1)	The Characteristic instance ID.

6.3.1.34 BAS

The following commands are for the GATT-based Battery Alert Service (BAS). The demo is configured as a BAS Server by using the 'RegisterBAS' command. Otherwise, the demo will function as a BAS Client.

6.3.1.34.1 RegisterBAS

This command registers a BAS instance. This command cannot be used while connected to a remote device.

6.3.1.34.2 UnregisterBAS

This command un-registers a BAS instance. This command cannot be used while connected to a remote device.

6.3.1.34.3 ConfigureRemoteBAS

This command configures notifications for the Battery Level Characteristic on a BAS Server. A BAS Client must configure the Battery Level Characteristic for notifications before it can receive notifications from a BAS Server.

Parameter Name	Type/Range	Description
Characteristic Type	(0= Digital, 1 = Analog)	The Characteristic type.
Characteristic ID	(0-1)	The Characteristic instance ID.
Configure Notifications	(0= Disable, 1 = Enable)	Enables or disables notifications.

6.3.1.34.4 GetBatteryLevel

This command reads the current value of a Battery Level Characteristic. If the demo is configured to support multiple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being read.

Parameter Name	Type/Range	Description
InstanceID	Positive, nonzero	The Battery Level instance. This is optional.

6.3.1.34.5 SetBatteryLevel

This command sets the value of a Battery Level Characteristic. This function is called by the BAS Client or BAS Server. If the demo is configured to support multiple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being written.

Parameter Name	Type/Range	Description
Battery Level	(0-100)	The Battery Level percentage.
InstanceID	Positive, nonzero	The Battery Level instance. This is optional (see following).

6.3.1.34.6 NotifyBatteryLevel

This command allows a BAS Server to send a notification for a Battery Level Characteristic to a remote AIOS Client.

Parameter Name	Type/Range	Description
Battery Level	(0-100)	The Battery Level percentage.
InstanceID	Positive, nonzero	The Battery Level instance. This is optional (see following).

If the demo is configured to support multiple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being notified.

A BAS Client must configure a Battery Level Characteristic for notifications before it can receive notifications from a BAS Server.

6.3.1.34.7 GetBatteryLevelPresentationFormat

This command reads the Battery Level Characteristic's Presentation Format Descriptor. If the demo is configured to support multiple Battery Level Characteristic instances, then the InstanceID parameter is required to identify the instance that is being read.

Parameter Name	Type/Range	Description
InstanceID	Positive, nonzero	The Battery Level instance. This is optional.

6.3.1.34.8 SetBatteryLevelPresentationFormat

This command sets the value of a Battery Level Presentation Format Descriptor. If the demo is configured to support multiple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being read.

Parameter Name	Type/Range	Description
Namespace	(0x00-0xFF)	The defined namespace.
Description	(0x0000-0xFFFF)	The field that identifies the Battery Level Characteristic instance.
InstanceID	Positive, nonzero	The Battery Level instance. This is optional.

6.3.1.35 GAPS

The following commands are for the GATT-based Generic Access Profile Service (GAPS). The demo is configured as a GAPS Server when the Bluetooth Protocol Stack is initialized by the 'InitializeBluetooth' Command. The demo will also function as a GAPS Client.

6.3.1.35.1 ReadLocalName

This command reads the GAPS device name on the local device.

6.3.1.35.2 SetLocalName

This command sets the GAPS device name on the local device.

Parameter Name	Type/Range	Description
InstanceID	Positive, nonzero	The Battery Level instance. This is optional.

6.3.1.35.3 ReadRemoteName

This command reads the GAPS device name on a GAPS Server.

6.3.1.35.4 ReadLocalAppearance

This command reads the GAPS device appearance on the local device.

6.3.1.35.5 SetLocalAppearance

This command sets the GAPS device appearance on the local device.

Parameter Name	Type/Range	Description
InstanceID	Positive, nonzero	The Battery Level instance. This is optional.

6.3.1.35.6 ReadRemoteAppearance

This command reads the GAPS device appearance on a GAPS Server.

6.3.1.36 HOGP

The following commands are for the GATT-based HID over GATT Service (HOGP). The demo is configured as a HOGP Server by using the 'RegisterHIDS' command. Otherwise, the demo functions as a HOGP Client (HID host).

6.3.1.36.1 RegisterHIDS

This command registers a Human Input Device Service (HIDS) instance. This command cannot be used while connected to a remote device.

6.3.1.36.2 UnregisterHIDS

This command un-registers a Human Input Device Service (HIDS) instance. This command cannot be used while connected to a remote device.

6.3.1.36.3 ConfigureHIDS

This command configures notifications for HIDS Characteristic instance on a HIDS Server.

A HIDS client must configure a characteristic instance for notifications before it can receive notifications from a HIDS Server.

Parameter Name	Type/Range	Description
InstanceID	Positive, non-zero	The HIDS instance identifier.
Report Notify	(0 = Disable, 1 = Enable)	Enables or disables report notifications.

6.3.1.36.4 ReadHIDSConfiguration

This command reads the current configuration for the HIDS Characteristic instance on a HIDS Server.

Parameter Name	Type/Range	Description
InstanceID	Positive, non-zero	The HIDS instance identifier.

6.3.1.36.5 GetReport

This command gets the current specified report for the HIDS Characteristic instance on a HIDS Server.

Parameter Name	Type/Range	Description
InstanceID	Positive, non-zero	The HIDS instance identifier.
Report Type	(1 = Input, 2 = Output, 3 = Feature, 4 = Boot Keyboard Input, 5 = Boot Keyboard Output, 6 = Boot Mouse Input)	The report type to retrieve.
Report ID	(0 = None)	The report ID of the report to retrieve.

6.3.1.36.6 SetReport

This command sets a report for the HIDS Characteristic instance on a HIDS Server.

6.3.1.36.7 SetSuspendMode

This command sets suspend mode on a HIDS Server.

Parameter Name	Type/Range	Description
InstanceID	Positive, non-zero	The HIDS instance identifier.
Suspend	(0 = Exit Suspend, 1 = Suspend)	Starts or exists suspend mode.

6.3.1.36.8 SetProtocolMode

This command sets protocol mode on a HIDS Server.

Parameter Name	Type/Range	Description
InstanceID	Positive, non-zero	The HIDS instance identifier.
Protocol Mode	(0 = Boot, 1 = Report)	Sets the protocol mode.

6.3.1.36.9 SendReport

This command notifies a report to a HIDS Client (HID host).

A HIDS client must configure a report for notifications before it can receive notification from a HIDS Server.

Parameter Name	Type/Range	Description
InstanceID	Positive, non-zero	The HIDS instance identifier.
Protocol Mode	(0 = Boot, 1 = Report)	Sets the protocol mode.

6.3.1.37 OTA

The following commands are for the GATT-based Over The Air (OTA) demo service. The OTA service is a demonstration of performing firmware upgrades over Bluetooth LE.

It can be used as a basis for customers to create their own implementation because this service is manufacturer-specific. The demo may be configured as a demo OTA service Server by using the 'RegisterOTA' command. Otherwise, the demo functions as an OTA client.

6.3.1.37.1 RegisterOTA

This command registers an OTA server instance. The server's role is to host image files that the client can download for firmware updates. The service uses a built-in list of default image files if the "UseDefaultImages" parameter is not set to 0. These default images must exist in the flash file system already for the server to use them. These image names are:

"/spinor/ImgConfig.bin", "/spinor/Quartz_HASHED.elf", "/spinor/ioe_ram_m0_threadx_ipt.mbn"

The "Fs ls" command can list the file system contents. Examples of these files do not exist in the default file system due to their potential size. The demo service may be modified to support different storage formats or means.

Parameter Name	Type/Range	Description
UseDefaultImages	(0= False, 1 = True)	Optional. Defaults to "True" if not provided.

6.3.1.37.2 UnregisterOTA

This command unregisters an OTA instance.

6.3.1.37.3 RegisterImage

This command registers an image file with the OTA server to host.

Parameter Name	Type/Range	Description
FileName	String	Enables or disables notifications.
FilePath	String	Path in the file system to the image.
Version	UINT32	File version to be hosted.
ImageID	UINT32	Optional. Specifies the ID to use; ID must not already be in use

6.3.1.37.4 UnregisterImage

This command unregisters an image file with the OTA server.

Parameter Name	Type/Range	Description
ImageID	UINT16	The Image ID to be unregistered.

6.3.1.37.5 GetRegisteredImages

This command retrieves the information of all registered images that the OTA server is currently hosting.

6.3.1.37.6 DiscoverOTA

This command discovers an OTA service on a remote device from a client.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	Bluetooth address of the remote device (OTA server.)

6.3.1.37.7 QueryOTAImage

This command queries for file updates from an OTA server. The OTA server must be discovered before this command can be run successfully.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	Bluetooth address of the remote OTA server.
FileName	String	Name of the file to be queried for updates.
Version	UINT32	Version number of the client's file. The query will be successful if the server is hosting the file with a higher version number.

6.3.1.37.8 ReadOTAImage

This command reads image data from an OTA server file using a queried image ID. An OTA server must be discovered before this command can be run successfully.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	Bluetooth address of the remote OTA server.
ImageID	UINT32	ID of the OTA server file returned from the query image command.
DataLength	UINT32	Length of the data to read.
FileOffset	UINT32	File offset to read.

6.3.1.38 SCPS

The following commands are for the GATT-based Scan Parameters Service (SCPS). The demo may be configured as an SCPS server by using the 'RegisterSCPS' command. Otherwise, the demo functions as an SCPS client.

6.3.1.38.1 RegisterSCPS

This command registers an SCPS instance. This command cannot be used while connected to a remote device.

6.3.1.38.2 UnregisterSCPS

This command unregisters an SCPS instance. This command cannot be used while connected to a remote device.

6.3.1.38.3 ConfigureRemoteSCPS

This command configures notifications for the Scan Refresh Characteristic on an SCPS Server.

A SCPS client must configure the Scan Refresh Characteristic for notifications before it can receive notifications from an SCPS Server.

Parameter Name	Type/Range	Description
Configure Scan Refresh Notification	(0= Disable, 1 = Enable)	Enables or disables notifications.

6.3.1.38.4 SCPSSetScanIntervalWindow

This command sets the Scan Parameters Characteristic on the SCPS Server.

Parameter Name	Type/Range	Description
LE Scan Interval	UINT16	The scan interval in milliseconds.
LE Scan Window	UINT16	The scan window in milliseconds.

6.3.1.38.5 NotifyScanRefresh

This command notifies the Scan Refresh Characteristic to an SCPS Client. A SCPS client must configure the Scan Refresh Characteristic for notifications before it can receive notification from an SCPS Server.

6.3.1.39 SPPLE

The following commands are for the GATT-based Serial Port Profile over LE (SPPLE). The demo may be configured as an SPPLE Server by using the 'RegisterSPPLE' command. Otherwise, the demo functions as an SPPLE Client.

6.3.1.39.1 RegisterSPPLE

This command registers an SPPLE instance. This command cannot be used while connected to a remote device.

6.3.1.39.2 UnregisterSPPLE

This command unregisters an SPPLE instance. This command cannot be used while connected to a remote device.

6.3.1.39.3 ConfigureSPPLE

This command configures a discovered SPPLE service on an SPPLE Server.

6.3.1.39.4 SendDataCommand

This command sends data to a remote device.

Parameter Name	Type/Range	Description
Number of bytes to send	UINT16	The number of bytes to send to the remote device.

6.3.1.39.5 ReadDataCommand

This command reads data received using SPPLE from a remote device.

6.3.1.39.6 Loopback

This command enables data to be sent back to the remote device (looped back) after it is received.

Parameter Name	Type/Range	Description
Loopback Mode	(0 = Disable, 1 = Enable)	Enables/Disables loopback mode.

6.3.1.39.7 DisplayRawDataMode

This command enables raw data to be displayed when data is received.

Parameter Name	Type/Range	Description
Display Raw Data Mode	(0 = Disable, 1 = Enable)	Enables/Disables Display Raw Data mode.

6.3.1.39.8 AutomaticReadMode

This command enables received data to automatically be read without using the 'ReadDataCommand.'

Parameter Name	Type/Range	Description
Automatic Read Mode	(0 = Disable, 1 = Enable)	Enables/Disables Automatic Read mode.

6.3.1.40 Demo configuration

This section describes the configuration that exists in the BLE demo.

6.3.1.40.1 GAP LE

The following configurations are default for the Generic Access Profile (GAP LE) and are located near the top of the main BLE demo's source file:

Parameter Name	Description
DEVICE_FRIENDLY_NAME	The default local device name. This name will be included in the advertising/scan response data. However, it can be updated using the GAPS 'SetLocalName' command.
DEFAULT_IO_CAPABILITY	The default GAP LE Input/Output Capability that is used for pairing.
DEFAULT_MITM_PROTECTION	The default Man in the Middle (MITM) protection mode that is used for pairing.
DEFAULT_SECURE_CONNECTIONS	The default Secure Connections mode that is used for pairing.

6.3.1.40.2 AIOS

The following configuration is by default for the Automation Input/Output Service (AIOS) and is located near the top of the main BLE demo's source file.

Parameter Name	Description
AIOP_NUMBER_OF_SUPPORTED_CHARACTERISTICS	The number of supported characteristic types for an AIOS instance.
AIOP_NUMBER_OF_SUPPORTED_INSTANCES	The number of supported instances for each characteristic type that is supported by an AIOS instance.
AIOP_DEFAULT_INPUT_CHARACTERISTIC_PROPERTY_FLAGS	The default input characteristic property flags that are used for registering an AIOS instance.
AIOP_DEFAULT_OUTPUT_CHARACTERISTIC_PROPERTY_FLAGS	The default output characteristic property flag that is used for registering an AIOS instance

6.3.1.40.3 BAS

The following configuration is by default for the Battery Alert Service (BAS) and is located near the top of the main BLE demo's source file.

Parameter Name	Description
MAX_SUPPORTED_BATTERY_INSTANCES	The number of supported BAS instances.

6.3.1.40.4 HOGP

The following configuration is by default for the HID over GATT Service (HOGP) and is located near the top of the main BLE demo's source file.

Parameter Name	Description
MAX_SUPPORTED_HID_INSTANCES	The number of supported HIDS instances.
HIDS_MAXIMUM_NUMBER_REPORTS	The maximum number of supported HIDS reports
HIDS_MAXIMUM_NUMBER_EXTERNAL_REPORT_REFERENCES	The maximum number of supported external reports.
HID_KEYBOARD_INPUT_REPORT_SIZE	The size of the application keyboard input report
HID_KEYBOARD_OUTPUT_REPORT_SIZE	The size of the application keyboard output report.
HID_MOUSE_INPUT_REPORT_SIZE	The size of the application mouse input report.

6.3.1.40.5 SPPLE

The following configuration is the default for the Serial Port Profile over LE (SPPLE) and are located near the top of the main BLE demo's source file.

Parameter Name	Description
SPPLE_DATA_BUFFER_LENGTH	The maximum size of the SPPLE data buffer.

6.3.1.41 Usage examples

This section contains examples for some common use cases of the BLE demo. Most of the examples will use two BLE demo devices and each command and printout will be prefixed as Dev1 or Dev2. If a third BLE demo device is used, then it will be labeled Dev3.

Dev1 will use the Bluetooth address: 0x000000000001, Dev2 will use the Bluetooth address: 0x000000000002, and Dev3 will use the Bluetooth address: 0x000000000003. These Bluetooth addresses are being used for demonstration purposes only.

Dev1 will be the master of the BLE connection (initiates the connection) and Dev2/Dev3 will be the slave (advertiser) for the BLE connections with Dev1.

All examples will use the default settings/configurations that are set when the Bluetooth Protocol Stack is initialized. All examples, unless otherwise specified, will assume that all BLE demo devices are not connected.

6.3.1.41.1 Initialization

The following commands demonstrates how to initialize the BLE demo for both devices. This initializes the Bluetooth Protocol Stack, initializes the GATT Profile, and registers the following GATT-based services: DIS, GAPS, and TPS.

Dev1:

```
BLE> InitializeBluetooth
BLE: OpenStack().
BLE: Bluetooth Stack ID: 1.
BLE: DeviceChipset: 4.2.
BLE: BD_ADDR: 0x000000000001
BLE: Connection Tx Power: 7.
```

Dev2:

```
BLE> InitializeBluetooth
BLE: OpenStack().
BLE: Bluetooth Stack ID: 1.
BLE: DeviceChipset: 4.2.
BLE: BD_ADDR: 0x000000000002
BLE: Connection Tx Power: 7.
```

6.3.1.41.2 Connecting

The following commands demonstrate how to establish a connection between Dev1 and Dev2 that have not been previously paired (bonded). If Dev1 and Dev2 have previously bonded, then security will be re-established after successful connection.

All GATT-based services should be registered before a connection is established. A GATT-based service cannot be registered/un-registered once a connection is established.

Since Dev2 is the slave, it must send connectable advertisements before Dev1 can send a connection request. This example will use un-directed connectable advertising, which is the default for the BLE demo.

Dev2:

```
BLE> AdvertiseLE 1
BLE: Advertising Channel Tx Power: 0.
```

```
BLE: GAP_LE_Advertising_Enable success, Advertising Interval Range: 100
- 200.
```

Advertising can be explicitly disabled using the command 'AdvertiseLE 0'. Advertising will automatically be disabled after a connection has been established.

Dev1:

```
BLE> ConnectLE 0 000000000002 0
BLE: Connection Request successful.
BLE: Scan Parameters:      Window 50, Interval 100.
BLE: Connection Parameters Interval Range 50-200, Slave Latency 0.
BLE: Using White List:     No.
```

This does not mean that the connection was successful. The connection request was successfully submitted to the Bluetooth controller. The following events are received on both BLE demo devices after the connection is successfully established.

Dev2:

```
BLE: etLE_Connection_Complete with size 24.
BLE:    Status:              0x00.
BLE:    Role:                Slave.
BLE:    Address Type:        QAPI_BLE_LAT_PUBLIC_E
BLE:    BD_ADDR:             0x000000000001.
BLE:    Connection Interval: 50.
BLE:    Slave Latency:       0

BLE: etGATT_Connection_Device_Connection with size 16:
BLE:    Connection ID:   1.
BLE:    Connection Type: LE.
BLE:    Remote Device:   0x000000000001.
BLE:    Connection MTU:  23.

BLE: Selected Remote Device
BLE:    Address: 0x000000000001
BLE:    ID:      1
```

Dev1:

```
BLE: etLE_Connection_Complete with size 24.
BLE:    Status:              0x00.
BLE:    Role:                Master.
BLE:    Address Type:        QAPI_BLE_LAT_PUBLIC_E
BLE:    BD_ADDR:             0x000000000002.
BLE:    Connection Interval: 50.
BLE:    Slave Latency:       0


BLE: etGATT_Connection_Device_Connection with size 16:
BLE:    Connection ID:   1.
BLE:    Connection Type: LE.
BLE:    Remote Device:   0x000000000002.
```

```
BLE:    Connection MTU:  23.

BLE: Selected Remote Device
BLE:    Address: 0x000000000002
BLE:    ID:      1
```

After a connection is established, the master of the connection (Dev1) will attempt to update the GATT MTU to the maximum supported size (517) from the current MTU for the connection 23 (This is also the minimum).

6.3.1.41.3 Pairing

The following command demonstrates how to pair (bond) Dev1 and Dev2 that are already connected and have not previously bonded. If Dev1 and Dev2 have previously bonded, then security will be re-established after successfully connecting and this command is not necessary.

For this example, the master of the connection (Dev1), will start the pairing procedure. If the slave of the connection (Dev2) calls the command 'PairLE', then a security request will be sent to Dev1 to request that it starts a pairing procedure. The master of the connection will always start the pairing procedure.

Dev1:

```
BLE> PairLE

BLE: Attempting to pair to 0x000000000002.
BLE:   I/O Capability:       No Input No Output.
BLE:   Bonding Type:        Bonding.
BLE:   MITM.                TRUE.
BLE:   Secure Connections:  TRUE.
BLE:   OOB:                 OOB Not Present.
BLE:   Encryption Key Size: 16.
BLE:   Sending Keys:
BLE:      LTK:              Yes.
BLE:      IRK:              Yes.
BLE:      CSRK:             No.
BLE:      Link Key:         No.
BLE:   Receiving Keys:
BLE:      LTK:              Yes.
BLE:      IRK:              Yes.
BLE:      CSRK:             No.
BLE:      Link Key:         No.
BLE:   GAP_LE_Extended_Pair_Remote_Device returned 0.
```

The pairing request has been sent to Dev2.

Dev2:

```
BLE: Authentication with size 60.
BLE: Extended Pairing Request: 0x000000000001.
BLE:   I/O Capability:       No Input No Output.
BLE:   Bonding Type:        Bonding.
```

```
BLE:   MITM.                TRUE.
BLE:   Secure Connections:  TRUE.
BLE:   OOB:                 OOB Not Present.
BLE:   Encryption Key Size: 16.
BLE:   Sending Keys:
BLE:      LTK:              Yes.
BLE:      IRK:              Yes.
BLE:      CSRK:             No.
BLE:      Link Key:         No.
BLE:   Receiving Keys:
BLE:      LTK:              Yes.
BLE:      IRK:              Yes.
BLE:      CSRK:             No.
BLE:      Link Key:         No.
BLE: Sending Pairing Response to 0x000000000001.
BLE: GAP_LE_Authentication_Response returned 0.
```

The pairing response has been sent to Dev1.

Dev1:

```
BLE: etLE_Authentication with size 60.
BLE: latExtendedConfirmationRequest.
BLE:    Secure Connections:     YES.
BLE:    Just Works Pairing:     Yes.
BLE:    Keypress Notifications: No.
BLE: Invoking Just Works.

BLE: etLE_Encryption_Change with size 12.
```

The connection with Dev2 is now encrypted. The LTK and IRK will now be exchanged over the encrypted link.

Dev2:

```
BLE: etLE_Authentication with size 60.
BLE: latExtendedConfirmationRequest.
BLE:    Secure Connections:     YES.
BLE:    Just Works Pairing:     Yes.
BLE:    Keypress Notifications: No.
BLE: Invoking Just Works.

BLE: etLE_Encryption_Change with size 12.
```

The connection with Dev1 is now encrypted. The LTK and IRK will now be exchanged over the encrypted link.

Dev1:

```
BLE: etLE_Authentication with size 60.
BLE:  Encryption Information from RemoteDevice: 0x000000000002
BLE:                              KeySize: 16
```

The LTK will be received in the Encryption Information event and can be used to re-establish security if Dev1 reconnects to Dev2.

Dev2:

```
BLE: etLE_Authentication with size 60.
BLE:  Encryption Information from RemoteDevice: 0x000000000001
BLE:  KeySize: 16
```

The LTK will be received in the Encryption Information event and can be used to re-establish security if Dev2 reconnects to Dev1.

Dev1:

```
BLE: etLE_Authentication with size 60.
BLE: Identity Information from RemoteDevice: 0x000000000002.
```

The IRK, Identity Address, and Identity Address Type will be received in the Encryption Information event and can be used to add Dev2 to the Resolving List in the Bluetooth controller with the command 'AddDeviceToResolvingList'.

Dev2:

```
BLE: etLE_Authentication with size 60.
BLE: Identity Information from RemoteDevice: 0x000000000001.
```

The IRK, Identity Address, and Identity Address Type will be received in the Encryption Information event and can be used to add Dev1 to the Resolving List in the Bluetooth controller with the command 'AddDeviceToResolvingList'.

Dev1:

```
BLE: etLE_Authentication with size 60.
BLE: Pairing Status: 0x000000000002.
BLE: Status: 0x00.
BLE: Key Size: 16.
```

The pairing procedure has successfully completed on Dev1.

Dev2:

```
BLE: etLE_Authentication with size 60.
BLE: Pairing Status: 0x000000000001.
BLE:         Status: 0x00.
BLE:         Key Size: 16.
```

The pairing procedure has successfully completed on Dev2.

6.3.1.41.4 Selecting a remote device

The following command demonstrates how to explicitly update the selected remote device if multiple remote devices are connected.

Most commands will use the selected remote device as the target for the command. An example would be the 'PairLE' command. Some commands may not use the selected remote device. Instead, the remote device address will be explicitly specified as a parameter for the command. An example would be the 'ConnectLE' command.

We will assume that Dev1 and Dev2 are already connected and there is no connection currently between Dev1 and Dev3. Dev1 will have the GATT Connection ID: 1, for the connection with

Dev2. Since the remote device that just connected will automatically become the selected remote device, Dev1 and Dev2 will currently be using each other as the selected remote device.

For this example, Dev1 establishes another connection to Dev3. Dev1 will have the GATT Connection ID: 2, for the connection with Dev3. Dev1 and Dev3 will now be using each other as the selected remote device. It is worth noting that Dev2 is still using Dev1 as the selected remote device. If Dev1 wants to issue a command to Dev2 such as the 'PairLE' command, the selected remote device must be explicitly updated to Dev2 using the 'SelectRemoteDevice' command. The 'SelectRemoteDevice' command takes the GATT Connection ID for the connection with the remote device as a parameter.

Dev1:

```
BLE> SelectRemoteDevice 1

BLE: Selected Remote Device
BLE:    Address: 0x000000000002
BLE:    ID:      1
```

6.3.2 Zigbee

The Zigbee demo is intended to demonstrate the operation of the Zigbee stack and how to use its APIs.

6.3.2.1 Initialize

This command initializes the Zigbee protocol stack and must be called before any other command within the Zigbee demo.

Parameter Name	Type/Range	Description
UsePersist	0-1	If set to 1, the demo will attempt to load persistent data from flash after initializing the stack (defaults to 0).

6.3.2.2 Shutdown

This command shuts down the Zigbee stack.

6.3.2.3 AddDevice

This command adds a device to the Zigbee demo device table which can be used by other commands within the demo to send packets.

The Index of the newly added device will be displayed if the device was added successfully.

Parameter Name	Type/Range	Description
Mode	1-3	The address mode for this device as group (1), network (2), or extended (3).
Address	16 or 64-bit depending on Mode	The address of the device to add. This is a 16-bit value for a group or network address and a 64-bit value for an extended address.
Endpoint	0-0xFF	The endpoint for the device.

6.3.2.4 RemoveDevice

This command removes a device from the Zigbee demo's device table. This function will not compress the remaining devices but will leave the device index unused.

Parameter Name	Type/Range	Description
DevID	Valid device index	The index of the device to be removed. Note that device zero is reserved as a NULL address and cannot be removed.

6.3.2.5 ShowDeviceList

This command lists the devices that are currently in the Zigbee device table.

6.3.2.6 Form

This command tells the Zigbee stack to form a Zigbee network, essentially becoming a HUB device. Other devices can then join this network.

A callback will occur when the network formation completes which displays the status of the form (0 is success) and the channel the network started on. The network address of the device will always be zero.

Parameter Name	Type/Range	Description
UseSecurity	0–1	Flag indicating if security is used (1) or not (0). Note that address zero is reserved as NULL address for sending packets using the binding table.
Distributed	0–1	Flag indicating if this should form a centralized (0) or distributed (1) network (optional: defaults to 0/Distributed).
Channel	11–26	Optional parameter to specify the channel to form the network on. If not specified, the FORM_CHANNEL MASK value is used.

6.3.2.7 Join

This command tells the Zigbee stack to join or rejoin a Zigbee network. The rejoin operations are only intended after the device has left the network and wishes to rejoin.

A callback will occur when the network join completes which displays the status of the join (0 for success), the network address assigned to the device, the extended PAN ID of the network, and the channel of the network.

Parameter Name	Type/Range	Description
AsCoordinator	0-1	Flag indicating if the device should join as a coordinator (1) or end device (0).
UseSecurity	0-1	Flag indicating if security is used (1) or not (0).
IsRejoin	0-1	Optional flag indicating if this is a Rejoin operation (1) or a join operation (0). Default is a join operation (0).
Channel	11-26	Optional parameter to specify the channel to join the network on. If not specified, the JOIN_CHANNEL_MASK value will be used.

6.3.2.8 Reconnect

Reconnects to a network after communication with the parent device has been lost.

6.3.2.9 Leave

This command tells the Zigbee stack to leave the network.

6.3.2.10 LeaveReq

This command sends a leave request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the leave request to.
LeaveDevId	Valid device index	Device ID of that should leave the network.
RemoveChildren	0–1	Flag indicating if the device should remove its children.
Rejoin	0–1	Flag indicating if the device should rejoin or not.

6.3.2.11 PermitJoin

This command tells the network to permit new devices to join for a given period. A value of zero effectively disables joining.

Parameter Name	Type/Range	Description
Duration	0–255	Time in seconds for join to be enabled.

6.3.2.12 GetNIB

This command reads a Network Information Base attribute.

Parameter Name	Type/Range	Description
AttrId	16-bit value	NWK Attribute ID to read.
AttrIndex	8-bit value	Index of the attribute to read (defaults to 0).
MaxLength	16-bit value	Maximum expected length of the attribute to read (defaults to 128 bytes).

6.3.2.13 SetNIB

This command writes a Network Information Base attribute.

Parameter Name	Type/Range	Description
AttrId	16-bit value	NWK Attribute ID to write.
AttrIndex	8-bit value	Index of the attribute to write.
Length	16-bit value	Length of the attribute in bytes.
Value	Hex string	Attribute to write. The number of bytes in the hexadecimal string should match the specified length.

6.3.2.14 GetAIB

This command reads an APS Information Base attribute.

Parameter Name	Type/Range	Description
AttrId	16-bit value	APS Attribute ID to read.
AttrIndex	8-bit value	Index of the attribute to read (defaults to 0).
MaxLength	16-bit value	Maximum expected length of the attribute to read (defaults to 128 bytes).

6.3.2.15 SetAIB

This command writes an APS Information Base attribute.

Parameter Name	Type/Range	Description
AttrId	16-bit value	APS Attribute ID to write.
AttrIndex	8-bit value	Index of the attribute to write.
Length	16-bit value	Length of the attribute in bytes.
Value	Hex string	Attribute to write. The number of bytes in the hexadecimal string should match the specified length.

6.3.2.16 GetBIB

This command reads a BDB Information Base attribute.

Parameter Name	Type/Range	Description
AttrId	16-bit value	BDB Attribute ID to read.
AttrIndex	8-bit value	Index of the attribute to read.
MaxLength	16-bit value	Maximum expected length of the attribute to read.

6.3.2.17 Set BIB

This command writes a BDB Information Base attribute.

Parameter Name	Type/Range	Description
AttrId	16-bit value	BDB Attribute ID to write.
AttrIndex	8-bit value	Index of the attribute to write.
Length	16-bit value	Length of the attribute in bytes.
Value	64-bit value	Attribute to write. The range of the value is dependent on the length specified.

6.3.2.18 SetExtendedAddress

This command sets the extended address of the device.

Parameter Name	Type/Range	Description
ExtAddr	64-bit value	Extended address to be set

6.3.2.19 GetAddresses

This command reads the NWK and extended address of the device.

6.3.2.20 ClearPersist

This command clears Zigbee persistent data from flash.

6.3.2.21 ZDP commands

6.3.2.21.1 NWKAddr

This command sends a ZDP network address request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the network address request to.
ExtAddr	64-bit value	Extended address of the device for which the network address is being requested.
RequestType	0-1	Type of request as either single (0) or extended (1).

6.3.2.21.2 IEEEAddr

This command sends a ZDP IEEE address request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the IEEE address request to.
NWKAddr	16-bit value	Network address of the device for which the IEEE address is being requested.
RequestType	0-1	Type of request as either single (0) or extended (1).

6.3.2.21.3 NodeDesc

This command sends a ZDP node descriptor request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the node descriptor request to.
NWKAddr	16-bit value	Network address of the node descriptor being requested. If not specified, the target address is used.

6.3.2.21.4 PowerDesc

This command sends a ZDP power descriptor request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the power descriptor request to.
NWKAddr	16-bit value	Network address of the power descriptor being requested. If not specified, the target address is used.

6.3.2.21.5 SimpleDesc

This command sends a ZDP simple descriptor request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the simple descriptor request to.
NWKAddr	16-bit value	Network address of the simple descriptor being requested. If not specified, the address of the target device is used.
Endpoint	8-bit value	Endpoint of the simple descriptor being requested. If not specified, the endpoint of the target device is used.

6.3.2.21.6 ActiveEP

This command sends a ZDP active endpoint request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the active endpoint request to.
NWKAddr	16-bit value	Network address of the active endpoints being requested. If not specified, the target address is used.

6.3.2.21.7 MatchDesc

This command sends a ZDP match descriptor request. To simplify the command line parameters, this command only allows a filter with one server or client cluster ID.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the match descriptor request to.
NWKAddr	16-bit value	Network address for the match descriptor request.
ClusterID	16-bit value	Cluster ID to be matched.
ServerClient	0-1	Flag indicating if the cluster ID specified is for a server (0) or client (1).

6.3.2.21.8 ComplexDesc

This command sends a ZDP complex descriptor request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the complex descriptor request to.
NWKAddr	16-bit value	Network address of the complex descriptor being requested. If not specified, the address of the target device is used.

6.3.2.21.9 UserDesc

This command sends a ZDP user descriptor request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the user descriptor request to.
NWKAddr	16-bit value	Network address of the user descriptor being requested. If not specified, the address of the target device is used.

6.3.2.21.10 UserDescSet

This command sends a ZDP user descriptor set request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the user descriptor ser request to.
NWKAddr	16-bit value	Network address of the user descriptor to set. If not specified, the address of the target device is used.

6.3.2.21.11 SysServerDisc

This command sends a ZDP system server discover request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the system server discover request to.
NWKAddr	16-bit value	Network address for the system server discover request. If not specified, the address of the target device is used.

6.3.2.21.12 ExtSimpleDesc

This command sends a ZDP extended simple descriptor request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the extended simple descriptor request to.
NWKAddr	16-bit value	Network address of the extended simple descriptor being requested. If not specified, the address of the target device is used.
Endpoint	8-bit value	Endpoint of the extended simple descriptor being requested. If not specified, the endpoint of the target device is used.

6.3.2.21.13 ExtActiveEP

This command sends a ZDP extended active endpoint request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the extended active endpoint request to.
NWKAddr	16-bit value	Network address of the extended active endpoints being requested. If not specified, the target address is used.

6.3.2.21.14 EndBind

This command sends a ZDP end bind request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the end bind request to.
SrcDevID	Valid device index	Device ID of the source of the bind. The address mode for this device must be an extended address
ClusterID	0-0xFFFF	The cluster ID for the bind.

Parameter Name	Type/Range	Description
IsServer	0-1	Network address of the extended active endpoints being requested. If not specified, the target address is used.

6.3.2.21.15 Bind

This command creates a binding between two endpoints.A callback occurs when the command completes displaying the status of the operation.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the bind request to. This is typically the source of the bind or its parent. The address mode for this device must be a network address.
SrcDevID	Valid device index	Device ID of the source of the bind. The address mode for this device must be an extended address
DestDevID	Valid device index	Device ID for the destination of the bind. The address mode for this device must be either a group or extended address.
ClusterID	0-0xFFFF	The cluster ID for the bind.

6.3.2.21.16 Unbind

This command removes a binding between two endpoints. A callback will occur when the command completes displaying the status of the operation.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the unbind request to. This is typically the source of the bind or its parent. The address mode for this device must be a network address.
SrcDevID	Valid device index	Device ID of the source of the bind. The address mode for this device must be an extended address
DestDevID	Valid device index	Device ID for the destination of the bind. The address mode for this device must be either a group or extended address.
ClusterID	0-0xFFFF	The cluster ID for the bind.

6.3.2.21.17 MgmtLqi

This command sends a ZDP management LQI request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the management LQI request to.
NWKAddr	16-bit value	Network address whose neighbor table is being requested. If not specified, the target address is used.

6.3.2.21.18 MgmtRtg

This command sends a ZDP management routing table request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the management routing table request to.

Parameter Name	Type/Range	Description
NWKAddr	16-bit value	Network address whose routing table is being requested. If not specified, the target address is used.

6.3.2.21.19 MgmtBindRequest

This command sends a ZDP management binding table request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the management binding table request to.
NWKAddr	16-bit value	Network address whose binding table is being requested. If not specified, the target address is used.

6.3.2.21.20 MgmtLeave

This command sends a ZDP management leave request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the management leave request to.
ExtAddr	64-bit value	IEEE address of the device which should leave the network.
RemoveChildren	0-1	Flag indicating if the device should also remove its children.
Rejoin	0-1	Flag indicating if the device should attempt o rejoin the network.

6.3.2.21.21 MgmtPermitJoining

This command sends a ZDP management permit joining request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the management permit joining request to.
Duration	0-0xFE	Duration for the device to permit joining.
UseTCPolicy	0-1	Flag indicating the request should also change the trust center policy.

6.3.2.21.22 MgmtNWKUpdate

This command sends a ZDP management network update request.

Parameter Name	Type/Range	Description
TargetDevID	Valid device index	Device ID to send the management network update request to.
ScanChannel	0-0x07FFF800	New scan channels value for the network.
ScanDuration	0-5, 254-255	Value of the scan duration for the network update request
ScanCount	0-255	Value for the scan count for the network update request. This value is ignored if the scan duration is 254 or 255.

6.3.2.21.23 SetLocalComplexDesc

Sets the local complex descriptor.

Parameter Name	Type/Range	Description
ManName	String	New value of the manufacturer name for the local device.
ModelName	String	New value of the model name for the local device.
SerialNum	String	New value of the serial number for the local device.

6.3.2.21.24 SetLocalUserDesc

This command sets a local user descriptor.

Parameter Name	Type/Range	Description
Descriptor	String	New value of the local user descriptor.

6.3.2.22 ZCL commands

6.3.2.22.1 ListClusterTypes

Lists the cluster names and ID that are supported by the Zigbee demo.

6.3.2.22.2 ListEndpointTypes

Lists the types of endpoints that can be created by the Zigbee demo.

6.3.2.22.3 CreateEndpoint

Creates an endpoint, including all clusters that the endpoint needs.

Parameter Name	Type/Range	Description
EndpointNumber	1–240	Endpoint to create.
EndpointType	1–13	Type of endpoint to create

6.3.2.22.4 RemoveEndpoint

Removes an endpoint and all clusters associated with it.

6.3.2.22.5 ListClusters

This command displays the current list of registered clusters. The IDs from this list can be used with the other generic cluster commands.

ReadLocalAttribute

Parameter Name	Type/Range	Description
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute
AttrId	0-0xFFFF	ID of the attribute
AttrLength	0–8	Length of the attribute

6.3.2.22.6 WriteLocalAttribute

Parameter Name	Type/Range	Description
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute
AttrId	0-0xFFFF	ID of the attribute
AttrLength	0–8	Length of the attribute
AttrValue	64-bit	Value to be written

6.3.2.22.7 ReadAttribute

This command is used to read a cluster attribute from a remote device.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the command to. Note that device zero can be specified to use the binding table.
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute
AttrId	0-0xFFFF	ID of the attribute
AttrValue	0–8	Length of the attribute

6.3.2.22.8 WriteAttribute

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the command to. Note that device zero can be specified to use the binding table.
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute
AttrId	0-0xFFFF	ID of the attribute
AttrType	0-0xFF	Type of the attribute
AttrLength	0–8	Length of the attribute
AttrValue	64-bit	Value to be written

6.3.2.22.9 ConfigReport

This command is used to configure attribute reporting on a remote device.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the command to. Note that device zero can be specified to use the binding table.
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute
AttrId	0-0xFFFF	ID of the attribute
AttrType	0-0xFF	Type of the attribute
MinInterval	0-0xFFFF	Minimum reporting interval for the attribute
MaxInterval	0-0xFFFF	Maximum reporting interval for the attribute
ChangeValue	64-bit	Value to be written

6.3.2.22.10 ReadReportConfig

This command is used to read the attribute reporting configuration on a remote device.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the command to. Note that device zero can be specified to use the binding table.
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute
AttrId	0-0xFFFF	ID of the attribute

6.3.2.22.11 DiscoverAttribute

This command is used to discover attributes on a remote device.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the command to. Note that device zero can be specified to use the binding table.
ClusterIndex	Valid cluster index	ID of the local cluster (in the cluster list) associated with the attribute

6.3.2.23 Basic

6.3.2.23.1 Reset

This command sends a reset request to a basic server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the reset to.
ClientEndpoint	Valid endpoint	Endpoint that contains the basic client cluster to use to send the command.

6.3.2.23.2 Read

Reads an attribute from the local basic server cluster.

Parameter Name	Type/Range	Description
AttrId	16-bit value	ID of the attribute to read.

6.3.2.23.3 Write

Writes an attribute to the local basic server cluster.

Parameter Name	Type/Range	Description
AttrId	16-bit value	ID of the attribute to write.
Value	8-bit value or string	Value to write to the attribute. It is either a byte or a string depending on which the attribute is being written.

6.3.2.24 Identify

6.3.2.24.1 Identify

This command sends an identify request to an Identify server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Identify request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Identify client cluster to use to send the command.
Time	16-bit value	The time for the device to identify itself

6.3.2.24.2 IdentifyQuery

This command sends an Identify Query request to an Identify server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Identify query request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Identify client cluster to use to send the command.

6.3.2.25 Groups

6.3.2.25.1 AddGroup

This command sends an Add Group request to a Groups server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Add Group request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Groups client cluster to use to send the command.

Parameter Name	Type/Range	Description
GroupID	16-bit value	ID of the group to add.
Name	String	Name of the group.
Identifying	0–1	Flag indicating if only endpoints that are identifying should be added to the group.

6.3.2.25.2 ViewGroup

This command sends a View Group request to a Groups server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the View Group request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Groups client cluster to use to send the command.
GroupID	16-bit value	ID of the group to view.

6.3.2.25.3 GetGroupMembership

This command sends a Get Group Membership request to a Groups server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Get Group Membership request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Groups client cluster to use to send the command.
GroupID	16-bit value	ID of the group to get membership for. If not specified, all groups will be requested.

6.3.2.25.4 RemoveGroup

This command sends a Remove Group request to Groups server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Remove Group request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Groups client cluster to use to send the command.
GroupID	16-bit value	ID of the group to remove.

6.3.2.25.5 RemoveAllGroups

This command sends a Remove All Groups request to a Groups server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Remove All Groups request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Groups client cluster to use to send the command.

6.3.2.26 Scenes

6.3.2.26.1 AddScene

Sends an Add Scene request to a Scenes server cluster. The extension field set used for this command is compiled from all clusters on the same endpoint which support scenes.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Add Scene request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scene to be added.
SceneID	8-bit value	ID of the Scene to add.
SceneName	String	Name of the Scene.
TransTime	16-bit value	Time to transition to the scene. This is in tenths of a second if IsEnhanced is 1, otherwise it is in seconds.
IsEnhanced	0–1	Flag indicating if this is an Enhanced Add Scene request.

6.3.2.26.2 ViewScene

Sends a View Scene request to a Scenes server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the View Scene request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scene to view.
SceneID	8-bit value	ID of the Scene to view.

6.3.2.26.3 RemoveScene

Sends a Remove Scene request to a Scenes server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Remove Scene request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scene to remove.
SceneID	8-bit value	ID of the Scene to remove.

6.3.2.26.4 RemoveAllScenes

Sends Remove All Scenes request to a Scenes server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Remove All Scenes request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scenes to remove.

6.3.2.26.5 StoreScenes

Sends a Store Scene request to a Scene server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Store Scene request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scenes to store.
SceneID	8-bit value	ID of the Scene to store.

6.3.2.26.6 RecallScenes

Sends a Recall All Scenes request to a Scene server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Recall Scene request to.

Parameter Name	Type/Range	Description
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scenes to remove.
SceneID	8-bit value	ID of the scene to recall.

6.3.2.26.7 GetSceneMembership

Sends a Get Scene Membership request to a Scenes server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the get Scene membership request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
GroupID	16-bit value	Group ID of the scenes to get membership for.

6.3.2.26.8 CopyScene

Sends a Copy All Scenes request to a Scenes server cluster.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the copy all Scenes request to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Scenes client cluster to use to send the command.
CopyAll	0–1	Flag indicating if all scenes should be copied
GroupFrom	16-bit value	Group ID to copy from.
SceneFrom	8-bit value	Scene ID to copy from.
GroupTo	16-bit value	Group ID to copy to.
SceneTo	8-bit value	Scene ID to copy to.

6.3.2.27 On/Off

6.3.2.27.1 On

This command sends an On/Off cluster On command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the On command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the On/Off client cluster to use to send the command.

6.3.2.27.2 Off

This command sends an On/Off cluster Off command to the device specified.

A callback is issued when the response is received which displays the status result of the command.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Off command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the On/Off client cluster to use to send the command.

6.3.2.27.3 Toggle

This command sends an On/Off cluster Toggle command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Toggle command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the On/Off client cluster to use to send the command.

6.3.2.27.4 SetSceneData

This command sets a On/Off scenes data that will be used with the AddScene command.

Parameter Name	Type/Range	Description
OnOff	0–1	Value of the On/Off attribute for the scene.

6.3.2.28 LevelControl

6.3.2.28.1 MoveToLevel

This command sends a Level Control cluster Move To Level command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveToLevel command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Level Control client cluster to use to send the command.
WithOnOff	0–1	Indicates if the command should be sent “with On/Off” (1) or not (0).
Level	0–255	Level the output should move to.
Time	0-0xFFFF	Time in tenths of a second the output should move.

6.3.2.28.2 Move

This command sends a Level Control cluster Move command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Move command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Level Control client cluster to use to send the command.
WithOnOff	0–1	Indicates if the command should be sent “with On/Off” (1) or not (0).
Direction	0–1	Indicates if the output should move up (1) or down (0).
Rate	0-0xFF	The rate the output should move.

6.3.2.28.3 Step

This command sends a Level Control cluster Step command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Step command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Level Control client cluster to use to send the command.
WithOnOff	0–1	Indicates if the command should be sent “with On/Off” (1) or not (0).
Direction	0–1	Indicates if the output should move up (1) or down (0).
StepSize	0-0xFF	The size of each step.
Time	0-0xFFFF	Time for each step.

6.3.2.28.4 Stop

This command sends a Level Control cluster Stop command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Stop command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Level Control client cluster to use to send the command.

6.3.2.28.5 SetLevelSceneData

This command sends a Level Control cluster Set level scene command to the device specified.

Parameter Name	Type/Range	Description
CurrentLevel	0–254	Value of the CurrentLevel attribute for the scene.

6.3.2.29 Alarms

6.3.2.29.1 ResetAlarm

This command sends an Alarm cluster Reset Alarm command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Reset Alarm command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Alarm client cluster to use to send the command.
SourceClusterId	16-bit integer	ID of the cluster which generated the alarm to reset
AlarmCode	8-bit integer	The alarm code to reset

6.3.2.29.2 ResetAllAlarms

This command sends an Alarm cluster Reset All Alarms command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Reset All Alarms command to.

Parameter Name	Type/Range	Description
ClientEndpoint	Valid endpoint	Endpoint that contains the Alarm client cluster to use to send the command.

6.3.2.29.3 GetAlarm

This command sends an Alarm cluster Get Alarm command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Get Alarm command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Alarm client cluster to use to send the command.

6.3.2.29.4 ResetAlarmLog

This command sends an Alarm cluster Reset Alarm Log command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the Reset Alarm Log command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Alarm client cluster to use to send the command.

6.3.2.29.5 Alarm

This command sends an Alarm cluster Alarm command to the device specified.

Parameter Name	Type/Range	Description
ServerEndpoint	Valid endpoint	Endpoint that contains the Alarm server cluster to use to send the command.
SourceCluster	Valid cluster index	Index of the cluster which is generating the alarm.
AlarmCode	8-bit integer	Alarm code to generate.

6.3.2.30 Time

6.3.2.30.1 Read

Reads an attribute from the local Time server cluster.

Parameter Name	Type/Range	Description
AttrId	16-bit value	ID of the attribute to read.

6.3.2.30.2 Write

Writes an attribute to the local Time server cluster.

Parameter Name	Type/Range	Description
AttrId	16-bit value	ID of the attribute to write.
Value	32-bit value	Value to write to the attribute. The supported length fo the value is dependent on the attribute being written.

6.3.2.31 OTA

The following commands perform Zigbee Over-the-Air (OTA) client cluster actions. Use the ZCL demo module to register the cluster. This ensures that no other commands are necessary. The demo uses a built-in list of default image files, which must exist in the flash file system so that they can be used. The image names are as follows:

- "/spinor/ImgConfig.bin"
- "/spinor/Quartz_HASHED.elf"
- "/spinor/ioe_ram_m0_threadx_ipt.mbn"

The "`Fs ls`" command can list the contents of the file system. Examples of these files do not exist in the default file system due to their potential size. The demo service can be modified to support different storage formats or means.

6.3.2.31.1 DiscoverServer

This command sends an OTA cluster Client Discover command to the specified device.

Parameter Name	Type/Range	Description
DstAddr	16-bit value	Network address to send the Client Discover command to.
Endpoint	Valid endpoint	Endpoint that contains the OTA client cluster to use to send the command.

6.3.2.31.2 QueryImage

This command sends an OTA cluster Query Next Image command to the OTA server previously discovered.

Parameter Name	Type/Range	Description
Endpoint	Valid endpoint	Endpoint that contains the OTA client cluster to use to send the command.
ImageType	16-bit integer	Image type to query. The built-in types the demo application uses are 0x0000, 0x0002, and 0x0002.
FileVersion	16-bit integer	Current version of the queried image.

6.3.2.31.3 StartTransfer

This command starts an OTA upgrade process. After a transfer has started, it automatically continues until completion or failure.

Parameter Name	Type/Range	Description
Endpoint	Valid endpoint	Endpoint that contains the OTA client cluster to use to send the command.

6.3.2.32 Qualcomm® Touchlink™

6.3.2.32.1 Start

This command starts the Touchlink process on either a client or server cluster.

Parameter Name	Type/Range	Description
Endpoint	Valid endpoint	Endpoint that contains the Touchlink cluster to start.
DeviceType	Valid endpoint	Type of device to start as a coordinator (0), router (1), RxOnWhenIdle end device (2) or sleepy end device (3).

6.3.2.32.2 Scan

This command starts a Touchlink scan on a Touchlink client cluster.

Parameter Name	Type/Range	Description
ClientEndpoint	Valid endpoint	Endpoint that contains the Touchlink client cluster to start scanning.

6.3.2.32.3 FactoryReset

This command Performs a factory reset on a Touchlink client cluster.

Parameter Name	Type/Range	Description
ClientEndpoint	Valid endpoint	Endpoint that contains the Touchlink client cluster to factory reset.

6.3.2.33 ColorControl

6.3.2.33.1 MoveToHue

This command sends a Color Control cluster MoveToHue command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveToHue command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Hue	0-0xFF	Hue for the light to move to.
Direction	0-3	Direction to move as Shortest(0), Longest(1), Up(2), or Down(3).
Time	0-0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.2 MoveHue

This command sends a Color Control cluster MoveHue command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveHue command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	0,1, or 3	Mode for the move command as Stop(0), Up(1), or Down(3).
Rate	0–0xFF	The rate at which the output should move.

6.3.2.33.3 StepHue

This command sends a Color Control cluster StepHue command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the StepHue command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	1 or 3	Mode for the move command as Up(1) or Down(3).
Step Size	0-0xFF	The size of each step.
Time	0-0xFFFF	Time for each step.

