Getting Started with Matter

The RA6W1 is a highly integrated, ultra-low-power Wi-Fi® microcontroller unit (MCU) designed to enable complete Wi-Fi solutions, while the RA6W2 extends this capability by combining Wi-Fi and Bluetooth® Low Energy (LE) in a single package for dual-connectivity applications. This FSP guide is intended for developers building Matter-compliant Wi-Fi devices (Matter Specification v1.4) using the RA6W1 EVK and FSP.

Contents

Figures

Tables

1. Terms and Definitions

AP	Access Point
ACL	Access Control List
BDX	Bulk Data Exchange
Bluetooth® LE	Bluetooth Low Energy
CASE	Certificate Authenticated Session Establishment
CD	Certification Declaration
COM	Communication Port
CSA	Connectivity Standard Alliance
DAS	Device Attestation Certificate
DPM	Dynamic Power Management
EVK	Evaluation Kit
FSP	Flexible Software Package
IDE	Integrated Development Environment
IP	Internet Protocol
MCU	Microcontroller Unit
OTA	Over the Air
PAA	Product Attestation Authority
PAI	Product Attestation Intermediate
PASE	Passcode-Authenticated Session Establishment
PMGR	Power Manger
RTOS	Real Time Operating System
SDK	Software Development Kit
SFDP	Serial Flash Discoverable Parameter
SSID	Service Set Identifier
UART	Universal Asynchronous Receiver/Transmitter
UTC	Universal Time Coordinated
TH	Test Harness

2. Introduction

Matter is a unified, open-source application-layer connectivity standard built to enable developers and device manufacturers to connect and build reliable, and secure ecosystems and increase compatibility among connected home devices. By building upon Internet Protocol (IP), Matter enables communication across smart home devices, mobile apps, and cloud services. For more information about Matter, go to the official site (https://project-chip.github.io/connectedhomeip-doc/index.html).

The Matter platform is ported on the top of RAFW generic Flexible Software Package (FSP), and the applications are built on the top of Matter platform. Using the RAFW Matter FSP, the RA6W1 or RA6W2 can be configured as a Matter accessory device which connects to the standard Matter network and can be controlled by Matter-enabled controllers. Figure 1 shows the structure of RA6W1/RA6W2 Matter.

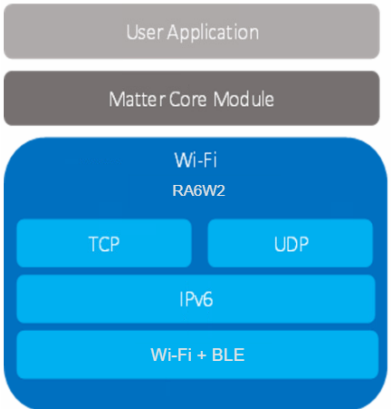

Figure 1. RA6W1/RA6W2 Matter structure

3. Hardware Setup

The hardware setup for RA6W1 and RA6W2 remains the same; therefore, RA6W1 is used as a reference example to describe the hardware configuration.

Figure 2 shows the basic element of Matter network (Hub, Device, and Controller) and how the network is linked together remotely. Matter accessory applications (lights and door locks) can be tested and developed on the device, RA6W1 Evaluation Kit (EVK). To find out more about setting up the RA6W1 EVK for the Matter application, see the Setting Up EK-RA6W1 section of RA6W2 Getting Started Guide.

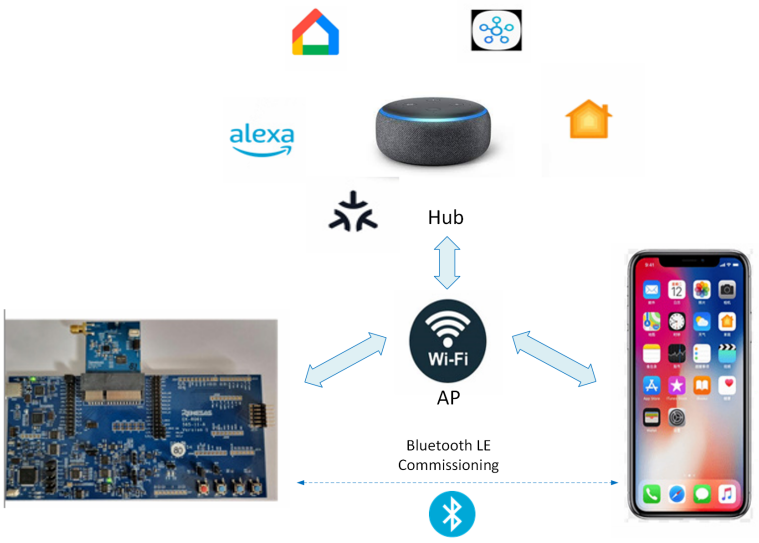

Figure 2. Matter hardware configuration

- **Hub**: Matter smart hubs such as Apple HomePod mini, Google Nest Hub, and Amazon Echo dot.
- **Controller**: Mobile applications such as Apple Home, Google Home, and Amazon Alexa.
- **Device**: RA6W1 EVK.

4. Software Setup

To develop Matter applications on the RA6W1 and RA6W2 using the Matter module, install and set up the Renesas Electronics e^2 studio IDE.
To set up the development environments:

1. Install and configure the e^2 studio IDE.
2. Download the RAFW FSP from the Renesas Electronics website.
3. Load the Matter Application project template in e^2 studio.
4. Develop New Application from Door Lock Application.
5. Customize Application Development.
6. Program firmware image.
7. Debug with J-Link.

NOTE
For software setup, see the *RA6W1 JTAG Manual* and *RA6W2 Getting Started Guide* documents.

4.1 Install e^2 studio IDE

To install the e^2 studio, download and run the e^2 studio installer on either Windows or Linux from the Renesas Electronics website (https://www.renesas.com/us/en/software-tool/e-studio). For more information, see the Install e^2 studio IDE for Windows and Linux sections of *RA6W1 Getting Started Guide*.

4.2 Download RAFW Matter FSP

Download the following files from the Renesas Electronics website (`RA FSP Pack - RAFW_FSP_Packs_<version>.zip`):

- Sample project template – `matter_wifi_at_lock_ek_ra6w1` available in `application_projects/r19an0438/Matter_Wifi_Lock_Application/" in the downloaded RA FSP Pack - RAFW_FSP_Packs_<version>.zip`.
- For using Bluetooth LE commissioning on RA6W2, use `matter_wifi_ble_lock_ek_ra6w2` available in `application_projects/r19an0438/Matter_Wifi_Lock_Application/" in the downloaded RA FSP Pack - RAFW_FSP_Packs_<version>.zip`.
- For using AT commands for Matter, use `matter_wifi_at_lock_ek_ra6w1`. available in `rafw_fsp_v1.0.0.example.1/application_projects/r19an0437/Matter_AT_Application/" in the downloaded RA FSP Pack - RAFW_FSP_Packs_<version>.zip`.

4.3 Load Matter Application Project Template in e^2 studio

1. Extract the project template `matter_wifi_lock_example_ek_ra6w1.zip`and launch e^2 studio.
2. To import project template, go to **File** > **Import**.
3. Under the **General** option, select **Rename & Import Existing C/C++ Project into Workspace**.

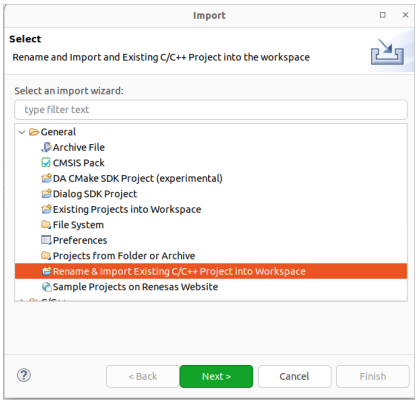

Figure 3. Rename and import

4. Browse to project template directory (unzip `matter_wifi_lock_example_ek_ra6w1.zip`) and enter **Project name**.
5. When the import is done, right-click the project, and select **Build Project** to build the project.

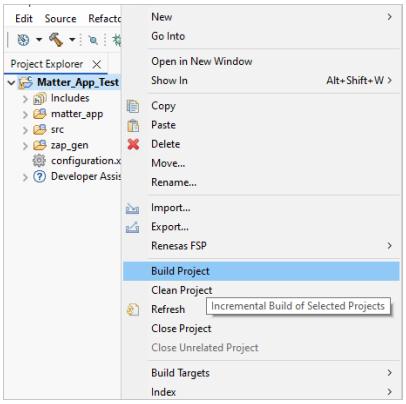

Figure 4. Build project

6. Check the console window to see if the build is proceeding.

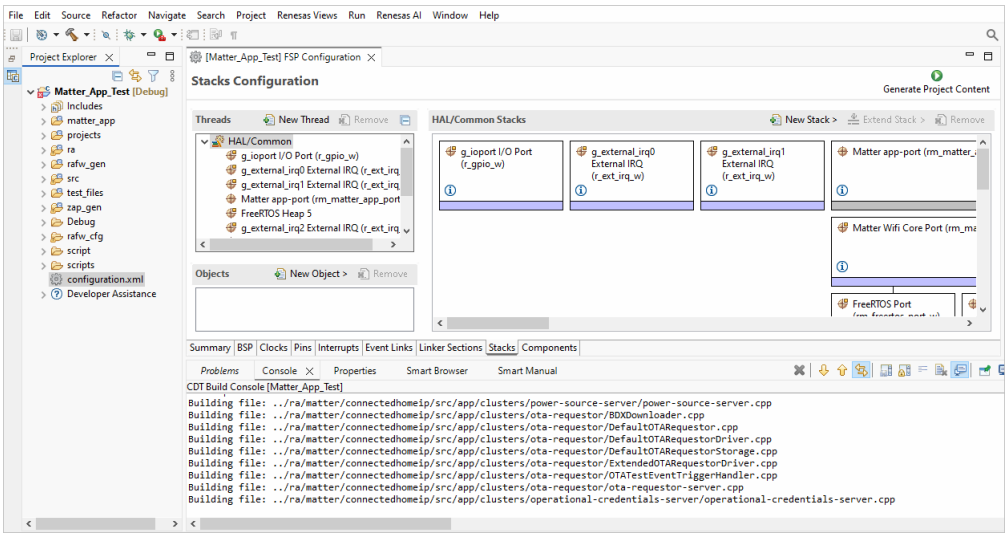

Figure 5. Console window

Figure 6 shows the build is completed successfully.