6.3.2.33.4 MoveToSaturation

This command sends a Color Control cluster MoveToSaturation command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveToSaturation command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Saturation	0–0xFF	Saturation value for the light to move to.
Time	0–0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.5 MoveSaturation

This command sends a Color Control cluster MoveSaturation command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveSaturation command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	0,1, or 3	Mode for the move command as either Up(1), or Down(3)
Rate	0–0xFF	Time for each step

6.3.2.33.6 StepSaturation

This command sends a Color Control cluster StepSaturation command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the StepSaturation command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	1, or 3	Mode for the move command as either Up(1), or Down(3)
StepSize	0–0xFF	The size of each step
Time	0–0xFF	Time in tenths of a second the output should move.

6.3.2.33.7 MoveToHueAndSaturation

This command sends a Color Control cluster MoveToHueAndSaturation command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveToHueAndSaturation command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Hue	0–0xFF	Hue for the light to move to.
Saturation	0–0xFF	Saturation value for the light to move to.
Time	0–0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.8 MoveToColor

This command sends a Color Control cluster MoveToColor command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveToColor command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
ColorX	0–0xFFFF	X coordinate of the destination color.
ColorY	0–0xFFFF	Y coordinate of the destination color.
Time	0–0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.9 MoveColor

This command sends a Color Control cluster MoveColor command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveColor command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
RateX	-32767 – 32768	Steps per second for the X coordinate
RateY	-32767 – 32768	Steps per second for the Y coordinate

6.3.2.33.10 StepColor

This command sends a Color Control cluster StepColor command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the StepColor command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
StepX	-32767 – 32768	The size of each step for the X coordinate.
StepY	-32767 – 32768	The size of each step for the Y coordinate.
Time	0–0xFFFF	Time for each step.

6.3.2.33.11 MoveToColorTemp

This command sends a Color Control cluster MoveToColorTemp command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveToColorTemp command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
ColorTempMireds	0–0xFFFF	Destination color temp in Mireds.
Time	0–0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.12 EnhancedMoveToHue

This command sends a Color Control cluster EnhancedMoveToHue command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the EnhancedMoveToHue command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Hue	0–0xFFFF	Hue for the light to move to.
Direction	0–3	Direction to move as Shortest(0), Longest(1), Up(2) or Down(3)
Time	0–0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.13 EnhancedMoveHue

This command sends a Color Control cluster EnhancedMoveHue command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the EnhancedMoveHue command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	0,1,or 3	Mode for the move command as Stop(0),Up(1), or Down(3).
Rate	0–0xFF	The rate the output should move.

6.3.2.33.14 EnhancedStepHue

This command sends a Color Control cluster EnhancedStepHue command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the EnhancedStepHue command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	1 or 3	Mode for the move command as Up (1) or Down (3).
StepSize	0-0xFFFF	The size of each step.
Time	0-0xFFFF	Time for each step.

6.3.2.33.15 EnhancedMoveToHueAndSaturation

This command sends a Color Control cluster EnhancedMoveToHueAndSaturation command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the EnhancedMoveToHueAndSaturation command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Hue	0–0xFFFF	Hue for the light to move to.
Saturation	0–0xFF	Saturation value for the light to move to.
Time	0–0xFFFF	Time in tenths of a second the output should move.

6.3.2.33.16 ColorLoopSet

This command sends a Color Control cluster ColorLoopSet command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the ColorLoopSet command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Flag	0–0xFF	Flags to control how the Color attributes should be updated: Bit 0: Update the ColorLoopActive attribute. Bit 1: Update the ColorLoopDirection attribute. Bit 2: Update the ColorLoopTime attribute. Bit 3: Update the ColorLoopStartEnhancedHue attribute.
ActMode	0–2	Action to be taken as de-activate (0), activate from color loop start enhanced hue (1), or activate from enhanced current hue (2).
DirectMode	0–1	Direction for color loop as decrement (0) or increment (1).
Time	0–0xFFFF	Time in tenths of a second the output should move.
StartHue	0–0xFFFF	Starting HUE for the color loop

6.3.2.33.17 StopMoveStep

This command sends a Color Control cluster StopMoveStep command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the StopMoveStep command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.

6.3.2.33.18 MoveColorTemp

This command sends a Color Control cluster MoveColorTemp command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the MoveColorTemp command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	0,1, or 3	Mode for the move command as Up (1) or Down (3).
Rate	0–0xFFFF	The rate the output should move.
MinMireds	0–0xFFFF	The minimum Mireds for this move operation.
MaxMireds	0–0xFFFF	The maximum Mireds for this move operation.

6.3.2.33.19 StepColorTemp

This command sends a Color Control cluster StepColorTemp command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the StepColorTemp command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the Color Control client cluster to use to send the command.
Mode	1 or 3	Mode for the move command as Up (1), or Down (3).
StepSize	0–0xFFFF	The size of each step.
Time	0–0xFFFF	Time of each step.
MinMireds	0–0xFFFF	The minimum Mireds for this step operation.
MaxMireds	0–0xFFFF	The minimum Mireds for this step operation.

6.3.2.34 Custom

6.3.2.34.1 SendCommand

This command sends a custom command to the device specified.

Parameter Name	Type/Range	Description
DevId	Valid device index	The device to send the command to.
ClientEndpoint	Valid endpoint	Endpoint that contains the custom client cluster to use to send the command.

6.3.2.35 Demo configuration

This section describes the configurations that exist in the Zigbee demo.

The following configuration are near the top of the main Zigbee demo’s source file:

Parameter Name	Description
DEFAULT_ZIGBEE_PAN_ID	The PAN ID that the demo uses during the formation of a network.
DEFAULT_END_DEVICE_TIME_OUT	End device timeout value.
DEFAULT_PAN_ID	This value controls the PAN ID used by the demo.
ZB_DEFAULT_EUI64_PREFIX	This value controls the prefix that is used for generating the devices extended address.
FORM_CHANNEL_MASK	The mask of channels that the demo can use when forming a network
JOIN_CHANNEL_MASK	The mask of channels scanned when joining a network.
DEV_ID_LIST_SIZE	Size (and number of devices) that can be added to the Zigbee demo's device list. This does not include device zero which is reserved for the NULL address.
DEFAULT_ZIGBEE_LINK_KEY	Default Link key for the Zigbee network (typically "ZigbeeAlliance09").
Default_ZB_Secuity	Structure containing the default security information used when forming or joining a network.
APP_ENDPOINT_NUM_LIGHT	Endpoint number used for lights.

The following configurations are in the Zigbee demo's header file:

Parameter Name	Description
APP_ENDPOINT_NUM_LIGHT	Endpoint number used for lights.
APP_ENDPOINT_NUM_SWITCH	Endpoint number used for switches.
APP_MAX_NUM_SCENES	Maximum number of scenes supported for each scenes server.

6.3.2.36 Usage examples

This section contains examples for some common use cases of the Zigbee demo. Most of the examples will use two devices. Each command and printout will be prefixed as Dev1 or Dev2. Dev1 will be initialized and used as a hub device (forming the network) and Dev2 will be another device on the network (joining the network).

Also, for demonstrating the device roles, Dev1 will be used as a Dimmer Light Switch and Dev2 will be used as a Dimmable Light.

6.3.2.36.1 Initialization

The following commands initialize Zigbee on both the devices, Dev1 and Dev2. Dev1 has its extended address set to 000000FFFE000001 and Dev2 has extended address set to 000000FFFE000002.

Dev1:

```
Zigbee> Initialize
Zigbee: Zigbee stack initialized.
Zigbee:    Extended Address: 0000000000000000

Zigbee> SetExtAddress 000000FFFE000001
Zigbee: qapi_ZB_Set_Extended_Address() success.
```

Dev2:

```
Zigbee> Initialize
Zigbee: Zigbee stack initialized.
Zigbee:    Extended Address: 0000000000000000

Zigbee> SetExtAddress 000000FFFE000002
Zigbee: qapi_ZB_Set_Extended_Address() success.
```

6.3.2.36.2 Endpoint creation

The device types are split into separate subgroups in the demo. There is one group for light device (on/off and dimmable) and the other for switch devices (on/off and dimmer). Initialization executes the command and determines the device type.

Dev1:

```
Zigbee\ZCL> CreateEndpoint 1 5

Zigbee\ZCL: qapi_ZB_CL_Register_Callback() success.
Zigbee\ZCL: Endpoint 1 Initialized as Color Dimmable Light.
```

Dev2:

```
Zigbee\ZCL> CreateEndpoint 1 6
Zigbee\ZCL: qapi_ZB_CL_Register_Callback() success.
```

6.3.2.36.3 Zigbee\ZCL: Endpoint 1 initialized as color dimmer switch forming and joining a network

Before a device can join a network, there must be a network for it to join. This may be a third-party hub or a Zigbee demo device. This demonstrates how to form a network using security:

Dev1:

```
Zigbee> Form 1

Zigbee: qapi_ZB_Form() success.

Zigbee>
Zigbee: Form confirm:
Zigbee:   Status:  0
Zigbee:   Channel: 23
```

Device joining is very similar to forming, but the hub device must be put into a state where join is permitted. For the Zigbee demo, this command will enable joining the network for 255 seconds (maximum allowed duration):

Dev1:

```
Zigbee> PermitJoin 255

Zigbee: qapi_ZB_Permit_Join() success.
```

Finally, to join a network, execute the following (note that the hub device must still be within the PermitJoin duration):

Dev2:

```
Zigbee> Join 1 1
Zigbee: qapi_ZB_Join() success.

Zigbee>
Zigbee: Join confirm:
Zigbee:   Status:        0
Zigbee:   NwkAddress:    0x13CF
Zigbee:   ExtendedPanId: 0011223344556677
Zigbee:   Channel:       23
```

On the devices on the network, the Device Annce is also seen:

Dev1:

```
Zigbee: Device Annce:
Zigbee:   ExtendedAddress: 8899AABBCCDDEEFF
Zigbee:   NetworkAddress:  0x13CF
Zigbee:   Capability:      0x8E
```

6.3.2.36.4 Finding devices

In Zigbee, scanning for devices that contain an endpoint is done using a match descriptor request that takes a list of clusters to be matched. The following will search for any On/Off server clusters (note that this is issued from Dev2 since it was initialized as a switch):

Dev2:

```
Zigbee\ZDP> MatchDescriptor 0xFFFD 0x0006 0

Zigbee\ZDP: Finding Cluster 0x0006.

Zigbee\ZDP>
Zigbee: Match Descriptor Response:
Zigbee:   Status:              0
Zigbee:   Address:             0x0000
Zigbee:   Number Of Endpoints: 1
Zigbee:   Matched Endpoints:   1

Zigbee\ZDP>
Zigbee: Match Descriptor Response:
Zigbee:   Status:              -150133
```

The first response received is from the light endpoint, advertising itself. The second indicates the operation timed out (this is because the example uses a broadcast address).

6.3.2.36.5 Device table

To simplify some of the commands, the Zigbee demo uses a device table to track addresses. Commands exist to add, remove, and view the device table. The device table will never be empty as index zero is reserved for the NULL address and sends packets using the binding table.

The following example adds three addresses to the device table of device 1 (the network address of the switch, the extended address of the switch, and the extended address of the light – these

will be used later in the binding section) and one address to device 2 (the network address of the light – which will be used to send cluster commands).

Dev1:

```
Zigbee> AddDevice 2 0x13CF 1
Zigbee: Registered device ID: 1
Zigbee> AddDevice 3 8899AABBCCDDEEFF 1
Zigbee: Registered device ID: 2
Zigbee> AddDevice 3 0011223344556677 1
Zigbee: Registered device ID: 3
```

Dev2:

```
Zigbee> AddDevice 2 0x0000 1
Zigbee: Registered device ID: 1
```

6.3.2.36.6 Binding

Binding can be created between two endpoints. These commands will establish a binding for the On/Off cluster (6) and Level Control Cluster (8):

Dev1:

```
Zigbee\ZDP> BindRequest 1 2 3 6

Zigbee: qapi_ZB_ZDP_Bind_Req() success.

Zigbee\ZDP>
Zigbee: ZDP Bind Response:
Zigbee:   Status: 0

Zigbee\ZDP> BindRequest 1 2 3 8

Zigbee: qapi_ZB_ZDP_Bind_Req() success.

Zigbee\ZDP>
Zigbee: ZDP Bind Response:
Zigbee:   Status: 0
```

6.3.2.36.7 Sending cluster commands

After everything is initialized and setup, commands are sent to turn on and off the the light. There are two ways to send commands at this point, either using the binding table entry that is setup in the previous step, or using the destination's address.

An "On" command to the destination device using its network address is listed as follows:

Dev2:

```
Zigbee\OnOff> On 1 1

Zigbee\OnOff: qapi_ZB_CL_OnOff_Send_On() success.
```

```
Zigbee\OnOff>
Zigbee\OnOff: OnOff Client Default Response:
Zigbee\OnOff:   Status:        0
Zigbee\OnOff:   CommandID:     0x01
Zigbee\OnOff:   CommandStatus: 0
```

Dev1:

```
Zigbee\LevelControl: LevelControl Server Transition Event:
Zigbee\LevelControl:   Level:          254
Zigbee\LevelControl:   TransitionTime: 0
Zigbee\LevelControl:   WithOnOff:      False
```

This generates a level control event on the server side. If the endpoint was initialized as an On/Off light without level control, an On/Off event would have been generated instead.

To send an off command using the binding table (this is done using the NULL address):

Dev2:

```
Zigbee\OnOff> Off 0 1

Zigbee\OnOff: qapi_ZB_CL_OnOff_Send_Off() success.

Zigbee\OnOff>
Zigbee\OnOff: OnOff Client Default Response:
Zigbee\OnOff:   Status:        0
Zigbee\OnOff:   CommandID:     0x01
Zigbee\OnOff:   CommandStatus: 0
```

Dev1:

```
Zigbee\LevelControl: LevelControl Server Transition Event:
Zigbee\LevelControl:   Level:          0
Zigbee\LevelControl:   TransitionTime: 0
Zigbee\LevelControl:   WithOnOff:      False
```

Other commands can be sent in the same way. The difference depends on the parameters they take.

6.3.3 Thread

The Thread demo is intended to demonstrate the operation of the Thread stack and how to use its APIs.

6.3.3.1 Initialize

Initializes the thread interface and sets the default device and network configurations.

Parameter Name	Type/Range	Description
ExtAddrSuffix	24-bit value	Deprecated
Type	0–1	Determines the type of device to initialize as a router capable device (0) or Sleepy end device (1)

6.3.3.2 Shutdown

Shuts down the thread interface.

6.3.3.3 Start

Attempts to become active on the specified network. This command will fail if network data has not been obtained, either by setting it manually or using commissioning.

6.3.3.4 Stop

Disconnects from the current Thread network.

6.3.3.5 GetDeviceConfiguration

Prints the current device information, including EUI-64, child timeout, and operating flags.

6.3.3.6 GetNetworkConfiguration

Prints the current Thread Network information.

6.3.3.7 SetExtendedAddress

This command sets a Extended address for this device. Normally, this is not necessary because Thread specifies the use of a randomized EUI-64.

Parameter Name	Type/Range	Description
ExtAddr	64-bit value	Extended address of the device.

6.3.3.8 SetChildTimeout

This command sets the child timeout in seconds for this device. The child timeout is how long the parent should wait before considering this device to have left the network due to lack of activity (polls, child updates).

Parameter Name	Type/Range	Description
Timeout	32-bit value	Child timeout in seconds.

6.3.3.9 SetLinkMode

This command sets the link mode to be used when connecting to the Thread network.

Parameter Name	Type/Range	Description
RxOnWhenIdle	0–1	Flag indicating if the device's receiver is on when the device is idle.
UseSecureDataRequests	0–1	Flag indicating if the device uses secure data requests
IsFFD	0–1	Flag indicating if the device is a FFD.
RequireNetworkData	0–1	Flag indicating if the device requires full network data.

6.3.3.10 SetChannel

This command sets the channel of the appropriate network.

Parameter Name	Type/Range	Description
Channel	11–26	Channel of the network.

6.3.3.11 SetPANID

This command sets the PAN ID of the appropriate network.

Parameter Name	Type/Range	Description
PANID	0–0FFFD	The 16-bit PAN ID of the network.

6.3.3.12 SetExtendedPANID

This command sets the extended PAN ID of the appropriate network.

Parameter Name	Type/Range	Description
ExtPANID	64-bit value	The 64-bit PAN ID of the network.

6.3.3.13 SetNetworkName

This command sets the Network Name of the appropriate network.

Parameter Name	Type/Range	Description
Name	String	String representing the networks name (16 characters maximum).

6.3.3.14 SetMasterKey

This command sets the Master Key of the appropriate network.

Parameter Name	Type/Range	Description
Key	16-byte hexadecimal string	The Master key in hexadecimal format

6.3.3.15 CommissionerStart

Starts acting as a commissioner on an active network. This device will fail to become the commissioner if there is another already on the network, or if it is not currently an active network.

6.3.3.16 CommissionerStop

Stops acting as a Commissioner.

6.3.3.17 CommissionerAddJoiner

This command adds a Joiner device to the steering information. This allows a device to scan for a network and determine if it is allowed to join the network.

Parameter Name	Type/Range	Description
Passphrase	String	Passphrase for the joiner
ExtAddr	64-bit value	Joiner extended address (optional: defaults to 0 or all devices)
Timeout	16-bit value	Timeout for the device to join in seconds (optional: defaults to 120 s)

6.3.3.18 CommissionerDelJoiner

This command removes a previously added joiner. The extended address must be provided and match the argument to a previous CommissionerAddJoiner command.

Parameter Name	Type/Range	Description
ExtAddr	64-bit value	Joiner extended address (optional, defaults to 0 or all devices)

6.3.3.19 JoinerStart

This command scans for and attempts to join a network using a given passphrase (PSKd).

Parameter Name	Type/Range	Description
Passphrase	String	Passphrase to use when joining.

6.3.3.20 JoinerStop

Stops the Joining process.

6.3.3.21 AddBorderRouter

This command adds prefix information to the Network Data. This is distributed to all active Thread devices on the network.

Parameter Name	Type/Range	Description
Prefix	IPv6 address	Prefix for use to route on the external network.
PrefixLength	0–128	Length of the prefix.
Pref	0–2	Router preference.
IsStable	0–1	Flag indicating if this is stable network data.
Flags	32-bit integer	Configuration flags for the border router: ▪ Bit 3: SLAAC Preferred ▪ Bit 4: SLAAC Valid ▪ Bit 5: Supports DHCPv6 address configuration ▪ Bit 6: Supports DHCPv6 other configuration ▪ Bit 7: Default route

6.3.3.22 RemoveBorderRouter

This command removes a border router prefix information. The prefix information specified must match that previously used in AddBorderRouter.

Parameter Name	Type/Range	Description
Prefix	IPv6 address	Prefix used to add the border router
PrefixLength	0–128	Length of the prefix

6.3.3.23 AddExternalRoute

This command adds an external route to the network data.

Parameter Name	Type/Range	Description
Prefix	IPv6 address	Prefix of the external route.
PrefixLength	0–128	Length of the prefix

Parameter Name	Type/Range	Description
IsStable	0–1	Flag indicating if this is stable network data.

6.3.3.24 RemoveExternalRoute

This command removes an external route from the network data. The prefix information specified must match the previously used in AddExternalRoute.

Parameter Name	Type/Range	Description
ExtAddrSuffix	24-bit value	Deprecated
Type	0–1	Determines the type of device to initialize as a router capable device (0) or Sleepy end device (1)

6.3.3.25 RegisterServerData

Forces registration of any pending network data update. This normally happens after a short timeout after updating the network data to avoid flooding the network with spurious traffic.

6.3.3.26 UseDefaultInfo

This is a convenience function to set up safe default parameters for a Thread Network. After it is issued, Start can be used to connect to (or form) the network.

6.3.3.27 StartBorderAgent

For Wi-Fi-enabled devices (QCA4020), this allows the device to act as a Border Agent if already connected to an existing Wi-Fi network. Note that an IP address must be associated with the WLAN interface before issuing this command.

Parameter Name	Type/Range	Description
Interface	String	Name of the interface to use for the border agent. It should be “wlan0” or “wlan1.”

6.3.3.28 StopBorderAgent

Stops acting as a border agent (Wi-Fi-enabled QCA4020 devices only).

6.3.3.29 UpdatePSKc

This updates the PSKc used for off-mesh commissioning. If the network data has already been set and not started, the PSKc can also be regenerated using a passphrase. This passphrase is entered on the Commissioning device (a smartphone running a commissioning app).

Parameter Name	Type/Range	Description
Passphrase	String	Commissioner passphrase.

6.3.3.30 ClearPersist

This clears the persistent Flash data that normally stores the network information for a Thread network. Note that this function will not clear the network data already cached by Thread. However, if the device is reset after issuing the command, its network data will be completely cleared.

6.3.3.31 Demo configuration

This section describes the configuration that exists in the Thread demo.

Parameter Name	Description
DEFAULT_CHILD_TIMEOUT	The default child timeout used for initializing the Thread interface.

6.3.3.32 Usage examples

This section contains examples for some common use cases of the Thread demo. Most of the examples will use two devices. Each command and printout will be prefixed as Dev1 or Dev2. Dev1 is used as router and border router and Dev2 is an end device.

For examples that use a border router, QCA4020 needs to be used.

6.3.3.32.1 Initialization

The following commands initialize Thread on both devices as either [(0=Router, 1=Sleepy, 2=End Device)] . Dev1 will be initialized as Router and Dev2 will be initialized as EndDevice.

Dev1:

```
Thread> Initialize 0

Thread: Thread Initialized Successfully:
Thread: Network Configuration:
Thread:    Channel:          11
Thread:    PAN_ID:           FFFF
Thread:    Extended_PAN_ID:  DEAD00BEEF00CAFE
Thread:    NetworkName:      OpenThread
Thread:    MasterKey:        00112233445566778899AABBCCDDEEFF.
```

Dev2:

```
Thread> Initialize 2

Thread: Thread Initialized Successfully:
Thread: Network Configuration:
Thread:    Channel:          11
Thread:    PAN_ID:           FFFF
Thread:    Extended_PAN_ID:  DEAD00BEEF00CAFE
Thread:    NetworkName:      OpenThread
Thread:    MasterKey:        00112233445566778899AABBCCDDEEFF.
```

6.3.3.32.2 Forming a network

The following commands will form a network using a device initialized as a router.

Dev1:

```
Thread> UseDefaultInfo

Thread: Network Configuration:
Thread:    Channel:          16
```

```
Thread:    PAN_ID:           8DA8
Thread:    Extended_PAN_ID:  0001020304050607
Thread:    NetworkName:      Test Network
Thread:    MasterKey:        F32B7B515AD61BFAF32B7B515AD61BFA

Thread> Start

Thread: qapi_TWN_Start() success.
Thread: Network State Changed: Detached

Thread: Network State Changed: Leader
```

6.3.3.32.3 On-Mesh commissioning

The following commands will add a device using an on-mesh commissioner. This assumes Dev1 has formed a network and Dev2 is joining it.

Dev1:

```
Thread> CommissionerStart
Thread: qapi_TWN_Commissioner_Start() success.

Thread> CommissionerAddJoiner mypassword
Thread: qapi_TWN_Commissioner_Add_Joiner() success.
```

Dev2:

```
Thread> JoinerStart mypassword
Thread: qapi_TWN_Joiner_Start() success.
Thread: Joiner Result: Success
Thread: Network Configuration:
Thread:    Channel:          16
Thread:    PAN_ID:           8DA8
Thread:    Extended_PAN_ID:  0001020304050607
Thread:    NetworkName:      Test Network
Thread:    MasterKey:        F32B7B515AD61BFAF32B7B515AD61BFA
Thread> Start
Thread: qapi_TWN_Start() success.
Thread: Network State Changed: Detached
Thread: Network State Changed: Child
```

6.3.3.32.4 Becoming a border agent

The following process is used for configuring the device as a border router and connecting a Thread commissioning app. This is only applicable if running a 4020 and assumes that the device has already connected to a Wi-Fi network. The “Thread 1.1 Commissioning App” is available for both Android and iPhone devices (this example follows the Android version). The device running the commissioning app should be connected to the same Wi-Fi network as the QCA4020.

Dev1:

```
Thread> AddBorderRouter 2000:baad:beef:: 64 0 1 0x98
Thread: qapi_TWN_Add_Border_Router() success.
```

```
Thread> RegisterServerData
Thread: qapi_TWN_Register_Server_Data() success.

Thread> StartBorderAgent wlan0
Thread: qapi_TWN_Start_Border_Agent() success.
```

From Thread 1.1 Commissioning App:

- When the app starts, it scans for border routers. Select the correct border router (Typically appears as “Quartz Border Router”).

2. Enter the password “12SECRETPASSWORD34” (this can be changed using the app).
3. Click OK and wait for it to connect.

6.3.3.32.5 Off-Mesh commissioning

The following process joins a device to the thread network using off-mesh commissioning. It assumes that a commissioning app has already joined the thread network.

From Thread 1.1 Commissioning App:

- Select “ENTER JOIN PASSPHRASE MANUALLY” near the bottom of the screen

4. Leave the EUI field blank and enter a passphrase of at least 6 characters (for this example “AAAAAA” is used)
5. Select Confirm (note that there is a timeout for the device to join after this step).

On the joining device:

Dev2:

```
Thread> JoinerStart AAAAAA
Thread: qapi_TWN_Joiner_Start() success.

Thread: Joiner Result: Success
Thread: Network Configuration:
Thread:    Channel:          16
Thread:    PAN_ID:           8DA8
Thread:    Extended_PAN_ID:  0001020304050607
Thread:    NetworkName:      Test Network
Thread:    MasterKey:        F32B7B515AD61BFAF32B7B515AD61BFA

Thread> Start
Thread: qapi_TWN_Start() success.
Thread: Network State Changed: Detached
Thread: Network State Changed: Child
```

6.3.4 HMI

The HMI demo is intended to demonstrate how to use the interface to the IEEE 802.15.4 MAC. It is assumed that developers using this demo are familiar with the MCPS and MLME service primitives of the IEEE 802.15.4-2006 specification.

6.3.4.1 Initialize

This command initializes the 802.15.4 MAC and must be called before any other command within the HMI demo. It also sets the initial state of several PIBs within the MAC including macShortAddress, macPanId, and macRxOnWhenIdle.

Parameter Name	Type/Range	Description
ShortAddr	0 – 0xFFFF	Initial short address of the MAC.
Type	0–1	The type of device as normal (0) or sleepy (1). This is mainly used to determine the initial state of the macRxOnWhenIdle PIB. It is set to TRUE for normal devices or FALSE for sleepy devices.

If initialization is successful, the command prints the initialization information of the MAC (essentially the initial state of the preceding PIBs plus the interface ID).

6.3.4.2 Shutdown

This command shuts down the 802.15.4 MAC interface.

6.3.4.3 SetPANID

This command sets the PAN ID of the MAC. It sets the macPanId PIB and the PAN ID that is used by other commands in the demo.

Parameter Name	Type/Range	Description
PAN_ID	0 – 0xFFFF	The PAN ID for the MAC interface.

6.3.4.4 SetAddressModes

This command sets the source and destination addressing modes that are used by other commands in the demo.

Parameter Name	Type/Range	Description
SrcAddrMode	0, 2, or 3	The mode for source addresses as none (0), short (2) or extended (3).
DstAddrMode	0, 2, or 3	The mode for destination addresses as none (0), short (2) or extended (3).

6.3.4.5 SetSecurity

This command sets up the security information for the MAC per the parameters specified.

Parameter Name	Type/Range	Description
SecurityLevel	0 – 7	The security level of the MAC according to the 802.15.4-2006 specification.
KeyIDMode	1 – 3	The Key ID mode to be used for sending data.

This includes the security structure that is used by other commands in the demo and the following PIBs.

- macSecurityEnabled is set to FALSE if the SecurityLevel specified was 0, otherwise it is set to TRUE.
- macDefaultKeySource is set per the default key source in the demo.

- Index 0 of the macKeyTable PIB is set up with the default key table information in the demo combined with the specified KeyIDMode.

The demo does not fully support the Implicit KeyIDMode.

6.3.4.6 AddDevice

This command adds a device to the HMI demo device table which is used by other commands within the demo and adds an entry to the macDeviceTable. The number of devices that can be added to the table is limited by the size of the macDeviceTable within the MAC.

Parameter Name	Type/Range	Description
ExtAddr	64-bit	The extended address of the device to add. The demo expects this entry to be a 64-bit hexadecimal number (the 0x prefix is optional). For correct operation with security, this value must be accurate even if the short addressing mode is used.
ShortAddr	0 – 0xFFFF	The short address of the device to add.
Type	0 – 1	The type of device that is added as normal (0) or sleepy (1). This is used to determine if the TxIndirect flag is set for packets sent to this device.
FrameCounter	0 – 0xFFFFFFFF	Optionally specifies the initial frame counter of the device. This defaults to zero if not specified.

The Index of the newly added device is displayed if the device was added successfully.

6.3.4.7 DelDevice

This command removes a device from the HMI device table and from the macDeviceTable PIB. This function will not compress the remaining devices, but will leave the device index unused.

Parameter Name	Type/Range	Description
DeviceIndex	Valid device index	The index of the device to be removed.

6.3.4.8 ListDevices

This command lists the devices that are currently in the HMI device table.

6.3.4.9 ListPIBs

This command lists out the PIBs (PIB number and description) that can be used with the SetPIBRequest command.

6.3.4.10 ResetRequest: cmd_MLME_Reset_Request()

This command issues an MLME-RESET.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
ResetPIBs	0 – 1	The value to be assigned to the SetDefaultPIB parameter of the MLME-RESET.request.

6.3.4.11 GetPIBRequest

This command issues an MLME-GET.request to the 802.15.4 MAC and displays its value to the console.

Parameter Name	Type/Range	Description
PIB_ID	0 – 0xFF	The ID of the PIB to be requested.
PIB_Index	0 – 0xFF	Optional parameter to specify the PIB Index to be requested. Defaults to 0.

6.3.4.12 SetPIBRequest

This command issues an MLME-SET.request to the 802.15.4 MAC. Only the PIBs listed using the ListPIBs command are supported.

Parameter Name	Type/Range	Description
PIB_ID	0 – 0xFF	The ID of the PIB to be requested.
PIB_Value	Determined by PIB length	The value to be set for the PIB.

6.3.4.13 ScanRequest

This command issues an MLME-SCAN.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
Type	0 – 1	The type of scan to start as Energy Detect (0) or Active (1).
ChannelMap	0 – 0xFFFFFFFF	The bitmask of channels to be scanned.
Duration	0 – 0x7FFFFFFF	The duration of the scan on each channel in milliseconds.

6.3.4.14 StartRequest

This command issues an MLME-START.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
Channel	0 – 31	The channel to start the PAN on.

6.3.4.15 AssociateRequest

This command issues an MLME-ASSOCIATE.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
Channel	0 – 31	The channel to associate on.
CoordDeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the coordinator.
Capability	0 – 0xFF	The capability information to be included in the associate request.

6.3.4.16 DisassociateRequest

This command issues an MLME-DISASSOCIATE.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
DeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the device.
Reason	0 – 0xFF	The disassociate reason.

6.3.4.17 PollRequest

This command issues an MLME-POLL.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
CoordDeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the coordinator.

6.3.4.18 RxEnableRequest

This command issues an MLME-RX-ENABLE.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
Duration	0 – 0x7FFFFFFF	The duration for the receiver to be turned on in milliseconds.

6.3.4.19 DataRequest

This command issues an MCPS-DATA.request to the 802.15.4 MAC. The data sent is hard coded to the hmi demo. The length of the data will be the maximum payload supported for the given parameters.

Parameter Name	Type/Range	Description
DstDeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the destination.
Acknowledge	0 – 1	Indicates if the acknowledge request flag should be set (1) or not (0).

The MSDUHandle of the packet will be displayed if the command is successful.

6.3.4.20 PurgeRequest

This command issues an MCPS-PURGE.request to the 802.15.4 MAC.

Parameter Name	Type/Range	Description
MSDUHandle	0– 0xFF	The MSDU handle of the packet to be purged.

6.3.4.21 StartSend

This command starts a continuous transmission.

Parameter Name	Type/Range	Description
DstDeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the destination.
Acknowledge	0 – 1	Indicates if the acknowledge request flag should be set (1) or not (0).

It will issue an MCPS-DATA.request to the requested device and keeps sending data as soon as the MCPS-DATA.confirm is received. While transmission is in progress, a periodic message will be displayed providing the amount of data sent, overall throughput seen, time the transmission has been in progress, and the number of non-critical errors that have occurred (such as no-ack).

6.3.4.22 StopSend

This command stops a continuous transmission that was started using the StartSend command

Parameter Name	Type/Range	Description
DeviceIndex	Valid device index	The index in the HMI demo device table to stop sending continuously to.

6.3.4.23 StartReceive

This command starts automatically receiving packets from a given device. While automatically receiving, the packets will not be displayed to the console, instead a periodic summary will be displayed.

Parameter Name	Type/Range	Description
SrcDeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the device to start receiving data from.
PollPeriod	500 – 60000	If receiving from a sleepy device (as determined by the SrcDeviceIndex), this parameter determines how often the device will be polled in milliseconds if no data is received.

This command behaves slightly differently based on the receive started on a device that is sleepy or normal (the determination is made based on the specified entry in the HMI demo device table). For a normal device, the command simply absorbs received packets and displays a periodic summary. For sleepy devices, however, the command will also start periodically polling the device. If data is received from the device as a result of a poll, this command will immediately issue another poll request. Otherwise it will issue a new poll at the specified interval.

6.3.4.24 StopReceive

This command stops a continuous transmission that was started using the StartReceive command.

Parameter Name	Type/Range	Description
SrcDeviceIndex	Valid device index	The index in the HMI demo device table to stop receiving continuously to.

6.3.4.25 EnableBBIF

This command enables BBIF on the 802.15.4 interface (support may be disabled).

Parameter Name	Type/Range	Description
Enable	0 – 1	Flag which specifies if BBIF should be enabled (1) or disabled (0).

6.3.4.26 AutoPoll

This command issues an VS-AUTOPOLL.request to the 802.15.4 MAC. The autopoll tells the make to start making periodic polls to the coordinator, allowing polls to continue in the background without host interaction.

Parameter Name	Type/Range	Description
CoordDeviceIndex	Valid device index	The index in the HMI demo device table that contains the information for the coordinator.
Period	0, 500 - 60000	The period to poll the coordinator in milliseconds. Disables autopoll if set to 0.

6.3.4.27 DUT subgroup

This subgroup contains commands for DUT testing such as continuous transmit and receive.

6.3.4.27.1 Enable

This command issues a VS-DUT- enables DUT mode and must be executed before the other DUT commands. Note that once DUT mode is entered, it can only be exited by issuing a ResetRequest: cmd_MLME_Reset_Request() with the ResetPIBs parameter set to 1.

6.3.4.27.2 TxTest

This command issues a VS-DUT-TX-TEST.request to the 802.15.4 MAC which starts a DUT transmit test.

Parameter Name	Type/Range	Description
Mode	0 – 2	The test mode to use: 0 = Carrier only 1 = Continuous modulated data 2 = Burst modulated data.
Channel	11 – 26	The channel number to start the transmit test.
Power	0 – 15	The transmit power level to use for the test.
PacketType	0 – 5	The type of packet to be used for the transmit test as one of the following: 1. PRBS9 1. PRBS16 ■ All 1's ■ All 0's ■ 01010101… ■ 10101010… This parameter is only necessary for mode 1 and 2 and is otherwise ignored.
Payload Length	1 – 127	The length of the payload for packets that are sent. The maximum value is 125 if CRC is included according to the flags parameter. This parameter is only necessary for mode 2 and is otherwise ignored.
Gap	0 – 65535	The spacing between packets. This parameter is only necessary for mode 2 and is otherwise ignored.

Parameter Name	Type/Range	Description
Flags	Bitmask	Bitmask of flags for the transmit test: Bit 0: Remove PHR when set Bit 1: Append CRC when set This parameter is only necessary for mode 2 and is otherwise ignored.

6.3.4.27.3 RxTest

This command issues a VS-DUT-RX-TEST.request to the 802.15.4 MAC which start a DUT receive test.

Parameter Name	Type/Range	Description
Mode	0 or 2	The test mode to be used: ■ Carrier only ■ Burst modulated data
Channel	0 – 31	The channel number to start the receive test on.
Packet Type	0 – 5	The type of packet to be used for the receive test: ▪ PRBS9 **CAUTION:** PRBS16 **CAUTION:** All 1's **CAUTION:** All 0's **CAUTION:** 010101… **CAUTION:** 101010… This parameter is only necessary for mode 2 and is otherwise ignored.
Payload Length	1 – 127	The expected length for received packets. The maximum value is 125 if CRC is included according to the flags parameter. This parameter is only necessary for mode 2 and is otherwise ignored.
Gap	0 – 65535	The spacing between packets. This parameter is only necessary for mode 2 and is otherwise ignored.
Flags	Bitmask	Bitmask of flags for the receive test: Bit 0: Remove PHR when set Bit 1: Append CRC when set This parameter is only necessary for mode 2 and is otherwise ignored.
RSSI	0 – 65535	The period between RSSI indications from the MAC in milliseconds. If set to 0, RSSI indications will be disabled. This parameter is only necessary for mode 2 and is otherwise ignored.

6.3.4.27.4 RxStat

This command issues a VS-DUT-Rx-STAT.request to the 802.15.4 MAC which is used to request the statistics during a receive test. The statistics is displayed to the console if the command executes successfully.

6.3.4.27.5 TestEnd

This command issues a VS-DUT-Test-End.request to the 802.15.4 MAC which terminates any current transmit or receive test.

6.3.4.28 Demo configuration

At the top of the HMI demo source file, there are several macros that control the configuration of the demo, including default states and the size of various structures. This section describes these configuration and how they can be modified.

Parameter Name	Description
CHANNEL_ PAGE	This value controls the page number for various commands throughout the demo. Different values may not be supported by the 802.15.4 interface on the device.
DEVICE_LIST_ SIZE	This value controls the size of the device list within the demo and can be changed if the user wishes to support more (or less) devices. The number of devices that can be added to the demo is restricted by the size of the macDeviceTable PIB in the 802.15.4 MAC.
DEFAULT_PAN_ ID	This value controls the PAN ID used by the demo.
DEFAULT_EUI6 4_PREFIX	This value controls the prefix that is used for automatically assigning the device's extended address when the demo is initialized (see Initialize).
SEND_DATA_ DISPLAY_ FREQUENCY	This value controls the rate at which the send status is displayed to the console in seconds during a continuous send (see StartSend).
RECEIVE_DATA _DISPLAY_ FREQUENCY	This value controls the rate at which the send status is displayed to the console in seconds during a continuous receive (see StartReceive).
Default_ KeyDescriptor	This structure contains the security information that is used when the SetSecurity command is called. It can be changed if the user wishes to use a specific KeySource, KeyIndex, or Key. The IdMode field of this structure is largely ignored as it is overwritten by the SetSecurity command.

6.3.4.29 Usage examples

This section contains examples for some common use cases of the 802.15.4 MAC. Most of the examples will use two devices. Each command and printout will be prefixed as Dev1 or Dev2. Dev1 will be initialized and used as a coordinator device and Dev2 will be a sleepy device.

6.3.4.29.1 Initialization

The following commands initialize the 802.15.4 MAC on both devices. Dev1 will have its short address set to 0x0001, its extended address is 0x000000FFFE000001, and its RxOnWhenIdle PIB set to true. Dev2 will have its short address set to 0x0002, its extended address is 0x000000FFFE000002, and its RxOnWhenIdle PIB set to false.

Dev1:

```
HMI> Initialize 0x0001 0
HMI: 802.15.4 MAC Initialized Successfully:
HMI:    Interface ID:     1 (0x00000001)
HMI:    Extended Address: 000000FFFE000001
HMI:    Short Address:    1 (0x0001)
HMI:    PAN ID:           1 (0x0001)
```

```
HMI:    Sleepy:          No
```

Dev2:

```
HMI> Initialize 0x0002 1
HMI: 802.15.4 MAC Initialized Successfully:
HMI:    Interface ID:     1 (0x00000001)
HMI:    Extended Address: 000000FFFE000002
HMI:    Short Address:    2 (0x0002)
HMI:    PAN ID:           1 (0x0001)
HMI:    Sleepy:           Yes
```

The extended address is pulled from OTP or NVM. If the extended address is zero, it is set using the SetPIBRequest command ("`SetPIBRequest 99 ________`").

6.3.4.29.2 Device table

Other commands in the HMI demo take the destination address as an entry in the device table. Before using any of these commands, an entry must be added to the table as follows.

Dev1:

```
HMI> AddDevice 000000FFFE000002 0x0002 1
HMI: Device added successfully (DeviceIndex=1).
```

Dev2:

```
HMI> AddDevice 000000FFFE000001 0x0001 0
HMI: Device added successfully (DeviceIndex=1).
```

6.3.4.29.3 Device joining

Association

To join two devices using association, there are a number of things that need to be setup. The PAN ID has already been taken care of (see 6.3.4.29.1). The Beacon Payload and permit join PIBs must be set and the PAN must be started. In this example, the beacon payload is being set to "My_Beacon" and the PAN is being started on channel 11. The beacon payload length is automatically set with the beacon payload.

Dev1:

```
HMI> SetPIBRequest 0x45 My_Beacon

HMI: qapi_HMI_MLME_Set_Request() success.

HMI> SetPIBRequest 0x41 1

HMI: qapi_HMI_MLME_Set_Request() success.

HMI> StartRequest 11

HMI: qapi_HMI_MLME_Start_Request() success.
```

After the coordinator is setup, the sleepy device can scan for the network (scanning is not necessary if the device already knows the coordinators address and channel). In this example, the sleepy device is scanning all supported channels (11–26) for 250 ms each.

Dev2:

```
HMI> ScanRequest 1 0x07FFF800 250

HMI: qapi_HMI_MLME_Scan_Request() success

HMI: MLME-BEACON-NOTIFY.indication:
HMI:   BSN:              20
HMI:   PANDescriptor:
HMI:     CoordAddrMode:   2
HMI:     CoordPANId:      0x0001
HMI:     CoordAddress:    0x0001
HMI:     LogicalChannel:  11
HMI:     ChannelPage:     0
HMI:     SuperframeSpec
HMI:       BeaconOrder:          15
HMI:       SuperframeOrder:      15
HMI:       FinalCAPSlot:         15
HMI:       BatteryLifeExtension: FALSE
HMI:       PANCoordinator:       FALSE
HMI:       AssociationPermit:    TRUE
HMI:     GTSPermit:       FALSE
HMI:     LinkQuality:     0
HMI:     Timestamp:       0
HMI:     SecurityFailure: 0
HMI:     Security:
HMI:       SecurityLevel: 0
HMI:       KeyIdMode:     0x00
HMI:       KeyIndex:      0x00
HMI:   NumShortAddr:      0
HMI:   NumExtendAddr:     0
HMI:   SDULength:         9
HMI:   SDU:
HMI:     0000 4D 79 5F 42 65 61 63 6F 6E
My_Beacon

HMI: MLME-SCAN.confirm:
HMI:   Status:            0x00 (SUCCESS)
HMI:   ScanType:          1
HMI:   ChannelPage:       0
HMI:   UnscannedChannels: 0x00000000
HMI:   ResultListSize:    1
HMI:   Scan results:
HMI:     Result  0:
HMI:       CoordAddrMode:   2
HMI:       CoordPANId:      0x0001
HMI:       CoordAddress:    0x0001
HMI:       LogicalChannel:  11
HMI:       ChannelPage:     0
```

The device associates with the PAN.

Dev2:

```
HMI> AssociateRequest 11 0x0001 0xC0

HMI: qapi_HMI_MLME_Associate_Request() success

HMI: MCPS-ASSOCIATE.confirm:
HMI:   AssocShortAddr:  0x0002
HMI:   Status:          0x00 (SUCCESS)
HMI:   Security:
HMI:     SecurityLevel: 0
HMI:     KeyIdMode:     0x00
HMI:     KeyIndex:      0x00
```

Dev1:

```
HMI: MCPS-ASSOCIATE.indication:
HMI:   DeviceAddress:   0x000000FFFE000002
HMI:   Capability:      0xC0
HMI:   Security:
HMI:     SecurityLevel: 0
HMI:     KeyIdMode:     0x00
HMI:     KeyIndex:      0x00

HMI: qapi_HMI_MLME_Associate_Response() success

HMI: MLME-POLL.indication received from 000000FFFE000002

HMI: MLME-COMM-STATUS.indication:
HMI:   PANId:       0x0001
HMI:   SrcAddrMode: 3
HMI:   SrcAddr:     000000FFFE000001
HMI:   DstAddrMode: 3
HMI:   DstAddr:     000000FFFE000002
HMI:   Status:      0x00 (SUCCESS)
HMI:   Security:
HMI:     SecurityLevel: 0
HMI:     KeyIdMode:     0x00
HMI:     KeyIndex:      0x00
```

No association

If association is not used, the join procedure is a bit simpler. As the PAN ID and short address of both devices have been set, only the channel number is required. This example sets the channel of both devices to 11.

Dev1:

```
HMI> SetPIBRequest 0x00 11

HMI: qapi_HMI_MLME_Set_Request() success.
```

Dev2:

```
HMI> SetPIBRequest 0x00 11

HMI: qapi_HMI_MLME_Set_Request() success.
```

Setting the beacon payload and scanning is performed similar to the Association case, (see 6.3.4.29.3) though the scan payload may be slightly different.

6.3.4.29.4 Sending data

Direct

After everything is setup, direct data transmission is very simple. The following example sends a packet from the sleepy device to the coordinator with acknowledgement (the end of the MSDU data has been truncated).

Dev2:

```
HMI> DataRequest 1 1

HMI: qapi_HMI_MCPS_Data_Request() success
HMI: MSDUHandle = 0

HMI: MCPS-DATA.confirmation:
HMI:   MSDUHandle: 0
HMI:   Status:     0x00 (SUCCESS)
HMI:   Timestamp:  0
```

Dev1:

```
HMI: MCPS-DATA.indication:
HMI:   SrcAddrMode:     2
HMI:   SrcPANID:        0x0001
HMI:   SrcAddr:         0x0002
HMI:   DstAddrMode:     2
HMI:   DstPANID:        0x0001
HMI:   DstAddr:         0x0001
HMI:   DSN:             201
HMI:   Timestamp:       0
HMI:   Security:
HMI:     SecurityLevel: 0
HMI:     KeyIdMode:     0x00
HMI:     KeyIndex:      0x00
HMI:   MPDULinkQuality: 0
HMI:   MSDULength:      116
HMI:   MSDU:
HMI:     0000 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E …
HMI:     0010 71 72 73 74 75 76 77 78 79 7A 41 42 43 44 …
HMI:     0020 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 …
HMI:     0030 57 58 59 5A 30 31 32 33 34 35 36 37 38 39 …
HMI:     0040 36 35 34 33 32 31 30 5A 59 58 57 56 55 54 …
```

```
HMI:     0050 51 50 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 …
HMI:     0060 41 7A 79 78 77 76 75 74 73 72 71 70 6F 6E …
HMI:     0070 6B 6A 69 68                               …
```

Indirect

Indirect data transmission is slightly more complicated since the data must be polled. The following example sends a data packet from the coordinator to the sleepy device with acknowledgement (the end of the MSDU data has been truncated). The data request sent in first will timeout if the poll request is not issued in time.

Dev1:

```
HMI> DataRequest 1 1

HMI: qapi_HMI_MCPS_Data_Request() success
HMI: MSDUHandle = 0
```

Dev2:

```
HMI> PollRequest 1

HMI: qapi_HMI_MLME_Poll_Request() success

HMI: MLME-POLL.confirm:
HMI:   Status: 0x00 (SUCCESS)

HMI: MCPS-DATA.indication:
HMI:   SrcAddrMode:     2
HMI:   SrcPANID:        0x0001
HMI:   SrcAddr:         0x0001
HMI:   DstAddrMode:     2
HMI:   DstPANID:        0x0001
HMI:   DstAddr:         0x0002
HMI:   DSN:             22
HMI:   Timestamp:       0
HMI:   Security:
HMI:     SecurityLevel: 0
HMI:     KeyIdMode:     0x00
HMI:     KeyIndex:      0x00
HMI:   MPDULinkQuality: 0
HMI:   MSDULength:      116
HMI:   MSDU:
HMI:     0000 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E …
HMI:     0010 71 72 73 74 75 76 77 78 79 7A 41 42 43 44 …
HMI:     0020 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 …
HMI:     0030 57 58 59 5A 30 31 32 33 34 35 36 37 38 39 …
HMI:     0040 36 35 34 33 32 31 30 5A 59 58 57 56 55 54 …
HMI:     0050 51 50 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 …
HMI:     0060 41 7A 79 78 77 76 75 74 73 72 71 70 6F 6E …
HMI:     0070 6B 6A 69 68                               …
```

Dev1:

```
HMI: MLME-POLL.indication received from 0x0002

HMI: MCPS-DATA.confirmation:
HMI:   MSDUHandle: 0
HMI:   Status:     0x00 (SUCCESS)
HMI:   Timestamp:  0
```

6.3.4.30 Configuring security

Half of setting up security is taken care of when the devices are added to the device table (see section 6.3.4.29.2). The following example sets up the remaining security information on both devices using security level 5 (32-bit MIC and encryption) and the default key source.

Dev1:

```
HMI> SetSecurity 5 1
HMI: Security set successfully
```

Dev2:

```
HMI> SetSecurity 5 1
HMI: Security set successfully
```

The security will be used for all applicable commands (data requests, association, and so on).

6.3.5 WLAN

This section demonstrates the commands related to Wi-Fi operation.

Glossary of abbreusingtions

AP: Access point.

P2P: Peer-to-peer network

Soft-AP: Device configured as an access point.

STA: Device configured in station mode.

TU: Time Units (1 TU = 1024 microseconds)

6.3.5.1 Ver

Displays the current QAPI version and CRM build version.

6.3.5.2 Help

This command is used to display a command list or usage for a command. It is the first command in the list for each menu level.

6.3.5.3 Up

This command is used to navigate up one group level. For instance, if the command is executed from within the "MyDemo\SubPart1" command group, QCLI will navigate to the "MyDemo" command group. This command is only available within a group or subgroup (not at the root level).

6.3.5.4 Root

This command is used to immediately navigate to the root group level. It is only available within a group or subgroup.

6.3.5.5 QueryVersion

This command displays the version of WLAN software and interface. Interface version is currently set to 1. This command will only work after WLAN is enabled.

6.3.5.6 Enable

This command enables the WLAN module. WLAN needs to be enabled before using WLAN related commands enlisted in this section such as scan, connect, disconnect and also networking commands such as DHCP, ping, and running traffic. This command also performs the following operations after WLAN is enabled.

6. Registers application callback with WLAN driver. This driver notifies various asynchronous events to application through this callback function.

CAUTION: Invokes qapi_WLAN_Add_Device() to register every WLAN virtual device to the networking stack.

Perform any other initialization operation in application which is necessary for a functional WLAN subsystem.