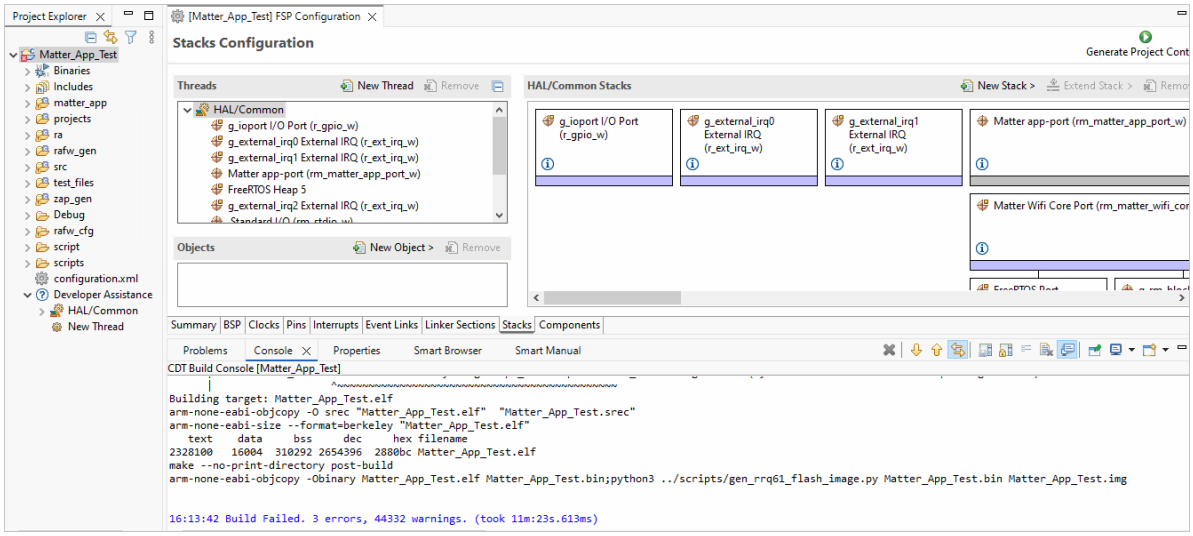

Figure 6. Build successfully

NOTE
The project **image** and **ELF** are generated correctly. After the build is completed, you may see three error messages in the output log. These are known **non-critical messages** and can be safely ignored, as they do **not** impact the final build output.

You can see the image in the project explorer as shown in Figure 7.

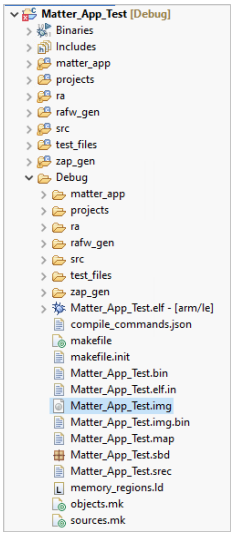

Figure 7. Image in Project Explorer tab

4.4 Develop New Application from Door Lock Application

Completion of all requirements in Section 4.4.1 Matter Environment Development is mandatory before proceeding further.

To develop a new application from the door lock application:

1. Import the reference Door Lock application template into **e²studio** and modify it to create the required Matter application.
2. Update the `.zap` file to reflect the new device configuration. Detailed instructions for updating the .zap file using a reference configuration are provided in .Section 4.4.4 Zap Code Generation.
3. After the Matter application project is created, double click the `configuration.xml` file.
4. Choose the clusters needed for your application as described in Section 4.4.3 Set Clusters.
5. In `matter_app/src` and `matter_app/include`, update, add, or remove application files needed to support your specific application requirements.

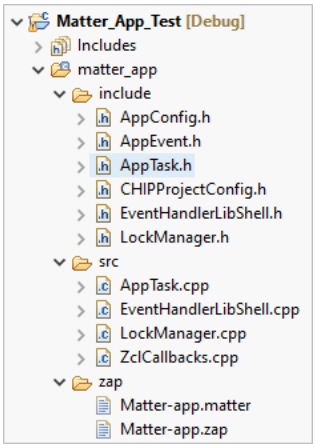

Figure 8. Application files to update

See Section 4.4.5 Light Application Source Files.

6. In **Project Explorer**, right-click the project name and open **C/C++ Project Settings** as shown in Figure 9. A property window appears.

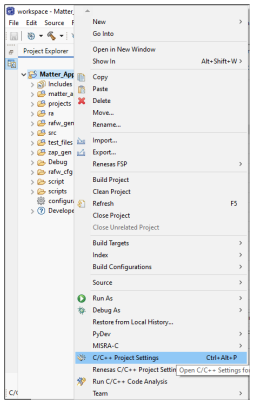

Figure 9. C/C++ Project Settings

7. Go to **C/C++ Build** > **Environment**. Update the ZAP_INSTALL_PATH to point to the zap CLI installation directory as shown in Figure 10.

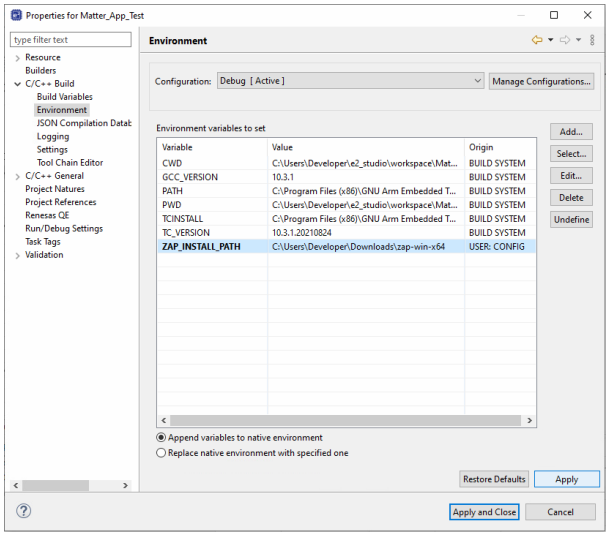

Figure 10. Zap CLI installation path

8. Right-click the project name and select **Build Project** as shown in Figure 11. This regenerates `ra/matter_app/zap/Matter-app.matter` along with the ZAP-generated code in `zap-gen/app/` by running a pre-build script. See Figure 11 and Figure 12. The main build then proceeds and generates the `.img` file in the **Debug** folder — this is the normal flow.

NOTE
In e² studio, the refreshed files are applied only after the first build completes. Therefore, a **second** build is required for the newly generated files to be used in compilation.

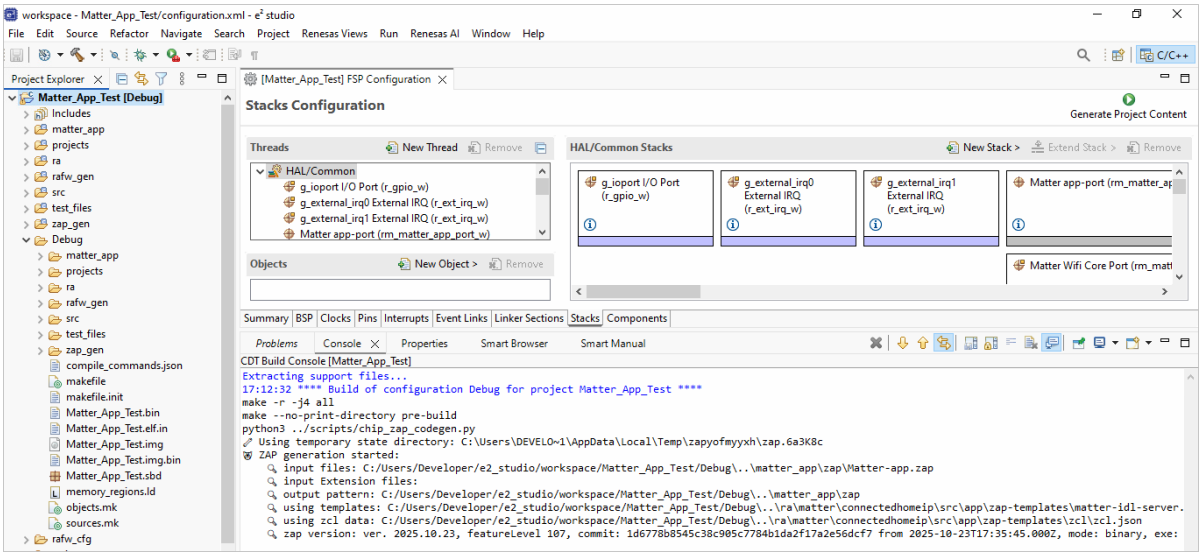

Figure 11. Pre-build script

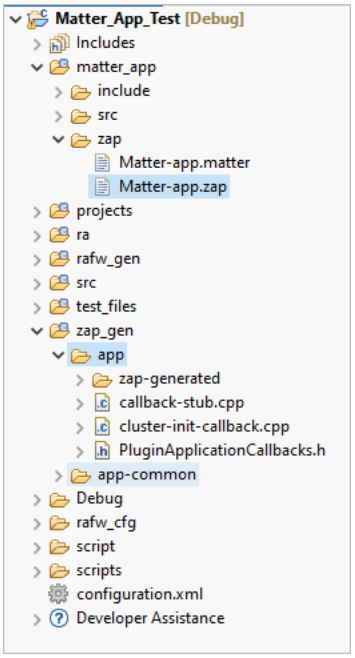

Figure 12. Generated files

4.4.1 Matter Environment Development

4.4.1.1 Repository Setup and Dependency Initialization

1. Clone the official Project CHIP repository using the command:
   `git clone https://github.com/project-chip/connectedhomeip.git`
2. Check out the required Matter release (for example, Matter 1.4):
   `git checkout tags/v1.4.0.0`
3. Initialize and update all submodules:
4. Install required build prerequisites:
   https://project-chip.github.io/connectedhomeip-doc/guides/BUILDING.html#prerequisites

4.4.1.2 Development Environment Bootstrap

1. Run the bootstrap script (one-time setup) using the command:
   `source ./scripts/bootstrap.sh`
   After running the `bootstrap.sh` script, you should see the window in Figure 13.