6.3.5.7 Disable

This command disables the WLAN module by de-registering the WLAN virtual devices from the networking stack and bringing down the WLAN driver. This turns off the WLAN subsystem. No WLAN operation can be performed after this, unless WLAN is enabled again.

6.3.5.8 Reset

This command is deprecated. To reset the WLAN module, use WLAN disable and enable commands in that order.

6.3.5.9 Info

This command displays network SSID, physical mode, power mode, MAC address, and operating mode for the currently selected device. If there are multiple devices, then use SetDevice command to change the device in use.

6.3.5.10 SetDevice

Parameter Name	Type/Range	Description
Device	0–1	To set the operating device ID

This command is used to set the device ID (0 or 1). All device specific WLAN commands such as info, connect, and disconnect are applied for the set device ID. WLAN module uses device 0 by default.

SoftAP and P2P operations Example: P2P-GO and P2P-STA should be used only on device 0.

In case of features like MCC and SCC, there is a requirement to set the parameters for both the operating devices (device 0 and device 1).

6.3.5.11 Scan

Parameter Name	Type/Range	Description
mode (optional)	0 – 2	0: Blocking mode. Application will block until the scan operation completes and scan results are received. This is the default scan mode if no input parameter is given. 1: Non-blocking scan mode. Application will not block, the host driver will buffer the results. After the scanning procedure completes, host driver will invoke the callback registered by the application which can then retrieve the results. 2: Non-blocking, no-buffering scan mode. Application will not block and the host driver will not buffer the scan results. Each BSS information received during the scan is directly given to the application as individual events and it is up to the application to manage the scan results.
ssid (optional)	String	Network SSID for a directed scan. Scan mode must be given as first parameter for using SSID parameter.

This command is used to scan for APs when the device is in station mode. This command does not apply to device if it is configured as a soft-AP. If P2P session is in progress, then this command returns an error. For mode 1 & 2, the maximum number of BSSs than can be buffered on WLAN driver is 12.

6.3.5.12 SetScanParameters

This command is used to configure specified parameters for scan procedure.

Parameter Name	Type/Range	Description
max_active_chan_dwell_time_ms	Milliseconds (uint16_t)	Dwell time at each channel during active scan mode. Station actively sends out probe requests and waits for probe responses from APs. Default value is 20ms. If 0 is given, the value is set to the default value.
passive_chan_dwell_time_ms	Milliseconds (uint16_t)	Channel dwell time for passive scan mode in which the client waits at each channel to receive beacon frames from APs within the proximity. Default value is 50ms. If 0 is given, value is set to the default value.

Parameter Name	Type/Range	Description
fg_start_period_in_sec	Seconds (uint16_t)	Foreground scan start period in seconds. Default value is 1 second. 0: Set to default value 0xFFFF: Disable periodic foreground scan
fg_end_period_in_sec	Seconds (uint16_t)	Foreground scan end period in seconds. Default value is 60 seconds. 0: Set to default value
bg_period_in_sec	Seconds (uint16_t)	Periodic background scan period in seconds. Stations can use this mode to scan for neighboring Aps without disconnecting from the AP it is currently connected to. Default value is 60 seconds. 0: Set to default value 0xFFFF: Disable periodic background scan
scan_ctrl_flags	uint8_t	0x00: Use current scan control settings 0xFF: Disable all scan control flags, this will set the flag to 0. For setting specific scan control flags, use each bit in this byte sized field as one flag. For each bit, 1=True, 0=False. Bit0: Scan allowed during connection. Bit1: Scan allowed for the SSID station is already connected. Bit2: Enable active scan. Bit3: Scan allowed for roaming when beacon misses or low signal strength is identified. Bit4: Customer BSSINFO reporting rule should be followed. Bit5: Device should scan after a disconnection. Bit6: Scan complete event with canceled status should be generated when a scan is preempted before completion. Bit7: Scanning DFS channels should be skipped.
min_active_chan_dwell_time_ms	Milliseconds (uint16_t)	Minimum dwell time at each channel in case there is no activity during active scan mode. Default value for it is 0ms. In this case, dwell time at each channel follows the configured duration even when there is no activity on the channel.
max_active_scan_per_ssid	Milliseconds (uint16_t)	Maximum number of scans allowed per channel to search for configured SSIDs during active mode. Default value is 1.
max_dfs_chan_active_time_in_ms	Milliseconds (uint16_t)	Maximum allowed time for scanning in DFS channels. Default is 2000 ms.

6.3.5.13 CustomScan

This command is reserved for future use.

6.3.5.14 Connect

This command is used to connect to the network.

Parameter Name	Type/Range	Description
ssid	String	SSID for the network
bssid (optional)	String	BSSID for specified ssid. Applicable only, whenever DUT is operating in STA mode.

The Bssid option can be used, whenever multiple BSSs operating with same SSID, and user wanted to connect to perticaulr AP. This is optional parameter and only used in STA mode.

Prerequisites:

CAUTION: Set the operating mode for the device to be connected. For example, if device 0 is configured as a soft-AP, it starts the AP operation with this command. Default is Infra-STA mode

CAUTION: If Security mode is required, key should be set using commands such as setWpaPassphrase, setWpaParams, setWepKeyPair, and setWepKey before using connect command.

6.3.5.15 GetRegulatoryDomain

This command is used to retrieve the current regulatory domain code.

6.3.5.16 SetCountryCode

Parameter Name	Type/Range	Description
country_code_string	String	Country code to be set.

This command sets the country code to operate. Country code should be two characters.

Example: SetCountryCode US

This command is only for OEM and should not be made available to the end customers.

6.3.5.17 GetCountryCode

Currently, this command is not supported. In future revisions, it will be used to get the country code of the operating device (typically STA mode).

6.3.5.18 GetLastError

Currently, this command is not supported. In future revisions, this command will be used to query the last error information in WLAN driver.

6.3.5.19 SetMacAddress

This command is not supported. MAC address can be set only from manufacturing tool.

6.3.5.20 SetWpaPassphrase

This command is used to configure the WPA or WPA2 Security Passphrase. This should be done before connecting to an AP or starting a Soft-AP.

Parameter Name	Type/Range	Description
passphrase	String	Security Passphrase

Example: SetWpaPassphrase 12345678

6.3.5.21 SetWpaParameters

This command is used to configure 802.11 Security Parameters WPA/WPA2 PSK/Enterprise. Mixed mode for unicast and multicast cipher is not supported. QCA402x WLAN can either be a pure WPA or WPA2 device.

Version: WPA/WPA2/WPACERT/WPA2CERT

Chiper: CCMP/TKIP

Parameter Name	Type/Range	Description
version	String	Version
ucipher	String	Type of Unicast cipher
mcipher	String	Type of Mulitcast cipher

Example: "SetWpaParameters WPA2 CCMP CCMP"

6.3.5.22 SetWepKeyPair

This command is used to configure WEP parameters such as keyIndex and Key in pair. WEP key length will be 10(WEP40)/26(WEP104) characters in Hexadeicmal or 5(WEP40)/13(WEP104) ASCII characters.

Parameter Name	Type/Range	Description
keyindex	1-4	WEP Key Index
key	String	WEP Key

Example: SetWepKeyPair 1 abcde

6.3.5.23 SetWepKey

This command is used to set the default WEP key index assuming that the key has been already populated in the given index.

Parameter Name	Type/Range	Description
keyindex	1 – 4	Default WEP key Index

Example: SetWepKey 3

6.3.5.24 AdhocConnect

This command is used to connect the unsecured Ad-hoc Network on specified SSID and channel.

Parameter Name	Type/Range	Description
ssid	String	SSID of Ad-hoc Network to Connect
channel (Optional)	uint32_t	Channel number

If an ad-hoc network already exists with given SSID, the device joins to the specified network. If an ad-hoc network doesn't exist, this command creates an ad-hoc network. Channel is optional parameter. If user doesn't provide the channel number, then device will scan all the channels with specified SSID and connect.

Example: AdhocConnect ADHOC_SSID 11

6.3.5.25 AdhocWepConnect

This command is used to connect the WEP Secured Ad-hoc Network on specified SSID and channel with the WEP Key index.

Channel is an optional parameter. If the user does not provide the channel number, then DUT will scan all the channels with the specified SSID and connect.

Parameter Name	Type/Range	Description
ssid	String	SSID of Ad-hoc Network to Connect
def_keyix	1-4	Wep Key Index
channel (Optional)	uint32_t	Channel Number

Example: AdhocWepConnect ADHOC_SSID 1 6

6.3.5.26 WpsPin

This command is used to setup and start a WPS connection using the pin method. The parameters SSID, MAC, and Channel are optional parameters.

Parameter Name	Type/Range	Description
connectFlag	0–1	WPS connection Flags 0-No Action to be taken After WPS inititation is successful 1-Setup a connection after WPS initiation is successful
pin	String	WPS Pin
ssid (Optional)	String	SSID
mac (Optional)	uint32_t	Mac Address
channel (Optional)	uint32_t	Channel Number

At any point of time, WPS operation can be performed in only one of the virtual devices. If WPS is being performed on device 1, the same cannot be performed on device 0 till device 1's WPS operation completes.

Example: WpsPin 1 99316334 linksys_WPS_x7ze 00226b68a86c 6

6.3.5.27 WpsPush

This command is used to setup and start a WPS connection using Push method.

Parameter Name	Type/Range	Description
connectFlag	0–1	WPS connection Flags 0-No Action to be taken After WPS initiation is successful 1-Setup a connection after WPS initiation is successful.
ssid (Optional)	String	SSID
mac (Optional)	uint32_t	Mac Address
channel (Optional)	uint32_t	Channel Number

At any point of time, WPS operation can be performed in only one of the virtual devices. If WPS is being performed on device 1, the same cannot be performed on device 0 till device 1's WPS operation completes.

Example: WpsPush 1 linksys_WPS 845dd741e083 1

6.3.5.28 Disconnect

This command is used to disconnect the device from the BSS. This command can be used to disconnect from the network irrespective of the operating mode (AP/Infra STA/IBSS).

6.3.5.29 SetTxRate

This command is used to configure Fixed Tx rate for data packet.

Parameter Name	Type/Range	Description
MCS	String	Specifies whether the given rate is MCS rate or not. MCS parameter is valid only if the user wants to set mcs data rate.
rate_in_Mbps / mcsIndex	uint32_t	This entry takes either Rate in Mbps or MCS rate index based on MCS parameter Rate in Mbps – 1,2,5,6,9,11,12,18,24,36,48,54 MCS rate Index – 0,1,2,3,4,5,6,7

Example: 'SetTxRate mcs 5' – sets the Fixed Tx Data rate to MCS5

If user wants to set the MCS rate as Fixed rate, user needs to give the 'mcs' string as first parameter and mcs rate index as second parameter.

If user want to set the 11a/b/g rates as Fixed rate, then the parameter is 'mcs' is invalid.

Example: 'SetTxRate 24' – Sets the Fixed Tx Data rate to 24 Mbps

6.3.5.30 GetTxRate

This command is used to get the modulation rate of the last transmitted packet.

6.3.5.31 SetTxPower

This command is used to configure the transmit power

Parameter Name	Type/Range	Description
txPower	0–63	Transmit power in dbm

Every increment here corresponds to 1 dBm increment in power. Even though the range allowed 0-63, the WLAN device supports only max transmit power 17 dBm.

6.3.5.32 SetPowerMode

This command configures the power mode for the device in use.

Parameter Name	Type/Range	Description
Mode	0–1	0 – Max Performance 1 – Power Save

There are two options available for the same, max performance and power save mode. In max performance mode, WLAN MAC/RF and CPU stays active, whereas in power save mode both WLAN MAC/RF and WLAN CPU enters power save mode. If WLAN is connected when in power save mode, the WLAN subsystem wakes up only for every beacon interval to receive beacons from the connected AP after which it again enters sleep.
When WLAN is enabled, by default, both virtual devices comes up in max performance mode and it is up to the application to put every virtual device to low power mode in order to achieve least power consumption. Also, when operating in IBSS/Soft-AP or P2P GO mode, WLAN subsystem should only happen in max performance mode.

6.3.5.33 SetChannel

This command configures the channel on which WLAN will operate. Application should ensure that the channel provided to be set complies with the regulatory domain restrictions.

Different regulatory restrictions apply.

Parameter Name	Type/Range	Description
Channel	1 – 14* 36 – 165*	Set Channel for the given device.

6.3.5.34 SetStaListenInterval

This command configures the Beacon Listen Interval Value for the station mode. The default value is 100.

Parameter Name	Type/Range	Description
Interval (in time units(TUs))	15–5000	Beacon Listen Interval for Station Mode

6.3.5.35 SetPhyMode

This command configures the physical mode in which wlan will operate. By default, the device will operate in 11n if g mode is selected.

Parameter Name	Type/Range	Description
Mode	a/b/g/ag/gonly	Set WLAN operating mode

6.3.5.36 Set11nHTCap

This command allows user to enable/disable 11n. When 11n is enabled, QCA402x WLAN only supports HT20 mode of operation.

Parameter Name	Type/Range	Description
HTCap	Disable/HT20	Set 11n HT parameter Disable – Disables 11n HT20 – Enables 11n and configures the same in 20Mhz mode.

6.3.5.37 GetRssi

This command is used to get RSSI (Received Signal Strength Indication) of current operating device in STA or P2P mode.

6.3.5.38 SetRoamThreshold

Parameter Name	Type/Range	Description
Lower Threshold	60 – 70	Set lower Roam Threshold (RSSI)
Upper Threshold	70 – 80	Set upper Roam Threshold (RSSI)
Weight		
Poll time		Time interval to poll for roaming

6.3.5.39 Roaming

Command to enable/disable roaming

Parameter Name	Type/Range	Description
Roaming	Enable/disable	Enable or Disable Roaming

6.3.5.40 EnableSuspend

This command is deprecated and suspend will be always enabled.

6.3.5.40.1 Suspend

Command to initiate suspend for WLAN module for the specified time. When WLAN is suspended, no WLAN operation is possible till it is resumed. WLAN suspend is not supported in IBSS/SoftAP/P2P GO modes.

Parameter Name	Type/Range	Description
time	Time in ms	Specify total suspend time.

6.3.5.41 SetPowerPolicy

Set power management parameter. This command helps in setting up the power policy. All the parameters are required.

Parameter Name	Type/Range	Description
idle_time_in_ms	ms(uint16_t)	Idle period during traffic, after which WLAN can go to sleep.
Pspollnum	uint16_t	Number of contiguous ps poll frames that wlan should send before sending 802.11 NULL frame to indicate power save exit to the AP.
Dtim_policy	1.Ignore 2.Normal 3.Stick 4.Auto	Set DTIM interval policy
Txwakeupolicy	1. wake host 2. do not wake host	Set the host wake up policy.
numTxToWakeup	uint16_t	Number of contiguous transmit packets after which WLAN firmware should exit protocol power save mode. This parameter is effective only if txwakeuppolicy is set to wake the host.
PsFailEventPolicy	1.Send Fail Event 2.Ignore Event	Set the power save failure event handling.

6.3.5.42 SetAggregationPolicy

Command to enable/disable Tx/Rx Aggregation on each TID

Parameter Name	Type/Range	Description
tx_tid_mask	0x0 – 0xFF	Set Tx tid bitmask. Bit 0 corresponds to TID 0, bit 1 to TID 1, and so on till TID 7
rx_tid_mask	0x0 – 0xFF	Set Rx tid bitmask. Bit 0 corresponds to TID 0, bit 1 to TID 1, and so on till TID 7

6.3.5.43 SetAggrx

This command sets Rx aggregation parameters and enable Rx aggregation.

Parameter Name	Type/Range	Description
buffer size	Frame count (uint8_t)	Set buffer size by frame count
reorder timeout	ms (uint8_t)	Set timeout in ms
session timeout	ms (uint8_t)	Set session timeout in ms
aggrx enable	Enable/disable	Enable aggrx
session timer	Enable/disable	Enable session timer

6.3.5.44 SetOperatingMode

This command sets the operating mode for a particular device. The device can operate in AP or station mode. Additionally, if configured in AP mode, the SSID can be configured in hidden mode with wps enabled or disabled.

Parameter Name	Type/Range	Description
Mode	ap/station	Set operating mode
param1	hidden/0	Set SSID in hidden or non hidden mode (only for AP mode)
Wps	wps/0	Enable WPS (only for AP mode if AP needs to support WPS).

6.3.5.45 Set AP beacon interval

This command sets AP beacon Interval.

Parameter Name	Type/Range	Description
Beacon_interval_in_time units	100 – 1500	Set AP beacon Interval

6.3.5.46 SetApDtimPeriod

This command setsAP DTIM period.

Parameter Name	Type/Range	Description
dtim_period	1 – 255	Set AP DTIM Period

6.3.5.47 SetApPowerSaveBufferParameters

This command is used to enable power save buffers and to configure the power save buffer count.

Parameter Name	Type/Range	Description
enable/disable	0–1	Enable (1) or Disable(0) power save buffer
buffer_count	Uint32_t	Total buffer count

6.3.5.48 SetApInactivityPeriod

This command sets AP inactivity period in min After this period, the AP will automatically shut down.

Parameter Name	Type/Range	Description
inactivity_period_in_mins	Uint32_t	AP inactivity period

6.3.5.49 setAPUapsd

This command is used to enable or disable UAPSD in soft-AP mode.

Parameter Name		Type/Range	Description
enable/disable		0 – 1	Enable (1) or disable (0) AP UAPSD.

6.3.5.50 setSTAUapsd

This command is used to configure UAPSD capability for each access category.

Parameter Name	Type/Range	Description
AC_mask	0 – 0xF	Configure U-APSD in station mode for each access category. Each bit corresponds to one access category as the following: Bit0: BE, Bit1:BK, Bit2:VI, Bit3:VO Bit set to 1 indicates U-APSD enabled and bit set to 0 indicates U-APSD disabled.

6.3.5.51 setMaxSPLen

Set the Max SP Len in QoS Info field.

Parameter Name	Type/Range	Description
maxSP_Len	0 – 3	0 - WMM AP may deliver all buffered frames. 1 - WMM AP may deliver a maximum of 2 buffered frames. 2 - WMM AP may deliver a maximum of 4 buffered frames. 3 - WMM AP may deliver a maximum of 6 buffered frames.

6.3.5.52 ForwardProbeReqToHost

Reserved for future use.

6.3.5.53 GetStats

This command is used to retrieve device specific and common WLAN statistics for data management and control traffic.

Parameter Name	Type/Range	Description
resetCounters	0, 1	1: Resets counters after the stats is obtained. 0: Continue counters without resetting.

6.3.5.54 SendRawFrame

This command is used to send raw packets with given attributes.

Parameter Name	Type/Range	Description
rate_index	0 – 18	rate index. 0- 1mbps; 1- 2mbps; 2- 5.5mbps
num_tries	1–14	number to times the frame to be sent
num_bytes	0–1400	Packet size.
channel	1–11	Sending channel.
type	1 – 2	0: Beacon, 1: QoS Data, 2: 4-addr data
addr1 (Optional)		MAC address if type is 2
addr2 (Optional)		MAC address if type is 2
addr3 (Optional)		MAC address if type is 2
addr4 (Optional)		MAC address if type is 2

6.3.5.55 EnableLPL

This command is used to enable/disable low power listen, which is a Rx power optimization feature. WLAN listen is compromised by operating some of the receive components with reduced power.

Parameter Name	Type/Range	Description
enable / disable	0 – 1	Enable (1) or disable (0) Low Power Listen mode.

6.3.5.56 EnableGTX

This command is used to enable/disable GreenTX which is a power optimization feature where WLAN transmit power is reduced. This is done while operating under good link conditions thereby reducing the power consumption.

Parameter Name	Type/Range	Description
enable / disable	0 – 1	Enable (1) or disable (0) Green Tx mode.

6.3.5.57 EnablePreferredNetworkOffload

This command is used to enable/disable preferred network offload (PNO) feature with given attributes.

Parameter Name	Type/Range	Description
enable / disable	0 – 1	Enable (1) or disable (0) Preferred Network Offload.
max_profiles	1 –15	Valid only for enable.
fast_scan_interval	Milliseconds (uint32_t)	Fast scan interval. Valid only for enable command.
fast_scan_duration	Milliseconds (uint32_t)	Fast scan duration.Optional. Valid only for enable.
slow_scan_interval	Milliseconds (uint32_t)	Slow scan interval. Valid only for enable.

6.3.5.58 SetPnoNetworkProfile

This command is used to add WLAN BSS profiles to the PNO scan list after the PNO feature is enabled. The added profile is used for matching during PNO scan after the station disconnects from AP. PNO scans are not performed when WLAN STA is connected.

Parameter Name	Type/Range	Description
index	0 – 5	Index of network profile.
ssid	String (length < 32)	ssid.
mode	String	open|wep|wpa|wpa2
wpa_cipher	Optional	TKIP or CCMP

6.3.5.59 OffloadTcpSessionKeepAlive

This command registers a TCP session with TCP KeepAlive offload manager. After this is enabled, WLAN firmware takes care of TCP Keepalive transmissions thereby allowing application processor to enter suspend for longer duration.

Parameter Name	Type/Range	Description
tcp_src_port	int16_t (> 0)	TCP source port.
tcp_dst_port	int16_t (> 0)	TCP destination port.
src_ip	uint32_t	Source IP address
dst_ip	uint32_t	Destination IP address.
dest_mac_addr	uint8_t	Destination MAC address.
seq_num	int32_t (> 0)	Sequence number.
ack_seq	int32_t (> 0)	ACK sequence number.
ip protocol	4, 6	4- IPV4; 6- IPV6

6.3.5.60 EnableTcpKeepaliveOffload

Parameter Name	Type/Range	Description
enable / disable	0–1	Enable (1) or disable (0) TCP keepalive offload.
keepalive_interval	Seconds (uint16_t)	Interval for periodic TCP keepalive packets.
keepalive_ack_threshold	uint16_t	Number of contiguous keepalive packets after which application processor is woken up if TCK ACK is not received from the peer for any of them.

6.3.5.61 SetStaMacKeepAliveTimeout

This command is used to configure MAC keepalive timeout in station mode. Station will send keepalive packets (NULL data frames) to AP if there is no other traffic for the given interval.

Parameter Name	Type/Range	Description
timeout_in_secs	Seconds (int32_t)	0: Disable the MAC keepalive feature for station. >0: Keep alive interval

6.3.5.62 SetPromiscuous

This command is used to enable/disable promiscuous mode and to configure filters. If application requires to set filter for promiscuous mode, the filter should be set first before enabling the promiscuous mode.

Parameter Name	Type/Range	Description
enable / filter	String	“enable”: Enables the promiscuous mode. “filter”: This option takes one more input parameter and adds or resets a promiscuous filter.
config / reset	String	This input parameter is valid only if first parameter to command is “filter”. “config”: Configures a filter with given attributes. “reset”: Resets all the promiscuous mode filters.

6.3.5.63 Enable80211v

This command is used for both station and soft-AP modes to enable 802.11v features, viz. WNM sleep and BSS max idle period. Other 802.11v features are not supported at this time.

Parameter Name	Type/Range	Description
enable / disable	0,1	Enable (1) or disable (0) 802.11v features.

6.3.5.64 SetWnmSleepPeriod

This command configures the WNM sleep period in station mode and does not apply to soft AP mode.

Parameter Name	Type/Range	Description
Action	0,1	1: Enter WNM sleep. 0: Exit WNM sleep.
period_in_DTIM_intervals	uint16_t	WNM sleep period in multiple of DTIM intervals.

For using this feature, station should be connected to an AP which supports 802.11v features. Sleep interval for the station should be less than BSS max idle period advertised by the AP in beacons. It is mandatory to enable 11v before issuing this command.

6.3.5.65 SetAPBssMaxIdlePeriod

Use this command to configure BSS max idle period. Command applies only to soft-AP mode.

Parameter Name	Type/Range	Description
period_in_1000TUs	uint16_t	BSS max idle period in terms of 1000TUs

6.3.5.66 SetWnmSleepResponse

Reserved for internal use.

6.3.5.67 EnableWow

This command enables or disables the WoW(wake-up on Wireless) functionality. User needs to enable this flag before adding/configuring WoW action in any of the packet filter patterns.

Parameter Name	Type/Range	Description
enable / disable	0–1	Enable (1) or disable (0) Wow. Default 0

6.3.5.68 EnablePktFilter

This command enables or disables the packet filter feature. Disable Packet filter will delete all packet filter patterns including default packet filters.

Parameter Name	Type/Range	Description
enable / disable	0–1	Enable (1) or disable (0) PacketFilter. Default 0

6.3.5.69 AddFilterPattern

This command is used to add the packet filter. If user wants to change the existing pattern index, then AddFilter works as update.

Parameter Name	Type/Range	Description
index	1–8	Packet Filter Index Index 0 is reserved for Default Filter in each protocol header type.
action	uint32_t	Action Flags 1- Reject, 2- Accept, 4- Defer
wow_flag	0–1	Enable (1) :Wakes up application processor throughout the band interrupt in case of pattern match. Disable (0)
priority	1–255	Priority of the pattern within the Header, higher the number, higher the priority. Priority 0 is reserved for Default Filter in each protocol header type
headerType	1–11	2. 802.3 MAC 3. SNAP 4. ARP 5. IPv4 6. IPv6 7. ICMP 8. ICMPv6 9. UDP 10. TCP 11. PAYLOAD
offset	uint32_t	Offset from the header to which pattern match algorithm is applied.
pattern_size	uint32_t	Size of pattern to search for in the received packet.
pattern_mask	uint32_t	Mask to do selective pattern match. Each bit in the mask corresponds to byte in the pattern. If a bit is not set, corresponding byte in the pattern is ignored, else it is matched against received byte.
pattern	uint32_t	Maximum support pattern length is 128 bytes (value should be given in hexadecimal)

Example: *addFilterPattern* 1 2 1 6 4 12 4 0f c0a80103

It creates the packet filter for protocol-4 in index 1 with action of WoW and Accept flag set.

Accept packet with source IP address of 192.168.1.3 and raise the wow interrupt to process.

6.3.5.70 DeleteFilterPattern

This command is used to delete the existing packet filers. Default filter pattern can be deleted only as part of packet filter disable.

Parameter Name	Type/Range	Description
index	1–8	Packet Filter Index Index 255: If the user wants to delete all packet filter patterns in specified protocol header except the default pattern.
header_type	1–11	Protocol Header Value.

Example: deleteFilterPattern 2 1

Deletes the pattern with index 2 in protocol heady type 1 (which is MAC).

Example: deleteFilterPattern 255 3

Deletes all patterns (except default pattern index 0) in protocol header type 3.

6.3.5.71 ChangeDefaultFilterAction

This command is used if user wants to change the any default packet filter action flags.

Parameter Name	Type/Range	Description
action	uint32_t	Action Flags 1- Reject,2- Accept,4- Defer
wow_flag	0–1	Enable (1) or Disable (0)
header_type	1–11	Protocol Header Value.

Example: changeDefaultFilterAction 2 1 4

Updates the default filter action flag to Accept with wow_flag enabled for header type 4 (which is IPV4).

6.3.5.72 Dbglog

Dbglog enable

Parameter Name	Type/Range	Description
enable	0,1	Enable (1) or disable (0) debug logs.

Dbglog config

Parameter Name	Type/Range	Description
debug port	0,1	Local port which corresponds to push the debug logs through WLAN subsystem's debug port. This is not supported currently. Remote port which corresponds to application processor's debug port.
report enable	0,1	report logs; 0- not report
report size	Byte	Byte number. When the log size reaches this number, it is sent to host. 0 is the default. Size is 1500Byte.

Dbglog loglevel

Parameter Name	Type/Range	Description
Moduleid	Integer	See the following definition. 0xFF indicates all modules.
Loglevel	1,2,4,8,15	1- info; 2- low; 4- high; 8- error;15- all levels.

```
#define DBGLOG_MODULE_BOOT                    0
#define DBGLOG_MODULEID_INF                   1
#define DBGLOG_MODULEID_WMI                   2
#define DBGLOG_MODULEID_MISC                  3
#define DBGLOG_MODULEID_PM                    4
#define DBGLOG_MODULEID_TXRX_MGMTBUF          5
#define DBGLOG_MODULEID_TXRX_TXBUF            6
#define DBGLOG_MODULEID_TXRX_RXBUF            7
#define DBGLOG_MODULEID_WOW                   8
#define DBGLOG_MODULEID_WHAL                  9
#define DBGLOG_MODULEID_DC                    10
#define DBGLOG_MODULEID_CO                    11
#define DBGLOG_MODULEID_RO                    12
#define DBGLOG_MODULEID_CM                    13
#define DBGLOG_MODULEID_MGMT                  14
#define DBGLOG_MODULEID_TMR                   15
#define DBGLOG_MODULEID_BTCOEX                16
#define DBGLOG_MODULEID_BOOT                  17
#define DBGLOG_MODULEID_P2P                   18
#define DBGLOG_MODULEID_IEEE                  19
#define DBGLOG_MODULEID_MAC_CONC              20
#define DBGLOG_MODULEID_WNM                   21
#define DBGLOG_MODULEID_RFKILL                22
#define DBGLOG_MODULEID_NET_AUTOIP            23
#define DBGLOG_MODULEID_NET_DHCPV4            24
#define DBGLOG_MODULEID_NET_PKT               25
#define DBGLOG_MODULEID_NET_IFDRV             26
#define DBGLOG_MODULEID_NET_PORT              27
#define DBGLOG_MODULEID_NET_IPMAIN            28
#define DBGLOG_MODULEID_NET_TCP               29
#define DBGLOG_MODULEID_WPS                   30
#define DBGLOG_MODULEID_DSET                  31
#define DBGLOG_MODULEID_BMI                   32
#define DBGLOG_MODULEID_MBOX                  33
#define DBGLOG_MODULEID_DFS                   34
#define DBGLOG_MODULEID_SECURITY              35
#define DBGLOG_MODULEID_OTA                   36
#define DBGLOG_MODULEID_PAL                   37
#define DBGLOG_MODULEID_USB                   38
#define DBGLOG_MODULEID_WREG                  39
#define DBGLOG_MODULEID_OFFLOAD               40
```

Dbglog example:

CAUTION: If debug logs and report needs to be enabled through the remote port printing all log levels for all modules, the following command should be used.

```
Dbglog enable 1  //enable Dbglog
Dbglog config 1 1 0    // report to remote port
Dbglog loglevel 255 15    //255 all modules, 15 all levels
```

CAUTION: If debug logs and reporting are required through remote port printing only error level logs of offload module, the following command should be used.

```
Dbglog enable 1  //enable Dbglog
Dbglog config 1 1 0    // report to remote port
Dbglog loglevel 40 8    //OTA module, error level
```

6.3.5.73 Pktlog

Reserved for future use.

6.3.5.74 Regquery

Reserved for internal use.

6.3.5.75 Memquery

Reserved for internal use.

6.3.5.76 SetDriverAssert

Reserved for internal use.

6.3.5.77 SetEventFilter

This command enables or disables the event filter feature.

Parameter Name	Type/Range	Description
Enable/Disable	0 – 1	Enable (1) or disable (0) event filter. When enabled, subsequent eventIDs should be provided. When disabled, following parameters are not required.
eventId1	Unsigned integer	The filterable event ID which user decides to filter out, when Enable is 1.
eventId2	Unsigned integer	The filterable event ID which user decides to filter out, when Enable is 1.
eventId3	Unsigned integer	The filterable event ID which user decides to filter out, when Enable is 1.

When SetEventFilter is enabled, the event Id should be provided. The events will be discarded in WLAN firmware thereby avoiding application processor wakeups due to filtered events. Only filterable event IDs are valid. At least 1 event ID should be provided, and at most 64 event IDs are allowed. When it is disabled, all the filterable events are processed as normal.

6.3.5.78 ChannelSwitch

This command can be used in Soft-AP mode to trigger the channel switch announcement on BSS before switching to new channel.

Parameter Name	Type/Range	Description
channel	uint32_t	Channel Number: The new channel number BSS should be operating on after the TBTT count.
tbtt_count	uint32_t	Number of TBTT intervals, after channel switch should occur. This value will be advertised in Channel Switch IE.

6.3.5.79 ARPOffload

This command enables or disables the ARP offload feature.

Parameter Name	Type/Range	Description
Enable/Disable	0 – 1	Enable (1) or disable (0) ARP offload feature. When enabled, subsequent command parameters are required. When disabled, no more parameters are required.
targetip	String	Target IP address in string format. Example: 192.168.0.10.
mac	String	Target MAC address in string format without colon. Example: 001122334455.

While enabling ARPOffload, target IP/MAC address pair should be provided for WLAN firmware to respond to the ARP request which matches the configured IP address. Other ARP requests which do not match the IP address is dropped within the firmware. When disabled, the ARP requests are processed in the IP stack.

6.3.5.80 nsOffload

This command enables or disables the Neighbor Solicitation offload feature.

Parameter Name	Type/Range	Description
Enable/Disable	0 – 1	Enable (1) or disable (0) Neighbor Solicitations offload feature. When enabled, subsequent command parameters are required. When disabled, no more parameters are required.
target_global_ip	String	Target global IPv6 address.
target_link_ip	String	Target link-local IPv6 address.
mac	String	Target MAC address in string format without colon. Example: 001122334455.
solicitationip	String	Solicitation IPv6 multicast address.

When enabled, target global/link-local IPv6 address and target mac address should be provided to offload the neighbor solicitation processing to WLAN firmware. Solicitation IPv6 multicast address parameter is optional which can be provided when it expects the firmware to respond to solicitation request from a specific IPv6 address. When disabled, the neighbor solicitation will be processed as normal in the IP stack.

6.3.5.81 setApplicationIe

This command configures the application specified IE in the given frame type. IE data is taken as a string and each character is converted into its ASCII value (in hexadecimal format) before putting it in the frame.

Parameter Name	Type/Range	Description
Management frame type	0 – 2	0: beacon; 1: probe request; 2: probe response.
IE data	String	IE data string, starting with “dd”.

Example: '1' will show up as '0x31' in the frame as ASCII of '1' converted to hexadecimal is 31.

6.3.5.82 setStaBmissConfig

This command configures the threshold time of reporting disconnect event after the AP with which we have associated is powered down. Both parameters should be provided. The user can configure one parameter with valid value and set the other to 0. If both parameters are provided with valid value, then beacon-miss number overrides beacon miss time.

Parameter Name	Type/Range	Description
bmiss_Time_in_ms	1000 – 10000	Beacon-miss time in milliseconds.
num_Beacons	10 – 100	Beacon-miss time in number of beacons.

6.3.5.83 GetSSID

If the active device is connected, this command should be used to get the network SSID.

6.3.5.84 GetPhyMode

This command should be used to get active device’s wireless mode.

6.3.5.85 GetPowerMode

This command should be used to get active device’s power mode (max perf or power save mode).

6.3.5.86 GetMacAddress

This command should be used to get MAC address of the active device.

6.3.5.87 GetOperatingMode

This command should be used to get active device’s operating mode (soft-AP or station).

6.3.5.88 GetChannel

This command should be used to get the channel used by active device if connected.

6.3.5.89 GetWepKeyPair

Parameter Name	Type/Range	Description
KeyIndex	1–4	Index for the key to be retrieved.

6.3.5.90 xGetDefaultWepKeyIndex

This command should be used to get the index of default WEP key. The default output for this command is 1.

6.3.5.91 getRegDomainChannelList

This command can be used to get the supported Channel list for current regulatory domain.

If user likes to operate and get supported channel list in particular Phy mode, the user should set the Phymode by using setPhyMode qcli command and run the getRegDomainChannelList command.

Example:

```
setPhyMode a
getRegDomainChannelList
```

6.3.5.92 setAntDiv

Parameter Name	Type/Range	Description
Enable/Disable	0 – 1	Enable (1) or disable (0).
TxFollowRx	0 -1	0: Ant diversity apply to Rx alone 1: Tx Ant follow Rx
AntDivMode	Number	0 : Auto Mode – Switch antenna whenever main antenna rssi is low compared to alt antenna rssi 1: packet Mode – Ant switch based on packet number strictly 2: interval Mode – Ant switch based on time interval strictly.
Interval/packet count	Number	This param is Optional if Automode, otherwise it takes packet count/ interval(in millisecond) based on AntDivMode

With this command, the user can enable or disable antenna diversity, set whether Tx antenna follows Rx antenna or not, set the antenna diversity mode and corresponding parameter.

If the mode is auto mode (0), the parameter is skipped;

If the mode is based on packet number strictly (1), the parameter is the number of packets which are used to calculate average RSSI and decide to antenna switch;

If the mode is based on time interval strictly (2), the parameter is the time interval in ms used to calculate average RSSI and antenna selection.

Example:

- Enable antenna diversit. Only Rx antenna configure to auto mode ant diversity.
  Wlan SetAntDiv 1 0 0
- Enable antenna diversity. Tx antenna will follow the Rx antenna and it will change based on Rx decision, ant diversity based on packet number strictly i.e 512 count Wlan SetAntDiv 1 1 1 512
- Enable antenna diversity, Tx antenna will follow the Rx antenna and it will change based on Rx decision ant diverisy based on time interval strictly, i.e Wlan SetAntDiv 1 1 2 60000
- Disable antenna diversity
  Wlan SetAntDiv 0

6.3.5.93 getAntDivStatus

With this command, the user can get the status of antenna diversity, such as the current Rx physical antenna and the current Tx physical antenna on both 2G or 5G, total packet count to calculate RSSI, the current average main RSSI, the current average alt RSSI and the count of antenna switching and so on.

6.3.5.94 SetAnt

With this command, the user can set the physical antenna number, it means that the user can switch to the specific antenna by this command.

Example: switch to antenna number 2

SetAnt 2

6.3.5.95 P2P

This subsection includes all commands related to peer-to-peer functionality.

6.3.5.95.1 On

This command enables P2P mode of operation and has no parameter. This command is a prerequisite to perform any other P2P operation. This command is executed only in device 0.

6.3.5.95.2 Off

This command disables P2P mode of operation and requires no parameter. P2P functionality ceases to work after this command. This command is executed only in device 0.

6.3.5.95.3 SetConfig

This command sets the parameters that are not covered by the set command, such as go intent, listen channel, operating channel, and so on.

Parameter Name	Type/Range	Description
GO_intent	1–15	This GO intent value is used in P2P GO Negotiation Request/Response frames. Higher the value, higher the intention to operate as a group owner.
listen_channel	1,6,11	The channel to listen during P2P find operation. The device remains on this channel during LISTEN phase of the *find* operation for a random duration. During this phase, device responds to probe requests from peer P2P devices.
operating_channel	The channel supported by the device	The preferred channel to operate the P2P group by indicating the same in GO negotiation action frames.
country	String	No longer used. Country code needs to be set by board data file or tunables.
node_timeout	0 – 0xFFFFFFFF	Timeout value (in seconds) for each node. After the timeout period expires, the nodes are deleted from P2P find results.

6.3.5.95.4 Connect

This command initiates connection request with a given peer MAC address using given WPS configuration method. At the end of this command, P2P connection should get established successfully provided the credentials are matching. This command requires a prerequisite of peer being already found during P2P find operation.

Parameter Name	Type/Range	Description
peer_dev_mac	any valid mac address	Device MAC address of the P2P peer to connect with. This MAC should address should be available in P2P find results at the time of issuing connect command
wps_method	String (push/display/keypad)	WPS configuration method for authentication.
WPS pin	any valid pin	(Optional) This parameter is set only if wps_method is keypad or display. The pin number is used for WPS keypad or display authentication.
persistent	Persistent	(Optional) This parameter indicates whether the connection with given peer_mac address is a persistent connection. Persistent connections are stored in a persistent media and can be reinvoked later using P2P invitation request. The persistent connections are maintained across the reboots of QCA4020.

6.3.5.95.5 Find

This command initiates P2P find operation. The *find* operation includes a sequence of SEARCH/LISTEN phases across the channels to look for available P2P peers.

Parameter Name	Type/Range	Description
channel_options	1,2,3	1: Scan all the channels from regulatory domain channel list. 2: Scan only the social channels (Channel 1, 6 and 11 in 2.4GHz band). This is the default option for the *find* command. 3: Continue channel scan from the last scanned channel index. Channel list is obtained from regulatory domain list.
timeoutInSecs	Integer	When the timeout period expires, the find operation is stopped. Default value is 60 seconds.

6.3.5.95.6 Provision

This command provisions the WPS configuration method between the DUT and the peer.

Parameter Name	Type/Range	Description
peer_dev_mac	any valid mac address	Device MAC address of the P2P peer to be provisioned. Peer which is provisioned should be found in the P2P search results.
wps_method	String (push/display/keypad)	Preferred WPS configuration method for authentication

6.3.5.95.7 Listen

This command initiates P2P listen process. This command maintains the P2P device in listen state which enables the peer to find QCA4020 device easily.

Parameter Name	Type/Range	Description
timeout	Integer	In units of second. When the timeout period expires, the listen operation is stopped. Default value is 300 seconds.

6.3.5.95.8 Cancel

This command cancels all ongoing P2P operation. The command requires no parameter.

6.3.5.95.9 Join

This command joins a P2P client to an existing P2P Group Owner. This command requires a prerequisite of GO to which the device requires to connect to is found as part of P2P find process. Care should be taken to provide GO's interface MAC address and not the P2P GO's device MAC address when joining an existing GO.

Parameter Name	Type/Range	Description
GO_intf_mac	any valid mac address	GO interface MAC address of the P2P peer to connect to.
wps_method	String (push/display/keypad)	WPS configuration method for authentication
WPS pin	any valid pin	(Optional) This parameter is set only if wps_method is keypad or display. The pin number is used for WPS keypad or display authentication.
persistent	Persistent	(Optional) This parameter indicates whether the connection with given peer_mac address is a persistent connection. Persistent connections are remembered in persistent media and can be reinvoked using P2P invitation request.

6.3.5.95.10 Auth

This command authenticates/rejects a connection request from a given peer MAC address using the given WPS configuration method. This command requires a prerequisite of given peer P2P device MAC address available in P2P find results.

Parameter Name	Type/Range	Description
peer_dev_mac	any valid mac address	Device MAC address of the P2P peer to accept or reject
wps_method	push, keypad, display	WPS configuration method for authentication
WPS pin	any valid pin	(Optional) This parameter is set only if wps_method is keypad or display. The pin number is used for WPS keypad or display authentication.
persistent	Persistent	(Optional) This parameter indicates whether the connection with given peer_mac address is a persistent connection. Persistent connections are remembered in persistent media and can be reinvoked using P2P invitation request.

6.3.5.95.11 AutoGO

This command starts P2P device in Autonomous Group Owner mode. This command can be used to start P2P device as a group owner without the need for the Group Owner negotiation procedure. This command has a prerequisite of configuring Auto GO SSID and passphrase using "SetPassphrase" command under P2P menu.

Parameter Name	Type/Range	Description
persistent	persistent	(Optional) This parameter indicates whether the start of autogo is a persistent connection. Persistent connections are remembered in persistent media and can be reinvoked using P2P invitation request.

6.3.5.95.12 Invite

This command invites a connection from persistent database. Reinvoking a persistent connection enables quicker association because the passphrase is available at database, saving the time taken for 8-way WPS handshake. Inviting a peer P2P device requires the peer P2P device to be available as part of P2P find results and the persistent network list should contain the group information which is being re-invoked. This requires the user to run "find" command and "ListNetworks" command prior to execution of invite command.

Parameter Name	Type/Range	Description
SSID	String	SSID of the persistent connection to be reinvoked
peer_dev_mac	any valid mac address	P2P device address of the peer that should be re-invoked
wps_method	push, keypad, display	WPS configuration method for authentication
persistent	persistent	(Optional) This parameter indicates whether the connection with given peer_mac address is a persistent connection. Persistent connections are remembered in persistent media and can be re-invoked using P2P invitation request.

6.3.5.95.13 ListNodes

This command displays the results of P2P find operation. This command is used after *find* and requires no parameter.

6.3.5.95.14 ListNetworks

Display the list of persistent P2P connections that are saved in persistent media. This command requires no parameter.

6.3.5.95.15 SetOPPSParams

This command enables and sets the parameters of Opportunistic Powersave when operating as GO. For more information on Opportunistic power save, see Wi-Fi P2P specification

Parameter Name	Type/Range	Description
ctwin	Integer	CTWindow parameter in OppPS for GO, in units of TU (1024 microseconds).
enable	1/0	Enable or disable OppPS feature for GO.

6.3.5.95.16 SetNOAParams

This command enables and sets the parameters of Notice of Absence when operating as GO. For more information on Notice of absence, see Wi-Fi P2P specification.

Parameter Name	Type/Range	Description
count	1–255	Indicate the number of absence interval for GO. 255 indicates continuous NoA without termination.
start_offset_in_usec	Integer	Units of millisecond. Indicate the offset time after the next TBTT, to calculate the absolute start time of GO's first absence period.
duration_in_usec	Integer	Indicates the maximum duration in units of milliseconds that the P2P GO can remain absent in one NoA interval.
interval_in_usec	Integer	Indicates the length of the Notice of Absence interval in units of milliseconds.

6.3.5.95.17 SetOpClass

This command sets Operating class parameters for the P2P device.

Parameter Name	Type/Range	Description
GO_intent	1–15	This GO intent value is used in P2P GO Negotiation Request/Response frames.
oper_reg_class	Integer	Regulatory class for operating channel.
oper_reg_channel	The channel the device supported	Operating channel.

6.3.5.95.18 Set

This command sets various P2P parameters.

Parameter Name	Type/Range	Description
p2pmode	p2pdev, p2pclient, p2pgo	Set P2P operating mode as p2p device, p2p client or p2p GO.
postfix	string	The string to be added to the SSID suffix when operating as P2P GO. P2P GO SSID starts with the string 'DIRECT-'. The string entered in this parameter is appended to 'DIRECT-'. Max length of this string is 22.
intrabss	0/1	Disable or enable intra-BSS transition distribution in capability field of P2P IE.
gointent	1-15	Integer value to be set as GO intent. This GO intent value is used in P2P GO Negotiation Request/Response frames.
cckrates	0/1	Disable or enable CCK rates.

6.3.5.95.19 StopFind

This command stops P2P find operation and requires no parameter.

6.3.5.95.20 SetPassphrase

This command sets the passphrase and SSID for the GO network. This is primarily used to set the parameters when P2P device operates as Auto-GO mode. For a P2P device which operates as GO, the SSID should have a prefix of "DIRECT-" followed by more characters.

Parameter Name	Type/Range	Description
passphrase	String	The passphrase to use.
SSID	String	SSID to use.

6.3.5.95.21 SetWpaCertParameters

This command is used to configure WPA-Enterprise security parameters.

Method: TLS/TTLS-MSCHAPV2/PEAP-MSCHAPV2

Parameter Name	Type/Range	Description
method	String	EAP method
id	String	Default identification
username	String	Account username, mandatory for TTLS-MSCHAPV2/PEAP-MSCHAPV2
Password	String	Account password, mandatory for TTLS-MSCHAPV2/PEAP-MSCHAPV2
Debug_level	Integer/0−5	Threshold for debug message
Ca_path	String	Path of certificate for centralized authenticator, mandatory for TLS
Cert_path	String	Path of certificate, mandatory for TLS
Key_path	String	Path of private key, mandatory for TLS if private key is separate from certificate
Key_password	String	Password of private key, mandatory for TLS if private key is encrypted and separate from certificate

Example: `wlan SetWpaCertParameters PEAP-MSCHAPV2 id user password`

6.3.5.96 Usage examples

The following sections demonstrate how to use the commands in WLAN demo.

6.3.5.96.1 Initialization

Enable and disable Wi-Fi module.

```
> WLAN
WLAN> Enable
WLAN> Disable
```

6.3.5.96.2 Connection procedure

The following sections demonstrate scan, connection, and disconnection procedure for soft-AP and station modes.

Station mode

```
WLAN> SetDevice 1
WLAN> Scan
WLAN: ssid = Test_AP
WLAN: bssid = 00:03:7f:59:09:56
WLAN: channel = 1
WLAN: indicator = 46
```

```
WLAN: security = NONE!
WLAN> Connect Test_AP
WLAN: Setting SSID to Test_AP
WLAN: devid - 1 1 CONNECTED MAC addr 00:03:7f:59:09:56
WLAN> Disconnect
WLAN: devId 1 Disconnected MAC addr 00:03:7f:59:09:56
```

Soft-AP mode

```
WLAN> SetDevice 0
WLAN> SetOperatingMode ap
WLAN> Connect Test_Soft_AP
WLAN: setting to ap mode
WLAN: AP in OPEN mode!
WLAN: Setting SSID to Test_Soft_AP
WLAN: devid - 0 1 CONNECTED MAC addr 00:03:7f:56:04:12
```

Power Modes

Power mode is per decide, use SetDevice command before using SetPowerMode to change the device in use.

```
WLAN> SetPowerMode 0
WLAN> SetPowerMode 1
```

Suspend modes

The WLAN device can be suspended for a certain period of time if no traffic is expected. Example:To suspend the WLAN device for 200ms, the following command should be used.

```
WLAN> EnableSuspend
WLAN> Suspend 200
```

6.3.6 Net

The networking demo demonstrates how the networking QAPIs are used to perform operations including creation of a TCP/SSL server or client, send and receive data, start, configure or stop a network service.

6.3.6.1 Ping

This command sends a Ping request to a host.

Parameter Name	Type/Range	Description
host	IPv4 address or host name	Remote host
-c	1 – 2^31	Amount of ping requests
-s	1 – 1536	Ping payload size in bytes

```
ping <host> [-c <count>] [-s <size>]
```

6.3.6.2 dhcpv4c

This command acquires a valid IPv4 address from a DHCPv4 server.

Parameter Name	Type/Range	Description
interface name	wlan0	Interface name
new	–	New DHCP request (get an IPv4 address)
release	–	Release an IPv4 address

```
dhcpv4c <interface name> [new|release]
```

6.3.6.3 autoipv4

This command generates a Link-local IPv4 address where a DHCPv4 server is not available

Parameter Name	Type/Range	Description
interface name	wlan0	Interface name

```
autoipv4 <interface name>
```

6.3.6.4 ifconfig

This command displays or configures network interface.

Parameter Name	Type/Range	Description
interface name	wlan0	Interface name (optional – shows all if not specified)
ipv4addr	x.x.x.x	IPv4 address
subnetmask	x.x.x.x	Subnet mask
default gateway	x.x.x.x	Default gateway (optional)

Note, this command allows to configure IPv4 addresses only. IPv6 addresses are generated automatically.

```
ifconfig [<interface name>]
```

6.3.6.5 ping6

This command sends a Ping (IPv6) request to a host.

Parameter Name	Type/Range	Description
host	IPv6 address or host name	Remote host
-c	1 – 2^31	Amount of ping requests
-s	1 – 1536	Ping payload size
-I	wlan0	Interface name must be specified if address is link-local

```
ping6 <host> [-c <count>] [-s <size>] [-I <interface name>]
```

6.3.6.6 dhcpv6c

This command acquires an IPv6 address from DHCPv6 server

Parameter Name	Type/Range	Description
interface name	wlan0	Interface name
enable	–	Enable DHCPv6 service on an interface

Parameter Name	Type/Range	Description
new	–	Acquire a global IPv6 address from server
confirm	–	Confirm current IPv6 address
disable	–	Disable DHCPv6 service
release	–	Release acquired IPv6 address

```
dhcpv6c <interface name> enable|new|confirm|disable|release
```

6.3.6.7 prefix

This command sets an IPv6 router prefix and effectively enable Router Advertisement.

Parameter Name	Type/Range	Description
interface name	wlan0	Interface name
ipv6addr	IPv6 address	Global IPv6 address of the interface
prefixlen	1–128	Prefix length in bits
prefix_lifetime	1–2^31	Prefix life time in seconds
valid_lifetime	1–2^31	Prefix valid life time in seconds

```
prefix <interface name> <ipv6addr> <prefixlen> <prefix_lifetime>
<valid_lifetime>
```

6.3.6.8 sntpc

This command enables time of day acquisition from SNTP server.

Parameter Name	Type/Range	Description
start	–	Start SNTP client service
stop	–	Stop SNTP client service
disable	–	Disable SNTP client service
addsrv	IPv4/IPv6 address	Add a server to the server list
delsrv	ID	Delete a server from the list
utc	–	Show UTC time

```
sntpc [start|stop|disable]
sntpc [addsvr <server> [<id>]]
sntpc [delsvr <id>]
sntpc [utc]
```

6.3.6.9 dnsc

This command is Dynamic name resolution service

Parameter Name	Type/Range	Description
start	–	Start DNS client service
stop	–	Stop DNS client service
disable	–	Disable DNS client service
addsrv	IPv4/IPv6 address	Add a server to the server list
delsrv	ID	Delete a server from the list

Parameter Name	Type/Range	Description
resolve	Host name v4/v6	Resolve a host name to an IPv4 or IPv6 address
gethostbyname	Host name	Resolve a host name to an IPv4 address
gethostbyname2	Host name v4/v6	Resolve a host name to an IPv4 or IPv6 address

```
dnsc [start|stop|disable]
dnsc addsvr <server> [<id>]
dnsc delsvr <id>
dnsc gethostbyname <host>
dnsc [resolve|gethostbyname2]<host> [v4|v6]
```

6.3.6.10 bridge

This command performs WLAN bridge operation.

Parameter Name	Type/Range	Description
enable	–	Enable WLAN bridge: Bridge traffic between wlan0 and wlan1 interfaces in Layer 2
disable	–	Disable WLAN bridge
mac_entry_age	1–3600	Configure aging of bridged addresses in seconds
showmacs	–	Display all bridged addresses

```
bridge [enable|disable]
bridge [mac_entry_age <val>]
bridge [showmacs]
```

6.3.6.11 profile

This command configures the network buffer queue profile.

Parameter Name	Type/Range	Description
set	1,2,3	Set active profile to: 1=performance, 2=best effort, 3=memory optimized
custom	see description	Set a custom profile <pool_size> : Total free queues (up to 3) <buf_size> <num_buf> : 1st queue tuple <buf_size> <num_buf> : 2nd queue tuple <buf_size> <num_buf> : 3rd queue tuple

```
profile custom <pool_size> <buf_size> <num_buf> <buf_size> <num_buf>
profile set [1=performance, 2=best effort, 3=memory optimized]
```

6.3.6.12 cert

This command performs certificate management operations.

Parameter Name	Type/Range	Description
Name		

```
15: cert <store|delete|list|get>
```

Type command name to get more info on usage. Example: "cert store".

6.3.6.13 SSL

Secure socket Layer (SSL) is a standard security technology for establishing encrypted link between a server and a client, typically a webserver (website) and a browser or a mail server and a mail client. After the client and server decide to use TLS, they negotiate a stateful connection by using a handshaking procedure. During this handshake, the client and server agree on various parameters used to establish the connection's security. The following conditions are met:

- The client sends the server the SSL version number, cipher settings, session-specific data, and other information of the client that the server needs to communicate with the client using SSL.
- The server sends the client the SSL version number, cipher settings, session-specific data, and other information of the server that the client needs to communicate with the server over SSL. The server also sends its own certificate. If the client is requesting a server resource that requires client authentication, the server requests the client's certificate.
- The client uses the information sent by the server to authenticate the server. Example: In the case of a web browser connecting to a web server, the browser checks whether the received certificate's subject name actually matches the name of the server being contacted, whether the issuer of the certificate is a trusted certificate authority, whether the certificate has expired, and, ideally, whether the certificate has been revoked. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client proceeds to the next step.
- Using all data generated in the handshake, the client (with the cooperation of the server, depending on the cipher in use) creates the pre-master secret for the session, encrypts it with the server's public key (obtained from the server's certificate, sent in step 2), and then sends the encrypted pre-master secret to the server.
- If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case, the client sends both the signed data and the client's own certificate to the server along with the encrypted pre-master secret.
- If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session ends. If the client can be successfully authenticated, the server uses its private key to decrypt the pre-master secret, and then performs a series of steps (which the client also performs, starting from the same pre-master secret) to generate the master secret.
- Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity (detect any changes in the data between the time it was sent and the time it is received over the SSL connection).
- The client sends a message to the server informing that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.
- The server sends a message to the client informing that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.

- The SSL handshake is now complete, and the session begins. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

The QCA402x SDK provides a rich library of cryptographic operations which are optimized for size, power, reliability and security, and are accelerated by dedicated hardware:

- Sign and Verify RSA/ECC/ECDSA/ED25519
- AES encryption and decryption: CBC/CTR/CCM/GCM/CHACHA20
- Digest and hashing operations: SHA1/SHA224/SHA256/SHA384/SHA512/MD5
- HMAC operations: MD5/SHA/Poly1305
- Key derivation: SRP/CURVE25519/ECDH/EC-JPAKE

Operations on private keys and certificates are done in TEE-Lite (trusted execution environment) which is not accessible to the application processor.

This command performs SSL operations

Parameter Name	Type/Range	Description
start	See description	<server | client>: Configure ssl instance as a client or server
stop	See description	<server | client>: Stop the ssl session and reset the client or server instance.
config	See description	<server | client>: Configure the ssl client or server instance. With the following parameters. protocol <protocol> : select protocol: TLS (default), TLS1.0, TLS1.1, TLS1.2, DTLS, DTLS1.0, DTLS1.2 time 0|1 : disable|enable certificate time validation (optional) domain 0|<name>: disable|enable validation of peer's domain name against <name> alert 0|1 : disable|enable sending of SSL alert if certificate validation fails. cipher <cipher>: select <cipher> (enum name) suite to use, can be repeated 8 times (optional) max_frag_len <bytes>: max fragment length in bytes neg_disable 0|1: disable|enable maximum fragment length negotiation sni <name>: configure name for server name indication (SNI) alpn <protocol_name: specify protocol name for ALPN, for example, "h2", "http/1.1"
cert	see description	<server|client>: add a certificate or CA list to either SSL server or client. <certificate|calist>: Certificate or CA list. <name>: Name of the file to load from secure storage.
psk	see description	<server|client>: This creates a psk table for either the SSL server or client. <identity>: Identity <psk>: psk
ecjpake	see description	<server|client>: Sets the password for ECJPAKE cipher suite for either client or server. <password>: Password to be set.

Parameter Name	Type/Range	Description
Max_clients	See description	<max_Clients>: Sets DTLS maximum number of clients per server connection. Defaults to 1 if not set.
Idle_timer	See description	<idle_timeout>: For DTLS servers, sets the number of seconds to wait before closing an idle client connection

```
16: ssl [start|stop|config|cert|psk|ecjpake|max_clients|idle_timer]
<argument>..
```

Type command name to get more info on usage. Example: "ssl start".

6.3.6.14 Benchrx

Parameter Name	Type/Range	Description
benchrx	See description	{tcp|tcpzc|ssl|sslzc}: Use TCP transport or ssl. Zc indicates zerocopy. <port>: Local port to listen on. [<local_ip>]: Local IP address to bind to.
Benchrx	See description	{udp|udpzc}: Use UDP transport. <port>: Local port to listen on. [<local_ip>|<multicast_ip>|<multicast_ip%if_name>|<multicast_ip> <local_ip>] : The local IP can be local interface address or a multicast IP address. If multicast address is specified, the interface name should be specified with %.
Benchrx	See description	{raw|rawh}: Use raw socket. <protocol> : Protocol id. [<local_ip>|<multicast_ip>|<multicast_ip%if_name>|<multicast_ip> <local_ip>]: The local IP can be local interface address or a multicast IP address. If multicast address is specified, the interface name should be specified with %.

```
17: benchrx  Enter 'benchrx' for help
    Run IPv4 receive traffic test
Net: Examples:
Net:  benchrx tcp 2390
Net:  benchrx tcpzc 5000 192.168.1.100
Net:  benchrx udp 6000 224.1.1.100%wlan1
Net:  benchrx udp 7000 224.1.1.100 192.168.1.100
Net:  benchrx rawh 26
```

6.3.6.15 Benchtx

Parameter Name	Type/Range	Description
benchtx	See description	<Rx IP>: Receive IP <port>: Server port {tcp|tcpzc|udp|udpzc|ssl}: Use tcp transport or UDP transport or ssl. Zc indicates if zerocopy should be used. <msg size>: Payload size. <mode>: can be 0 or 1. If <mode> is 0, <arg> is time to Tx in seconds. If <mode> is 1, <arg> is number of msgs to Tx. <arg>: Number of seconds if mode is 0 or number of packets if mode is 1. <delay in ms between msgs>: Delay between packets. [<tos>]: Type of Service. <source IP>: IP address of interface.
benchtx	See description	<Rx IP>: Receive IP <protocol>: Protocol ID raw: Indicates RAW packet. <msg size>: Payload size. <mode>: can be 0 or 1. If <mode> is 0, <arg> is time to Tx in seconds. If <mode> is 1, <arg> is number of msgs to Tx. <arg>: Number of seconds if mode is 0 or number of packets if mode is 1. <delay in ms between msgs>: Delay between packets. [<tos>]: Type of Service.
Benchtx	See description	<Rx IP>: Receive IP <protocol>: Protocol ID rawh: Indicates RAW packet with IP header included. <msg size>: Payload size. <mode>: can be 0 or 1. If <mode> is 0, <arg> is time to Tx in seconds. If <mode> is 1, <arg> is number of msgs to Tx. <arg>: Number of seconds if mode is 0 or number of packets if mode is 1. <delay in ms between msgs>: Delay between packets. [<tos>]: Type of Service. <source IP>: IP address of interface.

```
18: benchtx  Enter 'benchtx' for help
    Run IPv4 transmit traffic test
Net: Examples:
Net:  benchtx 192.168.1.20 2390 udp 1400 1 100 0 0xA0
Net:  benchtx 192.168.1.20 26 raw 1400 0 60 0
Net:  benchtx 192.168.1.20 26 rawh 1400 1 100 10 0xA0 192.168.1.100
```

6.3.6.16 benchquit

This command terminates bench test.

Parameter Name	Type/Range	Description
Rx	–	Terminate Server test
Tx	–	Terminate Client test

```
benchquit [rx|tx]
```

6.3.6.17 uapsdtest

This command runs the transmit traffic test to send packets with specified AC.

Parameter Name	Type/Range	Description
Name		

```
20: uapsdtest {Remote IP} <port> <the number of packets> <time interval>
<access category>
```

6.3.6.18 dhcpv4s

This command sets up DHCPv4 address pool.

Parameter Name	Type/Range	Description
Name		

```
dhcpv4s <interface name> pool <startip> <endip> [<lease_time_sec>|infinite]
```

6.3.6.19 nsstrrcl

Network stack store-recall unit test.

Parameter Name	Type/Range	Description
exit	–	Restore network stack state (Unit test only)

6.3.6.20 dnss

Perform DNS server operations.

Parameter Name	Type/Range	Description
Name		

```
dnss [start|stop|disable]
  dnss addhost <hostname> <host IP address> [<time-to-live>]
  dnss delhost <hostname>
```

6.3.6.21 benchrx6

Parameter Name	Type/Range	Description
Name		

```
24: benchrx6  Enter 'benchrx6' for help
    Run IPv6 receive traffic test
Net: benchrx6 {udp|udpzc|tcp|tcpzc|ssl|sslzc} <port>
Net: benchrx6 {udp|udpzc} <port> [<multicast_ip%if_name>]
Net: Examples:
Net:  benchrx6 tcp 2390
Net:  benchrx6 udp 2390 ff02::1:2%wlan1
```

6.3.6.22 benchtx6

Parameter Name	Type/Range	Description
benchtx6	See description	<Rx IP>: Server IPv6 address. <port>: Server port {tcp|tcpzc|udp|udpzc|ssl}: Use tcp transport or UDP transport or ssl. Zc indicates if zerocopy should be used. <msg size>: Payload size. <mode>: can be 0 or 1. If <mode> is 0, <arg> is time to Tx in seconds. If <mode> is 1, <arg> is number of msgs to Tx. <arg>: Number of seconds if mode is 0 or number of packets if mode is 1. <delay in ms between msgs>: Delay between packets. [<tos>]: Type of Service.

```
25: benchtx6  Enter 'benchtx6' for help
    Run IPv6 transmit traffic test
Net: Examples:
Net:  benchtx6 fe80::865d:d7ff:fe40:3498%wlan1 2390 udp 1400 1 100 0 0xA0
Net:  benchtx6 2001:db8:85a3:8d3:1319:8a2e:370:734 2390 tcpzc 512 0 30 5
```

6.3.6.23 httpsvr

Parameter Name	Type/Range	Description
Name		

```
httpsvr [init {v4|v6|v46} {http|https|http_https} [-c <cert_file>] [-p
<httpport>] [-s <httpsport>] [-i <ifname>] [-x <index_page>] [-r
<root_path>]]
httpsvr [start|stop]
httpsvr setbufsize <TX buffer size in bytes> <RX buffer size in bytes>
httpsvr {addctype|delctype} <content-type1> [<content-type2> ..]
httpsvr sslconfig [<keyword_1> <value_1> <keyword_1> <value_1> …]
Examples:
 httpsvr init v6 http
 httpsvr init v4 http_https -c cert.bin
 httpsvr init v46 https -c cert.bin -p 8080 -s 1443 -i wlan1 -x index.html
-r /mywebfolder
```

6.3.6.24 mdnss

This command is used to Set up and configure Multicast Domain Name System server.

Parameter Name	Type/Range	Description
start	See description	Start mDNS service. <ipv4 | ipv6 |ipboth> : IP version can be ipv4 or ipv6 or both. <interface>: Name of the interface <blocking>: Blocking or non-blocking mode.
stop	See description	Stop mDNS service.
sethostname	See description	<Hostname>: Hostname to be used for mDNS records.

Parameter Name	Type/Range	Description
addsvc	See description	<instance_name>: Name of the service to be registered. Should be of the form <instance._servicename.local>. Example: instance._appletv._tcp.local. <port>: Port on which the service is listening. <txt>: TXT related to the service. strings should be of the form “key=value” and a maximum of 10 key value pairs can be specified.
rmsvc	See description	<instance_name>: Name of the service instance to be unregistered or removed.
updatetxt	See description	<service_name>: Name of the service instance whose TXT record should be updated. <txt>: TXT record to be updated or added.

```
27: mdnss [start <ipv4|ipv6|ipboth>|stop] ifname hostname
mdnss [addsvc <instance_name> <host port> <txt>]
mdnss [rmsvc <instance_name>
```

6.3.6.25 httpc

Parameter Name	Type/Range	Description
Name		

```
httpc start
httpc stop
httpc connect <server> <port> <timeout>
httpc disconnect <client_num>
httpc get <client_num> <url>
httpc put <client_num> <url>
httpc post <client_num> <url>
httpc patch <client_num> <url>
httpc setbody <client_num> <len>
httpc addheaderfield <client_num> <hdr_name> <hdr_value>
httpc clearheader <client_num>
httpc setparam <client_num> <key> <value>
httpc config <httpc_demo_max_body_len> <httpc_demo_max_header_len>
```

6.3.6.26 dnssd

This command is used to Setup and configure Domain Name System Service Discovery

Parameter Name	Type/Range	Description
start	See description	Start DNS-SD service. <ipv4 | ipv6 |ipboth> : IP version can be ipv4 or ipv6 or both. <interface>: Name of the interface
stop	See description	Stop DNS-SD service.

Parameter Name	Type/Range	Description
init	See description	<timeout>: Timeout in milliseconds, denotes the maximum time allowed for a discovery request.
discover	See description	<instance_name>: Name of the service to be discovered.
gettargetinfo	See description	<instance_name>: Name of the service instance for which the server info(ip, port and so on.) should be requested.

```
29: dnssd Enter 'dnssd' for help
    Run dns service discovery
Net: dnssd start <ipv4|ipv6|ipboth> ifname
Net: dnssd stop
Net: dnssd init <timeout in milliseconds>
Net: dnssd discover <instance name>
Net: dnssd gettargetinfo <instance name>
```

6.3.6.27 free

This command shows status of queue of network buffers in the system.

6.3.6.28 mqttc

This command sets up and configures MQ Telemetry Transport client.

```
mqttc init [<calistfile>]
mqttc shutdown
mqttc new [<clientid_string>] [<clean_session_flag>]
mqttc destroy <session_id>
mqttc config <session_id> [-u <user> -P <pw>] [-t <will topic> -m <will
msg> -q <will QOS> -r]
mqttc connect <session_id> <svr> [-s] [-n] [-k <keepalive_sec>] [-w
<connack_wait_sec>] [-i <bind_if>]
mqttc subscribe <session_id> -t <topic filter> [-q <requested QOS>]
mqttc publish <session_id> -t <topic> [-q <QOS level>] [-m <message>] [-r]
[-d]
mqttc unsubscribe <session_id> -t <topic filter>
mqttc disconnect <session_id>
mqttc sslconfig <session_id> protocol <protocol> [<keyword_1> <value_1>
...]
```

6.3.6.29 route6

This command shows or configures the IPv6 routing table.

```
route6 add [<dest> <next_hop> <interface>]
route6 del [<dest> <interface>]
route6 show
Examples:
  route6 add fda8:6c3:ce53:a890::/64 fda8:6c3:ce53:a890::3e wlan1
  route6 del fda8:6c3:ce53:a890::/64 wlan1
```

6.3.6.30 route

This command shows or configures the IPv4 routing table.

```
route add <ifname> <ipv4addr> <mask> <gateway>
route del <ifname> <ipv4addr> <mask> <gateway>
route show <ifname>
```

6.3.6.31 eth

This command sends or receives the raw frames.

```
eth Tx <ifname> <dest mac addr> [-d <data bytes>] [-p <etherType>]
eth Rx [-i <ifname>] [-p <etherType>] [-q]
Examples:
  eth Tx wlan1 00:11:22:33:44:55 -p 0x888e -d "01 00 00 05 01 ab 00 05 01"
  eth Rx -i wlan1 -p 0x1234
  eth Rx -q
```

6.3.6.32 tcpka

This command shows or configures the TCP keepalive parameters.

```
tcpka [-i idle_time (in seconds)] [-r resp_timeout (in seconds)]
```

6.3.6.33 queuecfg

This command configures size of socket buffer for Tx or Rx.

```
queuecfg [tx|rx] <size_in_bytes>
```

6.3.6.34 coap

This command configures CoAP, client and server, and displays the configuration.

CoAP client

```
coap-client  <udp/tcp> <index> [-b [num,]size] [-B seconds]
[-e text] [-m method] [-N]
[-s duration] [-v ipv4/ipv6] URI
index            0 means create connect
                 1-4 client index
URI              can be an absolute or relative coap URI,
-b [num, size]  block size to be used in GET/PUT/POST requests
                (value must be a multiple of 16 not larger than 1024)
                If num is present, the request chain will start at
                block num
-B seconds      break operation after waiting given seconds(default is 90)
-e text         include text as payload (use percent-encoding for
                non-ASCII characters)
-m method       request method (get|put|post|delete), default is 'get'
-N              send NON-confirmable message
-s duration     subscribe for given duration [s]
```

```
-T token        include specified token
```

Example

```
coap client udp 0 -m get coap://192.168.1.100/
coap client udp 0 -m get coap://[2001::1]/
coap client tcp 1 -m get /time?ticks
coap client sslconfig [<keyword_1> <value_1>] [<keyword_2> <value_2>] ...
Where <keyword> <value> are:
 protocol <protocol> = select protocol: TLS (default), TLS1.0, TLS1.1,
                       TLS1.2, DTLS, DTLS1.0, DTLS1.2
 time 0|1            = disable|enable certificate time validation (optional)
 domain 0|<name>     = disable|enable validation of peer's domain name
                       against <name>
 alert 0|1           = disable|enable sending of SSL alert if certificate
                       validation fails
 cipher <cipher>     = select <cipher> (enum name) suite to use, can be
                       repeated 8 times (optional)
 max_frag_len <bytes>= max fragment length in bytes
 neg_disable 0|1     = disable|enable maximum fragment length negotiation
 sni <name>          = configure name for server name indication (SNI)
 alpn <protocol_name>= specify protocol name for ALPN, e.g. "h2", "http/1.1"
```

CoAP server

```
 coap_server <udp/tcp/dual>   -v v4  -p 443
    -v v4/v6/v4v6   support ipv4/ipv6
    -c cert.bin     certificate
    -i interface    interface wlan0/wlan1
```

Example

```
coap server udp  -v v4 -c cert.bin -p 443
coap server tcp  -v v4v6
coap server sslconfig [<keyword_1> <value_1>] [<keyword_2> <value_2>] ...
 where <keyword> <value> are:
 protocol <protocol> = select protocol: TLS (default), TLS1.0, TLS1.1,
                       TLS1.2, DTLS, DTLS1.0, DTLS1.2
 time 0|1            = disable|enable certificate time validation (optional)
 domain 0|<name>     = disable|enable validation of peer's domain name
                       against <name>
 alert 0|1           = disable|enable sending of SSL alert if certificate
                       validation fails.
 cipher <cipher>     = select <cipher> (enum name) suite to use, can be
                       repeated 8 times (optional)
 max_frag_len <bytes>= max fragment length in bytes
 neg_disable 0|1     = disable|enable maximum fragment length negotiation
 sni <name>          = configure name for server name indication (SNI)
 alpn <protocol_name>= specify protocol name for ALPN, e.g. "h2", "http/1.1"
```

6.3.6.35 user

This command adds, delete or list user account.

```
user add <username> <password> [-s <service>]
user del <username> <password>
user passwd <username> <password> <new password>
user addsvc <username> <password> <service>
user delsvc <username> <password> <service>
    where 'service' is a bitmask. 0x1 = HTTP service
user list
Examples:
  user add admin admin -s 0x1
  user del admin admin
  user passwd admin admin my_new_password
  user addsvc admin admin 0x1
```

6.3.6.36 arp

This command shows the IPv4 ARP table.

```
arp show [<ifname>]
```

6.3.6.37 websocketc

This command is used to set up and configures the websocket client.

Parameter Name	Type/Range	Description
Name		

```
websocketc new
   -o  <origin>
   -k  <subprotocol> (can appear more than once)
   -s  <enable ssl = 1 or 0>
   -l  CA list name
   -c  certificate name
   -t  <handshake_timeout ms>
   -z  <closing_timeout ms>
websocketc destroy client_id
websocket sslconfig client_id
   protocol <version, e.g. TLS1.2>
   time <0 for no cert time check or 1>
   cipher <cipher>
websocketc addhttpheader client_id <header_name> <header_value>
websocketc resethttpheaders client_id
websocketc connect client_id
   -h  <host>
   -p  <port>
   -r  <resource_path>
websocketc close client_id
websocketc dump client_id <tx/rx>  <on/off>
websocketc echo client_id <on/off>
```

```
websocketc ping client_id
   -d  <data>
websocketc pong client_id
   -d  <data>
websocketc send client_id
   -m  <message>
   -e  <end_of_message = 1 or 0>

Examples:
Start websocket server on Linux Machine:
   /pywebsocket-master/mod_pywebsocket# python standalone.py -p 9001 -d
../example/

Start websocket client on Quartz Device:
   net httpc start
   net websocketc new -o http://example.com -t 60000 -r 1024
   net websocketc echo 1 on
   net websocketc dump 1 rx on
   net websocketc connect 1 -h 192.168.2.130 -p 9001 -r /echo
   net websocketc ping 1 -d testping
   net websocketc send 1-m testtesttesttesttesttesttesttest -e 1
   net websocketc close 1
   net websocketc destroy 1
```

6.3.7 Certificate management demo

The certificate management module is an application that enables downloading the certificate to the device over-the-air.

The demo includes a PC side server called certcs, which listens on a dedicated port and allows the device to connect and download the certificates. The certificates must be converted to the internal SSL binary format which is an IoT optimized version for speed and size.

The following test procedure is an example to demonstrate how to download either an RSA or an ECC certificate to the device. The reference of "foo" is a name example.

- Generate a device certificate using OpenSSL.
  - Option 1: Generate an RSA certificate.
    - i Generate a private key.

      ```
      openssl genrsa -des3 -out foo.key 1024
      ```

      CAUTION: Generate a self-signed certificate.

      ```
      openssl req -new -key foo.key -out foo.pem -x509 -days 365
      ```

      CAUTION: Remove the passphrase from the key.

      ```
      mv foo.key foo.key.withpass
      openssl rsa -in foo.key.withpass -out foo.key
      ```

- Option 2: Generate an ECC certificate.

  ii Generate a private key.

  ```
  openssl ecparam -name secp224r1 -genkey -noout -out foo.key
  ```

CAUTION: Generate a self-signed certificate.

```
openssl req -new -key foo.key -out foo.pem -x509 -days 365
```

CAUTION: Convert certificate to the internal SSL binary format.

CAUTION: Generate binary certificate file.

```
SharkSSLParseCert foo.pem foo.key -b foo.cert.bin
```

CAUTION: Generate binary CA List file.

```
SharkSSLParseCAList -b foo.calist.bin foo.pem
```

CAUTION: Start the certificate server on a Linux PC.

CAUTION: Copy the **foo.cert.bin** and **foo.calist.bin** files to the **certcs** directory which also contains the **certcs** executable.

CAUTION: For AWS IOT Connection, along with the device certificates download the AWS IOT root CA.

CAUTION: Start the certificate server.

```
./certcs -s
```

CAUTION: Test the certificate server from another PC.

```
./certcs HOSTNAME/foo.calist.bin
```

HOSTNAME is the host name of the PC where the certificate server is started.

CAUTION: Download the certificate and CA list from the server and write them to the flash.

CAUTION: Download and flash the certificate.

```
Net> cert get foo.cert.bin IPADDR –p PORT -s cert.bin –t cert
```

CAUTION: Download and flash the CA list.

```
Net> cert get foo.calist.bin IPADDR –p PORT –s ca.bin –t ca_list
```

IPPADDR is the IP address of the certificate server.

CAUTION: In case of AWS IOT Connection, along with the device certificates download the AWS IOT root CA. Ensure the stored CA list is always named as “aws_ca_list.bin”.

CAUTION: Net> cert get root_ca.bin IPADDR –s aws_ca_list.bin -t <filetype>

CAUTION: The filetype can be ca_list or pem_ca_list based on type of device certificate.

CAUTION: Use the certificate and the CA list stored in the flash.

The cert.bin and ca.bin will be subsequently used with the other SSL demo commands. The following commands are used to attach the certificates to an SSL context, or to delete them.

- Add certificate to the server SSL context
  ```
  Net> ssl cert server certificate cert.bin
  ```
- Add CA list to the client SSL context.
  ```
  Net> ssl cert client calist ca.bin
  ```
- Delete the files.
  ```
  Net> cert delete ca.bin –t ca_list
  Net> cert delete cert.bin –t cert
  ```

Test prerequisites

The following settings are required for running the test examples.

- Enable WLAN and connect the device to an AP.

CAUTION: Assign either a static IPv4 address or a dynamic IPv4 address using DHCP client, or use IPv6.

CAUTION: Start the SNMP client if it is required to also check certificate expiration.

Test example, SSL/TLS server

```
Net> ssl start server
Net> ssl cert server certificate cert.bin
Net> benchrx ssl 1443
```

Test example, SSL/TLS client

7. On the QCA402x side, run the following commands:
   ```
   Net> ssl start client
   Net> ssl cert client calist ca.bin
   Net> ssl config protocol TLS1.2 time 1
   ```

CAUTION: Type `ssl config` to see the complete configuration list.

```
Net> benchtx 192.168.1.100 1443 ssl 1350 1 100 0
```

- On the Linux PC side, run the following command:
  ```
  ncat -l --ssl --ssl-key foo.key --ssl-cert foo.pem –k 1443
  ```

6.3.8 Coex

The coexistence demo shows how to use the coexistence API configuration functions and parameters through the use of general and advanced configuration commands.

6.3.8.1 Enable

This command enables or disables coexistence. It calls the API function qapi_COEX_Enable().

Parameter Name	Type/Range	Description
Enable	0 – 1	Enable (1) or disable (0) coexistence.

6.3.8.2 Configure

This command configures coexistence. Configuration flags are a bitmask of the definitions of the form QAPI_COEX_CONFIG_FLAG_XXX.

Parameter Name	Type/Range	Description
Config Flags	0 – 0xFFFFFFFF	Configuration flags.
Priority 1	1 – 4	The priority 1 wireless type value.
Priority 2	1 – 4	The priority 2 wireless type value.
Priority 3	1 – 4	The priority 3 wireless type value.
Priority 4	1 – 4	The priority 4 wireless type value.

The priority values set the priority order of the different wireless types with "Priority 1" being the highest priority. Values for the priority types are as follows:

- 1 – Qualcomm® Bluetooth Low Energy
- 2 – 802.15.4
- 3 – External PTA
- 4 – WLAN

6.3.8.3 ConfigureAdvanced

This command configures coexistence.

Parameter Name	Type/Range	Description
Config Flags	0 – 0xFFFFFFFF	Configuration flags.
Concurrency Flags	0 – 0xFFFFFFFF	Concurrency flags.

Configuration flags are a bitmask of the definitions of the form `QAPI_COEX_CONFIG_FLAG_XXX`. Concurrency flags are a bitmask of the definitions of the form `QAPI_COEX_XX_XX_CONCUR_ENABLE`.

Individual priority and class values are optionally set *before* calling this command using the following commands:

- SetBLEPriority
- SetLPWPriority
- SetEXTPriority

In using the preceding commands to set the priority, bear in mind that no two priority *values* across wireless types may be set equal with **ConfigureAdvanced**. For instance, Bluetooth Low Energy Scan and Bluetooth Low Energy Advertise may have the same value, but Bluetooth Low Energy Data Request and External PTA Low Request may not have the same value.

6.3.8.4 SetBLEPriority

This command sets the priority and class values for Bluetooth Low Energy. The new values passed to this command are *not* set in coexistence until a successful **ConfigureAdvanced** command is made.

Parameter Name	Type/Range	Description
Adv Value	0 – 15	Advertise priority value (default: 4).
Adv Class	0 – 2	Advertise priority class (default: 0).
Scan Value	0 – 15	Scan priority value (default: 3).
Scan Class	0 – 2	Scan priority class (default: 0).
DataReq Value	0 – 15	Data request priority value (default: 12).
DataReq Class	0 – 2	Data request priority class (default: 2).
DataAct Value	0 – 15	Data active priority value (default: 14).
DataAct Class	0 – 2	Data active priority class (default: 2).

6.3.8.5 SetLPWPriority

This command sets the priority and class values for 802.15.4. The new values passed to this command are *not* set in coexistence until a successful **ConfigureAdvanced** command is made.

Parameter Name	Type/Range	Description
RxReq Value	0 – 15	Receive request priority value (default: 5).
RxReq Class	0 – 2	Receive request priority class (default: 1).
TxReq Value	0 – 15	Transmit request priority value (default: 6).
TxReq Class	0 – 2	Transmit request priority class (default: 1).
RxAct Value	0 – 15	Receive active priority value (default: 9).
RxAct Class	0 – 2	Receive active priority class (default: 2).
TxAct Value	0 – 15	Transmit active priority value (default: 10).
TxAct Class	0 – 2	Transmit active priority class (default: 2).
Ack Value	0 – 15	Acknowledgment priority value (default: 15).
Ack Class	0 – 2	Acknowledgment priority class (default: 3).

6.3.8.6 SetEXTPriority

This command sets the priority and class values for external PTA. The new values passed to this command are *not* set in coexistence until a successful **ConfigureAdvanced** command is made.

Parameter Name	Type/Range	Description
LowReq Value	0 – 15	Low request priority value (default: 7).
LowReq Class	0 – 2	Low request priority class (default: 1).
HighReq Value	0 – 15	High request priority value (default: 11).
HighReq Class	0 – 2	High request priority class (default: 3).

Parameter Name	Type/Range	Description
LowAct Value	0 – 15	Low active priority value (default: 8).
LowAct Class	0 – 2	Low active priority class (default: 1).
HighAct Value	0 – 15	High active priority value (default: 13).
HighAct Class	0 – 2	High active priority class (default: 3).

6.3.8.7 StatisticsEnable

This command enables or disables coexistence statistics counters.

Parameter Name	Type/Range	Description
Enable	0 – 1	Enable (1) or disable (0) coexistence statistics.

6.3.8.8 StatisticsGet

This command retrieves coexistence statistics counters. The counters may optionally be reset with this command.

Parameter Name	Type/Range	Description
Reset	0 – 1	Reset (1) or retain (0) coexistence statistics.

6.3.8.9 WLANIFEnable

This command sets the WLAN coexistence interface mode. The interface may be disabled or enabled.

Parameter Name	Type/Range	Description
Mode	0 – 1	Interface mode: 0 (disable) or 1 (enable).

6.3.8.10 EPTAGPIOEnable

This command sets the external PTA coexistence interface mode. The interface may be disabled, set as PTA slave (external WLAN connected), or set as PTA master (external Bluetooth connected).

Parameter Name	Type/Range	Description
Mode	0 – 2	Interface mode: 0 (disable), 1 (slave), or 2 (master).
GPIO Option	0 - 1	EPTA GPIO option: 0 (standard GPIO 5,6, 7), 1(Alternate GPIO 16, 17, 60)

6.3.8.11 EnableWlanCoex

This command enables a particular mode of coex. Specifying 3-wire Coex Mode configures and enables the internal coex between WLAN and the on-chip BLE/15.4. Specifying EPTA Coex Mode configures and enables the coex between WLAN and an external wireless device. Either or both modes may be enabled.

Parameter Name	Type/Range	Description
Enablement	0 –1	WLAN coex enablement: 0 (disable), 1 (enable)

Parameter Name	Type/Range	Description
Coex Mode	0 –2	0: Internal 3-wire mode 1: Internal PTA mode (reserved) 2: External (EPTA) mode
Priority Levels	0–1	0: Two levels of priority (reserved) 1: Four levels of priority
Profile Type	1–13	1: SCO/eSCO 2: A2DP 3: Inquiry/Page Scan 4-7: Reserved 8: BLE/15.4 9-10: Reserved 11: WLAN always yields the medium (never stomp) 12: Reserved 13: Mesh scan, provisioning & network setup (connection setup, key bindings, and so on.)
Antenna Config	1–3	1: Single, shared antenna 2: Dual antenna, low isolation 3: Dual antenna, high isolation

6.3.8.12 GetWlanCoexStats

This command retrieves and optionally resets coex statistics counters in the WLAN coex firmware.

Parameter Name	Type/Range	Description
Reset Firmware Statistics Counters	0 – 1	0: retrieve counters without resetting statistic counters 1: retrieve counters and reset statistic counters

GetWlanCoexStats return values:

Stats Field Name	Type/Range	Description
High Rate Packet Count	Integer	Count of high rate (typically > 36Mbps) WLAN packets received. Used to assess impact of narrowband coex on WLAN performance
First Beacon Miss Count	Integer	Count of first beacon misses. Used to assess if narrowband is causing WLAN to miss beacons and debug WLAN loss of connection with AP
Beacon Miss Count	Integer	Overall count of first beacon misses. Used to assess if narrowband is causing WLAN to miss beacons and debug WLAN loss of connection with AP.
Ps-poll Failure Count	Integer	Count of failure to issue ps-poll to AP for traffic-shaping purposes. Used to debug situations where traffic shaping is in use. Traffic shaping is primarily used when SCO/ESCO and A2DP profiles are active
NULL Frame Failure Count	Integer	Count of failure to issue NULL frame to AP for traffic-shaping purposes. Used to debug situations where traffic shaping is in use. Traffic shaping is primarily used when SCO/ESCO and A2DP profiles are active

Stats Field Name	Type/Range	Description
Stomp Count	Integer	Count of times WLAN denies access to the medium by asserting the coexistence WLAN_ACTIVE line while the BT_ACTIVE line is also asserted

6.3.9 Firmware upgrade

The firmware upgrade demo is intended to demonstrate how to upgrade firmware using Firmware Upgrade API functions and parameters. It is assumed that network interface is established before running this demo.

6.3.9.1 Fwd

This command displays active Firmware Descriptor (FWD) index number and displays all Firmware Descriptor at flash.

6.3.9.2 Del

This command erases the FWD. After this command, the image space is available for download new image.

Parameter Name	Type/Range	Description
Fwd_num	0 – 2	Indicate which FWD to be deleted.

6.3.9.3 Trial

This command accepts or rejects the trial image. After this command, system will reboot based on [reboot] flag.

Parameter Name	Type/Range	Description
accept	0 – 1	Indicates if accept or reject the trial image. 0: reject the trial image; 1: accept the trial image.
reboot	0 – 1	Indicates if system reboot after this command. 0 : no reboot need, 1: reboot

6.3.9.4 Img

This command displays first 16 bytes of image at current active partition.

Parameter Name	Type/Range	Description
Id	0 – 255	Indicates which image content to display. 0: all images to display, other: image to display.

6.3.9.5 ftp

This command connects to FTP server and does Firmware Upgrade.

Parameter Name	Type/Range	Description
If_name	String	Network device name. Example: wlan1
ftp_paramter	String	This parameter includes Firmware Upgrade FTP related info. The format is [user]:[pwd]@[[ipv4]|[|ipv6|]:[port]. It includes FTP User Name, password, FTP Server IP Address, and FTP Server Port Number Example: user:password@192.168.1.108:21
File_name	String	This parameter is the firmware upgrade config file. Example:ota.bin
Flags	Bitmask	Bitmask of flags for Firmware Upgrade FTP: Bit0: Reboot after operate is done Bit1: 0: Don't duplicate File System from Current to Trial Images 1: Duplicate File System from Current to Trial Images. Bit2: This bit is used when Bit1 is set to 1 and files exist at Current Images and Trial Images 0: Copy and overwrite the files from Current Images to Trial images 1: Skip copying files from Current images to Trial images

6.3.9.6 http

This command connects to HTTP server and performs the firmware upgrade.

Parameter Name	Type/Range	Description
If_name	String	Network device name. Example: wlan1
http_paramter	String	This parameter includes Firmware Upgrade HTTP info. The format is <timeout>:<server>:<port>/<url>. It includes HTTP connection timeout, HTTP Server Address, HTTP Server Port Number, and HTTP URL. Example: 5000:192.168.1.108:80/firmware-download
File_name	String	This parameter is the firmware upgrade config file. Example: ota.bin

6.3.9.7 Cancel

This command cancels the ongoing firmware upgrade operation.

6.3.9.8 Suspend

This command pauses the ongoing firmware upgrade operation.

6.3.9.9 Resume

This command resumes the paused firmware upgrade operation.

6.3.9.10 Zigbee

This command starts a Zigbee OTA upgrade on the provided endpoint. The Zigbee OTA server must have been previously discovered using the "DiscoverServer" command in the Zigbee OTA ZCL demo module.

Parameter Name	Type/Range	Description
Endpoint	Valid endpoint	Endpoint that contains the OTA server cluster to use for the OTA upgrade.

6.3.9.11 BLE

This command starts a Bluetooth low energy (BLE) OTA upgrade from the provided OTA server. A BLE connection to this server must already exist using the BLE demo module.

Parameter Name	Type/Range	Description
BD_ADDR	Bluetooth Address (0x000000000000)	Bluetooth address of the connected remote OTA server.

6.3.10 ADSS

The ADSS demo is intended to demonstrate adss I2S and PCM functionality and APIs.

6.3.10.1 I2S play on Wi-Fi

This command plays audio so that audio data is downloaded from FTP Server.

Parameter Name	Type/Range	Description
url	string	ftp server URL
Buff_size	number	I2S one block data size, default is 1024
Packet_count	number	Number of block buffers to be reserved, default is 9

6.3.10.2 I2S record on Wi-Fi

This command records audio data and send to TCP server for saving. This command use codec on MB.

Parameter Name	Type/Range	Description
Svr_ip	IP address	Audio server IP address
port	number	TCP Port number
Buff_size	number	I2S one block data size, default is 1024
Packet_count	number	Number of block buffers to be reserved, default is 4

6.3.10.3 I2S audio echo

This command receive audio from MIC and send to speaker. This command uses codec on MB.

Parameter Name	Type/Range	Description
Buff_size	number	I2S one block data size, default is 1024
Packet_count	number	Number of block buffers to be reserved, default is 4
freq	number	I2S sample rate, default is 32K

6.3.10.4 PCM audio echo

This command receives and sends audio data with PCM. The command needs external codec for audio conversion.

Parameter Name	Type/Range	Description
Buf_size	number	PCM one block data size, default is 1024
Pkt_count	number	Number of block buffers to be reserved, default is 6
Sample_freq	number	Sample rate, default is 2 (32KHz)
slots	string	Channel 0~15 list

Example:

Pcmsr “buf_size=1024” “pkt_count=6” “sample_freq=2” “slots=”0,1”

6.3.11 Platform

The platform subgroup demo is intended to demonstrate time, device information, platform reset, watchdog reset, system reset reason, ramdump upload, gpio configuration, heap memory unit test and heap memory status and API’s

6.3.11.1 time

This command is used to set and get system time.

Parameter Name		Type/Range	Description
get			To get the system time
set			To set the system time
	year	number	Year: 1980 through 2100
	month	number	Month of the year: 1 through 12
	day	number	Day of the month: 1 through 31
	hour	number	Hour of the day: [0 through 23]
	minute	number	Minute of hour: [0 through 59]
	second	number	Second of minute: [0 through 59]
	Day_of_week	number	Day of the week [0 through 6] (corresponding to Monday through Sunday)

6.3.11.2 info

This command is used to get device specific information.

Parameter Name	Type/Range	Description
get	string	Data_type <data_type> can be one of the following: make - make of the device model - model of the device version - version of the device oem - OEM ID of the device serial - serial number of the device security - security state of the device soc - SoC HW version of the device

6.3.11.3 free

This command is used to get heap status.

Parameter Name	Type/Range	Description
none		display the heap size and an approximation of free amount of heap bytes

6.3.11.4 reset

This command is used to reset the system

Parameter Name	Type/Range	Description
none		reset the system

6.3.11.5 wdrst

This command is used to trigger watchdog reset

Parameter Name	Type/Range	Description
none		Trigger watchdog reset

6.3.11.6 geterrinfo

This command is used to get error information

Parameter Name	Type/Range	Description
none		Retrieve the fatal error debug information which captures minimal error information comprising of the source line number and module name prefixed with the core name.

6.3.11.7 gpio

This command is used to display and configure SoC GPIOs

Parameter Name	Type/Range	Description
show		Show the configuration table for all gpio
func	number	0 – 15
strength	number	0: 1.6mA 1: 2.7mA 2: 4.0mA
dir	number	0: input 1: output
pull	number	0: no pull 1: pull down 1: pull up
outval	number	[0 – 1] Specifies the value to be writer to gpio in case of a output gpio setting.

6.3.11.8 utmalloc

This command runs a malloc unit test

Parameter Name	Type/Range	Description
max_size	number	Max memory fragment size

6.3.11.9 reset_reason

This command is used to show previous reset reason

Parameter Name	Type/Range	Description
none		provides the boot or previous reset reason: 0: Power on cold boot or hardware reset 1: Reset from the Watchdog 2: Software cold reset

6.3.11.10 ramdump

This command is used to upload ramdump stored in flash to specific server.

See section RAM dump collection and debugging in the *QCA402x (CDB2x) Programmers Guide* (80-YA121-142).

Parameter Name	Type/Range	Description
Server_type	String	Protocol for upload. Only FTP is supported
Ip_version	String	IP version. Only v4 is supported.
Ftps_ip_address	String	FTP server's IP address. Only IPv4 addresses are supported.
Login_name	String	FTP login account name
Login_password	String	FTP login account password
Ramdump_path	String	Optional. Directory on FTP server to locate ramdump files. Directory under the root is supported.
ftpc_data_port	Integer	Optional. Data port for FTP client
ftps_cmd_port	Integer	Optional. Command port for FTP server
Ramdump_encryption	Integer/0,1	Optional. No encryption is supported yet.

6.3.12 Peripherals

The peripherals sub-group demo is intended to demonstrate ADC/PWM and sensors on MB functionality and APIs.

6.3.12.1 ADC

This command reads ADC converter and output the data.

Parameter Name	Type/Range	Description
None		

6.3.12.2 Humidity sensor

This command operates the humidity sensor.

Parameter Name	Type/Range	Description
Select	Number	0: activate sensor 1: red register 2: write register 3: read humidity measurement value

6.3.12.3 Pressure sensor

This command operates the pressure sensor.

Parameter Name	Type/Range	Description
Select	Number	0: set power status 1: read sensor parameters 2: set sensor parameters 3: read sensor measurement value

6.3.12.4 Compass sensor

This command operates compass sensor.

Parameter Name	Type/Range	Description
Select	Number	0: set operation mode 1: read sensor parameters 2: no operation 3: read the sensor measurement value

6.3.12.5 Gyroscope sensor

This command operates gyroscope sensor.

Parameter Name	Type/Range	Description
Select	Number	0: set operation mode 1: read the sensor parameters 2: no operation 3: read the sensor measurement value

6.3.12.6 Light sensor

This command operates light sensor.

Parameter Name	Type/Range	Description
Select	Number	0: set operation mode 1: read the sensor parameters 2: set the sensor parameters 3: read the sensor measurement value

6.3.12.7 PWM

This command operates PWM output control.

Parameter Name	Type/Range	Description
frequency	number	PWM output frequency
duty	Number/ 0-10000	PWM duty percent
phase	number	PWM phase
Channel	number	PWM channel
Time_interval	Number	PWM output time interval

6.3.12.8 SDIO/SPI master-slave interface

6.3.12.8.1 Configuration for SDIO Interface

To enable SDIO mode:

- Edit the file: quartz/demo/QCLI_demo/src/targetif/transport/QuRT/transport.c

CAUTION: Enable: #define SDIO_TRANSPORT

6.3.12.8.1.1 Pin mux configuration

SDIO Host controller configuration

Modify “sdio_gpio_arr” and “sdio_gpio_off_arr” to the values highlighted in the file:
target/quartz/demo/QCLI_Demo/src/export/DevCfg_master_fom_out.xml

```
<driver name="sdio">
  <global_def>
    <var_seq name="sdio_gpio_arr" type="0x00000002">
        0x12, 3, 0, 1, 1,
        0x13, 3, 0, 2, 1,
        0x14, 3, 0, 2, 1,
        0x15, 3, 0, 2, 1,
        0x16, 3, 0, 2, 1,
        0x17, 3, 0, 2, 1,
        end
    </var_seq>
    <var_seq name="sdio_gpio_off_arr" type="0x00000002">
        0x12, 3, 0, 1, 1,
        0x13, 3, 0, 2, 1,
        0x14, 3, 0, 2, 1,
        0x15, 3, 0, 2, 1,
        0x16, 3, 0, 2, 1,
        0x17, 3, 0, 2, 1,
        end
    </var_seq>
  </global_def>
```

```
  <device id="0x03000006">
    <props id="0x0010" type="0x00000018"> sdio_gpio_arr </props>
    <props id="0x0011" type="0x00000018"> sdio_gpio_off_arr </props>
    <props id="0x0020" type="0x00000002"> 96000000 </props>
    <props id="0x0021" type="0x00000002"> 2 </props>
    <props id="0x0022" type="0x00000002"> 10 </props>
    <props id="0x0023" type="0x00000002"> 1 </props>
  </device>
</driver>
```

6.3.12.8.1.2 Building the Demo

- Select peripheral configuration for the QCLI demo:
  - On Windows: setenv.bat periphenv.config
  - On Linux/Cygwin: setenv.sh periphenv.config

CAUTION: Build the demo as explained in the Build sample applications section.

6.3.12.8.2 HTCSlave

6.3.12.8.2.1 Start

This command Starts HTC Slave and initializes the SPI or SDIO interface.

Parameter Name	Type/Range	Description
Transport ID	Number	0: SDIO 1: SPI

6.3.12.8.2.2 Stop

Stop HTC Slave interface.

6.3.12.8.3 HTCHost

6.3.12.8.3.1 Init

Initialize HTC host interface.

6.3.12.8.3.2 Listtargets

List all HTC targets.

6.3.12.8.3.3 Open

Open connection to HTC slave.

Parameter Name	Type/Range	Description
Target ID	Number	ID of the target detected

6.3.12.8.3.4 Close

Close connection to HTC slave.

Parameter Name	Type/Range	Description
Target id	Number	ID of the target detected

6.3.12.8.3.5 shutdown

Stop HTC host interface.

6.3.12.8.4 Usage examples

The following CLI commands can be used to test master-slave communication:

```
//Start the target demo application in SDIO mode-
Peripherals\HTCSlave> start 0

//Or, to start the target demo application in SPI mode-
Peripherals\HTCSlave> start 1

//Run the following commands on the host device

//Initialize the HTC Host functionality
Peripherals\HTCHost> init

//To show the id of detected targets:
Peripherals\HTCHost> listtargets

//Open HTC:
Peripherals\HTCHost> open 0
   //Where 0 is the target id detected.

//To transmit from Host:
Peripherals\HTCHost> write 0 0 254
   //0 - target id
   //0 – endpoint id ( it can be from 0 to 3) for four endpoints.

//To receive from Host:
Peripherals\HTCHost> read 0 0 254
   //0 – target id
   //0 – endpoint id ( it can be from 0 to 3) for four endpoints.

//To Close the HTC connection:
Peripherals\HTCHost> close 0
   //0 – target id


//Shutdown HTC:
          Peripherals\HTCHost> shutdown
```

6.3.12.9 Keypad

6.3.12.9.1 Keypad

This command starts keypad demo. Each subsequent key press on the keypad displays the row and column information on CLI terminal.

6.3.12.9.2 Quit

This command stops keypad demo.

6.3.13 Low power

The Low Power demo is intended to demonstrate operating mode transition functionality and APIs.

6.3.13.1 Sleep

This command enables or disables the Deep-Sleep feature.

Parameter Name	Type/Range	Description
Mode	Number	0: Disable Sleep 1:Enable Sleep

6.3.13.2 RunOMTests

This command to triggers the OM transition tests.

Parameter Name	Type/Range	Description
dest_om	number	Destination operation mode. Takes value 1 or 2 1: FOM<=>SOM 2: FOM<=>MOM
duration_time	number	Timeout value in ms

6.3.13.3 ResetOMTests

This command is used to reset previous OM test Vectors.

6.3.14 File system

The FS demo is intended to demonstrate File System functions and File System APIs.

6.3.14.1 ls

This command lists all the files at file system.

6.3.14.2 format

This command erases all the files at file system.

6.3.14.3 Rm

This command removes the specified file at file system.