Figure 13. Bootstrap

2. Activate the environment for each new terminal: `source ./scripts/activate.sh`.
   After running the `activate.sh` script, you should see the window in Figure 14.

Figure 14. Matter environment setup

4.4.2 Application Development Flow

Figure 15 shows the development flow of the application.

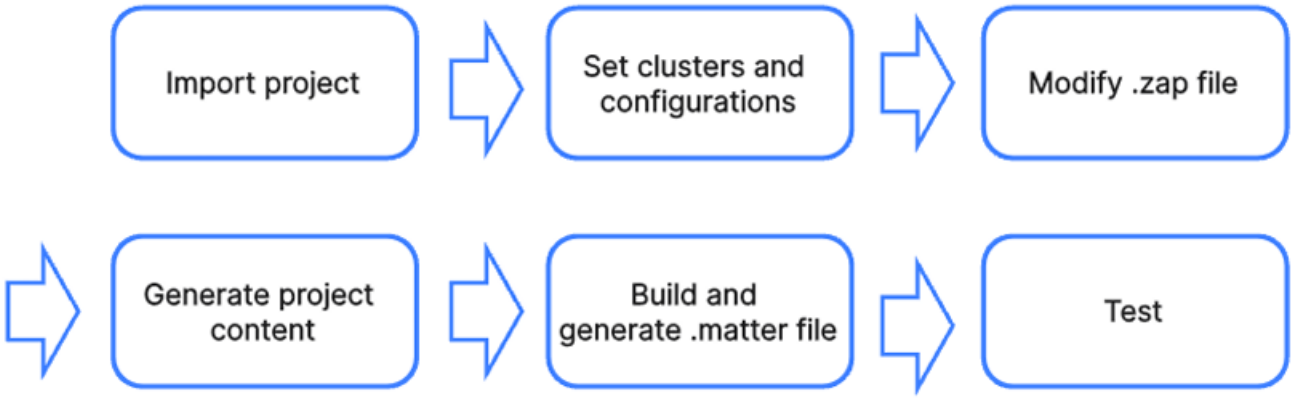

Figure 15. App development flow

4.4.3 Set Clusters

The current application already includes clusters related to the door lock functionality.
For more details, see the Select Endpoints and Clusters section in the *RA6W2 Matter Certification Manual*.

1. In e^2 studio, on the **Components Configuration** tab, select the required clusters components.

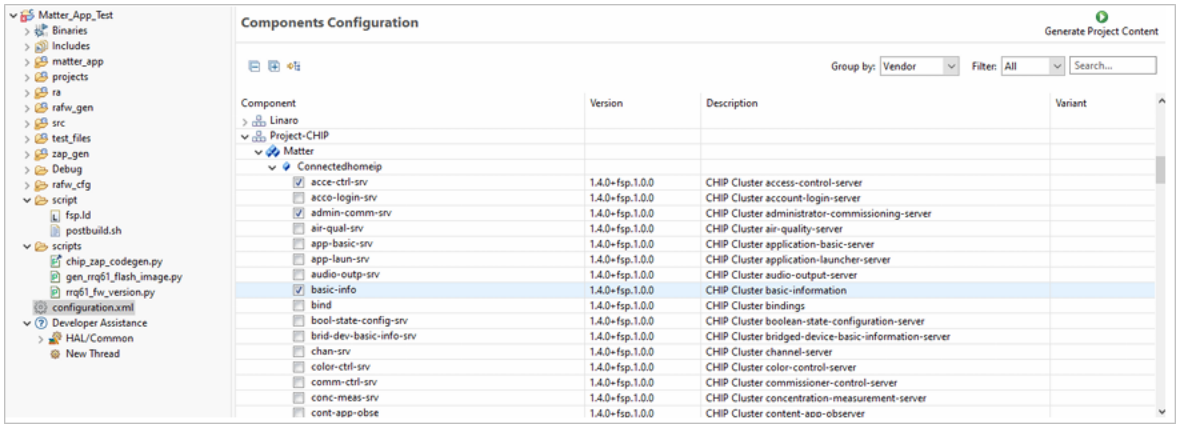

Figure 16. Cluster selection

2. For the **light application**, select the required clusters based on your application requirements. The clusters chosen in **e² studio** must match those selected in the **ZAP CLI application**. Remove any door lock–specific clusters from the template. The selected clusters for the lighting application are shown in Figure 17.

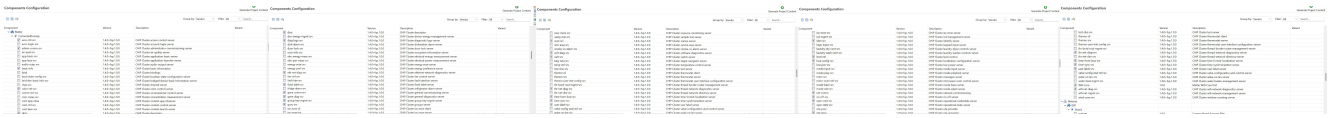

Figure 17. Cluster selection for Light application

3. Click **Generate Project Content**.
   Corresponding cluster code is retrieved from Matter FSP pack and made available under `ra/matter/connectedhomeip/src/app/clusters`.

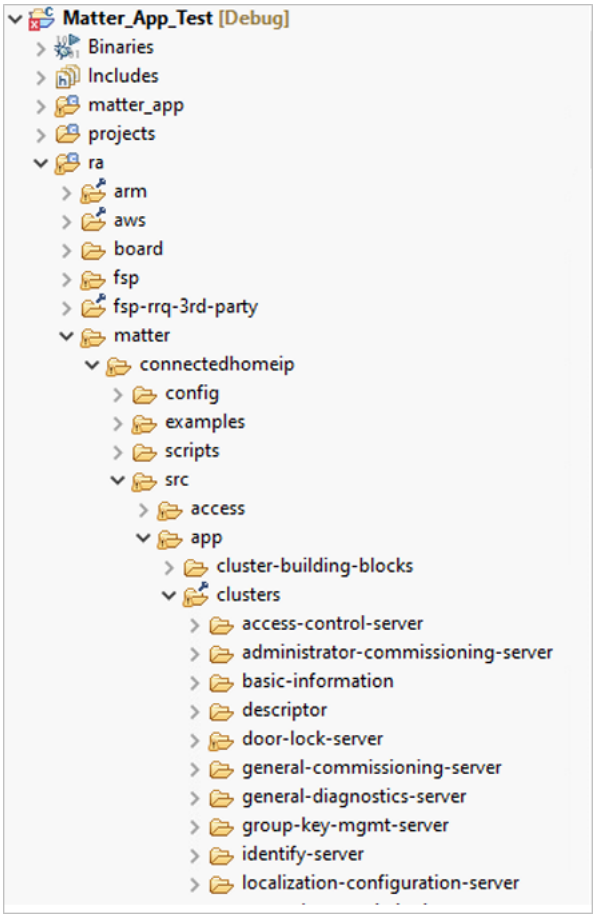

Figure 18. Generated Clusters

Figure 19 shows Generated clusters for Light application from FSP.

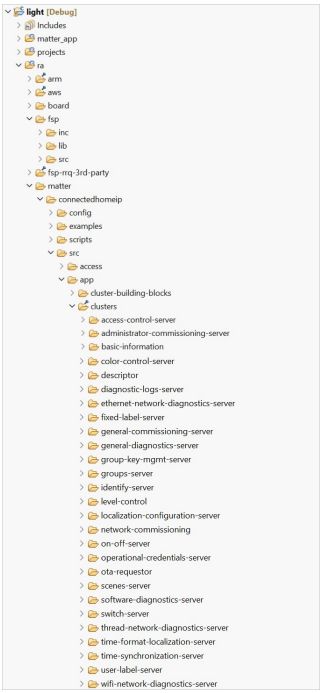

Figure 19. Generated Clusters for Light application

4.4.4 Zap Code Generation

1. Download the zap tool from https://github.com/project-chip/zap/releases/latest.
   - Download zap tool for Linux from https://github.com/project-chip/zap/releases/download/v2025.12.02/zap-linux-x6.zip.
   - Download zap tool for Windows from https://github.com/project-chip/zap/releases/download/v2025.12.02/zap-win-x64.zip.
2. In the ZAP tool, in the `connectedhomeip` repository, modify the `.zap` file based on your application requirements.
   See ZAP Tool section in in the *RA6W2 Matter Certification Manual*
   Various example applications are available in `connectedhomeip` repository.