Parameter Name	Type/Range	Description
Filename	String	Indicates file name to be removed.

6.3.14.4 Read

This command reads the specified file and display at terminal.

Parameter Name	Type/Range	Description
Filename	String	Indicates file name to be displayed.
Offset	Number	Indicates the start location to read.
Length	Number	Indicates the length to read

For example: read /spinor/readme.txt 0 100

6.3.14.5 Write

This command writes hexadecimal string to the specified file.

Parameter Name	Type/Range	Description
filename	String	Indicates file name to create at file system.
offset	Number	Indicates the start offset to write
Data_in_hex	Hexadecimal string	Hexadecimal string to be written to the file

Example: write /spinor/1.txt 0 1122334455667788

6.3.15 Secure FS

The Secure FS demo is intended to demonstrate Secure File System functions and Secure File System APIs.

6.3.15.1 ls

This command lists all the files matching the password in the file system.

Parameter Name	Type/Range	Description
password	Hexadecimal string/32	Indicates password used in creating the file

6.3.15.2 Rm

This command removes the specified file from the file system.

Parameter Name	Type/Range	Description
filename	String	Indicates file name to be removed.

6.3.15.3 Open

This command opens a secure storage file according to the specified flag in the file system.

Parameter Name	Type/Range	Description
filename	string	Indicates file name to be removed.
password	Hexadecimal string/32	Indicates password used in creating the file
flags	Number	Indicates describes how this file should be opened. For details, see qapi_securehs.h in SDK.

6.3.15.4 Seek

This command changes the file offset for the opened secure storage file.

Parameter Name	Type/Range	Description
offset	Number	Indicates the new offset within the secure storage file
whence	Number	Indicates how the new offset is computed: #define QAPI_FS_SEEK_SET 0 /**< Seek from beginning of file. */ #define QAPI_FS_SEEK_CUR 1 /**< Seek from current position. */ #define QAPI_FS_SEEK_END 2 /**< Seek from end of file. */

6.3.15.5 Tell

This command specifies the file offset for the opened secure storage file.

6.3.15.6 Read

This command reads the specified 'length' bytes of data from the opened secure storage file.

Parameter Name	Type/Range	Description
Length	Number	Indicates the length to read

6.3.15.7 Write

This command writes hexadecimal string to the opened secure storage file.

Parameter Name	Type/Range	Description
Data_in_hex	Hexadecimal string	Hexadecimal string to be written to the file

6.3.16 Crypto

Persistent objects

These commands are under the "Crypto" QCLI submenu.

6.3.16.1 run-tests

```
pobj run-tests
```

Description: Runs a fixed set of basic validation tests.

6.3.16.2 create

```
pobj create –h server –t file_type –f file_name –i obj_id –o obj_type -l
flags
```

Description: Creates a persistent object. The key material for a cryptographic object or data for a data object is initialized from a specified file. If the -h option is specified, the file is downloaded. Otherwise, we search for the file under /spinor/pobj-test.

- *server* (optional) IP address of remote server hosting the file. Server should run "certcs –s" in the directory that stores the file.
- *file_type* PEM (for ECDSA, ECDH, and RSA object types) or RAW (for HMAC, AES, and data objects)
- *file_name* name of the file (containing the key material or data for a pure data object)
- *obj_id* id of object (at least 1 character and at most 64 characters long)
- *obj_type* See following table "Object Types" of the specification. Set to 0 for pure data object (ones not associated with a key)
- *flags* See flags argument to TEE_CreatePersistentObject in the specification. Notes: We do not plan to support TEE_DATA_FLAG_SHARE_READ and TEE_DATA_FLAG_SHARE_WRITE, and this will return QAPI_ERR_NOT_SUPPORTED. TEE_DATA_FLAG_ACCESS_READ and TEE_DATA_FLAG_ACCESS_WRITE flags are only applicable to pure data objects.

Object Types

Object Type	ID
AES	0xA0000010
HMAC MD5	0xA0000001
HMAC SHA1	0xA0000002
HMAC SHA224	0xA0000003
HMAC SHA256	0xA0000004
HMAC SHA384	0xA0000005
HMAC SHA512	0xA0000006
RSA PUBLIC KEY	0xA0000030
RSA KEYPAIR	0xA1000030
ECDSA PUBLIC KEY	0xA0000041
ECDSA KEYPAIR	0xA1000041

Examples:

pobj create -h 192.168.1.101 -t RAW -f aes_key.bin -i aes_key_1 -o 0xA0000010 -l 0

pobj create -h 192.168.1.101 -t PEM -f private.pem -i rsa_key_pair_1 -o 0xA1000030

pobj create -h 192.168.1.101 -t PEM -f private.pem -i ecdsa_key_pair_1 -o 0xA1000041

6.3.16.3 list

```
pobj list
```

Description: Prints a table of persistent objects available including their object id and metadata. The field names of the table are specified on the first row. The first column is the object id. Subsequent columns are the object info fields. These are in the same order as listed in qapi_Crypto_Obj_Info_t.

6.3.16.4 sign

```
pobj sign –h server –f file_name –i obj_id –a signing_algorithm –d
hash_algorithm
```

Description: This command signs/hmacs a file and outputs the signature in base64. This signature/hmac can be verified with openssl.

- *server* (Optional) IP address of remote server hosting the file to sign. Server should run "certcs –s" in the directory that stores the file.
- *file_name* name of the file to sign/hmac
- *obj_id* id of object containing the signing/hmac key
  hash_algorithm Not applicable for HMACs. Only applicable to signing algorithms and not hmac. Id of hash algorithm used to compute the digest. SHA1, SHA256, SHA384, SHA512 are supported algorithms. See Table 6-8 of the specification for algorithm ids for these (algorithm id constants are also defined in qapi_crypto.h)
- *signing_algorithm* Signing/hmac algorithm id from table 6-8 of the spec (algorithm id constants are defined in qapi_crypto.h)

Hash algorithm ID

Hash algorithm	ID
MD5	0x50000001
SHA1	0x50000002
SHA224	0x50000003
SHA256	0x50000004
SHA384	0x50000005
SHA512	0x50000006

Signature algorithm ID

Signature algorithm	ID
ECDSA P192	0x70001042
ECDSA P224	0x70002042
ECDSA P256	0x70003042
ECDSA P384	0x70004042
ECDSA P521	0x70005042
RSASSA PKCS 1.5 SHA1	0x70002830
RSASSA PKCS 1.5 SHA256	0x70004830
RSASSA PKCS 1.5 SHA384	0x70005830
RSASSA PKCS 1.5 SHA512	0x70006830
HMAC MD5	0x30000001

HMAC SHA1	0x30000002
HMAC SHA224	0x30000003
HMAC SHA256	0x30000004
HMAC SHA384	0x30000005
HMAC SHA512	0x30000006

Examples:

```
pobj sign –h 192.168.1.100 –f input.txt -i rsa_key_pair_1 -a 0x70004830
-d 0x50000004

Crypto: INFO: downloaded key file of size 12 bytes
Crypto: Signature:

Crypto: T1JoBPkRocFzR2bLXS55HMr9NiD5osVu
Crypto: aKIYCWIEvMlRwz2D5d30sCadVhYQ5CQQ
Crypto: 1kSbssb6liu3adPSGoXSu0/qU8sPBOOh
Crypto: 3fxwCKDWh+VokeeM3mrrsGm54N70lfay
Crypto: Fdh7e+KZQXYcOF2zznLUsfe7/Yn02DcD
Crypto: Pjsq829Jj5Q=
Crypto> pobj sign -h 192.168.1.101 -f input.txt -i hmac_key_1 -a 0x30000004

Crypto: INFO: downloaded key file of size 12 bytes
Crypto: MAC:

Crypto: DORDUig0VtZyyM4c5dC89621rsaN1L49
Crypto: jrEk83dV50A=
```

6.3.16.5 verify

```
pobj verify –h server –f file_name –s signature_file_name -i obj_id
```

Description: Verifies a signature/hmac over a file.

- *server* (Optional) IP address of remote server hosting the files. Server should run “certcs –s” in the directory that stores the files.
- *file_name* name of the file that the signature was computed over.
- *signature_file_name* name of file binary signature/hmac. This can be produced using openssl for example.
- *obj_id* id of object containing the key pair or public key for verification
- *hash_algorithm* see pobj sign
- signing_algorithm see pobj sign

6.3.16.6 encrypt

```
pobj encrypt –h server –f file_name -i obj_id -a algorithm -v iv_file_name
-n nonce_file_name -t tag_file_name -d aad_file_name
```

Description: Encrypts a file. The object must be an AES or RSA key. For AES the file to encrypt can be an arbitrary size. For RSA the file can contain at most the number of bytes as the RSA key size (for example, for an RSA key of modulus 1024 it can contain at most 128 bytes). For authenticated encryption (AES CCM and GCM) in addition to the encrypted data, a tag is generated.

- *server* (Optional) IP address of remote server hosting the file to encrypt. Server should run "certcs –s" in the directory that stores the file.
- *file_name* name of the file to encrypt
- *obj_id* id of aes key object
- *aad_file_name* for authenticated encryption AES CCM and GCM only
- *algorithm* See table 6-8 of the spec or `qapi_crypto.h` (AES, RSA algorithms supported)
- *iv_file_name* binary file contains initialization vector for AES CBC or optionally for CTR
- *nonce_file_name* binary file containing nonce for authenticated encryption AES CCM (>=7 and <= 13 bytes) and GCM (12 bytes)

Algorithm	ID
AES CBC NO PAD	0x10000110
AES CTR	0x10000210
AES CCM	0x40000710
AES GCM	0x40000810
RSAES PKCS1.5	0x60000130
RSA NO PAD	0x60000030

Examples:

```
Crypto> pobj encrypt -h 192.168.1.101 -f message.bin -i aes_key_1 -a
0x10000110 -v iv.bin
Crypto: INFO: downloaded key file of size 48 bytes
Crypto: INFO: downloaded key file of size 16 bytes
Crypto: Ciphertext:
Crypto: AGM4aKT7ZpGpLjBnniM4t27qQAWMeIAt
Crypto: ewfxffw+uX7FFyzDdoL1gt4gA5FX9z1B
```

6.3.16.7 decrypt

```
pobj decrypt  -h server –f file_name -i obj_id -a algorithm -v iv_file_name
-n nonce_file_name -t tag_file_name -d aad_file_name
```

Description: Decrypts a file and prints the contents in base64.

- *server* (Optional) IP address of remote server hosting input files. Server should run "certcs –s" in the directory that stores the file.
- *file_name* name of the file to decrypt with encrypted data in base64 format.
- *obj_id* id of aes key object

- *aad_file_name* for authenticated encryption AES CCM and GCM only
- *algorithm* See table 6-8 of the spec or qapi_crypto.h (AES, RSA algorithms supported)
- *iv_file_name* binary file containing random initialization vector for AES CBC or optionally for CTR
- *nonce_file_name* binary file containing nonce for authenticated encryption AES CCM (<= 16 bytes) and GCM (12 bytes)
- *tag_filename* binary file containing tag for authenticated encryption AES CCM and GCM only (output from encrypt)

6.3.16.8 delete

```
pobj delete obj_id
```

Description: Deletes a persistent object.

6.3.16.9 rename

```
pobj rename obj_id new_obj_id
```

Description: Renames a persistent object to a new object id.

6.3.16.10 get-attributes

```
pobj get-attributes -a attribute-id -i obj_id
```

Description: Prints the attribute for a cryptographic persistent object. Attributes can be integer or buffer types. Buffer type attributes are printed in base64.

- *obj_id* id of object from which the attribute is read
- *attribute_id* attribute to read (only public attributes can be read). See table 6-11 of spec for list of public attributes

Attribute	ID
ECC PUBLIC VALUE X	0xD0000141
ECC PUBLIC VALUE Y	0xD0000241
ECC CURVE	0xF0000441
RSA PUBLIC EXPONENT	0xD0000230
RSA MODULUS	0xD0000130

Examples:

```
pobj get-attribute -a 0xD0000230 -i rsa_key_pair_1
pobj get-attribute -a 0xD0000130 -i rsa_key_pair_1
pobj get-attribute -a 0xD0000141 -i ecdsa_key_pair_1
pobj get-attribute -a 0xD0000241 -i ecdsa_key_pair_1
pobj get-attribute -a 0xF0000441 -i ecdsa_key_pair_1
```

6.3.16.11 clone

```
pobj clone obj_id obj_id_of_copy
```

Description: Creates a new persistent object from an existing persistent object

6.3.17 AWS demo

The AWS Demo is intended to demonstrate AWS IOT functions such as a simple shadow sample API and a simple jobs sample API.

6.3.17.1 aws_set_schema

This command sets the json schema, host name, thing name, device certificate, and interval for sending updates.

Before running this command, see step 3 of the Certificate management demo t section to download the device certificate using certcs server.

Parameter Name	Type/Range	Description
Host name	string	AWS IOT endpoint name
Thing Name	string	AWS IOT thing name (max 20 characters)
File Name	string	A file contains valid JSON schema. Note: Nested objects are not supported.
Device Cert Name	string	A cert file that is generated when a new Thing is created in AWS IOT console.
Interval Time	Number (time in milliseconds)	An interval which will be used to send new updates to AWS server. Note the demo uses random number to generate new data for each update.

6.3.17.2 aws_set_params

Parameter Name	Type/Range	Description
JSON Key	string	Valid json key based on schema used which setting schema
JSON type	Enum type (int)	See JsonPrimitiveType in documentation.
Value	Valid value based on Enum type	Provide a valid value based on enum type
and_mask	Integer	And_mask for random values.
Or_mask	Integer	Or_mask for random values

Usage of and/or mask is as follows:

Output = (random value & and_mask) | or_mask

6.3.17.3 aws_run

Runs the AWS shadow sample demo. This command will initiate a tls session with aws server and then transmit updates based on the parameters provided by preceding two commands. The user can then use AWS console to send updates to the device.

6.3.17.4 aws_quit

Stops the demo if it is running and resets all the parameters. If the demo is not running, then resets all the parameters supplied in preceding three commands.

6.3.17.5 jobs_run

Runs an AWS IoT job demo. This command enables to deploy and track management tasks in the device fleet. Use jobs to send remote actions to one or many devices at the same time, control the deployment of jobs to the devices, and track the current and past status of job executions for each device.

Parameter Name	Type/Range	Description
Host name	string	AWS IOT endpoint name
Thing Name	string	AWS IOT thing name(max 20 characters)
RootCA Name	string	A Root CA file signed by AWS for device authentication.
Device Cert Name	string	A cert file which is generated when a new Thing is created in AWS IOT console.

See AWS documentation before getting started with AWS IoT Job demo, located at https://docs.aws.amazon.com/iot/latest/developerguide/ios-sdk-create-job.html

6.3.17.6 jobs_quit

Stops the AWS jobs demo if it is running and resets all the parameters. If the demo is not running, then this command resets all the parameters supplied in preceding jobs_run command.

6.3.17.7 Usage examples

The following sections demonstrate how to setup connection and run commands in AWS demo.

6.3.17.7.1 Initialization

The procedure to download SDK and make changes are explained in the Build section. To build QCLI demo with AWS IoT:

8. Go to the target/quartz/demo/QCLI_demo/build/gcc folder.

CAUTION: Set ecosystem to AWS IOT:

On Windows: `ecosystem.bat 1`

On Linux: `export ecosystem=awsiot`

See the Build sample applications section to build QCLI demo, section Flash the image to flash the build and the AWS demo section on how to run AWS Demo.

See AWS documentation before getting started with AWS demo, located at http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html.

6.3.17.7.2 Connection procedure

The following sections demonstrate how to enable WLAN, connect to an access point, and configure a DHCPv4 client.

```
WLAN> Enable                  // Enable WLAN
WLAN> SetDevice 1             // Set device in Station Mode
WLAN> SetWpaPassphrase 12345678   // Enter Password of access point
WLAN> SetWpaParameters WPA2 CCMP CCMP  // Enter version, ucipher/mcipher
WLAN> Connect Test_AP         // Connect to access point
```

```
WLAN: Setting SSID to Test_AP
WLAN: devid - 1 1 CONNECTED MAC addr 00:03:7f:59:09:56
WLAN: 4 way handshake success for device=1

Net> dhcpv4c wlan1 new       // configuredhcpv4c to get valid IP
Net: DHCPv4c: IP=10.177.238.70  Subnet Mask=255.255.254.0
Gateway=10.177.238.1

//Set the sntp client server to 24.56.178.140 ,try to resolve AWS
Endpoint //

Net> sntpc start             // Start SNTP client
Net> sntpc addsvr pool.ntp.org    // Add server to SNTP client

Net> dnsc start    // Start DNS client and resolve AWS Endpoint
Net> dnsc resolve a3rvdj82eswxf2.iot.us-west-2.amazonaws.com
Net: a3rvdj82eswxf2.iot.us-west-2.amazonaws.com --> 52.34.127.18
```

6.3.17.7.3 Download the certificates and write to file system

This section demonstrates how to download certificates from certs server and write JSON lint text to file system.

On Linux Machine:

Connect to Linux Machine to same Access point of the Quartz Board.

Copy certificates (see point 3 in the Certificate management demo section) to the **certcs** directory which also contains the **certcs** executable.

Start the certificate server.

```
./certcs -s
```

On Quartz Device:

```
Net> cert get QZD1.cert.pem 10.177.244.133 -k QZD1.private.key -s QZD.pem -
t pem_cert
Net: Successfully downloaded QZD1.cert.pem
Net: Successfully downloaded QZD1.private.key
Net: Successfully converted and stored certificate
Net: QZD.pem is stored in NV memory

Net> cert get root_ca.pem 10.177.244.133 -s aws_ca_list.bin -t pem_ca_list
Net: Successfully downloaded root_ca.pem
Net: Successfully converted and stored CA list
Net: aws_ca_list.bin is stored in NV memory

// Write JSON file into file system, data in hex is converted json lint//
// Sample VALID JSON LINT //
{
   "test": "test",
   "test1": "test1"
}
```

```
// write the file into /spinor/json.txt path and set offset to 0 //
Fs> write /spinor/json.txt 0
"7b0d0a20200d0a092274657374223a20227465737422c0d0a09227465737431223a202274
65737431220d0a0d0a7d"
```

6.3.17.7.4 Set and run AWS

This section demonstrates how to set the schema and parameters and run AWS from QCA402x.

```
// set schema using endpoint thing_name file_name certificate_name //
Ecosystem> aws_set_schema a3eof03n80k7z0.iot.us-west-2.amazonaws.com QZD1
/spinor/json.txt client.pem 100

// Set parameters for aws based on json file option,value, endmask, ormask
//
Ecosystem> aws_set_params test 3 123 1111 0000
Ecosystem> aws_set_params test1 3 456 1111 0000
Ecosystem> aws_run          //Run the aws with provided options
Ecosystem: AWS IoT SDK Version 2.1.1-
Ecosystem: Shadow Init
Ecosystem: Shadow Connect
Ecosystem: Shadow Connection successful
Ecosystem:
================================================================================
============
Ecosystem: Update Shadow: {"state":{"reported":{"test":123,"test1":123}},
"clientToken":"QZD1-0"}
Ecosystem:
********************************************************************************
**************
Ecosystem: Randomizing Data
Ecosystem: Update Accepted !!
```

6.3.17.7.5 Run AWS jobs demo

This section demonstrates how to run AWS jobs demo from QCA402x.

```
// Ecosystem jobs_run [Host_Name] [Thing_Name] [RootCA_name] [Dev_cert]
> Ecosystem jobs_run acfg88q6accah.iot.us-west-2.amazonaws.com iot-t1-
entry1 aws_ca_list.bin QZD.pem
Ecosystem: Connecting...
Ecosystem: JOB_GET_PENDING_TOPIC callback
Ecosystem: topic: $aws/things/iot-t1-entry1/jobs/get/accepted
Ecosystem: payload:
{"timestamp":1541106370,"inProgressJobs":[],"queuedJobs":[{"jobId":"example
-jobs-
01","queuedAt":1541106354,"lastUpdatedAt":1541106354,"executionNumber":1,"v
ersionNumber":1}]}
Ecosystem: inProgressJobs: []
```

```
Ecosystem: queuedJobs: [{"jobId":"example-jobs-
01","queuedAt":1541106354,"lastUpdatedAt":1541106354,"executionNumber":1,"v
ersionNumber":1}]

Ecosystem: JOB_NOTIFY_NEXT_TOPIC / JOB_DESCRIBE_TOPIC($next) callback
Ecosystem: topic: $aws/things/iot-t1-entry1/jobs/$next/get/accepted
Ecosystem: payload: {"timestamp":1541106370,"execution":{"jobId":"example-
jobs-
01","status":"QUEUED","queuedAt":1541106354,"lastUpdatedAt":1541106354,"ver
sionNumber":1,"executionNumber":1,"jobDocument":{"operation":"customJob","o
therInfo":"someValue"}}}
Ecosystem: execution: {"jobId":"example-jobs-
01","status":"QUEUED","queuedAt":1541106354,"lastUpdatedAt":1541106354,"ver
sionNumber":1,"executionNumber":1,"jobDocument":{"operation":"customJob","o
therInfo":"someValue"}}
Ecosystem: jobId: example-jobs-01
Ecosystem: jobDocument: {"operation":"customJob","otherInfo":"someValue"}

Ecosystem: JOB_UPDATE_TOPIC / accepted callback
Ecosystem: topic: $aws/things/iot-t1-entry1/jobs/example-jobs-
01/update/accepted
Ecosystem: payload: {"timestamp":1541106371}

Ecosystem: JOB_NOTIFY_NEXT_TOPIC / JOB_DESCRIBE_TOPIC($next) callback
Ecosystem: topic: $aws/things/iot-t1-entry1/jobs/notify-next
Ecosystem: payload: {"timestamp":1541106372}
Ecosystem: execution property not found, nothing to do
```

6.3.18 JSON

This section demonstrates the commands related to JSON object creation, modification, query, and deletion operations.

6.3.18.1 Ver

This command displays the current QAPI version and CRM build version.

6.3.18.2 Help

This command is used to display a command list or usage for a command. It will be the first command in the list for every menu level.

6.3.18.3 Up

This command is used to navigate up one group level. For instance, if the command is executed from within the “MyDemo\SubPart1” command group, QCLI will navigate to the “MyDemo” command group. This command is only available while within a group or subgroup (not at the root level).

6.3.18.4 Root

This command is used to immediately navigate to the root group level. It is only available within a group or subgroup.

6.3.18.5 CreateObject

Parameter Name	Type/Range	Description
json formatted string	String	String input to create json object. String must follow JSON format restrictions.

This command is used to create a JSON object tree from the given string. If the input string follows valid JSON format, a handle to new object tree is displayed. Otherwise, the command returns an error. Displayed handle should be used for other operations on this object tree.

Json is a standard format, more information can be found at http://www.json.org.

Following are the restrictions/assumptions that apply to our JSON implementation:

- No space is allowed in the input as anything after the space will be considered as next argument.
- All keys are string whereas values can be of types strings, integers, floating point numbers, true, false or null.
- All strings should be enclosed in double quotes and quotes should be escaped using \ as shown in the following examples:
- Numbers, integers, true, false and null as values do not need double quotes, however, if quotes are used, the values will be treated like strings.
- An extra space character is required at the end if the string ends with “; otherwise, the command throws error. This limitation is only for this demo and may not be required for other demos.
- Any object with key-value pair/s must be enclosed in { and } brackets; otherwise, only the key will be used to create object and value will be ignores.

Following are the examples:

NOTE: A user must free the memory allocated for the object/tree by calling DeleteTree or DeleteAllObjects when it is not used. Failure to do this causes memory leak.

Examples

- Integer value without any key

```
JSON> CreateObject 1
JSON: Json object handle: 0.
```

- Floating point value without any key

```
JSON> CreateObject 0.5
JSON: Json object handle: 1.
```

- Single key-value pair
  - Invalid input: The value “John” is ignored here, only “name” is used.

```
JSON> CreateObject \"name\":\"John\"
```

```
JSON: Json object handle: 2.
```

- Valid input

```
JSON> CreateObject {\"name\":\"John\"}
JSON: Json object handle: 3.
```

i

- True, false and null objects − these values do not require double quotes

```
JSON> CreateObject true
JSON: Json object handle: 5.
```

- Array of values: Values in a an array can be of different type. Escaped double quotes required for strings. And complete input should be enclosed in [and].

```
JSON> CreateObject [\"value1\",2,3.5,true,false,null]
JSON: Json object handle: 7.
```

- Object with multiple key:value pairs. Each pair should be separated by a comma.

```
JSON> CreateObject {\"node\":\"washer\",\"soak\":true,\"time\":60}
JSON: Json object handle: 8.
```

- Nested objects

```
JSON> CreateObject
{\"node\":\"washer\",\"soak\":true,\"time\":{\"hours\":1,\"minutes\":30}
}
JSON: Json object handle: 9.
```

6.3.18.6 GetJsonString

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.

This command is used to display JSON object/array in string format. Input handle should be returned when the respective object was created.

Example

The following exmples use the objects created as per the examples in Section 6.3.18.5.

- Integer value without any key.

```
JSON> GetJsonString 0
JSON: 1
```

- Floating point value without any key

```
JSON> GetJsonString 1
JSON: 0.5000
```

- Single key-value pair
  - Invalid input: The value “John” was ignored here, only “name” was used.

```
JSON> GetJsonString 2
JSON: "name"
```

  - Valid input

```
JSON> GetJsonString 3
JSON: {
```

```
JSON:   "name": "John"
JSON: }
```

- True, false and null objects: These values do not require double quotes.

```
JSON> GetJsonString 5
JSON: true
```

- Array of values: Values in a an array can be of different type. Escaped double quotes required for strings. And complete input should be enclosed in [and].

```
JSON> GetJsonString 7
JSON: ["value1", 2, 3.5000, true, false, null]
```

- Object with multiple key:value pairs. Each pair should be separated by a comma.

```
JSON> GetJsonString 8
JSON: {
JSON:   "node": "washer",
JSON:   "soak": true,
JSON:   "time": 60
JSON: }
```

Nested objects

```
JSON> GetJsonString 9
JSON: {
JSON:   "node": "washer",
JSON:   "soak": true,
JSON:   "time": {
JSON:           "hours":        1,
JSON:           "minutes":      30
JSON:   }
JSON: }
```

6.3.18.7 Query

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
key / index	Key is of type string Index can be 0 – 0xFFFFFFFF	Key to search for OR Index of the required item.

This command is used for searching the specified JSON object/array to find an item with specified key or the item at specified index. Command returns the value of the item if found; else, the command returns error.

Example

These examples are based on the examples given in the previous sections.

- Query using key

```
JSON> Query 9 soak
JSON: Json query value: true
```

- Query using index

```
JSON> Query 7 2
JSON: Json query value: 3.5000
```

6.3.18.8 InsertValueByIndex

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
index	0 – 0xFFFFFFFF	Index at which new item is to be added.
new_item_string	String	JSON formatted string for new item to be added.

This command is used to insert a new item at specified index in the specified JSON object/array. New item string must be JSON formatted, and if it ends with **"**, extra space should be added at the end of the command, otherwise the command throws error.

As this command is specific for inserting entries with only value, not key:value, it is best used for arrays. Instead, the command InsertKeyValueByIndex should be used for inserting key:value pairs. Note that the command does not return an error when used for objects; however, the result may not be as expected.

If the index is more than the length of existing object tree/array, then the item is appended at the end.

This command only works for the outermost level of nested structures.

To insert a value in an array that is nested inside another array/object, use ReplaceValueByIndex, instead of this command, to replace the whole object/array which is at the outermost level.

Example

Insert string value in an array. There is an extra space at the end.

```
JSON> InsertValueByIndex 7 4 \"newvalue\"
JSON> GetJsonString 7
JSON: ["value1", 2, 3.5000, true, "newvalue", false, null]
```

6.3.18.9 InsertKeyValueByIndex

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
index	0 – 0xFFFFFFFF	Index at which new item is to be added.
key	String	Key field for the new item to be added, no formatting is needed.
new_item_string	String	Value field for the item to be added. Must be a JSON formatted string.

This command is used to insert a new item with specified key and value at given index in the given JSON object tree. If the index is more than the length of existing object tree/array, then the item is appended at the end.

Value string must be JSON formatted, which means if the value is a string, it must be enclosed in escaped double quotes and an extra space must be added at the end of the command; otherwise, the command throws error. Values of formats integers, floating point numbers, true, false, and null do not require formatting. The key does not need any formatting.

This command cannot be used for JSON arrays.

The command works for the outermost level of nested structures only.

To insert an entry in an object that is nested inside another array/object, use ReplaceValueByKey or ReplaceValueByIndex, instead of this command, to replace the whole object/array which is at the outermost level.

Example

- Invalid input

```
JSON> InsertKeyValueByIndex 8 0 newkey newValue
JSON: Json operation failed.
```

- Invalid input; this command cannot be used for arrays

```
JSON> 8 7 0 arrayKey \"arrayValue\" (Extra space here)
JSON: Json operation failed.
```

- Valid input

```
JSON> InsertKeyValueByIndex 8 0 newkey \"newValue\"
JSON> GetJsonString 8
JSON: {
JSON:   "newkey":       "newValue",
JSON:   "node": "washer",
JSON:   "soak": true,
JSON:   "time": 60
JSON: }
```

6.3.18.10 ReplaceValueByIndex

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
index	0 – 0xFFFFFFFF	Index of the item to be modified.
new_value	String	New value field for the queried item, must be a JSON formatted string.

This command is used to replace value of an existing item at given index in specified JSON object tree/JSON array with a new value. The command returns an error if the index is more than the length of existing object tree/array.

New value string must be JSON formatted, and if it ends with **"**, extra space should be added at the end of the command, otherwise the command throws error.

If used for JSON object, original key field is retained.

The command only works for the outermost level of nested structures.

For replacing values/objects/arrays in inner levels of nested objects, replace the complete array/inner object as shown in the following example:

Example:

- Change time from 60 to 30. Time is at index 2 as index starts at 0.

```
JSON> GetJsonString 8
JSON: {
JSON:   "node": "washer",
JSON:   "soak": true,
```

```
      JSON:   "time": 60
      JSON: }
JSON> ReplaceValueByIndex 8 2 30
JSON> GetJsonString 8
JSON: {
JSON:   "node": "washer",
JSON:   "soak": true,
JSON:   "time": 30
JSON: }
```

- Replacing value in an array inside an object: In this example, changing value '1' from the array to '5' is not possible directly as the array is inside the main object. Instead, whole array can be changed. The array is at index 1 in the main object as index start at 0.

```
JSON> CreateObject  {\"key1\":\"v1\",\"key2\":[0,1,2]}
JSON: Json object handle: 0.
JSON> GetJsonString 0
JSON: {
JSON:   "key1": "value1",
JSON:   "key2": [0, 1, 2],
JSON:   "key3": null
JSON: }

JSON> ReplaceValueByIndex 0 1 [0,5,2]

JSON> GetJsonString 0
JSON: {
JSON:   "key1": "value1",
JSON:   "key2": [0, 5, 2]
JSON:   "key3": null
JSON: }
```

- This command can also be used for inserting a new entry in the array which cannot be done by the insert commands as given here.

```
JSON> CreateObject  {\"key1\":\"v1\",\"key2\":[0,1,2]}
JSON: Json object handle: 0.
JSON> GetJsonString 0
JSON: {
JSON:   "key1": "value1",
JSON:   "key2": [0, 5, 2],
JSON:   "key3": null
JSON: }

JSON>  ReplaceValueByIndex 0 1 [0,5,2,null,\"NewValue\"]

JSON> GetJsonString 0

JSON: {
JSON:   "key1": "value1",
JSON:   "key2": [0, 5, 2, null, "NewValue"],
```

```
JSON:   "key3": null
JSON: }
```

6.3.18.11 ReplaceValueByKey

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
key	string	Key field for the item to be modified. No formatting needed.
new_value	String	New value field for the queried item, must be a JSON formatted string.

This command is used to replace value of an existing item with given key in specified JSON object tree/JSON array with a new value. The command returns an error if an item with given key is not found.

New value string must be JSON formatted, and if it ends with **“**, extra space should be added at the end of the command, otherwise the command throws error.

In case of repeated keys, only first occurrence is modified.

Nested objects can be modified as given in Section 6.3.17.10 .

Example

This example is in continuation of the example in Section 6.3.17.10 .

Change time from 30 to 15 using this command.

```
JSON> ReplaceValueByKey 8 time 15
JSON>  GetJsonString 8
JSON: {
JSON:   "node": "washer",
JSON:   "soak": true,
JSON:   "time": 15
JSON: }
```

6.3.18.12 DeleteEntryByIndex

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
index	0 – 0xFFFFFFFF	Index of the item to be deleted.

This command is used to delete the item at given index in specified JSON object/array. The command returns an error if the index or handle is invalid.

Example

This example is in continuation of the example in Section 6.3.18.11.

```
JSON>  GetJsonString 8
JSON: {
JSON:   "node": "washer",
JSON:   "soak": true,
```

```
JSON:   "time": 15
JSON: }

JSON> DeleteEntryByIndex 8 0
JSON> GetJsonString 8
JSON: {
JSON:   "soak": true,
JSON:   "time": 60
JSON: }
```

6.3.18.13 DeleteEntryByKey

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the required JSON object tree.
key	String	Key field of the item to be deleted.

This command is used to delete the item with the specified key in specified JSON object/array to find an item with specified key or the item at specified index. The command returns an error handle and is invalid if no item with the specified key is found.

In case of repeated keys, only the first item found is deleted.

Example

This example is in continuation of the example in Section 6.3.17.12.

```
JSON> GetJsonString 8
JSON: {
JSON:   "soak": true,
JSON:   "time": 60
JSON: }
JSON> DeleteEntryByKey 8 time
JSON> GetJsonString 8
JSON: {
JSON:   "soak": true,
JSON: }
```

6.3.18.14 DeleteTree

Parameter Name	Type/Range	Description
object handle	0 – 0xFFFFFFFF	Handle for the JSON object/array to be deleted.

This command is used to delete the complete JSON object/tree with given handle. The command returns an error that the handle is invalid if no item with given key is found. In case of nested structure, all the children items are deleted.

Example

This example is in continuation of the example in Section 6.3.18.13 .

```
JSON> DeleteTree 8
JSON: JSON object corresponding to handle 8 freed.
```

```
JSON> GetJsonString 8
JSON: Invalid JSON object handle.
```

6.3.18.15 GetHandleList

This command is used to retrieve a list of all objects/arrays created.

Example

```
JSON> GetHandleList
JSON: Handles:
JSON: 0, 1, 2, 3, 4, 5, 6, 7, 9,
```

6.3.18.16 DeleteAllObjects

This command is used to delete all created JSON objects/trees. nd. In case of nested structures, all the children items are deleted.

6.3.19 Azure

6.3.19.1 azure-twin

Parameter Name	Type/Range	Description
param	1 - 3	Azure twin demo sub-commands: 1 – connect to IOT hub with provided connection string 2 – update properties 3 – change transmit interval
Connection string	String	Device connection string (used with command parameter 1)
Maximum speed	Unsigned char	Input to “update properties”
version	signed int	Input to “update properties”
vanity	String	Input to “update properties”
Oil level %	float	Input to “update properties”
Interval	Int / seconds	Periodic interval in seconds at which message is sent to hub

6.3.19.2 azure-stop

Stop azure twin demo. Release all resources and close the connection.

6.3.19.3 Usage Example

```
/**** Enable WLAN and connect to AP *****/
> wlan enable
> wlan SetWpaPassphrase <PASSPHRASE>
> wlan SetWpaParameters WPA2 CCMP CCMP
> wlan connect <SSID>

/**** Obtain IP Address ****/
> net dhcpv4c wlan0 new
```

```
/**** Enable DNS Client, Add a Public DNS Server */
> net dnsc start
> net dnsc addsrv 8.8.8.8

/**** Enable SNTP Client, add a Public SNTP server*/
> net sntpc start
> net sntpc addsvr pool.ntp.org

/***Connect to IoT Hub  */
> ecosystem azure-twin  1 <connection string>

/**** On Successful connection, following message will be seen on
console*****/
Ecosystem: Reported state will be send to IoTHub
Ecosystem: IoTHub: reported properties delivered with status_code = 204

/*** Update Reported Properties */
> ecosystem azure-twin  2  <max Speed> <version>  <vanity>  <oil level
percent>
            //  Here, max speed – uint8_t
            // version – uint32_t
            // vanity- string
            // Oil level % - float

/*** Send reported properties at specified interval */
> ecosystem azure-twin  3 <interval in sec>
   //If interval is non-zero, reported properties are sent at specified
interval.
```

7 QCLI UART AT command demo

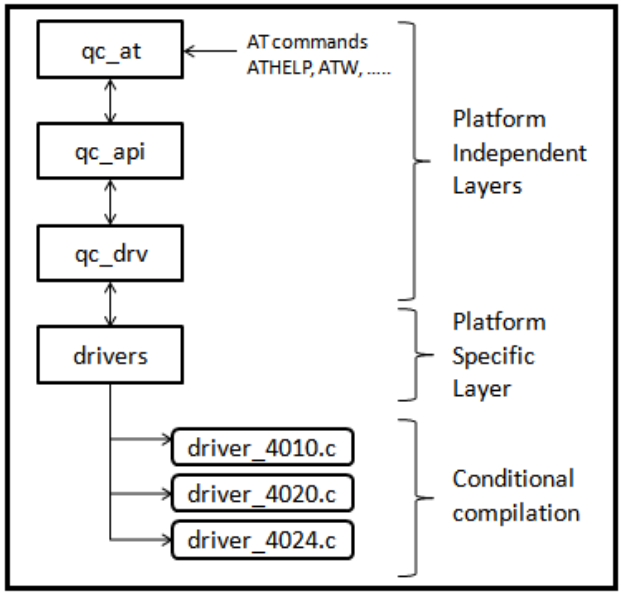
"QCLI_uart_at_demo" is a demo application that provides the AT commands for exercising the functionality of the boards and provides the capability for any MCU with low memory foot print to use CDB modules over UART interface with standard AT commands defined for the modules.

AT commands are used to communicate with the Quartz chipset loaded with the AT Command firmware connected to host using a serial interface. These commands provide access to the capabilities of the Quartz chipset to an external host over a serial interface. This offloads BLE, Zigbee, WLAN & TCP/IP (in QCA4020) and network management overhead to the Quartz chipset and facilitates small embedded host, based on microcontrollers like 8051, PIC, MSP430, AVR, and so on. that have a serial interface, to communicate with other hosts on the network using the available radio technologies. These serial hosts use simple command interface to configure and to create the wireless and network connections.

The necessary files for "QCLI_uart_at_demo" are under the "<sdk_source>/target/quartz/demo/QCLI_uart_at_demo" directory. There are four layers for handling the AT commands, which are "qc_at", "qc_api", "qc_drv" and "drivers"

7.1 Driver layers for AT command

There are four layers for handling the AT commands, which are "qc_at", "qc_api", "qc_drv" and "drivers".



qc_at, qc_api, qc_drv folders are always compiled in all platforms and act as clean abstractions to access platform-specific operations.

Drivers folder contains platform-specific files which may/may not contain few modules, may/may not contain few APIs from modules, may/may not contain API changes including but not limited to API name changes, API parameter name changes, API parameter count changes and so on.

The modules Wi-Fi, P2P, MQTT, OTA, Net, Thread has got the four level (qc_at, qc_api, qc_drv, drivers) layering hierarchy whereas the modules such as BLE, Zigbee has got the three level (qc_at, qc_drv, drivers) layering hierarchy.

- **Layer – "qc_at"**
  "qc_at" is the entry point for all AT commands for different modules like ATW, ATP, ATO, ATN, ATM, ATZ, ATT, and handle all command line validations before making a call to "qc_api" layer.

  Prints allowed in this layer shall be strictly usage/warning/error prints

  usage/warning/error prints cannot be compiled out

  Coding guidelines: All functions in "qc_at" has the following format:

  "qc_at_<MODULE_NAME>_<API_NAME>"

  Where, MODULE_NAME = {wifi, p2p, ota, network, mqtt, zigbee and so on.}

  API_NAME= derived from qcom_/qapi_ API nomenclature

- **qc_api layer:** This layer maintains any possible states which may be required during operation by saving the states and saving the configurations needed for connectivity operations. "qc_api" layer makes calls to "qc_drv" layer to set/get module specific parameters and perform any operations specific to a module. This layer does have debug prints which can be compiled out.

  Coding guidelines: All functions in "qc_api" are in the format:

  qc_api_<MODULE_NAME>_<API_NAME>

  Where:

  MODULE_NAME = {wifi, p2p, ota, network, mqtt, zigbee and so on}

  API_NAME= derived from qcom_/qapi_ API nomenclature

- **qc_drv layer:** This layer calls the platform-specific APIs (4010/4020/4024 and so on) which are registered as callbacks from driver_xxxx.c file (platform specific). This layer does have debug prints which can be compiled out.

  Coding guidelines: All functions in "qc_drv" are in the format:

  `qc_drv_<MODULE_NAME>_<API_NAME>`

  Where:

  MODULE_NAME = {wifi, p2p, ota, network, mqtt, zigbee and so on.}

  API_NAME= derived from qcom_/qapi_ API nomenclature

7.2 Command format and interface

- **AT command request:** All commands start with "AT", and are not case sensitive. However, the arguments for the commands (such as the SSID) are case-sensitive. Each command is terminated with a <CR>.
- **AT command response:** Command responses are in the form of ASCII strings. Command success returns an "OK" message. Command failure returns an "ERROR" message.

On boot up, the command interface will wait for command input on the UART.

The default UART settings of the adapter are:

- Baud rate: 115200
- Bits per character: 8
- Parity: none
- Number of stop bits: 1
- Flow control: off

7.3 Command reference

This section describes the command interface that is used between the host and the Quartz AT command application. This interface, for most cases, is defined by a set of human readable commands and responses. Thus, it should be simple to use the command interface either using a terminal or using a computer program. The objectives of the command interface are to keep the host programming requirement as lightweight as possible.

This command interface draws the inspiration from the "AT" command set commonly used in POTS and Wireless modems and extends the functionality to support BT, Zigbee and Wi-Fi connectivity.

- Wi-Fi related commands begin with the "ATW" prefix
- Wireless P2P related commands begin with the "ATP2P" prefix
- Network related commands begin with the "ATN" prefix
- MQTT related commands begin with the "ATMQTT" prefix
- OTA related commands begin with the "ATOTA" prefix
- BLE related commands begin with the "ATBLE" prefix
- Zigbee related commands begin with the "ATZB" prefix
- Thread related commands begin with the "ATTH" prefix

The `ATWENABLE` command to enable Wi-Fi module is shown here:

The consecutive sections explain the commands that are available to the host to control the QCA402x chipset.

7.4 AT commands

7.4.1 General commands

7.4.1.1 ATHELP

Syntax	ATHELP=
Description	Display the help menu and syntax of available commands
Parameters	None

7.4.1.2 ATVERSION

Syntax	ATVERSION=
Description	Display the current QAPI version and CRM build version
Parameters	None

7.4.2 WLAN commands

7.4.2.1 Generic Commands

7.4.2.1.1 ATWHELP

Syntax	ATW=
Description	Displays the help menu and syntax for the WLAN AT commands
Parameters	None

7.4.2.1.2 ATWQVERSION

Syntax	ATWQVERSION=
Description	This command displays the version of WLAN software version and interface version. Interface version is currently set to 1. This command will only work after enabling wlan.
Parameters	None

7.4.2.1.3 ATWENABLE

Syntax	ATWENABLE=
Description	This command enables the WLAN module
Parameters	None

7.4.2.1.4 ATWDISABLE

Syntax	ATWDISABLE=
Description	This command disables the WLAN module
Parameters	None

7.4.2.1.5 ATWINFO

Syntax	ATWINFO=
Description	Display the Wi-Fi details
Parameters	None

7.4.2.1.6 ATWSETDEV

Syntax	ATWSETDEV=<devId>	
Description	This command is used to set the device ID (0 or 1). All device-specific WLAN commands such as info, connect, disconnect are applied for the set device ID. WLAN module uses device 0 by default.	
Parameters	<devId>	0 – Interface1 1 – Interface2
Example	ATWSETDEV=0	

7.4.2.1.7 ATWPHYMODE

Syntax	ATWPHYMODE=<mode>	
Description	This command configures the physical mode in which wlan will operate. By default device will operate in 11n if g mode is selected	
Parameters	<mode>	1 – 11a mode 2 – 11b mode 3 – 11g mode (will operate in 11n mode) 4 – 11a/g mode 5 – 11g Only mode

7.4.2.1.8 ATWSETOPMODE

Syntax	ATWSETOPMODE=<mode>,[<hidden|0>,<wps|0>]		
Description	This command is used to set operating mode of interface, either Soft-AP or Station. Hidden and WPS parameters only apply to AP mode.		
Parameters	mode	AP	Interface will work as an AP
		Station	Interface will work as a station.
Example	ATWSETOPMODE=ap ATWSETOPMODE=ap,0,wps //Enable AP in WPS mode		

7.4.2.1.9 ATWGETRGDOM

Syntax	ATWGETRGDOM =	
Description	This command is used to retrieve the current regulatory domain code	
Parameters	None	
Example	ATWGETRGDOM =	

7.4.2.1.10 ATWCCODE

Syntax	ATWCCODE=[country_code_string]	
Description	This command is used to get and set the country code. If country code is not provided the current set country code will be returned	
Parameters	Country_code_string	Country code to be set.

Example	ATWCCODE=US //to set
	ATWCCODE= //to get the current country code set	

7.4.2.1.11 ATWGETLASTERR

Syntax	ATWGETLASTERR=
Description	This command is used to query the last error information in WLAN driver.
Parameters	None
Example	ATWGETLASTERR=

7.4.2.1.12 ATWCHANNEL

Syntax	ATWCHANNEL=[<channel_num>]	
Description	This command get/set the channel on which wlan will operate. Application should ensure that the channel provided to be set complies with the regulatory domain restrictions.	
Parameters	channel_num	1 – 14 : 2.4 GHz 36 – 165: 5 GHz
Example	ATWCHANNEL=6 ATWCHANNEL=	

7.4.2.1.13 ATWSET11NHTCAP

Syntax	ATWSET11NHTCAP=<disable|ht20>	
Description	This command allows user to enable/disable 892.11n. When enable, QCA402x WLAN only supports HT20 mode of operation.	
Parameters	Disable	Disable HT
	Ht20	Enables HT
Example	ATWSET11NHTCAP=1	

7.4.2.1.14 ATWSETPOWMOD

Syntax	ATWSETPOWMOD=[<powmod>]	
Description	Two options are available for the same: max performance and power save mode.	
Parameters	<powmod>	0 – disable //Max performance 1 – enable //power save enabled
Example	ATWSETPOWMOD=1 ATWSETPOWMOD *//retrieves configured power mode*	

7.4.2.1.15 ATWTXRATE

Syntax	ATWTXRATE=[[mcs],<fix_tx_rate_to_set>]	
Description	This command is used to set and get the Tx rates.	
Parameters	[mcs]	String "mcs". To specify whether given rate is MCS rate or not. mcs parameter is valid only if user want to set mcs data rate.
	<fix_tx_rate_to_set>	This entry takes either Rate in Mbps or MCS rate index based on MCS parameter Rate in Mbps – 1,2,5,6,9,11,12,18,24,36,48,54 MCS rate Index – 0,1,2,3,4,5,6,7
Example	ATWTXRATE=9 ATWTXRATE=mcs,0 ATWTXRATE= *//retrieves last transmitted packet Tx rate*	

7.4.2.1.16 ATWSETTXPOWER

Syntax	ATWSETTXPOWER=<txpower>	
Description	Every increment here corresponds to 0.5dBm increment in power. Even though the range of 0-63 corresponds to 0-32 dBm in transmit power, the WLAN device supports only transmit power between 12-18 dBm which means the supported Tx power index is between 24-36	
Parameters	<txpower>	0 to 63
Example	ATWSETTXPOWER=16	

7.4.2.1.17 ATWGETSTATS

Syntax	ATWGETSTATS=[<counters>]	
Description	This command is used to retrieve device specific and common WLAN statistics for data, management and control traffic.	
Parameters	Counters	0 – continue counters. 1 – reset counters.
Example	ATWGETSTATS=0 ATWGETSTATS=1	

7.4.2.2 WLAN STA Commands

7.4.2.2.1 ATWSCAN

Syntax	ATWSCAN=[<mode>,[ssid]]
Description	Scan for networks, using blocking/non-blocking/non-buffering modes. If SSID is provided, scan for specified SSID only

Syntax	ATWSCAN=[<mode>,[ssid]]	
Parameters	Mode	0: Blocking mode. Application will block until the scan operation completes and scan results are received. This is the default scan mode if no input parameter is given. 1: Non-blocking scan mode. Application will not block, instead host driver will buffer the results. When the scanning procedure completes, host driver will invoke the callback registered by the application which can then retrieve the results. 2: Non-blocking, no-buffering scan mode. Application will not block and host driver will not buffer the scan results. Every BSS information received during the scan is directly given to the application as individual events and it is upto the application to manage the scan results.
	Ssid	AP ssid
Example	ATWSCAN= ATWSCAN=1 ATWSCAN=1,MyAP	

7.4.2.2.2 ATWDISCONN

Syntax	ATWDISCONN=
Description	This command is used to disconnect for currently connected AP
Parameters	None

7.4.2.2.3 ATWRSSI

Syntax	ATWRSSI=
Description	This command is used to retrieve the RSSI for the current operating device
Parameters	None

7.4.2.2.4 ATWCONN

Syntax	ATWCONN=<ap_ssid>	
Description	This command is used to associate the STA to an AP	
Parameters	Ap_ssid	AP ssid to connect to
Example	ATWCONN=MyAP	

7.4.2.2.5 ATWWEPCONN

Syntax	ATWWEPCONN=<ap_ssid>,<key_index>,<key>	
Description	This command is used to associate the STA an AP enabled with WEP security. WEP key length will be 10(WEP40)/26(WEP104) characters in Hexadeicmal or 5(WEP40)/13(WEP104) ascii characters.	
Parameters	Ap_ssid	AP ssid to connect to
	Key_index	1 – first key 2 – second key 3 – third key 4 – fourth key
	key	WEP key (5 ASCII characters)

Syntax	ATWWEPCONN=<ap_ssid>,<key_index>,<key>
Example	ATWWEPCONN=MyAP,1, abcde ATWWEPCONN=MyAP,2 abcde ATWWEPCONN=MyAP,3, abcde ATWWEPCONN=MyAP,4 abcde

7.4.2.2.6 ATWWPACONN

Syntax	ATWWPACONN=<ap_ssid>,<wpa_ver>,<ucipher>,<mcipher>,<passphrase>	
Description	This command is used to associate the STA to an AP enabled with WPA security.	
Parameters	Ap_ssid	AP SSID to connect to
	Wpa_ver	1 – WPA 2 - WPA2
	Ucipher	1 - TKIP 2 - CCMP 3 - AES
	Mcipher	1 - TKIP 2 - CCMP 3 - AES
	passpharse	Password for the AP
Example	ATWWPACONN=MyAP,2,2,2,myappass	

7.4.2.2.7 ATWWPSCONN

Syntax	ATWWPSCONN=<connect>,<mode>,[pin],[<ssid>,<mac>,<channel>]	
Description	This command is used to connect to an AP with WPS pin or push mode operation	
Parameters	connect	0 – Does not attempt to connect to AP after WPS 1 – Attempts to connect to AP after WPS
	mode	0 – push 1 – pin
	pin	Pin code 8 bytes
	SSID	AP SSID
	MAC	MAC address of AP
	Channel	Channel Number
Example	ATWWPSCONN=1,1,15507782,MyAP,112233445566,8	

7.4.2.3 Wireless SoftAP commands

7.4.2.3.1 ATWSTARTOPEN

Syntax	ATWSTARTOPEN=<ssid>	
Description	Start AP on the default or set channel with provided ssid	
Parameters	ssid	SSID with which AP is to be started
Example	ATWSTARTOPEN=mySAP	

7.4.2.3.2 ATWSTARTWPA

Syntax	ATWSTARTWPA=<ssid>,<wpa_ver>,<ucipher>,<mcipher>,<passphrase>	
Description	This command is used to start SoftAP in WPA/WPA2 mode of operation	
Parameters	ssid	ssid for SoftAP operation
	Wpa_ver	1 – WPA 2 - WPA2
	Ucipher	1 - TKIP 2 - CCMP
	Mcipher	1 - TKIP 2 - CCMP
	passpharse	Password for the AP
Example	ATWSTARTWPA=mySAP,2,2,2,mysappass	

7.4.2.3.3 ATWSTARTWEP

Syntax	ATWSTARTWEP=<ssid>,<index>,<key>	
Description	This command is used to start SoftAP in WPA/WPA2 mode of operation	
Parameters	ssid	ssid for SoftAP operation
	index	Key index[1-4]
	key	Wep key
Example	ATWSTARTWEP=mySAP,1,abcde	

7.4.2.3.4 ATWSETBCONINT

Syntax	ATWSETBCONINT=<interval>	
Description	This command is used to set the beacon interval for SoftAP operation	
Parameters	interval	100 - 1500
Example	ATWSETBCONINT=300	

7.4.2.3.5 ATWSETDTIMINT

Syntax	ATWSETDTIMINT=<interval>	
Description	This command is used to set the DTIM interval for SoftAP operation	
Parameters	interval	1 – 255
Example	ATWSETDTIMINT=2	

7.4.2.4 Wireless P2P Commands

7.4.2.4.1 ATWP2PSERVICE

Syntax	ATWP2PSERVICE
Description	Enable or disable the P2P service
Parameters	0 – disable and 1- enable

Syntax	ATWP2PSERVICE
Example	ATWP2P=0 ATWP2P=1

7.4.2.4.2 ATWP2PSETCONFIG

Syntax	ATWP2PSETCONFIG=setconfig,<go_intent>,<listen_chl>,<operating_ch>,<country>,<node_timeout>	
Description	Set the parameters that are not covered by the set command.	
Parameters	go_intent	Integer value to be set as GO intent. This GO intent value is used in P2P GO Negotiation Request/Response frames. The value ranges from 0 to15.
	listen_chl	The channel to listen during P2P finds operation. The DUT remains on this channel during LISTEN phase of the find operation for random duration. During this phase, DUT responds to probe requests from peer P2P devices.
	operating_ch	The preferred channel to operate the P2P Group
	country	No longer used. Country code needs to be set by board data file or tunables.
	node_timeout	Timeout value (in seconds) for each node. After the timeout period expires, the nodes are deleted from P2P find results.