Figure 20. Connectedhomeip repository

For the lighting application, the `.zap` file is located at: `connectedhomeip/examples/lighting-app/lighting-common/lighting-app.zap`.

a. To open the `.zap` file run the following command: `./scripts/tools/zap/run_zaptool.sh <location of the .zap file>`.
   For Light Application, use: `./scripts/tools/zap/run_zaptool.sh <path-to-connectedhomeip>/examples/lighting-app/lighting-common/lighting-app.zap`.
b. Replace `<path-to-connectedhomeip>` with the actual directory where the `connectedhomeip` repository is downloaded on your system.

Zap Application is starting and now you can see the 2 endpoints:
Endpoint 0 is reserved for the general commissioning process and should not be modified.
Endpoint 1 is used for the lighting application.

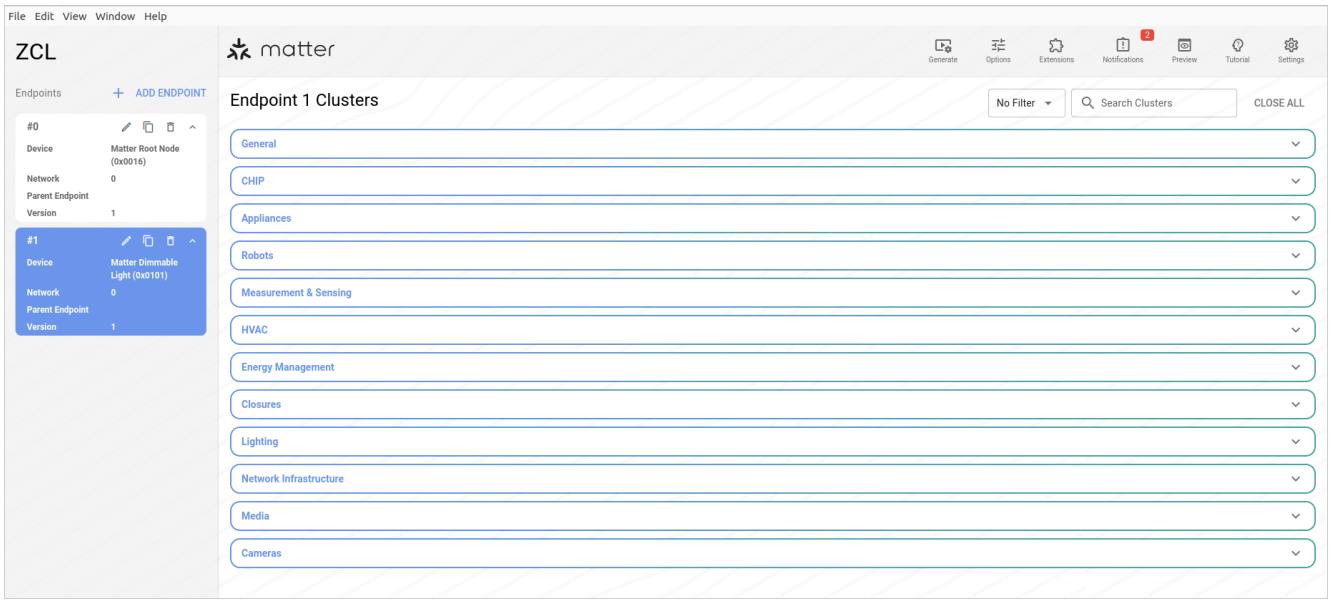

Figure 21. Zap application

c. Save the `.zap` file (CTRL+S). The changes saved to `onnectedhomeip/examples/lighting-app/lighting-common/lighting-app.zap`.

3. After doing required changes in Cluster and attribute of lighting app as your requirement like shown in Figure 23, Figure 24 and Figure 25. Save the changes and copy the updated `.zap` file into `ra/matter_app/zap/Matter-app.zap`.
4. Right-click the project and select **Properties**.
5. On the **Build Steps** tab, in **Pre-build steps**, check that the **Command** field has the needed command.
   The Python script scripts/chip_zap_codegen.py runs automatically as a pre-build step in e2 studio to generate the zap_generated files using zap.

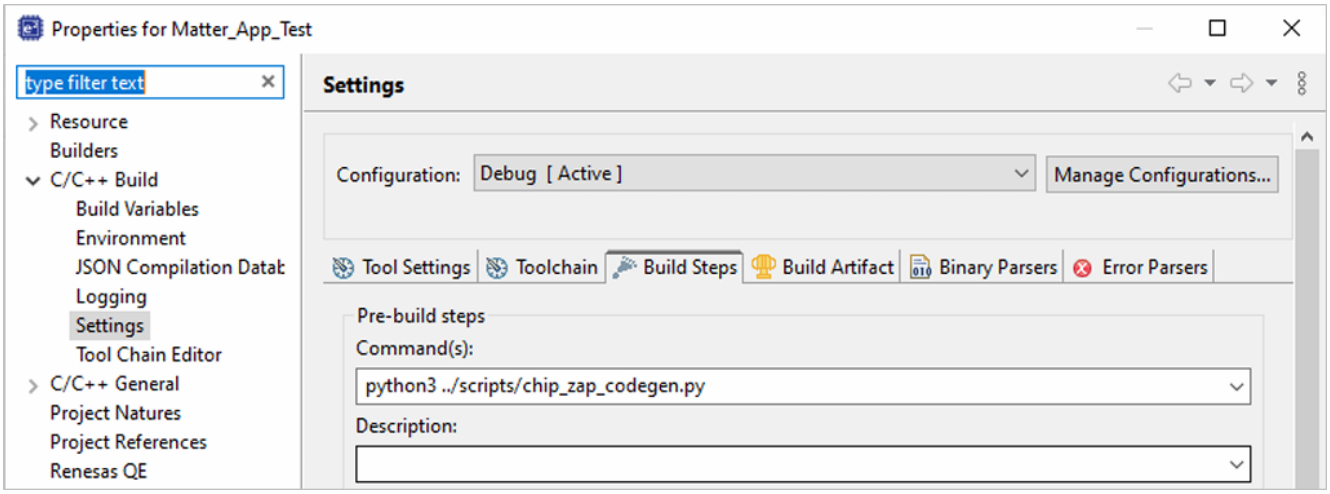

Figure 22. Pre-build step

6. Continue with step 3 of Section 4.4 Develop New Application from Door Lock Application. .
Figure 23 shows the clusters relevant to the lighting application development.

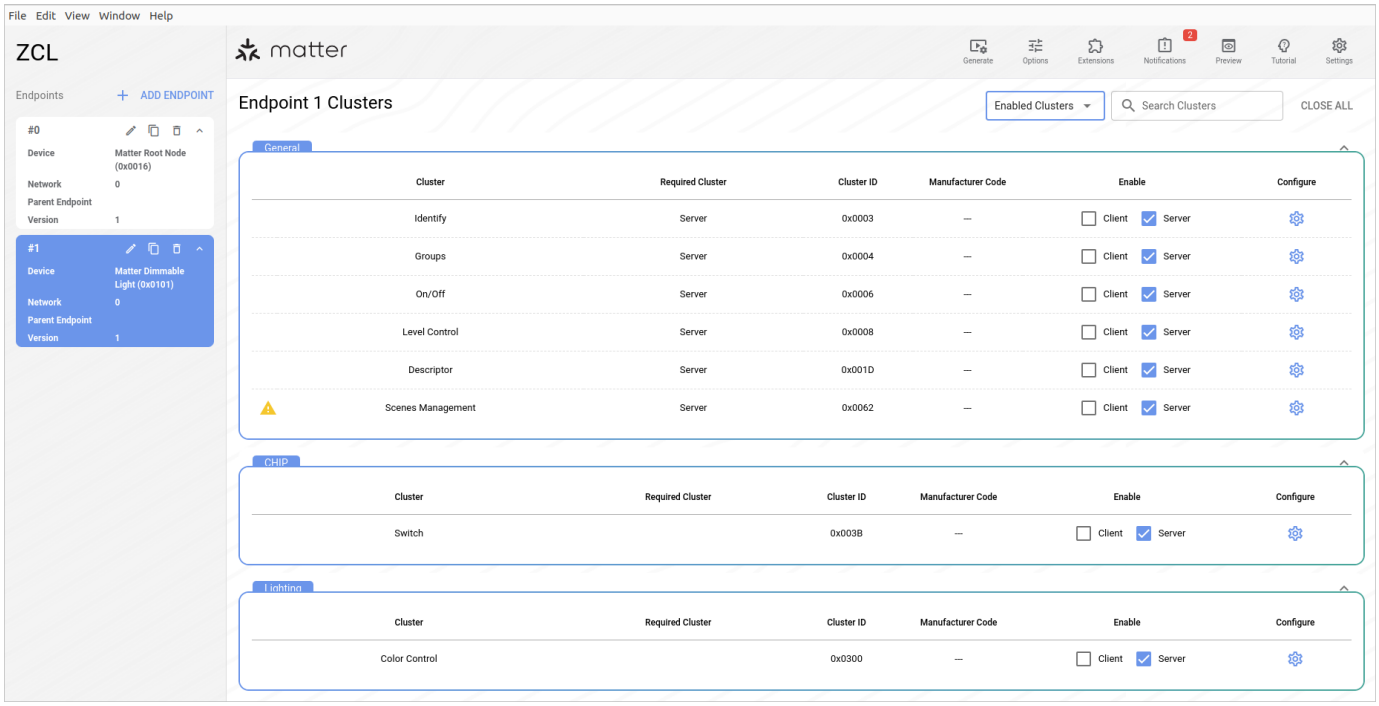

Figure 23. Clusters of Light application

You can edit the attributes of each cluster by selecting the required cluster. Figure 24 shows the corresponding attributes of Level Control.