7.4.2.4.3 ATWP2PCONNECT

Syntax	ATWP2PCONNECT=<peer_p2p_dev_mac>,<wps_method >,[pin-num],[persistent]		
Description	Initiate connection request with a given peer MAC address using given WPS configuration method.		
Parameters	peer_p2p_dev_mac	Device MAC address of the P2P peer to connect with	
	wps_method	push	Use WPS PBC method for authentication.
		display	Use WPS DISPLAY method for authentication.
		keypad	Use WPS KEYPAD method for authentication.
	pin-num	(Optional) This parameter is set only if wps_method is keypad. The pin number is used for WPS keypad authentication.	
	[persistent]	(Optional) This parameter indicates whether the start of autogo is a persistent connection. Persistent connections are remembered in a persistent media and can be reinvoked using P2P invitation request.	

7.4.2.4.4 ATWP2PFIND

Syntax	ATWP2PFIND=<channel_options>,<timeout>		
Description	Initiate P2P find operation. The find operation includes a sequence of SEARCH/LISTEN phase across the channels to look for available P2P peers.		
Parameters	channel_options	1	Scan all the channels from regulatory domain channel list.
		2	Scan only the social channels (Channel 1, 6 and 11 in 2.4GHzband). This is the default option for the find command.
		3	Continue channel scan from the last scanned channel index. Channel list is obtained from regulatory domain list.

Syntax	ATWP2PFIND=<channel_options>,<timeout>	
	timeout(in seconds)	When the timeout period expires, the find operation is stopped. Default value is 60 seconds.

7.4.2.4.5 ATWP2PWPSPROV

Syntax	ATWP2PWPSPROV=<peer_p2p_dev_mac>,<wps_method>		
Description	Provision the WPS configuration method between the DUT and the peer.		
Parameters	peer_p2p_dev_mac	Device MAC address of the P2P peer to connect with	
	wps_method	push	Use WPS PBC method for authentication.
		display	Use WPS DISPLAY method for authentication.
		keypad	Use WPS KEYPAD method for authentication.

7.4.2.4.6 ATWP2PLISTEN

Syntax	ATWP2PLISTEN=<timeout>	
Description	Initiate P2P listen process.	
Parameters	Timeout	In units of second. When the timeout period expires, the listen operation is stopped. Default value is 30 seconds.

7.4.2.4.7 ATWP2PCANCEL

Syntax	ATWP2PCANCEL=
Description	Cancel all ongoing P2P operation.
Parameters	None
Example	ATWP2PCANCEL=

7.4.2.4.8 ATWP2PJOIN

Syntax	ATWP2PJOIN=<peer_p2p_dev_mac>,<wps_method >,[pin-num],[persistent]		
Description	Join a P2P client to an existing P2P GO.		
Parameters	peer_p2p_dev_mac	Device MAC address of the P2P peer to connect with	
	wps_method	push	Use WPS PBC method for authentication.
		display	Use WPS DISPLAY method for authentication.
		keypad	Use WPS KEYPAD method for authentication.
	pin-num	Optional) This parameter is set only if wps_method is keypad. The pin number is used for WPS keypad authentication.	
	[persistent]	(Optional) This parameter indicates whether the start of autogo is a persistent connection. Persistent connections are remembered in a persistent media and can be reinvoked using P2P invitation request.	

7.4.2.4.9 ATWP2PAUTH

Syntax	ATWP2PAUTH=<peer_mac>,<wps_method >,[pin-num],[persistent]		
Description	Authenticate/Reject a connection request from a given peer MAC address using the given WPS configuration method.		
Parameters	peer_p2p_dev_mac	Device MAC address of the P2P peer to connect with	
	wps_method	push	Use WPS PBC method for authentication.

Syntax	ATWP2PAUTH=<peer_mac>,<wps_method >,[pin-num],[persistent]		
		display	Use WPS DISPLAY method for authentication.
		keypad	Use WPS KEYPAD method for authentication.
	pin-num	(Optional) This parameter is set only if wps_method is keypad. The pin number is used for WPS keypad authentication.	
	[persistent]	(Optional) This parameter indicates whether the start of autogo is a persistent connection. Persistent connections are remembered in a persistent media and can be reinvoked using P2P invitation request.	

7.4.2.4.10 ATWP2PAUTOGO

Syntax	ATWP2PAUTOGO=[persistent]	
Description	Start P2P device in Auto GO mode. This command can be used to start P2P device as a GO without the need for the GO negotiation procedure. In Auto GO mode, the default SSID is DIRECT-GO and the default passphrase is 1234567890. The default settings cannot be changed.	
Parameters	[persistent]	(Optional) This parameter indicates whether the start of autogo is a persistent connection. Persistent connections are remembered in a persistent media and can be reinvoked using P2P invitation request.

7.4.2.4.11 ATWP2PINVITE

Syntax	ATWP2PINVITE=<ssid>,<peer_p2p_dev_mac>,<wps_method>,[pin-num],[persistent]		
Description	Invite a connection from persistent database. Reinvoking a persistent connection enables quicker association because the passphrase is available at database, saving the 8-way WPS handshake.		
Parameters	ssid	SSID of the persistent connection to be reinvoked	
	peer_p2p_dev_mac	P2P device address of the peer that should be re-invoked	
	wps_method	push	Use WPS PBC method for authentication.
		display	Use WPS DISPLAY method for authentication.
		keypad	Use WPS KEYPAD method for authentication.
	[persistent]	(Optional) This parameter indicates whether the start of autogo is a persistent connection. Persistent connections are remembered in a persistent media and can be reinvoked using P2P invitation request.	

7.4.2.4.12 ATWP2PLIST

Syntax	ATWP2PLIST=
Description	Display the results of P2P find operation. This command is used after find.
Parameters	None
Example	ATWP2PLIST=

7.4.2.4.13 ATWP2PLISTPERSIST

Syntax	ATWP2PLISTPERSIST=
Description	Display the list of persistent P2P connections that are saved in the persistent media.
Parameters	None
Example	ATWP2PLISTPERSIST=

7.4.2.4.14 ATWP2PSETOPPS

Syntax	ATWP2PSETOPPS=<enable>,<ctwin>	
Description	Enable and set the parameters of OppPS for GO.	
Parameters	Enable	Enable or disable OppPS feature for GO.
	Ctwin	CTWindow parameter in OppPS for GO, in units of TU (1024 microseconds).

7.4.2.4.15 ATWP2PSETNOA

Syntax	ATWP2PSETNOA=<count> <start> <duration> <interval>	
Description	Enable and set the parameters of NoA for GO.	
Parameters	count	1-255. Indicate the number of absence interval for GO.
	start	In units of millisecond. Indicate the offset time after the next TBTT,to calculate the absolute start time of GO's first absence period.
	duration	Indicates the maximum duration in units of milliseconds that the P2P GO can remain absent in one NoA interval.
	interval	Indicates the length of the Notice of Absence interval in units of milliseconds.

7.4.2.4.16 ATWP2POPERATINGCLASS

Syntax	ATWP2POPERATINGCLASS=<go_intent>,<oper_reg_class>,<oper_reg_channel>	
Description	Set operating class parameters	
Parameters	Go_intent	Integer value to be set as GO intent. This GO intent value is used in P2P GO Negotiation Request/Response frames. The value ranges from 0 to15.
	oper_reg_class	Integer value. Operating register class
	oper_reg_channel	Integer Value. The preferred channel to operate the P2P

7.4.2.4.17 ATWP2PSET

Syntax	ATWP2PSET=<p2p_mode>,[<postifx>,<intrabss>,<gointent>,<cckrates>]	
Description	Set operating class parameters	
Parameters	p2p_mode	1-p2pdev 2-p2pclient 3-p2pgo
	postfix	Postfix_string
	intrabss	Flags for the intrabss
	Go_intent	In units of second. When the timeout period expires, the listen operation is stopped. Default value is 30 seconds.
	cckrates	1-enable 0-disable

7.4.2.4.18 ATWP2PSTOPFIND

Syntax	ATP2PSTOPFIND=	
Description	Stop P2P operation	
Parameters	None	

7.4.2.4.19 ATWP2PPASSPHRASE

Syntax	ATWP2PPASSPHRASE=<passphrase>,<SSID>	
Description	Set P2P passphrase	
Parameters	passphrase	String value – security passphrase
	SSID	String value – SSID of the network

7.4.2.5 Wi-Fi OTA commands

7.4.2.5.1 ATWOTAFWD

Syntax	ATWOTAFWD=
Description	This command displays active Firmware Descriptor (FWD) index number and displays all Firmware Descriptor at flash.
Parameters	None
Example	ATWOTAFWD=

7.4.2.5.2 ATWOTADEL

Syntax	ATWOTADEL=<fwd no.>		
Description	This command erases the FWD. After this command, the image space is available for download new image.		
Parameters	<fwd no.>	0/1/2	Indicates which FWD to be deleted
Example	ATWOTADEL=0		

7.4.2.5.3 ATWOTATRIAL

Syntax	ATWOTATRAIL=<accept>,<reboot>	
Description	This command accepts or rejects the trial image. After this command, system will reboot based on [reboot] flag.	
Parameters	accept	Indicates if accept or reject the trial image. 0: reject the trial image; 1: accept the trial image.
	reboot	Indicates if system reboot after this command. 0 : no reboot need, 1: reboot
Example	ATWOTADEL=0,0 ATWOTADEL=1,1	

7.4.2.5.4 ATWOTAIMG

Syntax	ATWOTAIMG=<ID>
Description	This command displays first 16 bytes of image at current active partition.

Syntax	ATWOTAIMG=<ID>	
Parameters	ID	0-255 Indicates which image content to display. 0: all images to display, other: image to display.
Example	ATWOTAIMG=0 ATWOTAIMG=3	

7.4.2.5.5 ATWOTAFTP

Syntax	ATWOTAFTP=<if_name>,<ftp_param>,<file_name>,<flags>	
Description	This command connects to FTP server and does Firmware Upgrade.	
Parameters	If_name	String value. Network device name. Example: wlan1
	ftp_param	String value: This parameter includes Firmware Upgrade FTP related info. The format is [user]:[pwd]@[[ipv4]|[ipv6|]:[port]. It includes FTP User Name, password, FTP Server IP Address, and FTP Server Port Number For example, user:password@192.168.1.108:21
	File_name	String value: This parameter is the firmware upgrade config file. For example, ota.bin
	flags	Bitmask Bitmask of flags for Firmware Upgrade FTP: Bit0: Reboot after operate is done Bit1: Duplicate File System from Current to Trial Images
Example	ATWOTAFTP=wlan0,test_user:password@10.14.0.21 ota.bin,1	

7.4.2.5.6 ATWOTAHTTP

Syntax	ATWOTAHTTP=<if_name>,<ftp_param>,<file_name>,<flags>	
Description	This command connects to HTTP server and does Firmware Upgrade.	
Parameters	If_name	String value. Network device name. Example: wlan1
	http_param	String value: This parameter includes Firmware Upgrade HTTP info. The format is <timeout>:<server>:<port>/<url>. It includes HTTP connection timeout, HTTP Server Address, HTTP Server Port Number, and HTTP URL. Example:. 100:192.168.1.108:80/fireware-download
	File_name	String value: This parameter is the firmware upgrade config file. For example, ota.bin
Example	ATWOTAHTTP=wlan0, 100:192.168.1.108:80/fireware-download ota.bin	

7.4.3 Networking commands

7.4.3.1 ATNHELP

Syntax	ATNHELP=
Description	Displays the help menu and syntax for the Networking AT commands
Parameters	None

7.4.3.2 ATNPING

Syntax	ATNPING= <host>,[<count>],[<size>]	
Description	This command sends a ping request to a host	
Parameters	host	IPv4 address
	Count	1-2^31: amount of ping requests
	size	1-1536: ping payload size in bytes

7.4.3.3 ATNIFCONFIG

Syntax	ATNIFCONFIG=[<interface_name>],[<Ipv4addr>],[<subnetmask>],[<default_gateway>]	
Description	Displays or configures the network interfaces	
Parameters	Interface_name	IPv4 address
	Ipv4addr	1-2^31: amount of ping requests
	subnetmask	x.x.x.x: subnet mask
	Defaul_gateway	x.x.x.x: Default gateway (optional)

7.4.3.4 ATNDHCPV4C

Syntax	ATNDHCPV4C=[<interface_name>],[new|release]	
Description	Command acquires a valid IPv4 address from a DHCPv4 server	
Parameters	Interface_name	IPv4 address
	New	New DHCP request (get an IPv4 address)
	Release	Release a DHCP IPv4 address

7.4.3.5 ATNAUTOIPV4

Syntax	ATNAUTOIPV4=[<interface_name>]	
Description	Generates a link-local IPv4 address where a DHCPv4 server is not available	
Parameters	Interface_name	IPv4 address

7.4.3.6 ATNPING6

Syntax	ATNPING6= <host>,[<count>],[<size>],[<interface_name>]	
Description	This command sends a ping (IPv6) request to a host	
Parameters	host	IPv6 address
	Count	1-2^31: amount of ping requests
	size	1-1536: ping payload size in bytes
	Interface_name	Interface name, must be specified if address is link-local. Ex: wlan0

7.4.3.7 ATNDHCPV6C

Syntax	ATNDHCPV6C=[<interface_name>],[enable|new|confirm|disable|release]	
Description	Command acquires a valid IPv6 address from a DHCPv6 server	
Parameters	Interface_name	IPv4 address
	enable	Enable DHCPv6 service on an interface
	new	Acquire a global IPv6 address from server
	confirm	Confirm current IPv6 address
	Disable	Disable DHCPv6 service
	Release	Release a DHCP IPv6 address
Example	ATNDHCPV6C=wlan0,enable	

7.4.3.8 ATNSNTPC

Syntax	ATNSNTPC=[start|stop|disable] ATNSNTPC=[addsrv <server> [<id>]] ATNSNTPC=[delsrv <id>] ATNSNTPC=[utc]		
Description	Command enables time of day acquisition from SNTP server		
Parameters	Start	Start SNTP client service	
	Stop	Stop SNTP client service	
	Disable	Disable SNTP client service	
	Addsrv	Ipv4/IPv6 address	Add a server to the server list
	Delsrv	ID - Deleted a server from the list	
	Utc	Show UTC time	

7.4.3.9 ATNDNSC

Syntax	ATNDNSC=[start|stop|disable] ATNDNSC=addsrv <server> [<id>] ATNDNSC=delsrv <id> ATNDNSC=[resolve <host> [v4|v6]]	
Description	Dynamic name resolution service	
Parameters	Start	Start DNS client service
	Stop	Stop DNS client service
	Disable	Disable DNS client service
	Addsrv	Add a server to the server list
	Delsrv	Deleted a server from the list
	resolve	Resolve a host name to an IPv4 or Ipv6 address

7.4.3.10 ATNBRIDGE

Syntax	ATNBRIDGE=[enable|disable] ATNBRIDGE=[mac_entry_age <val>] ATNBRIDEG=[showmacs]		
Description	Performs the WLAN bridge operation		
Parameters	enable	Enable WLAN bridge: bridge traffic between wlan0 and wlan1 interfaces in Layer2	
	disable	Disable the WLAN bridge	
	Mac_entry_age	1-3600	Configure aging of bridged addresses in seconds
	showmacs	Displays all bridged addresses	
Example	ATNBRIDGE=enable ATNBRIDGE=1000		

7.4.3.11 ATNCLOSE

Syntax	ATNCLOSE=
Description	Close a IPv4 socket
Parameters	None
Example	ATNCLOSE=

7.4.3.12 ATNCTCP

Syntax	ATNCTCP=<port>,<ip_address>,<per_packet_bytes>,<timeout>	
Description	TCP client connects to the server	
Parameters	Port	Server port number
	ip_address	IPv4 address
	Per_packet_bytes	Number of bytes sent per packet
	timeout	TCP client is alive for this timeout (in seconds)
Example	ATNCTCP=1234, 192.168.1.100, 1024,10	

7.4.3.13 ATNCTCPV6

Syntax	ATNCTCPV6=<port>,<ipv6_address>,<per_packet_bytes>,<timeout>	
Description	IPV6 TCP client connects to the IPV6 TCP server	
Parameters	Port	Server port number
	ip_address	IPv6 address
	Per_packet_bytes	Number of bytes sent per packet
	timeout	TCP client is alive for this timeout (in seconds)
Example	ATNCTCPV6=1234, fe80::9d37:33b:be06:32, 1024,10	

7.4.3.14 ATNCUDP

Syntax	ATNCUDP=<port>,<ip_address>,<per_packet_bytes>,<timeout>
Description	UDP client connects to the server

Syntax	ATNCUDP=<port>,<ip_address>,<per_packet_bytes>,<timeout>	
Parameters	Port	Server port number
	ip_address	IPv4 address
	Per_packet_bytes	Number of bytes sent per packet
	timeout	TCP client is alive for this timeout (in seconds)
Example	ATNCUDP=1234, 192.168.1.100, 1024,10	

7.4.3.15 ATNCUDPV6

Syntax	ATNCUDPV6=<port>,<ipv6_address>,<per_packet_bytes>,<timeout>	
Description	IPV6 UDP client connects to the IPV6 UDP server	
Parameters	Port	Server port number
	ip_address	IPv6 address
	Per_packet_bytes	Number of bytes sent per packet
	timeout	TCP client is alive for this timeout (in seconds)
Example	ATNCUDPV6=1234, fe80::9d37:33b:be06:32, 1024,10	

7.4.3.16 ATNTCPSERVER

Syntax	ATNTCPSERVER=<port>,<timeout>,[echo]	
Description	Start the TCP server	
Parameters	Port	port number in which server listens
	Timeout	Timeout value of server
	echo	Echoes back the data from the client
Example	ATNTCPSERVER=1234,10,echo ATNTCPSERVER=1234,10	

7.4.3.17 ATNTCPV6SERVER

Syntax	ATNTCPV6SERVER=<port>,<timeout>,[echo]	
Description	Start the IPV6 TCP server	
Parameters	Port	port number in which server listens
	Timeout	Timeout value of server
	echo	Echoes back the data from the client
Example	ATNTCPV6SERVER=1234,10,echo ATNTCPV6SERVER=1234,10	

7.4.3.18 ATNUDPSERVER

Syntax	ATNUDPSERVER=< port>,<timeout>,[echo]	
Description	Start the UDP server	
Parameters	Port	port number in which server listens
	Timeout	Timeout value of server
	echo	Echoes back the data from the client

Syntax	ATNUDPSERVER=< port>,<timeout>,[echo]
Example	ATNUDPSERVER=1234,10,echo ATNUDPSERVER=1234,10

7.4.3.19 ATNUDPV6SERVER

Syntax	ATNUDPV6SERVER=< port>,<timeout>,[echo]	
Description	Start the IPV6 UDP server	
Parameters	Port	port number in which server listens
	Timeout	Timeout value of server
	echo	Echoes back the data from the client
Example	ATNUDPV6SERVER=1234,10,echo ATNUDPV6SERVER=1234,10	

7.4.3.20 ATNDHCPV4S

Syntax	ATNDHCPV4S=<interface_name>,<pool|stop>,(<start_ip><end_ip>)[<lease_time_sec>|infinite]	
Description	Starts the DHCP server on SAP and assigns IP address to the clients from the pool (start IP to End IP). Also provides the lease time for the clients to hold the IP address	
Parameters	Interface_name	Name of the WLAN Interface
	pool	Range of IP addresses from which the IP addresses are leased to the clients
	stop	Stops the DHCP server
	Start_ip	Start IP address
	End_ip	End IP address
	Leasetime/infinite	lease time for the clients to hold the IP address
Example	ATNDHCPV4S=wlan0,pool,192.168.0.2,192.168.0.20,5000	

7.4.3.21 ATNMSOCKCREATE

Syntax	ATNMSOCKCREATE =<protocol>,<num_of_sockets>,<server_IP>,<port_num>,<time_msec>,<num_of_bytes>	
Description	This command supports the creation of multi-socket for TCP/UDP protocols, where user can create multiple sockets and trans-receive the data over the sockets created simultaneously.	
Parameters	protocol	TCP or UDP
	Num_of_sockets	Number of sockets to create
	Server_Ip	Ip address of the server
	Port_num	Sever port number to connect
	Time_ms	Timer interval to transfer the data By Default 1000 ms [Optional Parameter]
	Num of bytes	Number of bytes to transmit. By default 512 bytes [Optional Parameter]
Example	ATNMSOCKCREATE=192.168.0.1 6000 2000 1000	

7.4.3.22 ATNMSOCKCLOSE

Syntax	ATNMSOCKCLOSE=
Description	This command closes the multi-socket sessions for TCP and UDP protocols which are created.
Parameters	None
Example	ATNMSOCKCLOSE= // closes all the existing sessions with the server

7.4.3.23 ATNHTTPC

Syntax	ATNHTTPC=<start> ATNHTTPC=<stop> ATNHTTPC=<connect>,[<server>,<port>,<ssl-index>] ATNHTTPC=<disc>,[<client_num>] ATNHTTPC=<get>,[<client_num>,<url>] ATNHTTPC=<put>,[<client_num>,<url>] ATNHTTPC=<post>,[<client_num>,<url>] ATNHTTPC=<patch>,[<client_num>,<url>] ATNHTTPC=<setbody>,[<client_num>,<len>] ATNHTTPC=<addheader>,[<client_num>,<hdr_name>,<hdr_value>] ATNHTTPC=<clearheader>,[<client_num>] ATNHTTPC=<setparam>,[<client_num>] ATNHTTPC=<setparam>,[<client_num>] ATNHTTPC=<config>,[<httpc_demo_max_body_len>,<httpc_demo_max_header_len>]
Description	Configures the HTTP client at the run time

Syntax	ATNHTTPC=<start> ATNHTTPC=<stop> ATNHTTPC=<connect>,[<server>,<port>,<ssl-index>] ATNHTTPC=<disc>,[<client_num>] ATNHTTPC=<get>,[<client_num>,<url>] ATNHTTPC=<put>,[<client_num>,<url>] ATNHTTPC=<post>,[<client_num>,<url>] ATNHTTPC=<patch>,[<client_num>,<url>] ATNHTTPC=<setbody>,[<client_num>,<len>] ATNHTTPC=<addheader>,[<client_num>,<hdr_name>,<hdr_value>] ATNHTTPC=<clearheader>,[<client_num>] ATNHTTPC=<setparam>,[<client_num>] ATNHTTPC=<setparam>,[<client_num>] ATNHTTPC=<config>,[<httpc_demo_max_body_len>,<httpc_demo_max_header_len>]		
Parameters	Start	Start the HTTP client	
	Stop	Stop the HTTP client	
	Connect	Connect to the HTTP server, return if any error	
		server	Domain name / IP address of the HTTP server
		port	Port number (optional). Default port number is 80
		ssl_index	SSL index of SSL context if connecting to HTTPS server
	Disc	Disconnect the HTTP server and close the HTTP client session	
	Get	Request a page from the server	
		client_num	Page name to request
	Put	Send a PUT request to the HTTP server.	
		client_num	Page number to request
	Post	Send a POST request to the HTTP server	
		client_num	Page number to request
	Patch	Send a PATCH request to the HTTP server.	
		client_num	Page number to request
	Setbody	set customer http client body buffer	
		client_num	Page number to request
	Addheader	Add custom http client header buffer	
		client_num	Page number to request
		Hdr_name	Header name to request
		Hdr_value	Header value to request
	Clearheader	Clear custom http client header buffer	
		client_num	Page number to request
	Setparam	Set additional parameters	
		client_num	Page number to request
		key	Key to set
		value	Value to set
	Config	Configure the HTTP body and header length	
		http_demo_ Max_body_Len	Maximum HTTP body length to configure
		http_demo_ Max_header_Len	Maximum HTTP header length to configure

7.4.3.24 ATNHTTPS

Syntax	ATNHTTPS=[init <v4|v6|v46>,<http|https|http_https>,<cert_file>],[<httpport>] [<httpsport>],[<ifname>],[<index_page>],[<root_path>]] ATNHTTPS=<start> ATNHTTPS=<stop> ATNHTTPS=<setbufsize>,<TX_buffer_size>,<RX_buffer_size> ATNHTTPS=<addctype>,[<content-type1>,[<content-type2> ..] ATNHTTPS=<delctype>,[<content-type1>,[<content-type2> ..] ATNHTTPS=<sslconfig> [<keyword_1> <value_1> <keyword_1> <value_1> …]		
Description	Configures the HTTP server at the run time		
Parameters	Init	Initialize the HTTP server	
		V4/v6/v46	Ipv4 or IPv6 or both IPv4 and IPv6
		http/https/ http_https	HTTP or HTTPS or support both HTTP and HTTPS
		Httpport	HTTP Port number to connect
		httpsport	HTTPS Port number to connect
		Ifname	Interface name to connect
		Index_page	Index page number to connect
		Root_path	Root path to connect
	Start	Start the HTTP server	
	Stop	Stop the HTTP server	
	Setbufsize	Set the buffer size	
		Tx buffer size	Transmit buffer size in bytes
		Rx buffer size	Receive buffer size in bytes
	Addctype	Add content type to HTTP server	
		Content_type	Indicate the media type of resource
	Delctype	delete content type to HTTP server	
		Content_type	Indicate the media type of resource
	Ssl_config	SSL configuration to the HTTP server	
		Keyword	Place holder of the key
		Value	Value of the place holder
		client_num	Page number to request

7.4.3.25 ATNSSL

Syntax	ATNSSL=<start>,[<server|client>] ATNSSL=<stop>,[<server|client>] ATNSSL=<config>,[<server|client>],[<keyword_1>,<value_1>], [<keyword_2>,<keyword_2>]..... ATNSSL=<cert>,[<server>|<client>],<certificate|calist>,<name> ATNSSL=<psk>,[<server>|<client>],<identity>,<psk> ATNSSL=<ecjpake>,[<server>|<client>],<password> ATNSSL=<max_clients> ATNSSL=<idle_timer>,<idle_timout>	
Description	This command performs the SSL operations	
Parameters	start	<server | client>: Configure ssl instance as a client or server
	stop	<server | client>: Stop the ssl session and reset the client or server instance.
	config	<server | client>: Configure the ssl client or server instance. With the following parameters. protocol <protocol> : select protocol: TLS (default), TLS1.0, TLS1.1, TLS1.2, DTLS, DTLS1.0, DTLS1.2 time 0|1 : disable|enable certificate time validation (optional) domain 0|<name>: disable|enable validation of peer's domain name against <name> alert 0|1 : disable|enable sending of SSL alert if certificate validation fails. cipher <cipher>: select <cipher> (enum name) suite to use, can be repeated 8 times (optional) max_frag_len <bytes>: max fragment length in bytes neg_disable 0|1: disable|enable maximum fragment length negotiation sni <name>: configure name for server name indication (SNI) alpn <protocol_name: specify protocol name for ALPN, for example, "h2", "http/1.1"
	cert	<server|client>: add a certificate or CA list to either SSL server or client. <certificate|calist>: Certificate or CA list. <name>: Name of the file to load from secure storage.
	psk	<server|client>: This creates a psk table for either the SSL server or client. <identity>: Identity <psk>: psk
	ecjpake	<server|client>: Sets the password for ECJPAKE cipher suite for either client or server. <password>: Password to be set.
	Max_clients	<max_Clients>: Sets DTLS maximum number of clients per server connection. Defaults to 1 if not set.
	Idle_timer	<idle_timeout>: For DTLS servers, sets the number of seconds to wait before closing an idle client connection

7.4.4 MQTT commands

7.4.4.1 ATMQTTHELP

Syntax	ATMQTTHELP=	
Description	Displays list of MQTT commands	
Example	ATMQTTHELP=	

7.4.4.2 ATMQTTCINIT

Syntax	ATMQTTINIT=[ca_file]	
Description	Initialize the MQTT service	
Parameters	ca_file	Encrypted support is provided by using the certificate based
Example	ATMQTTCINIT=	

7.4.4.3 ATMQTTCSHUT

Syntax	ATMQTTCSHUT=
Description	shutdown MQTT service
Parameters	None
Example	ATMQTTCSHUT=

7.4.4.4 ATMQTTCNEW

Syntax	ATMQTTCNEW=[<client_id>,<is_clean_session >]	
Description	Creates a MQTT session, taking the client name and the session ID	
Parameters	Client_id	Client ID for creating a session. This is of type string format
	is_clean_session	1 – for clean session
Example	ATMQTTCNEW=test 1	

7.4.4.5 ATMQTTCDESTROY

Syntax	ATMQTTCDESTROY=<session_ID>	
Description	Destroys the MQTT session, taking the session ID as the parameter for which session to destroy	
Parameters	Session_ID	Session number for the session to be deleted. Valid positive number for which session is already created
Example	ATMQTTCDESTROY= 1	

7.4.4.6 ATMQTTCCONFIG

Syntax	ATMQTTCCONFIG=<user_name>,<password>,[<topic>,<message>,<qos>]	
Description	This command configures the MQTT client parameters like username, password, topic, message, QOS	
	user_name	Any valid username. Generally, a string of characters
	Password	Valid password. Generally, a string of characters
	topic	Valid topic name. Generally, a string of characters

Syntax	ATMQTTCCONFIG=<user_name>,<password>,[<topic>,<message>,<qos>]	
	message	Valid message. Generally, a string of characters
	qos	QoS: 0/1/2
Example	ATMQTTCCONFIG=test,password ATMQTTCCONFIG=test,password,wheather,hello_message,2	

7.4.4.7 ATMQTTCCONNECT

Syntax	ATMQTTCCONNECT=<session_ID>,<server_IP>,[<ss>,<nb>,<keep_alive>,<wait_time>,<interface_name>]	
Description	Connect to the MQTT server	
Parameters	session_ID	Session number which is already created
	Server_IP	The URL on the server side
	ss	1. Enable Secure session connection, 0 –disable Secure session connection
	nb	2. Enable Non-blocking connect, 0 – disable Non-blocking connect
	Keep_alive	The keep-alive interval, in unit of seconds
	Wait_time	Wait time till “connack” is received
	Interface_name	An interface name which is bind for the server_IP
Example	ATMQTTCCONNECT=1,192.168.43.245,1,1,60,5,wlan0 ATMQTTCCONNECT=1,192.168.43.245	

7.4.4.8 ATMQTTCSUB

Syntax	ATMQTTCSUB=<session_ID>,<topic_filter>,[<qos>]	
Description	Subscribe the topic	
Parameters	Session_ID	Session number which is already created
	Topic_filter	The topic string. This is a string value
	qos	QoS: 0/1/2
Example	ATMQTTCSUB=1,hello,1	

7.4.4.9 ATMQTTCPUB

Syntax	ATMQTTCSUB=<session_ID>,<topic_filter>,[<qos>],[<message>,<retain>,<duplicate>]	
Description	Publish the message about some certain topics	
Parameters	Session_ID	Session number which is already created
	Topic_filter	The topic string. This is a string value
	qos	QoS: 0/1/2
	message	Message. Typically strings, can be alphanumeric
	retain	1-retained, 0 – non retained
	duplicate	1-duplicate, 0 – non duplicate
Example	ATMQTTCSUB=1,hello,1,”Hello world”,0,0	

7.4.4.10 ATMQTTCUNSUB

Syntax	ATMQTTCUNSUB=<session_ID>,<topic_filter>	
Description	Un-subscribe from a topic	
Parameters	Session_ID	Session number which is already created
	Topic_filter	The topic string. This is a string value
Example	ATMQTTCUNSUB=1,hello	

7.4.4.11 ATMQTTCDISCONNECT

Syntax	ATMQTTCDISCONNECT=<session_ID>	
Description	Disconnect from the MQTT server	
Parameters	Session_ID	Session number which is already created
Example	ATMQTTCDISCONNECT=1	

7.4.4.12 ATMQTTCSSLCONFIG

Syntax	ATMQTTCSSLCONFIG=<session_ID>[<keyword_1>,<value_1>], [<keyword_2>,<keyword_2>].....		
Description	For configuring the SSL server/client parameters		
Parameters	<session_ID>	Session number which is already created	
	<keyword>	**<value>**	**description**
	Protocol	<protocol>	TLS(default), TLS1.0, TLS1.1, TLS1.2, DTLS, DTLS1.0, DTLS1.2
	Time	0|1	Disable|enable certificate time validation (optional)
	Domain	0|<name>	Disable|enable validation of peer's domain name against <name>
	Alert	0|1	Disable|enable sending of SSL alert if certificate validation fails
	Cipher	<cipher>	Select <cipher> suite to use, can be repeated 8 times (optional)
	Max_frag_len	<bytes>	Max fragment length in bytes
	Neg_disable	0|1	Disable|enable maximum fragment length negotiation
	sni	<name>	Configure name of server name indication (SNI)
	alpn	<protocol>	Specify protocol name for ALPN. Example h2, http/1.1
Example	ATMQTTCSSLCONFIG=1,alert,0,domain,0....		

7.4.5 BLE commands

7.4.5.1 Generic commands

7.4.5.1.1 ATBLEHELP

Syntax	ATBLEHELP=
Description	Displays list of BLE generic commands
Example	ATBLEHELP=

7.4.5.1.2 ATBLESERVICE

Syntax	ATBLESERVICE=start|stop	
Description	Start command initializes the Bluetooth Protocol stack and must be called before any command. If the initialization is successful, command prints initialization information of Bluetooth Protocol stack. Also prints the version of Bluetooth Controller Stop command shuts down the Bluetooth Protocol Stack.	
Parameters	start	Initialize the Bluetooth Protocol Stack
	stop	Shutdown the Bluetooth Protocol stack
Example	ATBLESERVICE=start ATBLESERVICE=stop	

7.4.5.1.3 ATBLESETRADIO

Syntax	ATBLESETRADIO=<radio>	
Description	This command selects the BLE radio that used by the Bluetooth Protocol Stack	
Parameters	radio	1-2 : Selects the BLE radio
Example	ATBLESETRADIO=1 ATBLESETRADIO=2	

7.4.5.1.4 ATBLESETDISCCONN

Syntax	ATBLESETDISCCONN=discover,<mode> ATBLESETDISCCONN=connect,<mode>			
Description	This command sets the GAP LE Discoverability Mode that will be used when the local device is advertising. By default, GAP LE Discoverability Mode is set to 'General Discoverable' when the Bluetooth Protocol Stack is initialized. This command also sets the GAP LE Connectability Mode that will be used when the local device is advertising. By default, GAP LE Connectability Mode is set to 'Connectable' when the Bluetooth Protocol Stack is initialized			
Parameters	discover	<mode>	0 - Non – discoverable	sets the GAP LE Discoverability Mode
			1- Limited discoverable	
			2 - General Discoverable	
	connect	<mode>	0 - Non – connectable	Sets the GAP LE Connectability Mode
			1 - Connectable	
			2 - Direct Connectable	
Example	ATBLESETDISCCONN=discover,2 ATBLESETDISCCONN=connect,1			

7.4.5.1.5 ATBLESETPAIR

Syntax	ATBLESETPAIR=<mode>	
Description	This command sets the GAP LE Pairability Mode that will be used when the local device is advertising. By default, GAP LE Pairability Mode is set to 'Pairable w/Secure Connections' when the Bluetooth Protocol Stack is initialized.	
Parameters	<mode>	0 - Non – discoverable
		1- Pairable
		2 – Pairable with security connections
Example	ATBLESETPAIR=2	

7.4.5.1.6 ATBLECHNGPAIR

Syntax	ATBLECHNGPAIR=<IO_capability>[yes|no],<MITM>,<secure_connection>		
Description	This command sets GAP LE Pairing Parameters that will be used during pairing. By default, GAP LE I/O Capability is set to 'No Input/Output' when the Bluetooth Protocol Stack is initialized. By default, GAP LE MITM requirement is set to 'Yes' when the Bluetooth Protocol Stack is initialized. By default, GAP LE Secure Connections requirement is set to 'Yes' when the Bluetooth Protocol Stack is initialized. For pairing of secure connections to take place, both the local and remote devices must support Bluetooth 4.2 and the GAP LE Pairability Mode must be set to 'Pairably w/Secure Connections'.		
Parameters	<IO_capability>	0 – Display Only	Sets the GAP LE Input/Output capability of the local device
		1- Display (Yes/No)	
		2 – Keyboard only	
		3- Direction (Yes/No), yes – input and no - output	
		4 – interface (Yes/No), yes – keyboard no - display	
	<MITM>	0 –no, 1 - yes	Sets if Man in the Middle (MITM) is required during pairing
	<secure_connection>	0-no, 1- yes	Sets if secure connections is required during pairing
Example	ATBLECHNGPAIR=3,0,1		

7.4.5.1.7 ATBLEPASSKEYRESP

Syntax	ATBLEPASSKEYRESP=<pass_key>		
Description	This command responds to passkey request		
Parameters	<pass_key>	(000000-999999)	the GAP LE passkey

7.4.5.1.8 ATBLEQUERYENCRYPT

Syntax	ATBLEQUERYENCRYPT=<none>
Description	Queries the encryption state of BLE connection
Parameters	None
Example	ATBLEQUERYENCRYPT=

7.4.5.1.9 ATBLEPASSKEY

Syntax	ATBLEPASSKEY=<mode>,<pass_key>		
Description	This command Sets fixed passkey to use when IO capabilities are display only		
Parameters	<mode>	0 – clear	Clears the current GAP LE Passkey or sets the GAP LE Passkey.
		1- Set	
	<pass_key>	(000000-999999)	Sets the GAP LE passkey that will be used during pairing. This is required if the 'Clear/Set' parameter is set to '1'.
Example	ATBLEPASSKEY=1,123456		

7.4.5.1.10 ATBLEGETLOCADDR

Syntax	ATBLEGETLOCADDR=
Description	This command queries the Bluetooth address of the local Bluetooth controller.
Parameters	None
Example	ATBLEGETLOCADDR=

7.4.5.1.11 ATBLEADV

Syntax	ATBLEADV=<disable|enable>,[<BD_ADDR>]		
Description	This command enables/disables GAP LE Advertising. If advertising is enabled, this command can be used to enable un-directed or directed advertising. The GAP LE Connectability Mode must be set to 'Direct Connectable' to use direct advertising. If this is NOT the case, then this command will revert to un-direct advertising. If the optional Direct BD_ADDR is specified and the preceding requirements for directed advertising are met, then this command enables the directed advertising.		
Parameters	disable	0 – disable, Disable the advertising	
	enable	1 – enable the channels 37 and 38	
		2 - enable the channels 37	
		3- enable the channels 38	
		4- enable the channels 39	
	<BD_ADDR>	Direct BD Address - (0x000000123456)	The Bluetooth address of the target remote device that will receive directed advertisements. This is optional
Example	ATBLEADV=0 ATBLEADV=1, 000000123456 ATBLEADV=4, 000000123456		

7.4.5.1.12 ATBLESCAN

Syntax	ATBLESCAN=<enable|disable>,<Filter_policy>		
Description	This command enables/disables GAP LE Scanning. This command will use the Filter Policy of 'No Filter' if the Filter Policy is not optionally specified by the Filter Policy parameter. If the Filter Policy parameter is specified, then it may be used to filter advertisements based on remote devices that have been added to the White List and/or Resolving List, in the Bluetooth controller.		
Parameters	<enable|disable>	1 – enable, 0 - disable	Enables or disables un-directed or directed GAP LE Scanning.
	<Filter_policy>	0-No filter 1-White List 2-No white list directed RPA 3-White list directed RPA	The Filter Policy that will be used to filter advertisement. This is optional (see the following). RPA – Resolvable Private Address.
Example	ATBLESCAN=1,1		

7.4.5.1.13 ATBLECONN

Syntax	ATBLECONNLE=connect|disconnect|cancel,[<white_list>,<BD_ADDR>,<ADDR_type>]			
Description	This command sends a GAP LE Connection/ disconnection request to a remote device. If the white list parameter is enabled, then the BD_ADDR and ADDR Type parameters will not be used. The first Bluetooth address of a remote device, found in the White List, in the Bluetooth controller, will receive the connection request. If the White List parameter is disabled, the BD_ADDR and ADDR Type parameters must be used to specify the remote device that will receive the connection request. The following GATT-based services: AIOS, BAS, HOGP, SCPS, and SPPLE cannot be registered/un-registered after a connection has been established with a remote device.			
Parameters	connect	<white_list>	1 – enable, 0 - disable	Enables or disables using the White List in the Bluetooth Controller for connecting to a remote device.
		<BD_ADDR >	Bluetooth Address (0x000000000000)	The Bluetooth address of the remote device that will receive the connection request. This is optional
		< ADDR_type>	0 = Public 1 = Random 2 = Public Identity 3 = Random Identity	The Bluetooth address type of the remote device that will receive the connection request. This is optional
	disconnect	Disconnect the active LE connection		
	cancel	Cancels the active LE connection		
Example	ATBLECONNLE=connect,1, 000000123456,1 ATBLECONNLE=disconnect ATBLECONNLE=cancel			

7.4.5.1.14 ATBLEDSERVICE

Syntax	ATBLEDSERVICE=
Description	This command starts a process to discover all GATT services on the selected remote device. A GATT service must be discovered before read/write requests may be issued for the Characteristics and Descriptors of a service.
Parameters	None
Example	ATBLEDSERVICE=

7.4.5.1.15 ATBLEPAIR

Syntax	ATBLEPAIR=<value>	
Description	This command starts pairing/unpairing process with the active BLE connection	
Parameters	0	Unpairs the active BLE connection
	1	Pairs the active BLE connection

Example	ATBLEPAIR=1 // Pairs the BLE connection
	ATBLEPAIR=0 // Unpairs the BLE connection	

7.4.5.1.16 ATBLEREMDEV

Syntax	ATBLEREMDEV=[<GATT_Connection_ID>] ATBLEREMDEV=		
Description	This command explicitly sets a connected remote device as the selected remote device. The selected remote device is used by many BLE commands as the target remote device since multiple remote devices may be connected. This command displays information about each remote device that is stored in the remote device information.		
Parameters	<GATT_connection_ID >	Positive non-zero number. The GATT Connection ID for the connection with the remote device that will be set as the selected remote device.	

Examples	ATBLEREMDEV=7 // sets remote device with ID 7 as the default remote device
	ATBLEREMDEV= // displays all the remote devices		

7.4.5.1.17 ATBLEDEVWLIST

Syntax	ATBLEDEVWLIST=<BD_ADDR>,[<Addr_type>] ATBLEDEVWLIST=<BD_ADDR>		
Description	This command adds/removes a remote device to/from the White List in the Bluetooth controller.		
Parameters	<BD_ADDR >	The Bluetooth address of the remote device that will be added to the White List in the Bluetooth controller.	
	<Addr_type>	0-public	The Bluetooth address type of the Bluetooth address that will be added to the White List in the Bluetooth controller.
		1-Random	
		2-Public Identity	
		3-Random Identity	

Examples	ATBLEDEVWLIST=798701F901FA 1 // adds remote device to white list
	ATBLEDEVWLIST=798701F901FA // removes remote device from resolve list		

7.4.5.1.18 ATBLEDEVRESOLVLIST

Syntax	ATBLEDEVRESOLVLIST=<val>,<BD_ADDR> ATBLEDEVRESOLVLIST=<val>,<BD_ADDR>	
Description	This command adds/removes a remote device to/from the Resolving List in the Bluetooth controller.	
Parameters	<val>	0-remove, 1-add
	<BD_ADDR >	The Bluetooth address of the remote device that will be added to the White List in the Bluetooth controller.

Examples	ATBLEDEVRESOLVLIST=1,798701F901FA // adds remote device to resolve list
	ATBLEDEVRESOLVLIST=0,798701F901FA // removes remote device from resolve list	

7.4.5.1.19 ATBLESETAUTHPAYTOUT

Syntax	ATBLESETAUTHPAYTOUT=<BD_ADDR>,<timeout>	
Description	This command sets the authentication payload timeout for a specified remote device.	
Parameters	<BD_ADDR >	The Bluetooth address of the remote device that will be added to the White List in the Bluetooth controller.
	<timeout>	Positive non-zero number. The specified timeout in milliseconds.
Examples	ATBLESETAUTHPAYTOUT=798701F901FA,100	

7.4.5.1.20 ATBLEMTU

Syntax	ATBLEMTU=[<MTU>]	
Description	This command queries the maximum GATT MTU. This command can be also used to set the maximum GATT MTU.	
Parameters	<MTU>	(23-517) The MTU that will be set as the maximum GATT MTU.