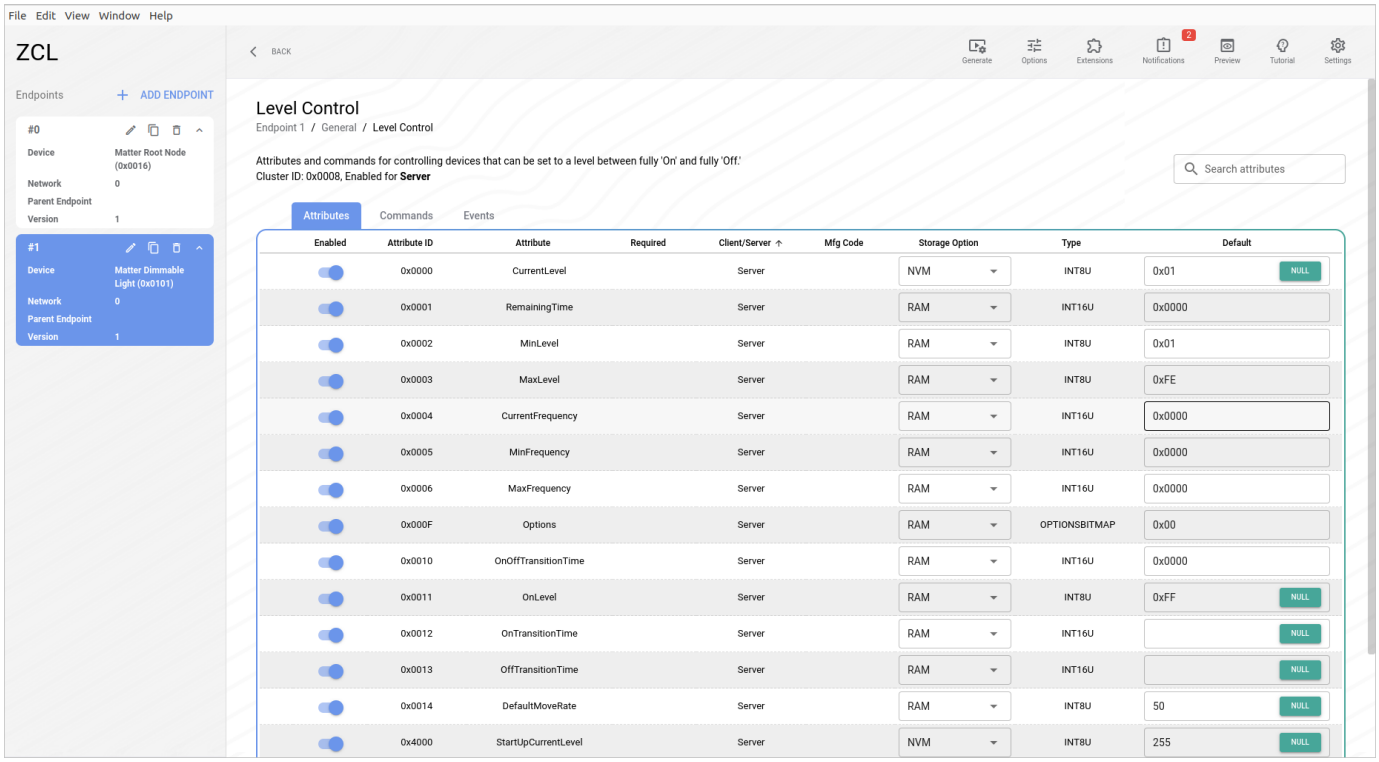

Figure 24. Attribures of Level Control Cluster

You can edit the commands of your application in window in Figure 25.

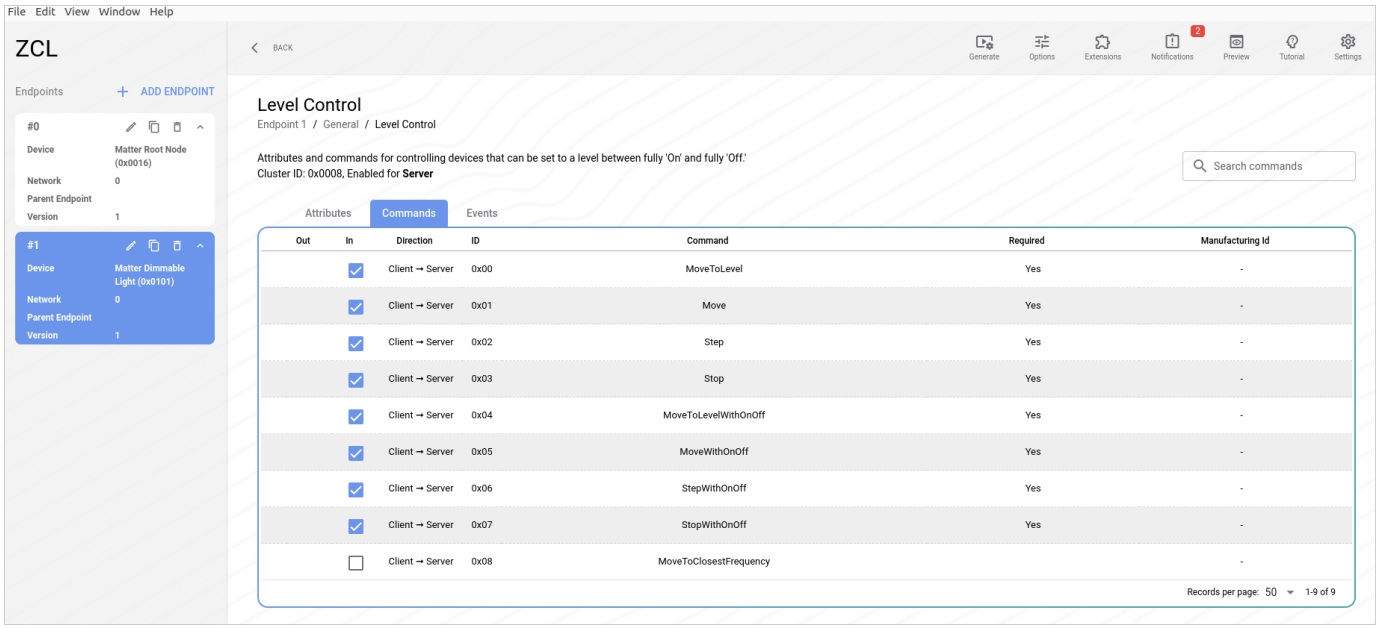

Figure 25. Commands of Level Control

4.4.5 Light Application Source Files

The Light application follows the same application framework and execution model as the existing Door Lock application. Only the **application-specific logic** and **Matter data model** are replaced.

Reused from Door Lock Application:

- Application task framework
- Platform initialization
- Build and runtime structure

Reused from `lighting-common`
Lighting behavior logic
Matter cluster handling patterns
File structure include/
```
├── AppConfig.h
├── AppEvent.h
├── AppTask.h
├── CHIPProjectConfig.h
└── LightingManager.h
src/
├── AppTask.cpp
├── LightingManager.cpp
└── ZclCallbacks.cpp
```
Header Files (`include/`):
- **AppConfig.h** – defines application configuration parameters. Updated for the Light application to map LED GPIOs instead of lock actuators.
- **AppEvent.h** – defines the application event structure, reused from the Door Lock application for uniform event handling.
- **AppTask.h** – declares the main application task interface, enabling lighting-specific event dispatch.
- **CHIPProjectConfig.h** – defines Matter compile-time configuration, updating device identity from Door Lock to Light.
- **LightingManager.h** – declares the core lighting control interface, reused from `lighting-common`.

Source Files (`src/`):
- **AppTask.cpp** – implements the application execution flow. Reused from the Door Lock application with event routing updated to `LightingManager`.
- **LightingManager.cpp** – implements lighting behavior such as ON, OFF, and TOGGLE, reused or adapted from `lighting-common`.
- **ZclCallbacks.cpp** – implements Matter cluster callbacks for the Light application, forwarding On/Off commands to `LightingManager`.

4.5 Customize Application Development

A Matter device-type application can be developed by leveraging the Door Lock reference application as a baseline and following the application flow outlined in Figure 26.

4.5.1 Describe Reference Lock Application

The Lock application consist of Application Start, Application Main Task, and Attribute Change Callback parts as shown Figure 26.