Examples	ATBLEMTU= //Gets the MTU
	ATBLEMTU=517 //sets the MTU	

7.4.5.1.21 ATBLESCANPARAMS

Syntax	ATBLESCANPARAMS=[<scan_interval>,<scan_window>]	
Description	This command queries/sets the GAP LE scan Parameters	
Parameters	<scan_interval>	Positive non-zero number. The scan interval in milliseconds
	<scan_window>	Positive non-zero number. The scan window in milliseconds

Examples	ATBLESCANPARAMS= //Gets the scan parameters
	ATBLESCANPARAMS=100,60 //sets the scan parameters	

7.4.5.1.22 ATBLEADVPARAMS

Syntax	ATBLEADVPARAMS=[<min_adv_interval>,< max_adv_interval >]	
Description	This command queries/sets the GAP LE advertising Parameters	
Parameters	< min_adv_interval >	Positive non-zero number. The minimum advertising interval in milliseconds
	< max_adv_interval >	Positive non-zero number. The maximum advertising interval in milliseconds

Examples	ATBLEADVPARAMS= //Gets the advertising parameters
	ATBLEADVPARAMS=50,100 //sets the advertising parameters	

7.4.5.1.23 ATBLECONNPARAMS

Syntax	ATBLECONNPARAMS=[<min_adv_interval>,< max_adv_interval >,<slave_latency>]	
Description	This command queries/sets the GAP LE connection Parameters	
Parameters	< min_adv_interval >	Positive non-zero number. The minimum advertising interval in milliseconds.
	< max_adv_interval >	Positive non-zero number. The maximum advertising interval in milliseconds
	<slave_latency>	Positive non-zero number. The number of connection events

Examples	ATBLECONNPARAMS= //Gets the connection parameters
	ATBLECONNPARAMS=50,100,2 //sets the connection parameters	

7.4.5.1.24 ATBLESETTESTDATAPERIOD

Syntax	ATBLESETTESTDATAPERIOD=[<BD_ADDR>,<period>,<size_of_data>]	
Description	Sets the periodic interval to wake up and send data on connected device. This command also can be used to set BLE packet size for the connection	
	< BD_ADDR >	The Bluetooth address of the device to send data

Syntax	ATBLESETTESTDATAPERIOD=[<BD_ADDR>,<period>,<size_of_data>]	
Parameters	< period >	Positive non-zero number. The period to wake up and send data (in milliseconds)
	<size_of_data >	Size of data to send (bytes should be greater than 1)
Examples	ATBLESETTESTDATAPERIOD=798701F901FA,100,10 ATBLESETTESTDATAPERIOD=517	

7.4.5.1.25 ATBLEPERSISTDATA

Syntax	ATBLEPERSISTDATA=store|load|delete,[<force>]	
Description	Store/load/delete application persistent storage data	
Parameters	store	Store application data to persistent storage
	Load	<force> 1=true Load application from persistent storage
	delete	Delete application data from persistent storage
Example	ATBLEPERSISTDATA=store ATBLEPERSISTDATA=load,1 ATBLEPERSISTDATA=delete	

7.4.5.1.26 ATBLECFGADVREPDISP

Syntax	ATBLECFGADVREPDISP=<verbose>	
Description	This command configures how much advertisement data can be displayed.	
Parameters	<verbose>	0 – limited advertising display 1 - verbose advertising display
Example	ATBLECFGADVREPDISP=1	

7.4.5.1.27 ATBLEGETLOCOOBDATA

Syntax	ATBLEGETLOCOOBDATA=
Description	This command queries the local out of band data to send to remote device
Parameters	None
Example	ATBLEGETLOCOOBDATA=

7.4.5.1.28 ATBLESETREMOOBDATA

Syntax	ATBLESETREMOOBDATA=[confirmation] [randomizer]	
Description	Stops data transmit/receive packet test	
Parameters	confirmation	
	randomizer	
Example	ATBLESETREMOOBDATA =	

7.4.5.1.29 ATBLELOWPOWTRANS

Syntax	ATBLELOWPOWTRANS=<enable>,<TX_power_level>,<FEM_mode>	
Description	This command overrides the Tx power by choosing Tx step and FEM control settings. Tx power level and FEM Mode parameters are only used when the mode is enabled	
Parameters	<enable>	1-enable

Syntax	ATBLELOWPOWTRANS=<enable>,<TX_power_level>,<FEM_mode>	
	<TX_power_level>	Transmit power level (-128 to 127), 127 means don't care
	FEM_mode	0-shutdwon 1-rx
Example	ATBLELOWPOWTRANS=1,120,1	

7.4.5.1.30 ATBLESTARTTEST

Syntax	ATBLESTARTTEST=transmit|receive,<BD_ADDR>,[<packet_size>]	
Description	Starts HCI transmit packet test Starts data receive packet test	
Parameters	trasmit	BD_ADDR - Bluetooth address
	receive	BD_ADDR - Bluetooth address <packet_size> - positive number greater than 1
Example	ATBLESTARTTEST= transmit,798701F901FA,517 ATBLESTARTTEST= receive,798701F901FA	

7.4.5.1.31 ATBLESTPTXRXTEST

Syntax	ATBLESTPTXRXTEST=
Description	Stops data transmit/receive packet test
Parameters	None
Example	ATBLESTPTXRXTEST=

7.4.5.1.32 ATBLEV5PHY

Syntax	ATBLEV5PHY=[<Tx_phy>,<Rx_phy>]	
Description	Reads the current PHY for the specified connection Sets the PHY for the specified connection	
Parameters	Tx_phy	Bitmask 0=no preference, 0x0001=1M, 0x0002=2M
	Rx_phy	Bitmask 0=no preference, 0x0001=1 M, 0x0002=2M

7.4.5.1.33 ATBLEV5EXTADV

Syntax	ATBLEV5EXTADV=set_ext_adv_params, [AdvertisingHandle],[EventProperties], [Adv_Tx_pow] ATBLE5EXTADV=en_ext_adv,[Enable],[num_of_sets],[adv_handle],[duration],[max_ext_adv_events]		
Description	This command sets the Extended advertising parameters Enables/disables extended advertising		
Parameters	set_ext_adv_params	AdvertisingHandle	Non-zero number - Integer (uint8_t) Advertising handle
		EventProperties	Non-zero number – integer (uint16_t) EventProperties
		Adv_Tx_pow	-127 -126, 127 = Don't care
	en_ext_adv	Enable	0-disable, 1-enable
		num_of_sets	Non-zero number – integer (uint8_t) num_of_sets
		Adv_handle	Non-zero number – integer (uint8_t) adv_handle
		duration	Non-zero number – integer (uint32_t) duration
		max_ext_adv_events	Non-zero number – integer (uint8_t) max_ext_adv_events

7.4.5.1.34 ATBLEV5EXTSCAN

Syntax	ATBLEV5EXTSCAN=set_ext_scan_params, [scan_interval],[scan_window], [filter] ATBLEV5EXTSCAN=en_ext_scan,[Enable],[duration],[period]		
Description	This command sets the Extended scanning parameters Enables/disables extended scanning		
Parameters	set_ext_scan_params	scan_interval	UNIT16 The scan interval in milliseconds.
		scan_window	UNIT16 The scan window in milliseconds.
		filter	0-No filter 1-filter using the white list
	en_ext_scan	Enable	0-disable, 1-enable
		duration	In milliseconds
		period	In milliseconds

7.4.5.1.35 ATBLEV5EXTCON

Syntax	ATBLEV5EXTCON=[BD_ADDR],[ADDR_Type]	
Description	Sends an Extended Connection request to a remote device	
Parameters	BD_ADDR	Bluetooth Address
	ADDR_type	0-public 1-random 2-public identity 3-random identity

7.4.5.2 BLE: AIOS profile

7.4.5.2.1 ATBLEAIOSSERVICE

Syntax	ATBLEAIOSSERVICE=register|unregister	
Description	This command registers/unregisters an AIOS instance. This command cannot be used while connected to a remote device.	
Parameters	Register/unregister	Register/unregister an AIOS instance

7.4.5.2.2 ATBLEAIOSCFGREMAIOS

Syntax	ATBLEAIOSCFGREMAIOS=<Characteristic_Type>,<Characteristic_ID>,<Configure_Notifications>	
Description	This command configures notifications for an Analog or Digital Characteristic instance on an AIOS Server.An AIOS client must configure a characteristic instance for notifications before it can receive notifications from an AIOS Server.	
Parameters	Characteristic_Type	0-Digital, 1- Analog The characteristic type
	Characteristic_ID	0-1 The characteristic ID
	Configure_Notifications	0-Disable, 1-Enable Enables or disables notifications.

7.4.5.2.3 ATBLEAIOSREAD

Syntax	ATBLEAIOSREAD=ReadChar,<Characteristic_Type>,<Characteristic_ID> ATBLEAIOSREAD=ReadPresent,<Characteristic_Type>,<Characteristic_ID> ATBLEAIOSREAD=< ReadNumber >,<Characteristic_ID> ATBLEAIOSREAD=<NotifyAIOSChar>,<Characteristic_ID>,<Characteristic_ID>		
Description	This command reads an Analog or Digital Characteristic instance on an AIOS Server. This command reads an Analog or Digital Characteristic instance's Presentation Format Descriptor on an AIOS Server. This command reads a Digital Characteristic instance's Number of Digitals Descriptor on an AIOS Server. This command allows the AIOS Server to send a notification for a Digital or Analog		
Parameters	ReadChar/ ReadPresent/ ReadNumber/ NotifyAIOSChar	Characteristic_Type	0-Digital, 1- Analog The characteristic type
		Characteristic_ID	0-1 The characteristic ID

7.4.5.2.4 ATBLEAIOSWRITE

Syntax	ATBLEAIOSWRITE=WriteDigOutput,<Characteristic_ID>,<Digital_oct_1>,<Digital_oct_2> ATBLEAIOSWRITE=WriteAnOutput,<Characteristic_ID>,<Analog_Value> ATBLEAIOSWRITE=SetDigOutput,<Characteristic_ID>,<Digital_oct_1>,<Digital_oct_2> ATBLEAIOSWRITE=SetAnOutput,<Characteristic_ID>,<Analog_Value>
Description	This command writes a Digital/Analog (Output) Characteristic instance's value on an AIOS Server. This command allows the AIOS server to set the value for a Digital/Analog (Input) Characteristic instance.

Syntax	ATBLEAIOSWRITE=WriteDigOutput,<Characteristic_ID>,<Digital_oct_1>,<Digital_oct_2> ATBLEAIOSWRITE=WriteAnOutput,<Characteristic_ID>,<Analog_Value> ATBLEAIOSWRITE=SetDigOutput,<Characteristic_ID>,<Digital_oct_1>,<Digital_oct_2> ATBLEAIOSWRITE=SetAnOutput,<Characteristic_ID>,<Analog_Value>		
Parameters	WriteDigOutput / WriteAnOutput / SetDigOutput / SetAnOutput	Characteristic_ID	0-1 The characteristic ID
		Digital Octet 1	Integer value The first four digital signals. Each digital signal is a 2-bit value.
		Digital Octet 2	Integer value (UINT8) The last four digital signals. Each digital signal is a 2-bit value.
		Analog Value	Integer value (UNIT16) The analog value

7.4.5.3 BLE: BAS profile

7.4.5.3.1 ATBLEBAS

Syntax	ATBLEBAS=register|unregister	
Description	This command registers/ un-registers a BAS instance. This command cannot be used while connected to a remote device	
Parameters	register	Register a BAS instance
	unregister	Unregister a BAS instance
Example	ATBLEBAS=register ATBLEBAS=unregister	

7.4.5.3.2 ATBLEBASCFGREMBAS

Syntax	ATBLEBASCFGREMBAS=<type>,<ID>,<notification>	
Description	This command configures notifications for the Battery Level Characteristic on a BAS Server. A BAS Client must configure the Battery Level Characteristic for notifications before it can receive notifications from a BAS Server	
Parameters	<type>	0-digital, 1-analog. The characteristic type
	<ID>	The characteristic instance ID
	<notification>	0-diable, 1-enable. Either enable/disable notifications
Example	ATBLEBASCFGREMBAS =0,1,1	

7.4.5.3.3 ATBLEBASBATTLVL

Syntax	ATBLEBASBATTLVL=get_bat,<ID> ATBLEBASBATTLVL =set_bat,<bat_level>,[<ID>]	
Description	This command will get/set the value of a Battery Level Characteristic. If the demo is configured to support muliple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being read / written. Setting the battery level can be called by BAS server or BAS client	
Parameters	get_bat	0 – to get the battery level instance
	set_bat	1-to set the battery level instance
	<bat_level>	(0-100), the battery level percentage
	<ID>	A non-zero positive number. The battery level instance which is optional

Syntax	ATBLEBASBATTLVL=get_bat,<ID> ATBLEBASBATTLVL =set_bat,<bat_level>,[<ID>]
Example	ATBLEBASBATTLVL=0,7 ATBLEBASBATTLVL=1,35,7

7.4.5.3.4 ATBLEBASNOTIBATTLVL

Syntax	ATBLEBASNOTIBATTLVL=<bat_level >[,<ID>]	
Description	This command allows a BAS Server to send a notification for a Battery Level Characteristic to a remote AIOS Client. If the demo is configured to support muliple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being notified. A BAS Client must configure a Battery Level Characteristic for notifications before it can receive notifications from a BAS Server	
Parameters	<bat_level>	(0-100), the battery level percentage
	<ID>	A non-zero positive number. The battery level instance which is optional
Example	ATBLEBASNOTIBATTLVL=22 ATBLEBASNOTIBATTLVL=22,4	

7.4.5.3.5 ATBLEBASBATTLVLPSTFMT

Syntax	ATBLEBASBATTLVLPSTFMT=get_bat,[,<ID>] ATBLEBASBATTLVLPSTFMT=set_bat,<name_space>,<description>[,<ID>]	
Description	This command gets/sets the Battery Level Characteristic's Presentation Format Descriptor. If the demo is configured to support muliple Battery Level Characteristic instances, then InstanceID parameter is required to identify the instance that is being read/written	
Parameters	get_bat	0 – to get the battery level characteristic
	set_bat	1 - to set the battery level characteristic
	<ID>	A non-zero positive number. The battery level instance which is optional
	<name_space>	(0x00-0xFF). The defined namespace
	<description>	(0x0000-0xFFFF). The field that identifies the battery level characteristic instance
Example	ATBLEBASBATTLVLPSTFMT=0,4 ATBLEBASBATTLVLPSTFMT=0,0x10,0x1111,4	

7.4.5.4 BLE: GAPS profile

7.4.5.4.1 ATBLEGAPSLOCNAME

Syntax	ATBLEGAPSLOCNAME=[<device_name>]	
Description	This command gets/sets the GAPS device name on the local device.	
Parameters	<device_name>	A string value, which can be set to the device name
Example	ATBLEGAPSLOCNAME = ATBLEGAPSLOCNAME =CDB24	

7.4.5.4.2 ATBLEGAPSRDREMPARAMS

Syntax	ATBLEGAPSRDREMPARAMS=		
Description	This command reads the GAPS device name and appearance on a GAPS Server.		
Parameters	None		
Example	ATBLEGAPSRDREMPARAMS=		

7.4.5.4.3 ATBLEGAPSLOCAPP

Syntax	ATBLEGAPSLOCAPP=[<ID >]		
Description	This command gets/sets the GAPS device appearance on the local device.		
Parameters	<ID>	A positive number in the displayed list	
Example	ATBLEGAPSLOCAPP= ATBLEGAPSLOCAPP=1		

7.4.5.5 BLE: HOGP (HID over GATT service) profile

7.4.5.5.1 ATBLEHIDSERVICE

Syntax	ATBLEHIDSSERVICE=register|unregister ATBLEHIDSSERVICE=configure,<InstanceID>,<ReportNotify> ATBLEHIDSSERVICE=ReadHIDSconfig,<InstanceID>		
Description	This command registers/unregisters/configure a Human Input Device Service (HIDS) instance.		
Parameters	Register/unregister	Register/unregister an HIDS instance	
	Configure/ ReadHIDSconfig	InstanceID	Positive non-zero number The HIDS instance identifier.
		Report Notify	0-Disable, 1-Enable Enables or disables report notifications.

7.4.5.5.2 ATBLEHIDSREPORT

Syntax	ATBLEHIDSREPORT=Get,< InstanceID>,<Report_Type>,<Report_ID> ATBLEHIDSREPORT=Set ATBLEHIDSREPORT=send,< InstanceID>,<Protocol_Mode>		
Description	This command gets the current specified report for the HIDS Characteristic instance on a HIDS Server. This command sets a report for the HIDS Characteristic instance on a HIDS Server. This command notifies a report to a HIDS Client (HID Host). A HIDS client must configure a report for notifications before it can receive notification from a HIDS Server.		
Parameters	get/ set/ send/	InstanceID	Positive non-zero number The HIDS instance identifier.
		Report_Type	1=input, 2= output, 3=feature, 4= boot keyboard input, 5=boot keyboard output, 6=boot mouse input The report type to retrieve
		Report_ID	0=None The report ID of the report to retrieve.
		Protocol_Mode	0-boot, 1=report Sets the protocol mode

7.4.5.5.3 ATBLEHIDSMODE

Syntax	ATBLEHIDSMODE=Suspend,< InstanceID>,<suspend_mode> ATBLEHIDSMODE=Protocol,< InstanceID>,<protocol_mode>		
Description	This command sets suspend/protocol mode on a HIDS Server.		
Parameters	Suspend / Protocol	InstanceID	Positive non-zero number The HIDS instance identifier.
		Suspend mode	0=exit suspend, 1=suspend Starts or exists suspend mode
		Protocol_Mode	0-boot, 1=report Sets the protocol mode

7.4.5.6 BLE: HRS profile

7.4.5.6.1 ATBLEHRSCONFIG

Syntax	ATBLEHRSCONFIG=[HRS_notify]	
Description	This command configures the remote HRS service	
Parameters	HRS_notify	0-disble, 1 -enable

7.4.5.7 BLE: DIS profile

7.4.5.7.1 ATBLEDISSERVICE

Syntax	ATBLEDISSERVICE=<mode>	
Description	This command registers/un-registers the DIS service	
Parameters	mode	0-disable , 1-Enable

7.4.5.8 BLE: TPS profile

7.4.5.8.1 ATBLETPSSERVICE

Syntax	ATBLETPSSERVICE=<mode>	
Description	This command registers/un-registers the TPS service	
Parameters	mode	0-disable , 1-Enable

7.4.5.9 BLE: SCPS profile

7.4.5.9.1 ATBLESCPSSERVICE

Syntax	ATBLESCPSSERVICE=register|unregister ATBLESCPSSERVICE=configure		
Description	This command registers/unregisters/configure an SCPS instance.		
Parameters	Register/unregister	Register/unregister an SCPS instance	
	Configure	Configure Scan Refresh Notification	0=Disable, 1=Enable Enables or disables notifications.

7.4.5.9.2 ATBLESCPSSETSCANWIN

Syntax	ATBLESCPSSETSCANWIN=<Scan_Int>,<Scn_win>	
Description	This command sets the Scan Parameters Characteristic on the SCPS Server.	
Parameters	Scan_Int	UNIT16 The scan interval in milliseconds.
	Scan_win	UNIT16 The scan window in milliseconds.

7.4.5.10 BLE: SPPLE profile

7.4.5.10.1 ATBLESPPLESERVICE

Syntax	ATBLESPPLESSERVICE=register|unregister|configure	
Description	This command registers/unregisters/configure an SPPLE instance.	
Parameters	Register/unregister/configure	Register/unregister/configure an SPPLE instance

7.4.5.10.2 ATBLESPPLEDATA

Syntax	ATBLESPPLEDATA=[Num_Of_Bytes]	
Description	This command sends/reads data to/from a remote device.	
Parameters	Num_Of_Bytes	UNIT16 The number of bytes to send to the remote device.

7.4.5.10.3 ATBLESPPLEEXT

Syntax	ATBLESPPLEEXT=Loopback,<mode> ATBLESPPLEEXT=DispRawData,<mode> ATBLESPPLEEXT=AutoReadMode,<mode>	
Description	This command enables data to be sent back to the remote device (looped back) after it is received. This command enables raw data to be displayed when data is received. This command enables received data to automatically be read without using the 'ReadDataCommand.'	
Parameters	Loopback/ DispRawData/ AutoReadMode	0-Disable, 1-Enable Enables/Disables Display loopback mode/Raw Data mode/automatic read mode

7.4.6 Zigbee commands

7.4.6.1 Generic commands

7.4.6.1.1 ATZBHELP

Syntax	ATZBHELP=
Description	This command lists the zigbee commands

7.4.6.1.2 ATZBSERVICE

Syntax	ATZBSERVICE=start|stop	
Description	This command initializes or shuts down the zigbee stack	
Parameters	start	Initialize zigbee stack
	stop	Shutdown the zigbee stack
Example	ATZBSERVICE=start ATZBSERVICE=stop	

7.4.6.1.3 ATZBDEVICE

Syntax	ATZBDEVICE=add|delete|show		
Description	This command adds a device to the Zigbee demo device table which can be used by other commands within the demo to send packets. This command removes a device from the Zigbee demo's device table. This function will not compress the remaining devices, but will leave the device index unused This command lists the devices that are currently in the Zigbee device table.		
Parameters	add	Mode	1-3 The address mode for this device as group (1), network (2), or extended (3).
		Address	16 or 64-bit depending on Mode The address of the device to add. This is a 16-bit value for a group or network address and a 64-bit value for an extended address
		Endpoint	0-0xFF The endpoint for the device.
	delete	DevID	Valid device index The index of the device to be removed. Note that device zero is reserved as a NULL address and cannot be removed.
	show	lists the devices that are currently in the Zigbee device table	

7.4.6.1.4 ATZBFORM

Syntax	ATZBFORM=<useSecurity>,<Distributed>,<channel>	
Description	This command tells the Zigbee stack to form a Zigbee network. A callback will occur when the network formation completes which displays the status of the form (0 is success) and the channel the network started on. The network address of the device will always be zero.	
	UseSecurity	Flag indicating if security is used (1) or not (0). Note that address zero is reserved as NULL address for sending packets using the binding table
	Distributed	Flag indicating if this should form a centralized (0) or distributed (1) network (optional: defaults to 0/Distributed).

Syntax	ATZBFORM=<useSecurity>,<Distributed>,<channel>	
	Channel	Optional parameter to specify the channel to join the network on. If not specified, the FORM_CHANNEL_MASK value will be used.

7.4.6.1.5 ATZBJOIN

Syntax	ATZBJOIN=<CoOrdinator>,<useSecurity>,<IsRejoin>	
Description	This command tells the Zigbee stack to join or rejoin to a Zigbee network. The rejoin operations are only intended for after the device has left the network and wishes to rejoin.	
Parameters	Coordinator	Flag indicating if the device should join as a coordinator(1) or end device(0).
	UseSecurity	Flag indicating if security is used (1) or not (0)
	IsRejoin	Optional flag indicating if this is a Rejoin operation (1) or a join operation (0). Default is a join operation (0).
	Channel	Optional parameter to specify the channel to join the network on. If not specified, the JOIN_CHANNEL_MASK value will be used.

7.4.6.1.6 ATZBLEAVE

Syntax	ATZBLEAVE=Leave|LeaveRequest,[<TargetDevID>,<LeaveDevID>,<RemoveChildren>,<Rejoin>]		
Description	This command tells the Zigbee stack to leave the Zigbee network. Sends a leave request		
Parameters	Leave	Leave the zigbee network	
	LeaveRequest	TargetDevID	Valid device index Device ID to send the leave request to.
		LeaveDevId	Valid device index Device ID of that should leave the network.
		RemoveChildren	0-1 Flag indicating if the device should remove its children.
		Rejoin	0-1 Flag indicating if the device should rejoin or not.

7.4.6.1.7 ATZBPERMITJOIN

Syntax	ATZBFORM=<duration>	
Description	This command tells the network to permit new devices to join for a given period. A value of zero effectively disables joining.	
Parameters	duration	0-255: Time in seconds for join to be enabled

7.4.6.1.8 ATZBBIND

Syntax	ATZBBIND=bind,<TargetDevID>,<SrcDevID>,<DestDevID>,<ClusterID> ATZBBIND=unbind,<TargetDevID>,<SrcDevID>,<DestDevID>,<ClusterID>
Description	This command creates/removes a binding between two endpoints. A callback will occur when the command completes displaying the status of the operation.

Syntax	ATZBBIND=bind,<TargetDevID>,<SrcDevID>,<DestDevID>,<ClusterID> ATZBBIND=unbind,<TargetDevID>,<SrcDevID>,<DestDevID>,<ClusterID>		
	Bind/unbind	TargetDevID	Valid device index Device ID to send the bind/unbind request to. This is typically the source of the bind or its parent. The address mode for this device must be a network address.
		SrcDevID	Valid device index Device ID of the source of the bind. The address mode for this device must be an extended address
		DestDevID	Valid device index Device ID for the destination of the bind. The address mode for this device must be either a group or extended address.
		ClusterID	0-0xFFFF The cluster ID for the bind.

7.4.6.1.9 ATZBATTR

Syntax	ATZBATTR=GetNIB,<AttID>,<AttrIndex>,<MaxLength>,<ClusterID> ATZBATTR=SetNIB,<AttID>,<AttrIndex>,<Length>,<Value> ATZBATTR=GetAIB,<AttID>,<AttrIndex>,<MaxLength> ATZBATTR=SetAIB,<AttID>,<AttrIndex>,<Length>,<Value> ATZBATTR=GetBIB,<AttID>,<AttrIndex>,<MaxLength> ATZBATTR=SetBIB,<AttID>,<AttrIndex>,<Length>,<Value>		
Description	Reads/writes a network information base attribute Reads/writes an APS information base attribute Reads/writes a BDB information base attribute		
Parameters	GetNIB/SetNIB GetAIB/SetAIB GetBIB/SetBIB	AttrID	16-bit value NWK/APS /BDB Attribute ID to read/write
		AttrIndex	8-bit value Index of the attribute to read /write (defaults to 0).
		Maxlength	16-bit value Maximum expected length of the attribute to read (defaults to 128 bytes).
		ClusterID	0-0xFFFF The cluster ID for the bind.
		Length	16-bit value Length of attribute in bytes
		Value	Hex string Attribute to write. The number of bytes in the hexadecimal string should match the specified length.

7.4.6.1.10 ATZBEXTADDR

Syntax	ATZBEXTADDR=[ExtAddr]	
Description	Gets/sets the extended address of the device	
Parameters	ExtAddr	64-bit value Extended address to be set

7.4.6.1.11 ATZBCLEARPERSIST

Syntax	ATZBCLEARPERSIST=
Description	Clears zigbee persistent data
Parameters	None

7.4.6.2 Zigbee Cluster commands

7.4.6.2.1 ATZBCLENDP

Syntax	ATZBCLENDP=ListClusterTypes ATZBCLENDP=ListEndpointtypes ATZBCLENDP=CreateEndpoint,<EndpointNumber>,<EndpointType> ATZBCLENDP=RemoveEndpoint ATZBCLENDP=ListClusters		
Description	Lists the cluster names and ID that are supported by the Zigbee demo. Lists the types of endpoints that can be created by the Zigbee demo. Creates an endpoint, including all clusters that the endpoint needs. Removes an endpoint and all clusters associated with it. This command displays the current list of registered clusters. The IDs from this list can be used with the other generic cluster commands.		
Parameters	ListClusterTypes	Lists the cluster names and ID	
	ListEndpointtypes	Lists the endpoints	
	CreateEndpoint	EndpointNumber	1-240 Endpoint to create
		EndpointType	1-13 Type of endpoint to create
	RemoveEndpoint	Removes an endpoint and all clusters associated to it	
	ListClusters	displays the current list of registered clusters	

7.4.6.2.2 ATZBCLATTR

Syntax	ATZBCLATTR=ReadLocAttr,<ClusterIndex>,<AttrID>,<AttrLength> ATZBCLATTR=WriteLocAttr,<ClusterIndex>,<AttrID>,<AttrLength>,<AttrValue> ATZBCLATTR=ReadAttr,<DevID>,<ClusterIndex>,<AttrID>,<AttrValue> ATZBCLATTR=WriteAttr,<DevID>,<ClusterIndex>,<AttrID>,<AttrType>,<AttrLength>, <AttrValue> ATZBCLATTR=DiscAttr,<DevID>,<ClusterIndex>		
Description	This command reads/writes a cluster local attribute This command is used to reads/writes a cluster attributes from a remote device. This command is used to discover attributes on a remote device		
Parameters	ReadLocAttr/ WriteLocAttr	ClusterIndex	Valid Cluster Index ID of the local cluster (in the cluster list) associated with the attribute
		AttrID	0-0xFFFF ID of the attribute
		AttrLength	0-8 Length of the attribute
		AttrValue	64-bit value Value to be written
	ReadAttr/ WriteAttr	DevID	Valid device index The device to send the command to. Note that device zero can be specified to use the binding table.

Syntax	ATZBCLATTR=ReadLocAttr,<ClusterIndex>,<AttrID>,<AttrLength> ATZBCLATTR=WriteLocAttr,<ClusterIndex>,<AttrID>,<AttrLength>,<AttrValue> ATZBCLATTR=ReadAttr,<DevID>,<ClusterIndex>,<AttrID>,<AttrValue> ATZBCLATTR=WriteAttr,<DevID>,<ClusterIndex>,<AttrID>,<AttrType>,<AttrLength>, <AttrValue> ATZBCLATTR=DiscAttr,<DevID>,<ClusterIndex>		
		ClusterIndex	Valid cluster index ID of the local cluster (in the cluster list) associated with the attribute
		AttrID	0-0xFFFF ID of the attribute
		AttrType	0-0xFF Type of the attribute
		AttrLength	0-8 Length of the attribute
		AttrValue	64-bit Value to be written
	DiscAttr	DevID	Valid device index The device to send the command to. Note that device zero can be specified to use the binding table.
		ClusterIndex	Valid cluster index ID of the local cluster (in the cluster list) associated with the attribute

7.4.6.2.3 ATZBCLCFGREP

Syntax	ATZBCLCFGREP=ConfigReport,<DevID>,<ClusterIndex>,<AttrID>,<AttrType>, <Mininterval>, <MaxInterval>,<ChangeValue> ATZBCLCFGREP=ReadReportConfig,<DevID>,<ClusterIndex>,<AttrID>,<AttrLength>, <AttrValue>		
Description	This command is used to configure/read attribute reporting on a remote device.		
Parameters	ConfigReport / ReadReportConfig	DevID	Valid device index The device to send the command to. Note that device zero can be specified to use the binding table.
		ClusterIndex	Valid cluster index ID of the local cluster (in the cluster list) associated with the attribute
		AttrID	0-0xFFFF ID of the attribute
		AttrType	0-0xFF Type of the attribute
		Mininterval	0-0xFFFF Minimum reporting interval for the attribute
		MaxInterval	0-0xFFFF Maximum reporting interval for the attribute
		ChangeValue	64-bit: Value to be written

Syntax	ATZBCLCFGREP=ConfigReport,<DevID>,<ClusterIndex>,<AttrID>,<AttrType>, <Mininterval>, <MaxInterval>,<ChangeValue> ATZBCLCFGREP=ReadReportConfig,<DevID>,<ClusterIndex>,<AttrID>,<AttrLength>, <AttrValue>		
		ClusterIndex	Valid cluster index ID of the local cluster (in the cluster list) associated with the attribute

7.4.6.3 Zigbee Basic

7.4.6.3.1 ATZBBASIC

Syntax	ATZBBASIC=Reset,<DevID>,<ClientEndpoint> ATZBBASIC=Read, <AttID> ATZBBASIC=Write,<AttID>,<Value>		
Description	Sends a reset request to a basic server cluster. Reads/writes an attribute from/to the local basic server cluster.		
Parameters	Reset	DevId	Valid device index The device to send the command to. Note that device zero can be specified to use the binding table
		ClientEndpoint	Valid end point Endpoint that contains the basic client cluster to use to send the command
	Read/write	AttrId	16-bit value ID of the attribute to read /write
		Value	8-bit value or string Value to write to the attribute. It is either a byte or a string depending on which the attribute is being written.

7.4.6.4 Zigbee Identify

7.4.6.4.1 ATZBIDENTIFY

Syntax	ATZBIDENTIFY=Ident,<DevID>,<ClientEndpoint>,<Time> ATZBIDENTIFY=IdentQuery,<DevID>,<ClientEndpoint>		
Description	Sends a identify request/identify query request to a server cluster.		
Parameters	Ident/ IdentQuery	DevId	Valid device index The device to send the command to
		ClientEndpoint	Valid end point Endpoint that contains the identify client cluster to use to send the command
		Time	16-bit value The time for the device to identify itself

7.4.6.5 Zigbee Group

7.4.6.5.1 ATZBGROUP

Syntax	ATZBGROUP=add,<DevID>,<ClientEndpoint>,<GroupID>,<Name>,<Identifying> ATZBGROUP=view,<DevID>,<ClientEndpoint>,<GroupID> ATZBGROUP=remove,<DevID>,<ClientEndpoint>,<GroupID>		
Description	Sends a add group/view group/remove group to a group server clusters		
Parameters	add/view/ remove	DevId	Valid device index The device to send the Add /view/remove Group request to.
		ClientEndpoint	Valid end point Endpoint that contains the groups client cluster to use to send the command
		GroupID	16-bit value The ID of the group to add/view/remove
		Name	String Name of the group
		Identifying	0-1 Flag indicating if only endpoints that are identifying should be added to the group

7.4.6.5.2 ATZBGROUPEXT

Syntax	ATZBGROUPEXT=GroupMem,<DevID>,<ClientEndpoint>,<GroupID> ATZBGROUPEXT=RemAll,<DevID>,<ClientEndpoint>		
Description	Sends a Get Group Membership request to a Groups server cluster. Sends a Remove All Groups request to a Groups server cluster.		
Parameters	GroupMem /RemAll	DevId	Valid device index The device to send the Get group membership / remove all groups request to
		ClientEndpoint	Valid end point Endpoint that contains the groups client cluster to use to send the command
		GroupID	16-bit value ID of the group to view

7.4.6.6 Zigbee Scenes

7.4.6.6.1 ATZBSCENE

Syntax	ATZBSCENE=add,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID>,<SceneName> ,<TransTime>,<IsEnhanced> ATZBSCENE=view,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID>,<IsEnhanced> ATZBSCENE=remove,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID> ATZBSCENE=store,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID>		
Description	Sends an Add/view/remove/store Scene request to a Scenes server cluster. The extension field set used for this command is compiled from all clusters on the same endpoint which support scenes.		
Parameters	add/view/ remove/store	DevId	Valid device index The device to send the Add /view/remove/store Group request to.

Syntax	ATZBSCENE=add,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID>,<SceneName>,<TransTime>,<IsEnhanced> ATZBSCENE=view,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID>,<IsEnhanced> ATZBSCENE=remove,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID> ATZBSCENE=store,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID>		
		ClientEndpoint	Valid end point Endpoint that contains the scenes client cluster to use to send the command
		GroupID	16-bit value The ID of the scene to add/view/remove/store
		SceneID	8-bit value ID of the scene to add/view/remove/store
		SceneName	String Name of the scene
		TransTime	16-bit value Time to transition to the scene. This is in tenths of a second if IsEnhanced is 1, otherwise it is in seconds.
		IsEnhanced	0-1 Flag indicating if this is an Enhanced Add Scene request.

7.4.6.6.2 ATZBSCENEEXT

Syntax	ATZBSCENEEXT=RemAllScenes,<DevID>,<ClientEndpoint>,<GroupID> ATZBSCENEEXT=RecallScenes,<DevID>,<ClientEndpoint>,<GroupID>,<SceneID> ATZBSCENEEXT=GetSceneMem,<DevID>,<ClientEndpoint>,<GroupID> ATZBSCENEEXT=CopyScene,<DevID>,<ClientEndpoint>,<CopyAll>,<GroupFrom>,<SceneFrom>,<GroupTo>,<SceneTo>		
Description	[Sends Remove All Scenes request to a Scenes server cluster. Sends a Recall All Scenes request to a Scene server cluster. Sends a Get Scene Membership request to a Scenes server cluster. Sends a Copy All Scenes request to a Scenes server cluster.		
Parameters	RemAllScenes RecallScenes GetSceneMem CopyScene	DevId	Valid device index The device to send the RemAllScenes /RecallScenes/GetSceneMem/copyScene Group request to.
		ClientEndpoint	Valid end point Endpoint that contains the scenes client cluster to use to send the command
		GroupID	16-bit value Group ID of the scene to remove/Get membership
		SceneID	8-bit value ID of the scene to recall
		CopyAll	0-1 Flag indicating if all scenes should be copied
		GroupFrom	16-bit value Group ID to copy from
		SceneFrom	8-bit value Scene ID to copy from
		GroupTo	16-bit value Group ID to copy to
		SceneTo	8-bit value Scene ID to copy to

7.4.6.7 Zigbee On/Off/Toggle

7.4.6.7.1 ATZBONOFF

Syntax	ATZBONOFF=on,<DevID>,<ClientEndpoint> ATZBONOFF=off,<DevID>,<ClientEndpoint> ATZBONOFF=toggle,<DevID>,<ClientEndpoint> ATZBONOFF=SetSceneData,<OnOff>		
Description	This command sends an On/Off cluster On/Off/Toggle command to the device specified.		
Parameters	on/off/toggle /SetSceneData	DevId	Valid device index The device to send the On/Off/Toggle command to
		ClientEndpoint	Valid end point Endpoint that contains the On/Off client cluster to use to send the command
		OnOff	0-1 Value of the On/Off attribute for the scene

7.4.6.8 Zigbee Level control

7.4.6.8.1 ATZBLVL

Syntax	ATZBLVL=MoveToLevel,<DevID>,<ClientEndpoint>,<WithOnOff>,<Level>,<Time> ATZBLVL=Move,<DevID>,<ClientEndpoint>,<WithOnOff>,<Direction>,<Rate> ATZBLVL=Step,<DevID>,<ClientEndpoint>,<WithOnOff>,<Direction>,<StepSize>,<Time> ATZBLVL=stop,<DevID>,<ClientEndpoint>		
Description	This command sends a Level Control cluster MoveToLevel/Move/Step/stop command to the device specified		
Parameters	MoveToLevel /Move /Step /stop	DevId	Valid device index The device to send the MoveToLevel/Move/Step/stop command to
		ClientEndpoint	Valid end point Endpoint that contains the Level control client cluster to use to send the command
		WithOnOff	0-1 Indicates if the command should be sent “with On/Off” (1) or not (0)
		Level	0-255 Level the output should move to
		Direction	0-1 Indicates if the output should move up (1) or down (0)
		Time	0-0xFFFF Time in tenths of second the output should move
		Rate	0-0xFF The rate the output should move
		StepSize	0-0xFF The size of each step

7.4.6.8.2 ATZBSETLVLSCENEDATA

Syntax	ATZBLVLSCENEDATA=<DevID>,<ClientEndpoint>	
Description	This command sends a Level Control set level scene command to the device specified	
Parameters	CurrentLevel	0-254 Value of the CurrentLevel attribute for the scene

7.4.6.9 Zigbee Alarm

7.4.6.9.1 ATZBALARM

Syntax	ATZBALARM=ResetAlarm,<DevID>,<ClientEndpoint>,<SourceClusterId>,<AlarmCode> ATZBALARM=ResetAllAlarms,<DevID>,<ClientEndpoint> ATZBALARM=GetAlarm,<DevID>,<ClientEndpoint> ATZBALARM= ResetAlarmLog,<DevID>,<ClientEndpoint> ATZBALARM=Alarm,<ServerEndpoint>,<SourceCluster>,<AlarmCode>		
Description	This command sends an Alarm cluster Reset Alarm command to the device specified. This command sends an Alarm cluster Reset All Alarms command to the device specified. This command sends an Alarm cluster Get Alarm command to the device specified.		
Parameters	ResetAlarm/ ResetAllAlarms/ GetAlarm/ ResetAlarmLog/ Alarm	DevId	Valid device index The device to send the ResetAlarm / ResetAllAlarms/ GetAlarm command to
		ClientEndpoint	Valid end point Endpoint that contains the Alarm client cluster to use to send the command
		SourceClusterId	16-bit integer ID of the cluster which generates/generated the alarm to reset
		AlarmCode	8-bit integer The alarm code to reset/generate
		ServerEndpoint	Valid endpoint Endpoint that contains the Alarm server cluster to use to send the command

7.4.6.10 Zigbee Time

7.4.6.10.1 ATZBTIME

Syntax	ATZBTIME=Read,<AttrID> ATZBTIME=Write,<AttrID>,<Value>		
Description	Reads/Writes an attribute from the local Time server cluster.		
Parameters	Read/ Write	AttrID	16-bit value ID of the attribute to read/write
		Value	32-bit value Value to write to the attribute. The supported length of the value is dependent on the attribute being written.

7.4.6.11 Zigbee Touchlink

7.4.6.11.1 ATZBTOUCHLINK

Syntax	ATZBTOUCHLINK=Start,<Endpoint>,<DeviceType> ATZBTOUCHLINK=Scan,<ClientEndpoint> ATZBTOUCHLINK=FactoryReset,<ClientEndpoint>		
Description	This command starts the Touchlink process on either a client or server cluster. This command starts a Touchlink scan on a Touchlink client cluster. This command Performs a factory reset on a Touchlink client cluster.		
Parameters	Start/ Scan/	Endpoint	Valid Endpoint Endpoint that contains the Touchlink cluster to start

Syntax	ATZBTOUCHLINK=Start,<Endpoint>,<DeviceType> ATZBTOUCHLINK=Scan,<ClientEndpoint> ATZBTOUCHLINK=FactoryReset,<ClientEndpoint>		
	FactoryReset	DeviceType	Valid Endpoint Type of device to start as a coordinator (0), router (1), RxOnWhenIdle end device (2) or sleepy end device (3).
		ClientEndpoint	Valid Endpoint Endpoint that contains the Touchlink client cluster to start scanning/FactoryReset

7.4.6.12 Zigbee ColorControl

7.4.6.12.1 ATZBCOLHUE

Syntax	ATZBTCOLHUE=MoveToHue,<DevId>,<ClientEndpoint>,<Hue>,<Direction>,<Time> ATZBTCOLHUE =MoveHue,<DevId>,<ClientEndpoint>,<Mode>,<Rate> ATZBTCOLHUE =StepHue,<DevId>,<ClientEndpoint>,<Mode>,<StepSize>,<Time>		
Description	This command sends a Color Control cluster MoveToHue/MoveHue/StepHue command to the device specified.		
Parameters	MoveToHue / MoveHue / StepHue/	DevId	Valid device index The device to send the MoveToHue / MoveHue / StepHue command to
		ClientEndpoint	Valid end point Endpoint that contains the Color control client cluster to use to send the command
		Hue	0-0xFF Hue for the light to move to
		Direction	0-3 Direction to move as Shortest(0), Longest(1), Up(2), Down(3)
		Time	0-0xFFFF Time in tenths of a second the output should move.
		Mode	0-3 Mode for the move command as Up(1) or Down(3).
		StepSize	0-0xFF The size of each step

7.4.6.12.2 ATZBCOLSAT

Syntax	ATZBTCOLSAT=MoveToSat,<DevId>,<ClientEndpoint>,<Saturation>,<Time> ATZBTCOLSAT =MoveSat,<DevId>,<ClientEndpoint>,<Mode>,<Rate> ATZBTCOLSAT =StepSat,<DevId>,<ClientEndpoint>,<Mode>,<StepSize>,<Time> ATZBTCOLSAT =MoveToHueSat,<DevId>,<ClientEndpoint>,<Hue>,<Saturation>,<Time>		
Description	This command sends a Color Control cluster MoveToSat / MoveSat / StepSat/ MoveToHueSat command to the device specified.		
Parameters	MoveToSat / MoveSat / StepSat / MoveToHueSat	DevId	Valid device index The device to send the MoveToHue / MoveHue / StepHue / MoveToHueSat command to
		ClientEndpoint	Valid end point Endpoint that contains the Color control client cluster to use to send the command
		Saturation	0-0xFF Saturation value for the light to move to

Syntax	ATZBTCOLSAT=MoveToSat,<DevId>,<ClientEndpoint>,<Saturation>,<Time> ATZBTCOLSAT =MoveSat,<DevId>,<ClientEndpoint>,<Mode>,<Rate> ATZBTCOLSAT =StepSat,<DevId>,<ClientEndpoint>,<Mode>,<StepSize>,<Time> ATZBTCOLSAT =MoveToHueSat,<DevId>,<ClientEndpoint>,<Hue>,<Saturation>,<Time>		
		Hue	0-0xFF Hue for the light to move to
		Time	0-0xFFFF Time in tenths of a second the output should move.
		Mode	0-3 Mode for the move command as Up (1) or Down (3).
		Rate	0-0xFF Time for each step
		StepSize	0-0xFF The size of each step

7.4.6.12.3 ATZBCOL

Syntax	ATZBTCOL=MoveToCol,<DevId>,<ClientEndpoint>,<ColorX>,<ColorY>,<Time> ATZBTCOL =MoveCol,<DevId>,<ClientEndpoint>,<RateX>,<RateY> ATZBTCOL =StepCol,<DevId>,<ClientEndpoint>,<StepX>,<StepY>,<Time> ATZBTCOL =MoveToColTemp,<DevId>,<ColorTempMireds>,<Time>		
Description	This command sends a Color Control cluster MoveToCol / MoveCol / StepCol / MoveToColTemp command to the device specified.		
Parameters	MoveToCol / MoveCol / StepCol / MoveToColTemp	DevId	Valid device index The device to send the MoveToCol / MoveCol / StepCol / MoveToColTemp command to
		ClientEndpoint	Valid end point Endpoint that contains the Color control client cluster to use to send the command
		ColorX	0-0xFFFF X co-ordinate of the destination color
		ColorY	0-0xFFFF Y co-ordinate of the destination color
		Time	0-0xFFFF Time in tenths of a second the output should move.
		RateX	-32767 – 32768 Steps per second for the X coordinate
		RateY	-32767 – 32768 Steps per second for the Y coordinate
		StepX	-32767 – 32768 The size of each step for the X coordinate
		StepY	-32767 – 32768 The size of each step for the Y coordinate
		ColorTempMireds	0-0xFFFF Destination color temp in Mireds.

7.4.6.12.4 ATZBCOLENH

Syntax	ATZBTCOLENH=EnhMoveToHue,<DevId>,<ClientEndpoint>,<Hue>,<Direction>,<Time> ATZBTCOLENH=EnhMoveHue,<DevId>,<ClientEndpoint>,<Mode>,<Rate> ATZBTCOLENH=EnhStepHue,<DevId>,<ClientEndpoint>,<Mode>,<StepSize>,<Time> ATZBTCOLENH=EnhMoveToHueSat,<DevId>,<ClientEndpoint>,<Hue>, <Saturation>,<Time>		
Description	This command sends a Color Control cluster EnhMoveToHue / EnhMoveHue / EnhStepHue / EnhMoveToHueSat command to the device specified.		
Parameters	EnhMoveToHue / EnhMoveHue / EnhStepHue / EnhMoveToHueSat	DevId	Valid device index The device to send the EnhMoveToHue / EnhMoveHue / EnhStepHue / EnhMoveToHueSat command to
		ClientEndpoint	Valid end point Endpoint that contains the Color control client cluster to use to send the command
		Direction	0-3 Direction to move as Shortest(0), Longest(1), Up(2) or Down(3)
		Hue	0-0xFF Hue for the light to move to
		Time	0-0xFFFF Time in tenths of a second the output should move.
		Mode	0-3 Mode for the move command as Stop (0), Up (1) or Down (3).
		Rate	0-0xFF Time for each step
		StepSize	0-0xFF The size of each step
		Saturation	0-0xFF Saturation value for the light to move to

7.4.6.12.5 ATZBCOLEXT

Syntax	ATZBTCOLEXT=ColorLoopSet,<DevId>,<ClientEndpoint>,<Flag>,<Actmode>, <DirectMode><Time>,<StartHue> ATZBTCOLEXT=StopMoveStep,<DevId>,<ClientEndpoint> ATZBTCOLEXT=MoveColorTemp,<DevId>,<ClientEndpoint>,<Rate>,<MinMireds>, <MaxMireds> ATZBTCOLEXT= StepColorTemp,<DevId>,<ClientEndpoint>,<Mode>, <StepSize>,<Time>,<MinMireds>,<MaxMireds>		
Description	This command sends a Color Control cluster ColorLoopSet / StopMoveStep / MoveColorTemp / StepColorTemp command to the device specified.		
Parameters	ColorLoopSet / StopMoveStep / MoveColorTemp / StepColorTemp	DevId	Valid device index The device to send ColorLoopSet / StopMoveStep / MoveColorTemp / StepColorTemp command to
		ClientEndpoint	Valid end point Endpoint that contains the Color control client cluster to use to send the command

Syntax	ATZBTCOLEXT=ColorLoopSet,<DevId>,<ClientEndpoint>,<Flag>,<Actmode>, <DirectMode><Time>,<StartHue> ATZBTCOLEXT=StopMoveStep,<DevId>,<ClientEndpoint> ATZBTCOLEXT=MoveColorTemp,<DevId>,<ClientEndpoint>,<Rate>,<MinMireds>, <MaxMireds> ATZBTCOLEXT= StepColorTemp,<DevId>,<ClientEndpoint>,<Mode>, <StepSize>,<Time>,<MinMireds>,<MaxMireds>		
		Flag	0-0xFF Flags to control how the Color attributes should be updated: Bit 0: Update the ColorLoopActive attribute. Bit 1: Update the ColorLoopDirection attribute. Bit 2: Update the ColorLoopTime attribute. Bit 3: Update the ColorLoopStartEnhancedHue attribute.
		ActMode	0-2 Action to be taken as de-activate (0), activate from color loop start enhanced hue (1), or activate from enhanced current hue (2).
		DirectMode	0-1 Direction for color loop as decrement (0) or increment (1).
		StartHue	0-0xFFFF Starting HUE for the color loop
		Time	0-0xFFFF Time in tenths of a second the output should move.
		Mode	0-3 Mode for the move command as Stop (0), Up (1) or Down (3).
		Rate	0-0xFFFF The rate the output should move
		StepSize	0-0xFF The size of each step
		MinMireds	0-0xFFFF The minimum Mireds for this move operation.
		MaxMireds	0-0xFFFF The maximum Mireds for this move operation.

7.4.7 Thread commands

7.4.7.1 ATTHHELP

Syntax	ATTHHELP=
Description	Displays list of Thread commands
Example	ATTHHELP=

7.4.7.2 ATTHSERVICE

Syntax	ATTHSERVICE=initialize,<ExtAddrSufix>,<Type> ATTHSERVICE=shutdown ATTHSERVICE=start ATTHSERVICE=stop		
Description	Initialize the thread interface and sets default configurations Shutdown the thread interface Attempts to become active on the specified network. This command will fail if network data has not been obtained, either by setting it manually or using commissioning. Disconnects from the current Thread network		
Parameters	initialize	ExtAddrSufix	24-bit value: deprecated
		Type	Determines the type of device to iniitlaize as either a router capable device (0) or Sleepy end device (1)
	shudown	None	
	start	None	
	stop	None	

7.4.7.3 ATTHSETPOLLPERIOD

Syntax	ATTHSETPOLLPERIOD=<period>	
Description	Set the maximum Poll period	
Parameters	Period	Integer value. Number of seconds

7.4.7.4 ATTHGETCONFIG

Syntax	ATTHGETCONFIG=getDevConfig|getNetConfig	
Description	Prints out the current device information, including EUI-64, child timeout, and operating flags. Prints the current Thread network information	
Parameters	getDevConfig	None
	getNetConfig	None

7.4.7.5 ATTHSETCONFIG

Syntax	ATTHSETCONFIG=setExtAdd,<extAddr > ATTHSETCONFIG=setChildTimeout,<Timeout> ATTHSETCONFIG=setChannel,<chann>		
Description	Sets the Extended address for this device. Normally, this is not necessary because Thread specifies the use of a randomized EUI-64. Sets the child timeout in seconds for the device. The child timeout is how long the parent should wait before considering this device to have left the network due to lack of activity (polls, child updates). Sets the channel of the appropriate thread network		
Parameters	setExtAdd	ExtAddr	64-bit value: Extended address of the device.
	setChildTimeout	Timeout	32-bit value: child timeout in seconds
	setChannel	chann	11-26: sets the channel of the network

7.4.7.6 ATTHSETLINKMOD

Syntax	ATTHSETLINKMOD=< RxOnWhenIdle>,< UseSecureDataRequests>,<isFFD>, < RequireNetworkData>	
Description	Sets the link mode to be used when connecting to the Thread network.	
Parameters	RxOnWhenIdle	0-1: Flag indicating if the device's receiver is on when the device is idle
	UseSecureDataRequests	0-1: Flag indicating if the device uses secure data requests
	IsFFD	0-1: Flag indicating if the device is an FFD
	RequireNetworkData	0-1: Flag indicating if the device requires full network data

7.4.7.7 ATTHSETCONFIGEXT

Syntax	ATTHSETCONFIGEXT=setPANID,<PANID> ATTHSETCONFIGEXT=setExtPANID,<ExtPANID> ATTHSETCONFIGEXT=setNetName,<name> ATTHSETCONFIGEXT=setMasterKey,<key>		
Description	Sets the PAN ID of the appropriate thread network Sets the Extended PAN ID of the appropriate thread network Sets the network name of the appropriate thread network Sets the master key of the appropriate thread network		
Parameters	setPANID	PANID	0-FFFD: The 16-bit PAN ID of the network
	setExtPANID	ExtPANID	64-bit value: The 64-bit PAN ID of the network
	setNetName	name	String: string representing the network name
	setMasterKey	Key	16-bit hexadecimal key: The master key in hexadecimal format

7.4.7.8 ATTHCOMMISSIONER

Syntax	ATTHCOMMISSIONER=commStart ATTHCOMMISSIONER=commStop ATTHCOMMISSIONER=commAddJoiner,<Passphrase>,<ExtAddr>,<Timeout> ATTHCOMMISSIONER=commDelJoiner		
Description	Starts acting as a commissioner on an active network Stops acting as a commissioner Adds a joiner device to the steering information Removes the previously added joiner		
Parameters	commStart	None	
	commStop	None	
	commAddJoiner	Passphrase	String: passphrase of the joiner
		ExtAddr	64-bit address: joiner extended address
		Timeout	16-bit value: timeout for the device to join in seconds
	commDelJoiner	ExtAddr	64-bit address: joiner extended address

7.4.7.9 ATTHJOINER

Syntax	ATTHSETJOINER=joinerStart,<Passphrase> ATTHSETJOINER=joinerStop		
Description	Scans for and attempts to join a network using passphrase (PSKd) Stops the joining process		
Parameters	joinerStart	Passphrase	string: passphrase to use when joining
	joinerStop	None	

7.4.7.10 ATTHBORDERROUTER

Syntax	ATTHADDBORDROUTER=addBordRouter,<prefix>,<prefixLen>,<Pref>,<Isstable>,<Flags> ATTHADDBORDROUTER=delBordRouter,<prefix>,<prefixLen>			
Description	Make device act as a border router by adding prefix to network data Removes a border router prefix information			
Parameters	addBordRouter, delBordRouter	Prefix	IPv6 address: Prefix for use to route on external network	
		PrefixLen	0-128: Length of the prefix	
		Pref	0-2: Router preference	
		IsStable	0-1: Flag indicating if this is stable network data	
		Flags	32-bit integer:	Configuration flags for the border router
				Bit3 : SLAAC preferred
				Bit4: SLAAC valid
				Bit5: Supports DHCP v6 address configuration
				Bit6: supports DHCPv6 other configuration
				Bit7: default route

7.4.7.11 ATTHEXTROUTE

Syntax	ATTHEXTROUTE=addExtRoute,<prefix>,<PrefixLen>,<IsStable> ATTHEXTROUTE=delExtRoute,<ExtAddrSuffix>,<type>		
Description	Adds an external route to the network data Removes an external route from the network data. The prefix information should match the previously used in the AddExternalRoute		
Parameters	addExtRoute	Prefix	IPv6 address: Prefix to the external router
		PrefixLen	0-128: Length of the prefix
		IsStable	0-1: Flag indicating if this is stable network data
	delExtRoute	ExtAddrSuffix	24bit value: Deprecated
		Type	0-1: Determines the type of device to iniitlaize as either a router capable device (0) or Sleepy end device (1)

7.4.7.12 ATTHREGSERVDATA

Syntax	ATTHREGSERVDATA=	
Description	Forces registration of any pending network data updates. This normally happens after a short timeout after updating the network data, to avoid flooding the network with spurious traffic.	
Parameters	None	

7.4.7.13 ATTHUSEDEFINFO

Syntax	ATTHUSEDEFINFO=	
Description	This is a convenience function to set up some safe default parameters for a Thread Network. After it is issued, Start can then be used to connect to (or form) the network	
Parameters	None	

7.4.7.14 ATTHBORDERAGENT

Syntax	ATTHBORDERAGENT=[interface] ATTHBORDERAGENT=	
Description	This allows the device to act as a border agent if already connected to Wi-Fi network Stops acting as a border agent	
Parameters	Interface	String: Name of the interface for to use for the border agent. Should be either “wlan0” or “wlan1

7.4.7.15 ATTHUPDATEPSK

Syntax	ATTHUPDATEPSK=<Passphrase >	
Description	This updates the PSKc used for off-mesh commissioning. If the network data has already been set and not started, the PSKc can also be regenerated using a passphrase. This passphrase is entered on the Commissioning device (a smartphone running a commissioning app).	
Parameters	Passphrase	String: commissioner passphrase

7.4.7.16 ATTHCLEARPERSIST

Syntax	ATTHCLEARPERSIST=	
Description	This clears the persistent Flash data that normally stores the network information for a Thread network. Note that this function will not clear the network data already cached by Thread. However, if the device is reset after issuing the command, its network data will be completely cleared.	
Parameters	None	

7.4.7.17 ATTHLEADERROUTER

Syntax	ATTHLEADERROUTER=<mode>	
Description	Attempts to upgrade to a router Attempts to become a Leader	
Parameters	Mode	3. Router, 1- Leader

7.4.7.18 ATTHIPSTACKINTEG

Syntax	ATTHIPSTACKINTEG=<mode>	
Description	Enable/disables the use of the QAPI sockets with the thread interface	
Parameters	Mode	0-Disable, 1-Enable

7.4.7.19 ATTHUNICAST

Syntax	ATTHUNICAST=add,<address>,<prefix_len>,<prefer> ATTHUNICAST=remove,<address>		
Description	Adds/Removes a static IP to/from the thread interface		
Parameters	add / remove	address	IPv6 address
		PrefixLen	0-128: Length of the prefix
		Prefer	0-1: Router preference

7.4.7.20 ATTHMULTICAST

Syntax	ATTHMULTICAST=sub|unsub,<address>		
Description	Subscribes/Unsubscribes the to/from a multicast address		
Parameters	sub / unsub	address	IPv6 address

7.4.7.21 ATTHPINGPROV

Syntax	ATTHPINGPROV=ping,<mode> ATTHPINGPROV=prov,<URL>		
Description	Subscribes / Un-subscribes the to/from a multicast address		
Parameters	ping	mode	0-disable, 1-enable
	prov	URL	URL link

7.4.7.22 ATTHLOGGING

Syntax	ATTHLOGGING=<mode>	
Description	Sets the Commissioner provision URL for joiner filtering	
Parameters	mode	0-disable, 1-enable

8 HTC demo

This chapter describes communication over SPI or SDIO between a system and a QCA402x device.

This chapter provides details of **RTOS-based System** support on QCA402x. In this mode, an additional system is connected to the QCA402x.

This additional system serves as a SPI Master or SDIO Master and connects to QCA402x which serves as a SPI Slave or SDIO Slave. Such a system is called as an “external Host” or “Host” system and the QCA402x is called as a “Target” system.

It is possible for a single host system to connect to multiple targets. Each target connects to exactly one host system.

It describes the software layers involved in host-target communications and how to build and execute *demonstration* software with:

- a host-side application that runs on an RTOS-based host system plus
- a target-side application that runs on a QCA402x.

The scope of this document is limited to **RTOS-based** host systems.

8.1 Software architecture

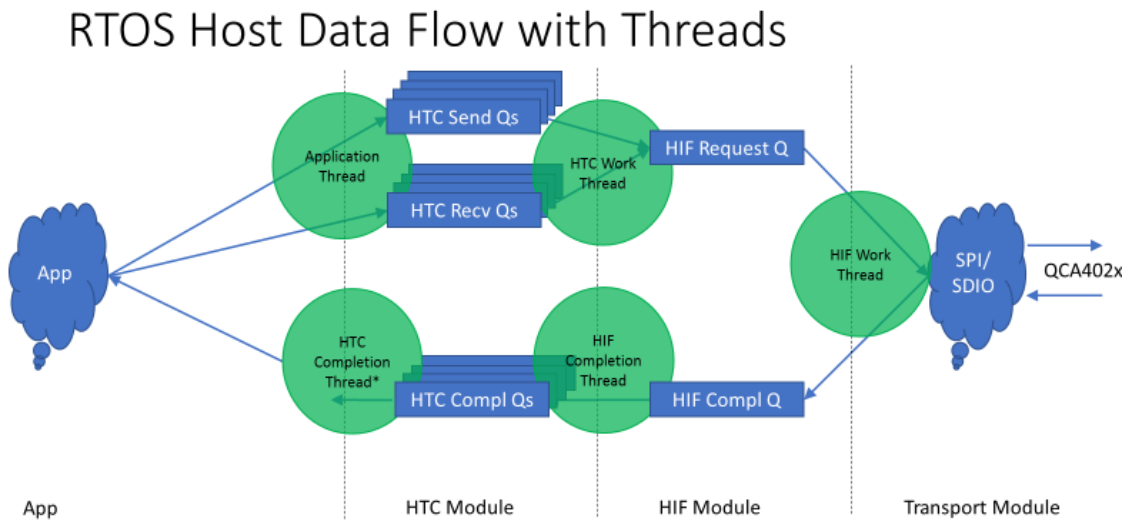
On the external host system, software that communicates with a QCA402x target is arranged into layers:

- **Application**: A customer-specific application provides top-level application-specific semenatics. The application invokes HTC APIs in order exchange messages with the QCA402x target.
- **HTC**: The Host-Target Communication layer understands the protocol used to communicate with a QCA402x target. It provides HTC APIs to the application which sits above it. The HTC layer understands QCA402x’s SPI and SDIO registers and address space so that it knows how to handle flow control (via TX Credit Counters), End-Of-Message signaling, RX Lookahead, interrupt management, and data transfers through FIFO mailboxes with appropriate block-size padding. HTC sits on top of the HIF layer which handles low-level communication.
- **HIF**: The Host InterFace layer provides an asynchronous read/write API for use by HTC. Using this API, HTC can remain completely independent of the details of SPI vs SDIO. The main job of HIF is to keep the SPI/SDIO interconnect occupied with useful activity – to avoid idle time on the bus.
- **Transport**: The transport layer provides an API for use by HIF, which is bound to either a SPI implementation or an SDIO implementation. If a host must simultaneously support *both* SPI- and SDIO-connected QCA402x’s device, then the transport layer handles that

multiplexing. Calls to system-specific SPI and SDIO device drivers are made from this layer only; this allows HIF to remain platform-independent.

- **SPI/SDIO driver**: This layer understands the SPI or SDIO host controller and details of host-side DMA hardware. It is a host platform-specific layer or stack.

Although these layers are described on the host-side, there are also analogous layers on the target-side. In particular, there might be a target-side application which exchanges messages with a host-side application. The host-side application invokes host-side HTC APIs, whereas the target-side application invokes target-side HTC QAPIs to implement this message exchange.



8.2 Software overview of SDK

For the HTC demo, the QCA402x SDK includes:

- A host-side application, **sdhost**, which demonstrates communication with a target
- A target-side application, **HTC_demo**, which demonstrates communication with a host
- **Host-side source code** with an implementation of HTC, HIF, and transport layers with both SPI and SDIO transport bindings
- **Target-side binary libraries** containing implementations of HTC along with the rest of the communication stack.
- A host-side header file, **htca.h**, which includes the HTC API definition
- A target-side header file, **qapi_htc_slave.h**, which includes the target-side HTC QAPI definition.

8.3 Build and flash HTC demo

This section describes how to build the host side demo software, targer side demo software with flashing instruction. HTC demo build is supported on Linux machine.

8.3.1 How to build HTC demo software

This section presents the build instructions for the host demonstration software build instruction for the host ported to QCA402x and targert demonstration software which runs on QCA402x.

8.3.1.1.1 Build configuration

Set the following environmental variable based on the SPI/SDIO demo:

- export CHIPSET_VARIANT=qca4020 or qca4024 depending on board type
- export BOARD_VARIANT=cdb or carrier depending on the board type.
- export RTOS=freertos or threadx

For SDIO:

- export CFG_INTERCONNECT_SDIO=true
- export CFG_INTERCONNECT_SPI=false

For SPI:

- export CFG_INTERCONNECT_SPI=true
- export CFG_INTERCONNECT_SDIO=false

See the supported toolchains section or installing the appropriate toolchain to build the demo.

8.3.1.1.2 Build Host-side sdhost application

cd $SDK/exthost/rtos/transport

make

cd $SDK/exthost/rtos/hif

make

cd $SDK/exthost/rtos/htca

make

cd $SDK/exthost/rtos/demo/sdhost/build/gcc

make prepare

Disable sleep by modifying "***devcfgSleepData***" value to 0 in file
$SDK/exthost/rtos/demo/sdhost/src/export/DevCfg_master_devcfg_out_cdb.xml

```
<driver name="Sleep">
  <global_def>
    <var_seq name="devcfgSleepData" type="0x00000003">
      0, 0, 0,
      180, 166, 200,
```

```
    end
  </var_seq>
 </global_def>
 <device id="0x02000018">
  <props id="0x1" oem_configurable="false" type="0x00000014"> devcfgSleepData </props>
  <props id="0x2" oem_configurable="false" type="0x00000002"> 0 </props>
  <props id="0x3" oem_configurable="false" type="0x00000002"> 632 </props>
  <props id="0x4" oem_configurable="false" type="0x00000002"> 96 </props>
 </device>
</driver>
```

While using SDIO as the interconnect:

SDIO Host controller configuration

Modify "**sdio_gpio_arr**" and "**sdio_gpio_off_arr**" to the values highlighted in the file:
$SDK/quartz/demo/QCLI_Demo/src/export/DevCfg_master_fom_out_cdb.xml

```
<driver name="sdio">
 <global_def>
  <var_seq name="sdio_gpio_arr" type="0x00000002">
    0x12, 3, 0, 1, 1,
    0x13, 3, 0, 2, 1,
    0x14, 3, 0, 2, 1,
    0x15, 3, 0, 2, 1,
    0x16, 3, 0, 2, 1,
    0x17, 3, 0, 2, 1,
    end
  </var_seq>
  <var_seq name="sdio_gpio_off_arr" type="0x00000002">
    0x12, 3, 0, 1, 1,
    0x13, 3, 0, 2, 1,
    0x14, 3, 0, 2, 1,
    0x15, 3, 0, 2, 1,
    0x16, 3, 0, 2, 1,
    0x17, 3, 0, 2, 1,
    end
  </var_seq>
 </global_def>
```

<device id="0x03000006">

cd $SDK/exthost/rtos/demo/sdhost/build/gcc

make

8.3.1.1.3 Build target-side HTC_demo application

cd $SDK/quartz/demo/HTC_demo/build/gcc

make prepare

Disable sleep by modifying "***devcfgSleepData***" value to 0 in file ***$SDK/exthost/rtos/demo/sdhost/src/export/DevCfg_master_devcfg_out_cdb.xml***

<driver name="Sleep">

<global_def>

<var_seq name="devcfgSleepData" type="0x00000003">

0, 0, 0,

180, 166, 200,

end

</var_seq>

</global_def>

<device id="0x02000018">

<props id="0x1" oem_configurable="false" type="0x00000014"> devcfgSleepData </props>

<props id="0x2" oem_configurable="false" type="0x00000002"> 0 </props>

<props id="0x3" oem_configurable="false" type="0x00000002"> 632 </props>

<props id="0x4" oem_configurable="false" type="0x00000002"> 96 </props>

</device>

</driver>

cd $SDK/quartz/demo/HTC_demo/build/gcc

make

8.3.2 Flash host/target demo application

To flash host/target demo application, copy the host and target images to windows PC and follow the instructions in the Flash the image section.

8.4 Run demonstration software

Demonstration software is easy to use; it requires no user input.

To run demonstration software, program the host-side sdhost software to the host board and program the target-side HTC_demo software to the target board. Then, power cycle or reset both boards and the test runs automatically.

The Autoboot mode section provides information on setting up the serial console.

NOTE: Ensure that no JTAG jumpers are connected on the CDB2x boards.

NOTE: It is recommended to use power adaptor to supply power to the board rather than using micro USB cable connected to a PC.

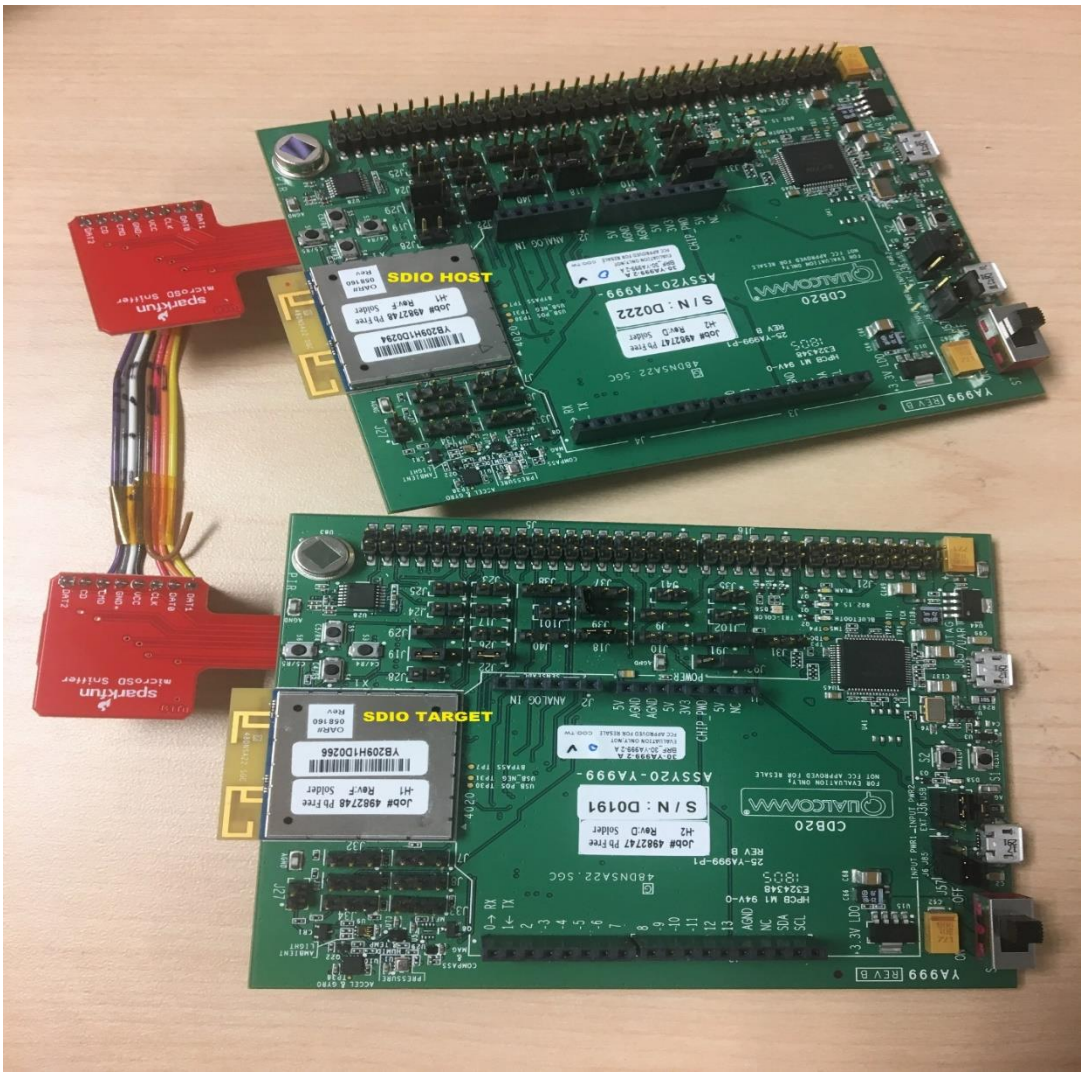
8.4.1 SPI connection between two QCA4020 boards

The following is the GPIO/Pin connection information to connect cables between two CDB20 boards to run SPI demonstration.

SPI Target			SPI Host	
PIN	**GPIO**		**PIN**	**GPIO**
J5.31	GPIO19_BE	CS	J5.2	GPIO24_BE
J5.29	GPIO18_BE	CLK	J5.4	GPIO25_BE
J5.39	GPIO23_BE	MOSI	J5.6	GPIO26_BE
J5.33	GPIO20_BE	MISO	J5.8	GPIO27_BE
J5.35	GPIO21_BE	INTR	J5.35	GPIO21_BE
J5.40	GPIO40_BE	GND	J5.40	GPIO40_BE

8.4.2 SDIO connection between two QCA4020 boards

SDIO connection between two CDB2x boards is made using the MicroSD-to-MicroSD cable as shown in the following image:



8.5 HTC host porting guidelines

The implementation of HTC/HIF/Transport that is provided in the QCA402x SDK is bound to the reference host RTOS system which is a QCA402x host running standard SDK software. However, this reference software must be portable to other systems. This chapter surveys the items that might require attention when porting this code.

The reference source code is 32-bit Little-endian. When porting to 64-bit or Big-endian, various changes are required.

The reference source code uses QuRT, which is Qualcomm's Real Time Operating System (RTOS) abstraction. When porting this code to a different system, decide to port QuRT APIs to the platform or choose to replace QuRT calls with native RTOS calls.

The reference source code uses QAPI calls (qapi_*) for low-level system services. These are especially found in the Transport layer which makes SPI QAPI calls to the “SPIM” (SPI Master) driver and to the “SDCC” (SDIO Card Controller) driver.

The reference source uses types like qbool_t and uint16_t. It might be required to define these types or replace them with native types on the platform.

The reference source operates entirely in kernel mode. To support a user mode application, an additional layer of software is required to handle transitions between user and kernel mode.

The platform may place restrictions on DMA that are not present on the reference system. For instance, some systems do not permit DMA directly to/from stack-based buffers.

The platform may support virtual memory, which presents to this software as virtually contiguous but physically discontinuous buffers. Such systems are likely to support SPI/SDIO controllers with scatter/gather; but this may require additional support in the transport layer.

The platform may require set up to enable SPI/SDIO, to map hardware controllers to appropriate pins, and so on. It may require adjustments to the heap, to permit allocation of buffers.

Generally, the top layers – HTC and HIF – must be easy to port.

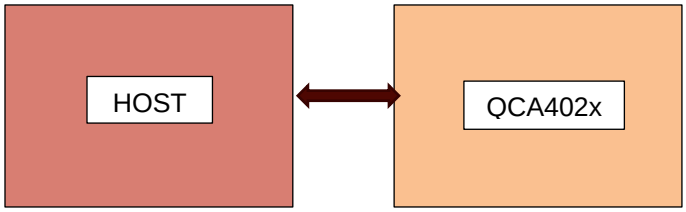
9 Hosted mode demo

An external host running Linux can interface with a QCA402x platform over a serial interface and invoke narrowband features using a standardized Qualcomm API interface. The SDK includes all the software components required to connect to a QCA402x device.

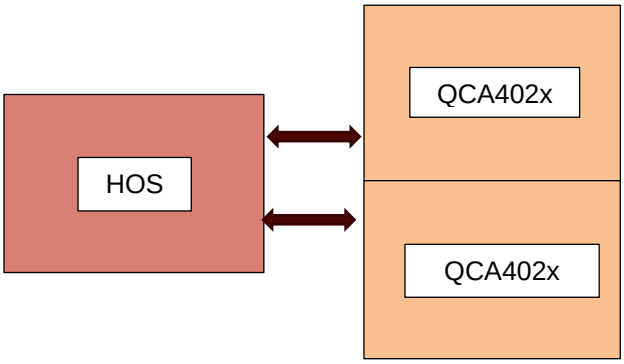
9.1 Supported configurations

NOTE: Here the target refers to QCA402x and host refers to a Linux-based system.

Configuration 1. One target, one active interface



Configuration 2. Multiple targets, multiple active interfaces



9.1.1 Supported features

Following narrow band protocols are supported-

- BLE -Both versions 4.2 and 5.0 are supported.
- ZigBee
- Thread- includes support for 6LoWPAN

9.1.2 Other features

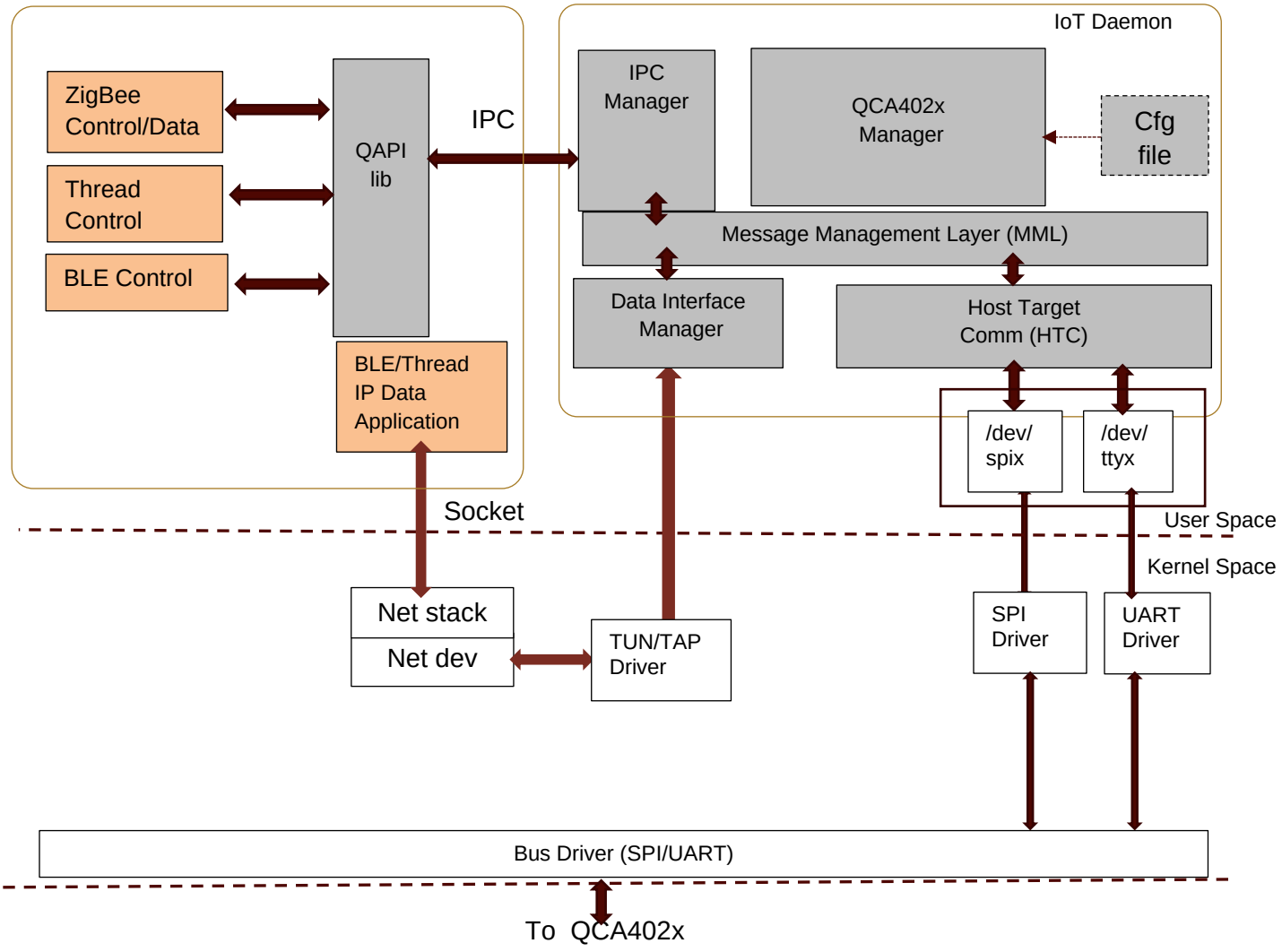
- QCA402x firmware upgrade over the serial bus
- Coexistence support for narrowband technologies.

9.1.3 Supported interfaces

A host CPU can connect to QCA402x module over the following interfaces:

- UART: up-to 2 MBaud with flow control supported.
- SPI: SPI Slave frequency up to 48 MHz supported.

9.1.4 System architecture



The preceding figure shows the high-level architecture block diagram. All software modules run in user space and make use of standard Linux POSIX library and APIs.

9.1.5 System modules

9.1.5.1 IoT Daemon

This is the core module that runs on the Linux host in user space.

The IoT daemon:

- Manages interaction with QCA402x target via supported serial interfaces.
- Manages IPC to and from other user space application processes via POSIX message queues.
- Manages the TUN/TAP interface used for IP data by BLE and thread (6LoWPAN).
- Manages heartbeat. A QCA402x device sends a periodic heartbeat message to the daemon. If the daemon does not receive the message for a preconfigured time, it assumes that QCA402x device has failed and informs the associated applications.

9.1.5.1.1 Daemon Config file

The IoT Daemon is highly configurable, and its behavior can be modified by adjusting the 'iotd_config.ini' file provided in the SDK.

The configuration is divided into two categories:

- SYSTEM- Includes system-wide parameters
- INTERFACE- Includes parameters specific to an interface

The following is the summary and a brief description of important daemon configuration parameters:

Parameter Name	Type	Description
num_device	SYSTEM	Number of QCA402x devices attached to host. Default value is 1.
num_interface	SYSTEM	Number of serial bus interfaces in use.
num_clients	SYSTEM	Maximum number of applications that the daemon can support simultaneously
num_buffer	SYSTEM	Number of buffers used by daemon
heart_beat_enable	SYSTEM	Flag to enable Heartbeat monitoring
heart_beat_interval	SYSTEM	Interval in seconds beyond which a heartbeat miss triggers a failure
dbg_level	SYSTEM	Debug message verbosity level, default is 1
enable	INTERFACE	1 - Enable interface, 0- disable interface
type	INTERFACE	0 – UART interface, 1 – SPI Interface
speed	INTERFACE	Baud rate for UART, Frequency in Hz for SPI
flow_control	INTERFACE	0 – Disable Flow Control, 1 – Enable Flow Control, only applicable for UART Interface
name	INTERFACE	Interface device file e.g. /dev/tty0

9.1.5.1.2 Build the Daemon

A reference daemon Makefile is provided in SDK at [target]/exthost/Linux/daemon. Update compiler flags for different cross compilers.

```
localhost:~/target/exthost/Linux/daemon$ make
```

9.1.6 QAPI library

This library exposes the QCA402x QAPI set to an application. Additionally, it performs the following tasks:

- Abstracts all communications with the QCA402x Manager Daemon. It uses a POSIX message queue to exchange messages with the daemon.
- Serializes all API calls and transmits them to the daemon in the transmit path.
- Manages all asynchronous events and callbacks. It creates a dedicated event handler thread that blocks on the message queue receive function provided by IPC framework.

See QCA402x QAPI reference specification for information on supported QAPIs.

Build QAPI library

A sample Makefile is provided in the SDK at [target]/exthost/Linux/qapi.

```
localhost:~/target/exthost/Linux/qapi$ make
```

9.1.7 Application

Any application that intends to use the features and technologies provided by a QCA402x chipset must link with the QAPI library. The library provides the QAPI definitions that map to remote QAPI calls on the target. The application developer need not be aware of any internal details of the QAPI library or the QCA402x daemon. All details such as message formats, Inter Process Communication, packet serialization, synchronization and so on are abstracted out by the QAPI library interface.

To initialize the QAPI library infrastructure and establish a connection to the daemon, an application must invoke the following APIs:

API	Parameters	Description
qapi_Qs_Init	***serverName*** – daemon msg-q name, default "/iotdq" ***maxMsgSize*** – Maximum message size, suggested calue = 2000 ***maxMsgCnt*** – Maximum number of messages to queue, Suggested - 10	Initializes QAPI interface, establishes a connection (message queue) with the daemon.
qapi_Qs_Register_Cb	***cb*** – application callback function. The format for callback is – typedef void (* eventCb_t)(uint32_t eventId, void *param); ***Parameter***- pointer to user specified parameter	Registers callbacks with QAPI library. The callback is invoked for certain events

9.1.7.1 Build host application

Any application that intends to use QAPIs must link with the QAPI library.

See the sample application Makefile provided in the SDK at: [target]/exthost/Linux/apps/NB_demo/build.

```
localhost:~/target/exthost/Linux/app/NB_QCLI_demo/build$ make
```

9.1.8 QCA402x firmware

The SDK includes source and binary for QCA402x firmware image. The firmware is configured to use SPI interface by default. User has the option to reconfigure and rebuild the firmware image to use UART.

To reconfigure the firmware, edit the following file:

`[target]\quartz\demo\HostedMode_demo\src\export\DevCfg_master_fom_out_[CDB].xml`

Search for `"exhost_scheme"` driver.

Modify the following parameters:

Prop ID	Parameter	Possible values	Explanation
0x100001	Heart Beat	Enable – 1, Disable - 0	If enabled, it will send a heart-beat message to host
0x100002	Interval		Heart-beat interval in ms
0x110001	Uart mode	Enable – 1, Disable - 0	Enable or disable UART interface
0x110003	UART port	High Speed UART – 0, Debug UART - 1	Select UART interface based on board configuration
0x110004	UART Baud rate		
0x110005	UART Flow control	Disable – 0, Enable - 1	
0x210001	SPI mode	Disable – 0, Enable - 1	Control SPI mode

```
<driver name="exhost_scheme">
   <device id="0x03000019">
      <props id="0x100001" id_name="EXHOST_SYS_SCHEME_HEART_BEAT_ENABLE"
oem_configurable="true" helptext="0: Heart Beat Disable; 1:Heart Beat
Enable" type="0x00000002"> 1 </props>
      <props id="0x100002" id_name="EXHOST_SYS_SCHEME_HEART_BEAT_INTERVAL"
oem_configurable="true" helptext="Heart Beat Interval in ms"
type="0x00000002"> 1000 </props>
      <props id="0x110001" id_name="EXHOST_UART_SCHEME_PROP_ENABLE"
oem_configurable="true" helptext="0: Uart Disable; 1:Uart Enable)"
type="0x00000002"> 1 </props>
      <props id="0x110002" id_name="EXHOST_UART_SCHEME_PROP_DEVICE_ID"
oem_configurable="true" helptext="Device ID Setting" type="0x00000002"> 0
</props>
      <props id="0x110003" id_name="EXHOST_UART_SCHEME_PROP_UART_PORT"
oem_configurable="true" helptext="0: High Speed Uart Port; 1: Debug Uart
Port)" type="0x00000002"> 1 </props>
      <props id="0x110004" id_name="EXHOST_UART_SCHEME_PROP_BAUD_RATE"
oem_configurable="true" helptext="Uart Baud Rate" type="0x00000002"> 115200
</props>
      <props id="0x110005"
id_name="EXHOST_UART_SCHEME_PROP_FLOW_CONTROL_ENABLE"
oem_configurable="true" helptext="0: Uart Flow Control Disable; 1:Uart Flow
Control Enable)" type="0x00000002"> 0 </props>
      <props id="0x110006"
```

To build and flash the QCA402x firmware, see Build sample applications section.

9.1.9 Run the application

1. Ensure that the QCA402x firmware is programmed on QCA402x.
2. If UART interface is used, connect the QCA402x UART (USB) port to the Linux host UART interface.
3. Ensure that the configured interface parameters (port, baud rate etc.) match in daemon config and firmware configurations.
4. Power on the QCA402x device first.
5. Run the daemon executable on the host. The executable 'iotd' is generated in daemon/output directory. Pass the iotd_config.ini location as an input parameter.

   For example:

   ```
   ./iotd    <path_to_config>/iotd_config.ini
   ```

6. Edit iotd_config.ini file to match the QCA402x firmware image configuration.

   Example: Modify the config file as follows:

   ```
   localhost:~/target/exthost/Linux/daemon$ vi iotd_config.ini
   [INTERFACE]
   enable=1                          # 0:Disable 1:Enable
   type=0                            # 0:UART, 1:SPI, 2:SDIO
   speed=115200                      # Baud/frequency
   flow_control=0                    # UART Flow Control: 0-disable, 1-
   enable
   block_size=1024                   # SPI block Size
   name=/dev/ttyUSB2                 # File name
   device_id=0                       # Instance of Quartz device it is
   associated with
   num_service_q=6                   # Number of associated service queues
   ```

7. If the interface parameters are configured correctly, the handshake between daemon and QCA402x must succeed.

   Example:

   ```
   localhost:~/target/exthost/Linux/daemon$ sudo output/iotd
   iotd_config.ini
   IOTD: Opening config file iotd_config.ini
   Uart RX thread started for target ID 0
   Iotd Manager: Recv MGMT_MSG_HELLO resp
   Target QAPI Ver: 2.0.1    CRM  Num: 71
   ***** IOT Daemon started ****
   ```

8. Now run the sample application. A message queue is established between the application and the daemon and a list of CLI commands is now available to exercise various narrowband features.

```
localhost:~/target/exthost/Linux/app/NB_QCLI_demo/build$ sudo ./nb_demo
Sending Hello Message
Received HELLO response from the server.

> 0
QAPI Ver: 2.0.1
CRM  Num: 71
> 1
Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit
  Subgroups:
     3. BLE
     4. ZigBee
     5. Debug
     6. HMI
     7. Thread
     8. FwUp
     9. Coex
```

A CDB2x board setup

This chapter describes the procedure to set up the CBD2x development board. After the set up is complete, the preinstalled demo applications can be run to evaluate the performance of the supported hardware for the CDB2x Wi-Fi, BLE, 802.15.4 modules.

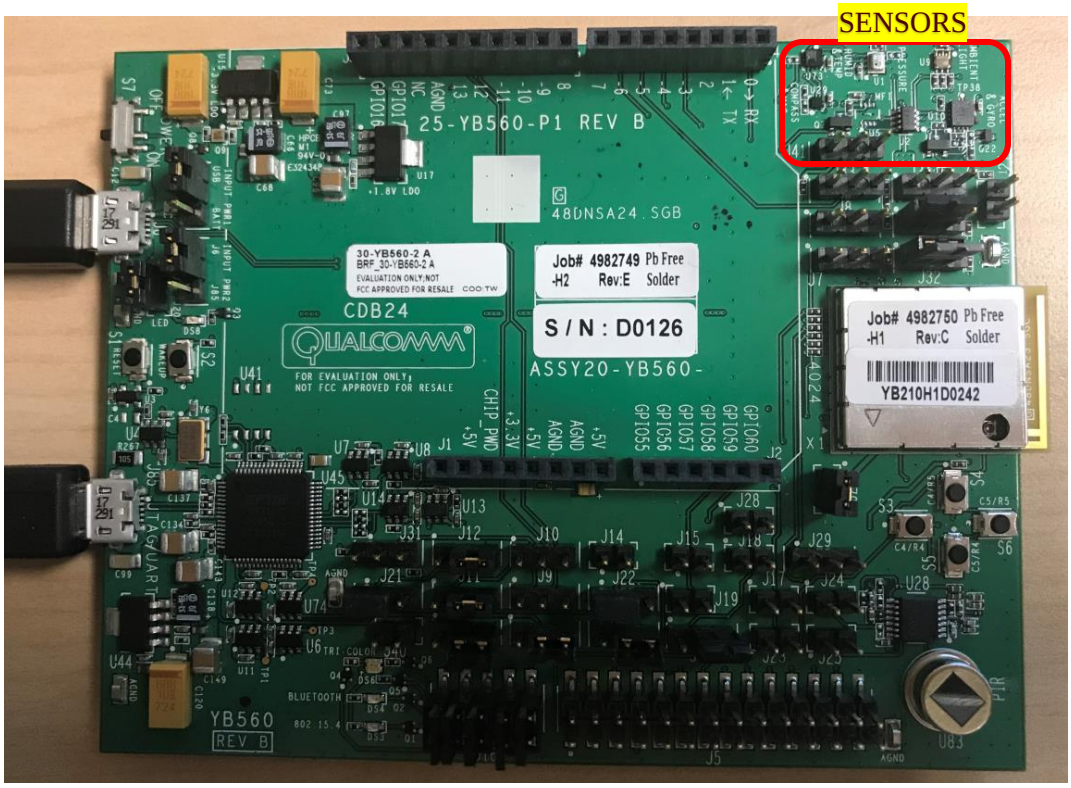


Figure A-1 CDB24 development board

Figure A-2 shows a high-level view of the jumper configurations.

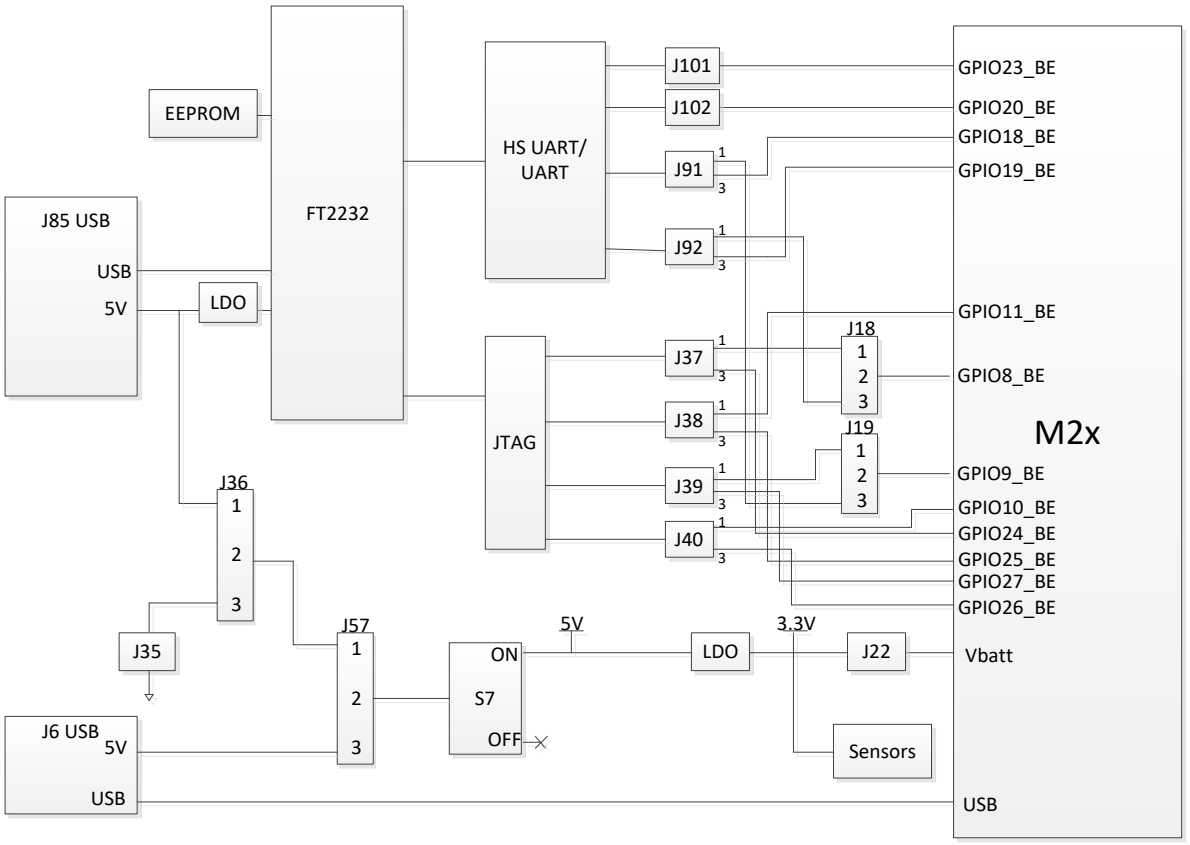


Figure A-2 CDB20 jumper configuration

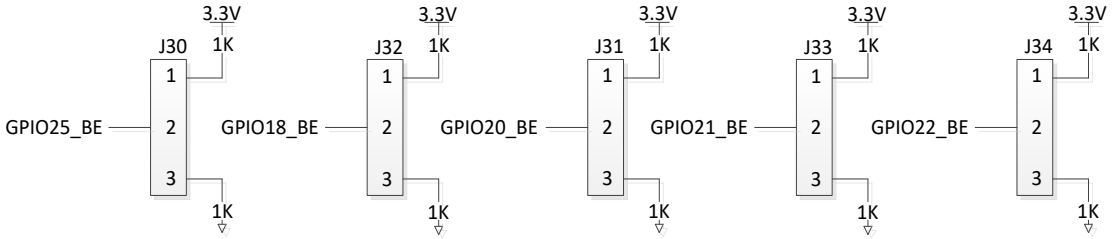


Figure A-3 CDB20 bootstrap

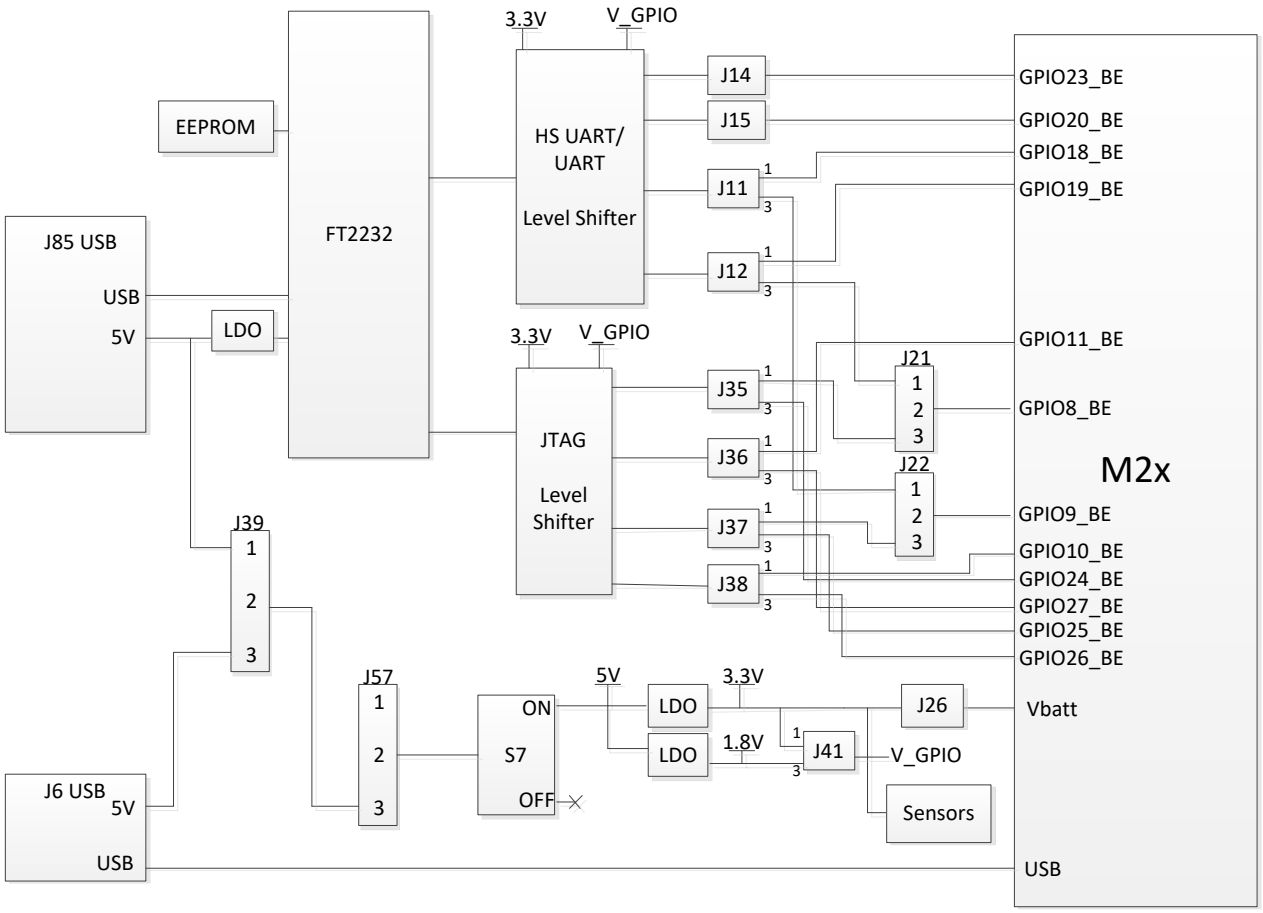


Figure A-4 CDB24 jumper configuration

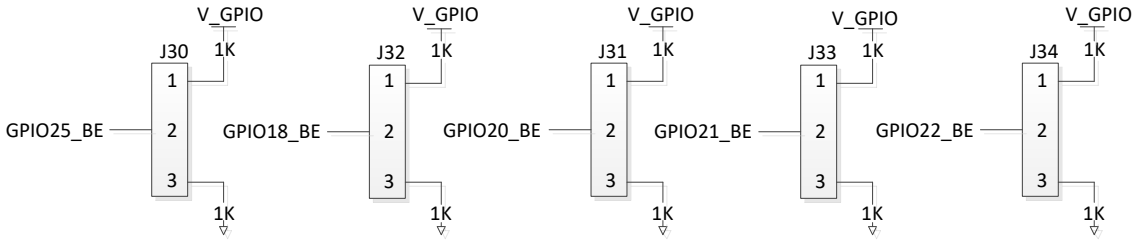


Figure A-5 CDB24 bootstrap

CAUTION: To configure the CDB2x board, ensure that the development board jumper, and switch settings are adjusted to match the information given in the following tables:

Table A-1 CDB20 customer development board jumper settings

Category	Description	Ref	Default position
Power	J6 USB Power Source (Option1)	J57	Connect pins 2 and 3 for J6 Power
	J85 USB Power Source (Option 2)	J57 J36	Connect pins 1 and 2 for J85 Power Connect pins 1 and 2 for J85 Power
	LED_PWR	J20	Connect pins 1 and 2 to power up LED
	Vbatt Select	J22	Connect pins 1 and 2 for fixed +3.3V
	Reset Switch	S7	OFF
EDL	Download Mode	J34	For autoboot mode, remove jumper For EDL mode, connect pins 1 and 2
LED	LED source	J16	Connect pins 1 and 2 for the WLAN white LED
			Connect pins 3 and 4 for the 802.15.4 red LED
			Connect pins 5 and 6 for the Bluetooth blue LED
			Connect pins 7 and 8 for the red Tri LED
			Connect pins 9 and 10 for the blue Tri LED
			Connect pins 10 and 12 for the green Tri LED
Serial	Debug UART(GPIO 9:8)	J18	Connect pins 2 and 3 for Debug UART
		J19	Connect pins 2 and 3 for Debug UART
		J91	Connect pins 1 and 2 for Debug UART
		J92	Connect pins 1 and 2 for Debug UART
JTAG	4-pin JTAG (GPIO27:24)	J30	Connect pins 1 and 2 for JTAG
		J31	For autoboot mode, remove jumper For JTAG debug/flashing, connect pins 1 and 2
		J32	Connect pins 1 and 2 for JTAG
		J37	Connect pins 2 and 3 for JTAG
		J38-J39	Connect J38 pins 2 and J39 pins 3 for JTAG Connect J38 pins 3 and J39 pins 2 for JTAG
		J40	Connect J40 pins 2 and 3 for JTAG

Table A-2 CDB24 customer development board jumper settings

Category	Description	Ref	Default position
Power	J6 USB power source (Option 1)	J39	Connect pins 2 and 3 for J6 USB power
	J85 USB power source (Option 2)	J39	Connect pins 1 and 2 for J85 USB power
	Power source	J57	Connect pins 2 and 3 for external power
	LED_PWR	J20	Connect pins 1 and 2 to power up LED
	Vbatt Select	J26	Connect pins 1 and 2 for fixed +3.3V
	Reset Switch	S7	OFF
EDL	Download Mode	J34	For autoboot mode, remove jumper For EDL mode, connect pins 1 and 2
LED	LED source	J16	Connect pins 1 and 2 for the 802.15.4 red LED
			Connect pins 3 and 4 for the Bluetooth blue LED
			Connect pins 5 and 6 for the red Tri LED
			Connect pins 7 and 8 for the blue Tri LED
			Connect pins 9 and 10 for the green Tri LED
Serial	Debug UART(GPIO 9:8)	J11	Connect pins 2 and 3 for Debug UART
		J12	Connect pins 2 and 3 for Debug UART
		J21	Connect pins 1 and 2 for Debug UART
		J22	Connect pins 1 and 2 for Debug UART
JTAG	4-pin JTAG (GPIO27:24)	J30	Connect pins 1 and 2 for JTAG
		J31	For autoboot mode, remove jumper For JTAG debug/flashing, connect pins 1 and 2
		J32	Connect pins 1 and 2 for JTAG
		J35	Connect pins 2 and 3 for JTAG
		J36	Connect pins 2 and 3 for JTAG
		J37	Connect pins 2 and 3 for JTAG
		J38	Connect pins 2 and 3 for JTAG

- For serial communication, connect the micro USB at J85 on the customer development board and connect the USB side to a USB port on a PC.
- For power and EDL mode, connect the micro USB at J6 on the customer development board and connect the USB side to a USB port on a PC.