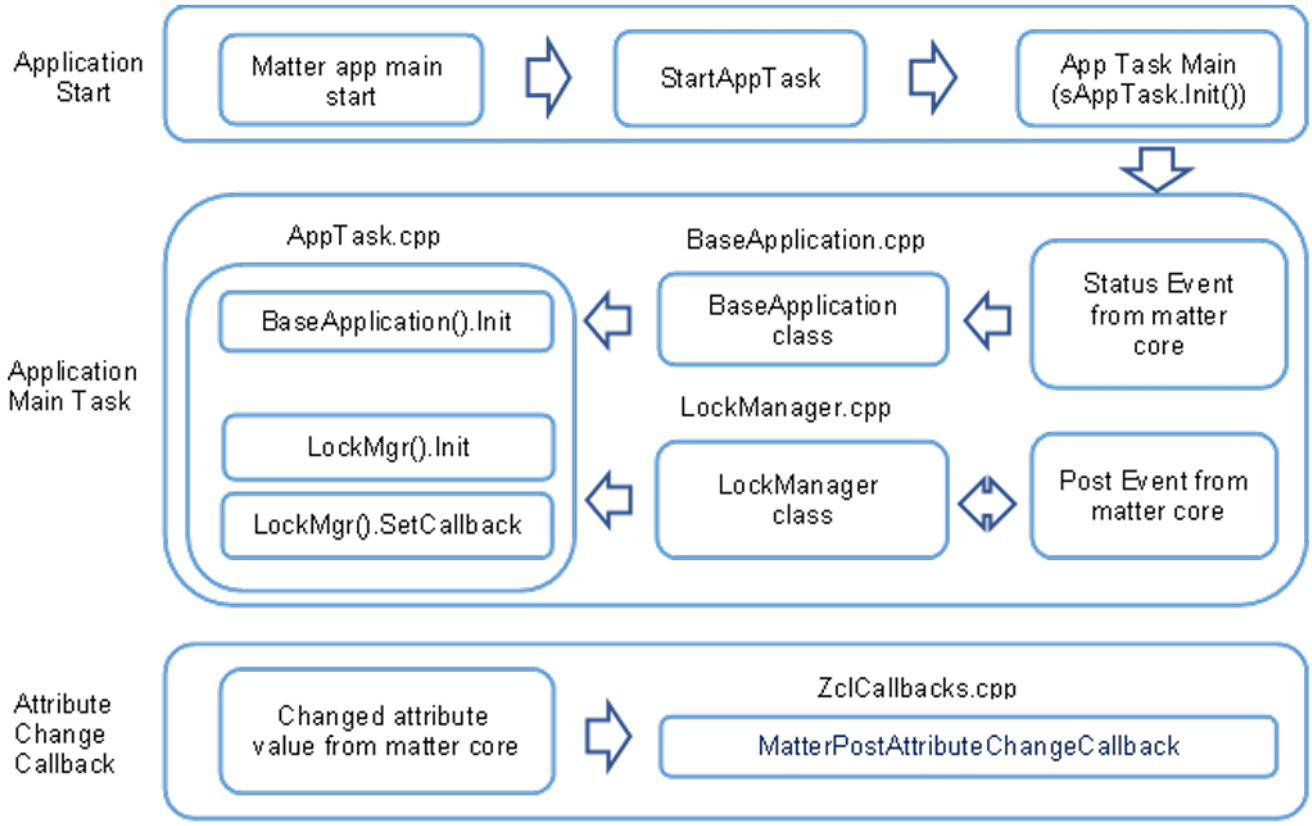

Figure 26. Lock application structure

4.5.1.1 Wi-Fi Configuration

In the NVRAM, there is a section called **WIFI PROFILE** where Wi-Fi configuration parameters can be stored. Upon rebooting, the application checks to see if a saved profile exists. If a saved profile exists, it loads the configuration from the WIFI PROFILE; otherwise, it uses hardcoded default values. These values are then saved to the WIFI PROFILE for future use. Currently, the application supports only WPA2 Security mode.

4.5.1.2 Bluetooth® LE

During Bluetooth LE commissioning, the device advertises itself over Bluetooth LE and can be discovered by a Matter controller (for example, chip-tool or mobile app) using the Bluetooth LE advertisement. The controller connects to the device over Bluetooth LE and exchanges information, such as setup codes and network credentials. After pro visioning is complete, the device uses these credentials to join the network.
The Bluetooth LE transport is only used for the commissioning process. After successful provisioning, communication with the device continues over Wi-Fi.
To enable Bluetooth LE Commissioning:

1. Use the QE for Bluetooth LE tool to create the required Bluetooth LE profile.
2. Import the generated profile into the project.

3. In the Matter module (`rm_matter_app_port_w`) configuration, under **Properties** tab, enable the **CHIPOBLE** option to allow the Matter stack to use Bluetooth LE for commissioning and device communication.

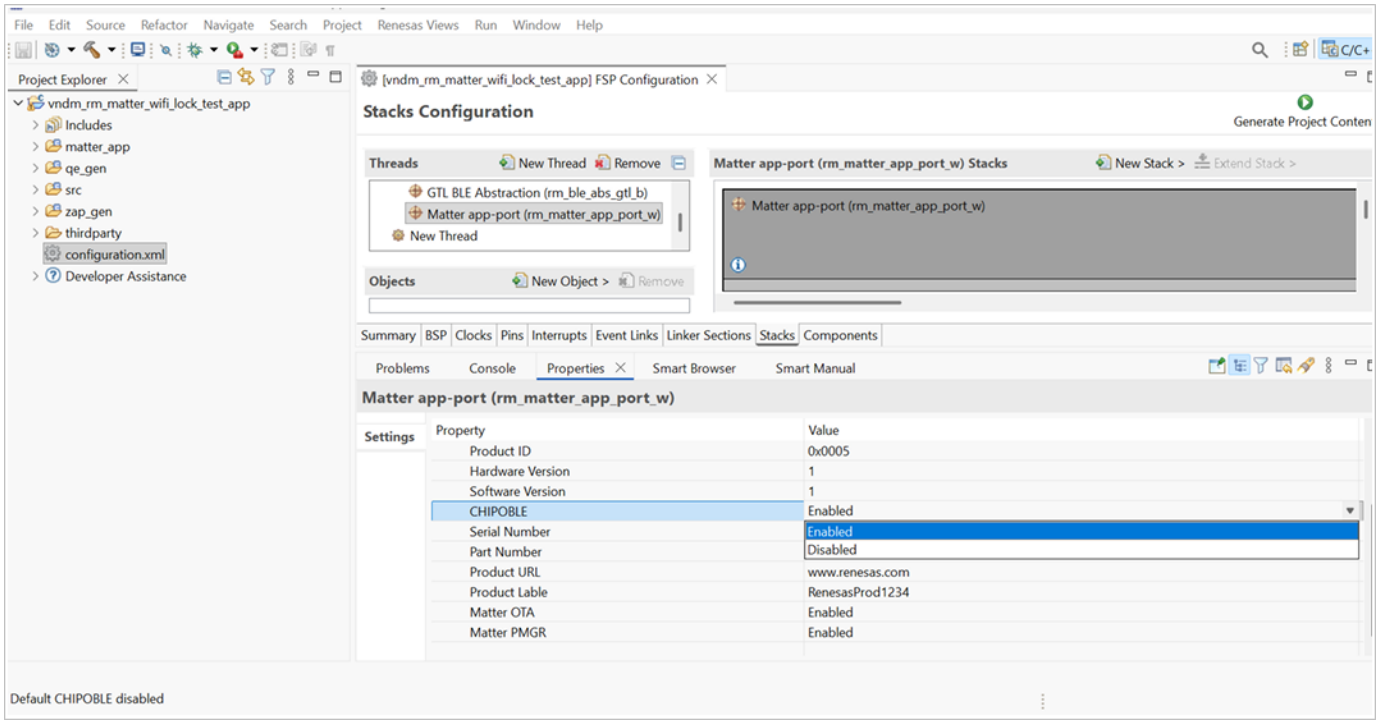

Figure 27. Bluetooth LE configuration property in e^2 studio

4.5.1.3 Application Main Task

This is the main task where the lock device type is initialized.

- **Base Application**: the Base Application class provides functions to set and get basic information such as serial number, vendor ID, product ID, product name, hardware version, pin code, discriminator manufacturing date, device type, QR code, manual pairing code, and information for commissioning.
- **Lock Manager**: The Lock Manager class provides functions for managing the lock device. This includes con figuring, initializing, registering callback functions for receiving messages from the Matter core, and registering timers.
- **Attribute Change Callback**: The Attribute Change Callback is called by the Matter application framework when one of the lock device attribute values changes. This is where the application performs an action based on the value of the attribute

4.5.2 Matter DPM

Power Manager (PMGR) is the power management module of RA6W1. It is used to configure Dynamic Power Man agement (DPM) mode for optimizing energy consumption for devices. Matter uses the configuration parameter Matter PMGR to enable or disable DPM for Matter. By default, Matter PMGR is enabled in the Matter Application template. To disable Matter PMGR:

1. Double-click `configuration.xml`.
2. Click **Matter app-port**(**rm_matter_app_port_w**).

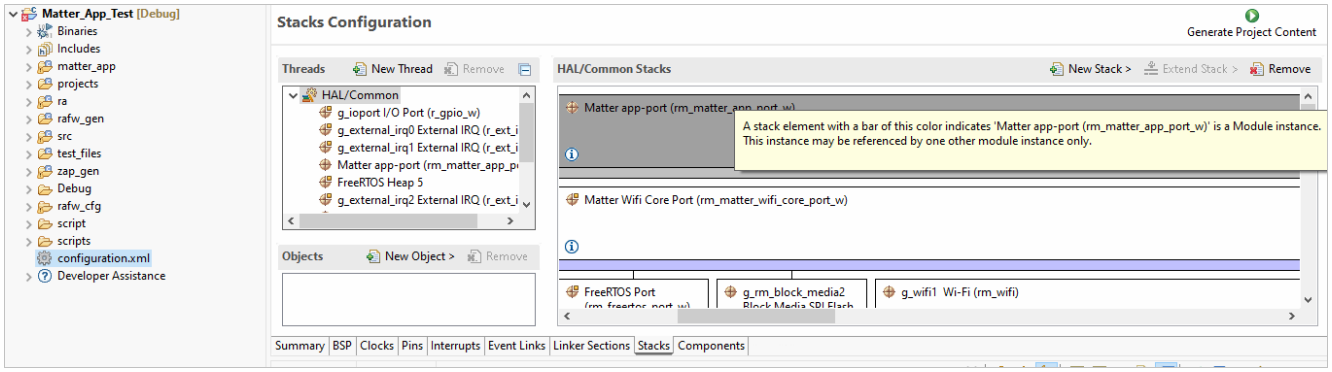

Figure 28. rm_matter_app_port_w module

3. In **Stacks** > **Properties window**, for Matter PMGR select **Disabled**.

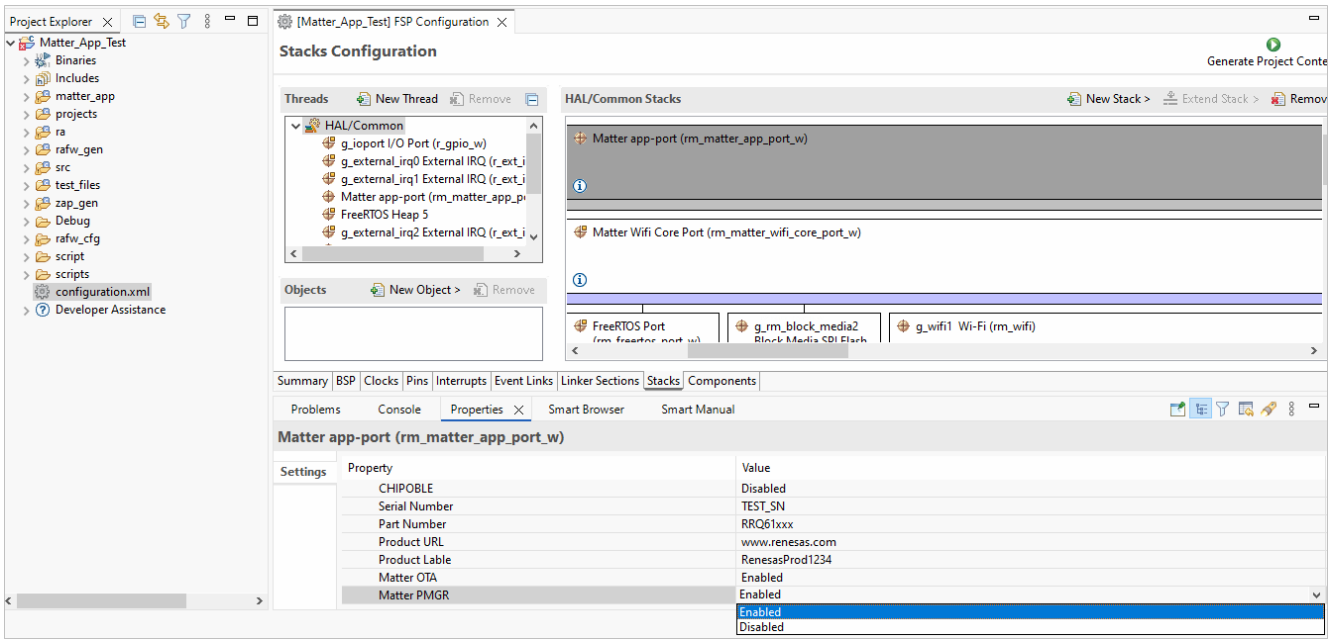

Figure 29. Property: Matter PMGR

4. Click **Generate Project content**.
5. Right-click the project name and select **Build Project**.
   With Matter PMGR enabled, the RA6W1 Lock enters DPM mode after commissioning, wakes up upon receiving a uni cast packet, and performs the lock or unlock operation.
   From the application's perspective, the only difference when DPM is enabled is a change in the sequence flow. Figure 30 shows that DPM is enabled so that the Server Ready Event is sent only after the application started. When DPM is disabled, Server Ready Event is sent before the application starts, during server initialization as shown in Figure 31.

Figure 30. With DPM

Figure 31. Without DPM

4.6 Program Firmware Image

The Matter lock application image is created in the e^2 studio project folder (`e2_studio/Matter_App_Test /Debug/Matter_App_Test.img.bin`).
For more information about firmware downloading, see the Programming Firmware Images Using cli_programmer.exe section of *RA6W1 Getting Started Guide*.

4.7 Debug with J-Link Debug Probe

For more information about firmware downloading, see the e^2 studio Debug Mode with GDB of *RA6W1 Getting Started Guide*.
To debug Matter application, complete the following steps after setting up the debugger:

1. Reset the target system by entering command: `monitor reset`.
2. Set a breakpoint at the Matter entry function MATTER_On by entering command: `b MATTER_On`.
3. Start or continue execution by entering command: `c`.

5. Matter Application Setup

Figure 32 shows how to start testing each application in the Matter environment.

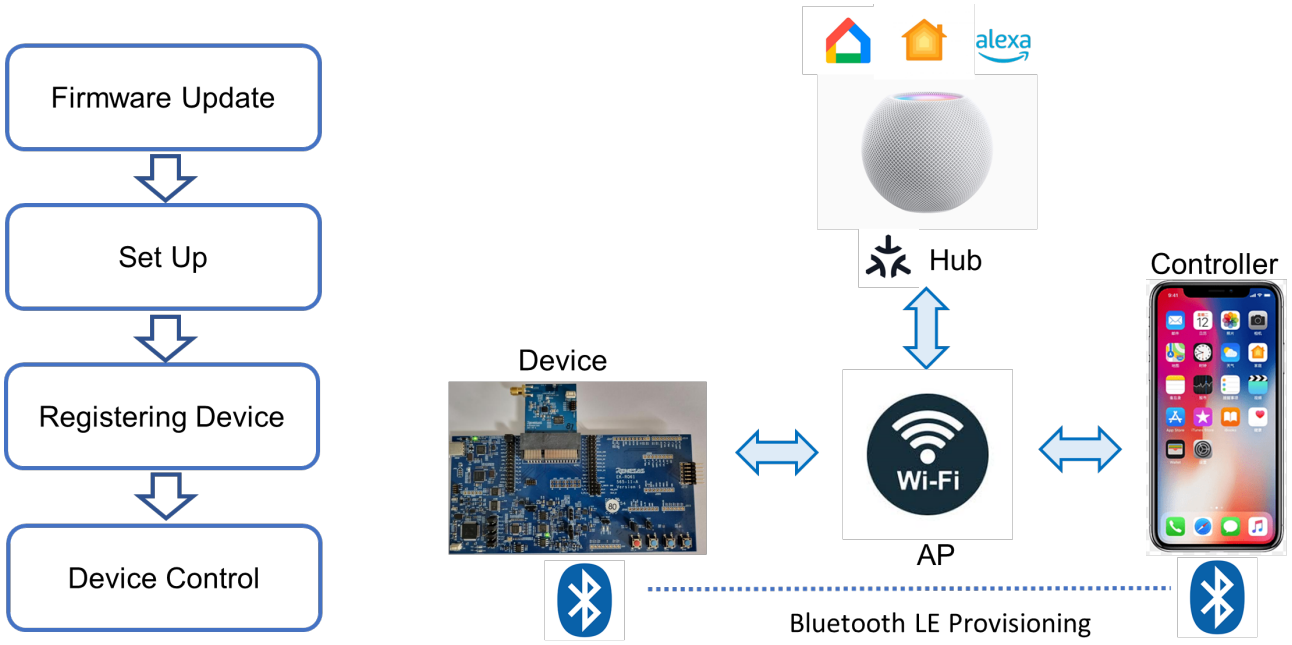

Figure 32. Quick start

- **Firmware update**: Matter application firmware image update for RA6W1.
- **Setup**: Smart Hub and mobile controller application configuration and connecting home AP.
- **Registering device**: RA6W1 pairing (commissioning) as Matter accessory.
- **Device control**: RA6W1 is controlled through mobile controller application or Hub.

5.1 Apple Home App

To set up HomePod mini to Apple Home app:

1. Get your iPhone or iPad ready to set up HomePod.
2. Plug HomePod into power and wait for a chime.
3. Hold your iPhone or iPad close to HomePod.
4. Customize your HomePod settings.
5. Wait for setup and configuration to complete.

For more information on HomePod mini setup, go to the official website (https://support.apple.com/en-us/111110).

To pair and manage your Matter accessories:

1. Pair a Matter accessory.
2. Add a Matter accessory to the Home app.
3. View your paired Matter accessories.
4. Remove Matter accessories.

For more information on pairing Matter accessories, go to the official website (https://support.apple.com/en-us/102135).

5.1.1 Wi-Fi Provisioning and Commissioning for Registering Device to Apple Home

For using Wi-Fi provisioning and commissioning with RA6W1, you need to flash RA6W1 with the image of **matter_wifi_lock_example_ek_ra6w1**. To register an accessary device to Home App, connect the RA6W1 using manual code/scan the QR code which is shared with product. During this process, the RA6W1 is connected to AP. Then, the device is registered to a Home application.

5.1.2 Bluetooth LE Provisioning and Commissioning for Registering Device to Apple Home

For using Bluetooth LE provisioning and commissioning with RA6W2, you need to flash RA6W2 with the image of **matter_wifi_ble_lock_ek_ra6w2**.
The Bluetooth LE provisioning and commissioning process starts automatically when you try to add matter device to the controller. During this process, AP information is transferred to the RA6W2 over Bluetooth LE network, and the RA6W2 can be connected to AP. Then, the device is registered to the Home application.

5.2 Google Home Application

Google Nest needs to create a developer project and Matter integration in advance:
- Create a developer project (https://developers.home.google.com/matter/project/create)
- Create a Matter integration (https://developers.home.google.com/matter/integration/create)

To register Google Nest to Google Home app:
- Set up your Google Nest or Home speaker or display (https://support.google.com/googlenest/answer/7029485?hl=en&sjid=5926392987998707780-AP)
- Pair a Matter Device (https://developers.home.google.com/matter/integration/pair)

5.2.1 Wi-Fi Provisioning and Commissioning for Registering Device to Google Home

Procedure is same as in Section 5.1.1 Wi-Fi Provisioning and Commissioning for Registering Device to Apple Home.

5.2.2 Bluetooth Provisioning & Commissioning for Registering Device to Google Home

Procedure is same as in Section 5.1.2 Bluetooth LE Provisioning and Commissioning for Registering Device to Apple Home.

5.2.3 Google Home Control

After setting up the device in the Google Home app, the Controller can control the accessory.
1. Open the Google Home app.
2. Tap Favourites or Devices and find the tile for your device.
3. Tap or drag the tile to take actions such as turning lights on and off or locking and unlocking a door.

5.3 Amazon Alexa App

In Amazon Alexa App, before testing, you need to change the Vendor ID (VID) and the Product ID(PID) and to flash test certificates.
For more details, see the e^2 studio Test VID/PID Modification and Certification Regeneration section in *RA6W2 Matter Certification Manual*.
To change VID and PID for Amazon Alexa App:
1. Load the template into workspace.
2. Click `Configuration.xml`.
3. In stack, select **Matter app-port**, and then click **Show properties**.
   a. For Vendor ID, set 0xFFF1
   b. For Product ID, set 0x8005.
4. Click the **Generate Project Content** button.
5. Right-click the project, and then select **Build**.
6. For AT application use the below commands:

```
AT+MCONFIG=VID,65521
AT+MCONFIG=PID,32773
```

To flash the test certificate, follows the steps of the Flash Matter Certificates on to S-Flash of RA6W1 section in *RA6W2 Matter Certification*.

To register Amazon smart speaker (Echo Dot 5th) to Amazon Alexa:
1. Search for Amazon Alexa in the Google Play Store and install the Alexa app.
2. Sign in with your Amazon account or create a new Amazon account if you do not have one.
3. To connect Matter device with Alexa:
   a. Open the Alexa app.
   b. Open **Devices**.
   c. Select the plus icon.
   d. Select **Add Device**.
   e. Select the type of smart home device that you want to connect.
   f. Select the brand and follow the on-screen instructions.

For more information, see the official documents about Amazon Alexa (https://www.amazon.com/gp/help/customer/display.html?nodeId=G3RKPNRKF33ECTW7).

NOTE
For Wi-Fi/Bluetooth LE provisioning and scanning QR code of the RA6W1, see Section 5.1.1 Wi-Fi Provisioning and Commissioning for Registering Device to Apple Home and Section 5.1.2 Bluetooth LE Provisioning and Commissioning for Registering Device to Apple Home.

5.3.1 Amazon Alexa Control

After setting up the device in the Amazon Alexa application2, the Controller can control the accessory.
1. Open the Alexa application.
2. Tap the icon of the device.
3. Tap or drag the device to take actions such as turning the lights on and off or lock and unlock a door.

5.4 Security and Certificate

Certificates and keys are important for secured communication in Matter. For details, see the Flash Matter Certificates on to S-Flash of RA6W1 in *RA6W2Matter Certification Manual*.

5.5 AT Commands for Matter from External MCU

5.5.1 Overview

External MCU can control RA6W1 using AT commands. Basically, AT commands are available for all Device type of Matter using Device Type and Cluster ID and Attribute ID. For AT UART pin configuration, see Introduction of *AT Commands Manual*.
Figure 33 shows the AT command flow for lock and unlock control.

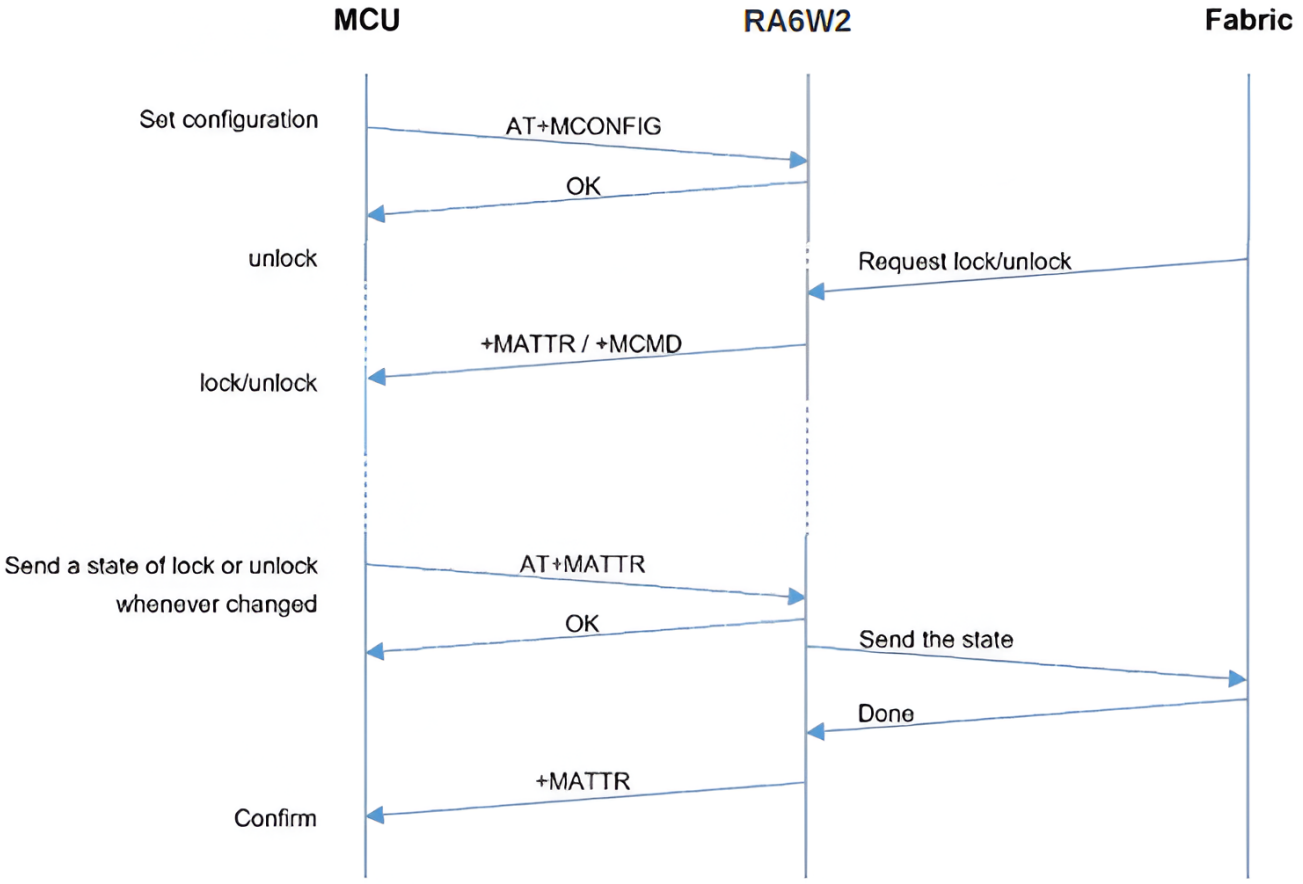

Figure 33. AT command flow for lock control

5.5.2 AT Command List

Table 1. AT commands from MCU to RA6W2

Command	Parameters	Description
AT+MCONFIG	<parameter>,<value>	Sets configurations for commissioning. <parameter>: configurable parameter ■ DISC: discriminator ■ VID: vendor ID ■ PID: product ID ■ HWVER: Hardware version ■ SPKPCNT: spake2p iteration count ■ SPKPSALT: spake2p salt ■ SPKPVF: spake2p verifier ■ PINCODE: pin code ■ DEVTYPE: matter device type <value>: data of string type ■ '?' for <value>: is responded current value of the parameter with "+MSTATUS=CFG" Response: OK or ERROR

	Example
	`AT+MCONFIG=DISC,3840`
	`OK`
	`AT+MCONFIG=VID,FFF1`
	`OK`
	`AT+MCONFIG=DISC,?`	

Command	Parameters	Description
	+MSTATUS=CFG,DISC,3840 OK	
AT+MATTR	<endpoint-id>, <cluster-id>, <attribute-id>, <attribute-data>, <data-type(w),data-length (r)>, <write/read>	<endpoint-id>: device endpoint id <cluster-id>: cluster id <attribute-id>: attribute id <attribute-data>: attribute data <data-type(w), data-length(r)>: data type for write, data length for read <write/read>: write or read to attribute

	Example
	//write lock to lockstate_attribute for lock cluster
	
	AT+MATTR=1,256,0,1,36,1
	
	//read 1 byte from lockstate_attribute for lock cluster
	
	AT+MATTR=1,256,0,0,1,0	

Table 2. AT commands from RA6W2 to MCU

Command	Parameters	Description
AT+MCONFIG	<parameter>,<value>	Sets configurations for commissioning. <parameter>: configurable parameter ■ DISC: discriminator ■ VID: vendor ID ■ PID: product ID ■ HWVER: Hardware version ■ SPKPCNT: spake2p iteration count ■ SPKPSALT: spake2p salt ■ SPKPVF: spake2p verifier ■ PINCODE: pin code ■ DEVTYPE: matter device type <value>: data of string type ■ '?' for <value>: is responded current value of the parameter with "+MSTATUS=CFG" Response: OK or ERROR
	Example AT+MCONFIG=DISC,3840 OK AT+MCONFIG=VID,FFF1 OK AT+MCONFIG=DISC,? +MSTATUS=CFG,DISC,3840 OK	
AT+MATTR	<endpoint-id>, <cluster-id>, <attribute-id>, <attribute-data>, <data-type(w),data-length (r)>, <write/read>	<endpoint-id>: device endpoint id <cluster-id>: cluster id <attribute-id>: attribute id <attribute-data>: attribute data <data-type(w), data-length(r)>: data type for write, data length for read <write/read>: write or read to attribute

	Example
	//write lock to lockstate_attribute for lock cluster
	
	AT+MATTR=1,256,0,1,36,1
	
	//read 1 byte from lockstate_attribute for lock cluster	

Command	Parameters	Description
	AT+MATTR=1,256,0,0,1,0	

5.5.3 AT Command Examples for Lock Application

Figure 34 shows the actual flow detailed for lock application.

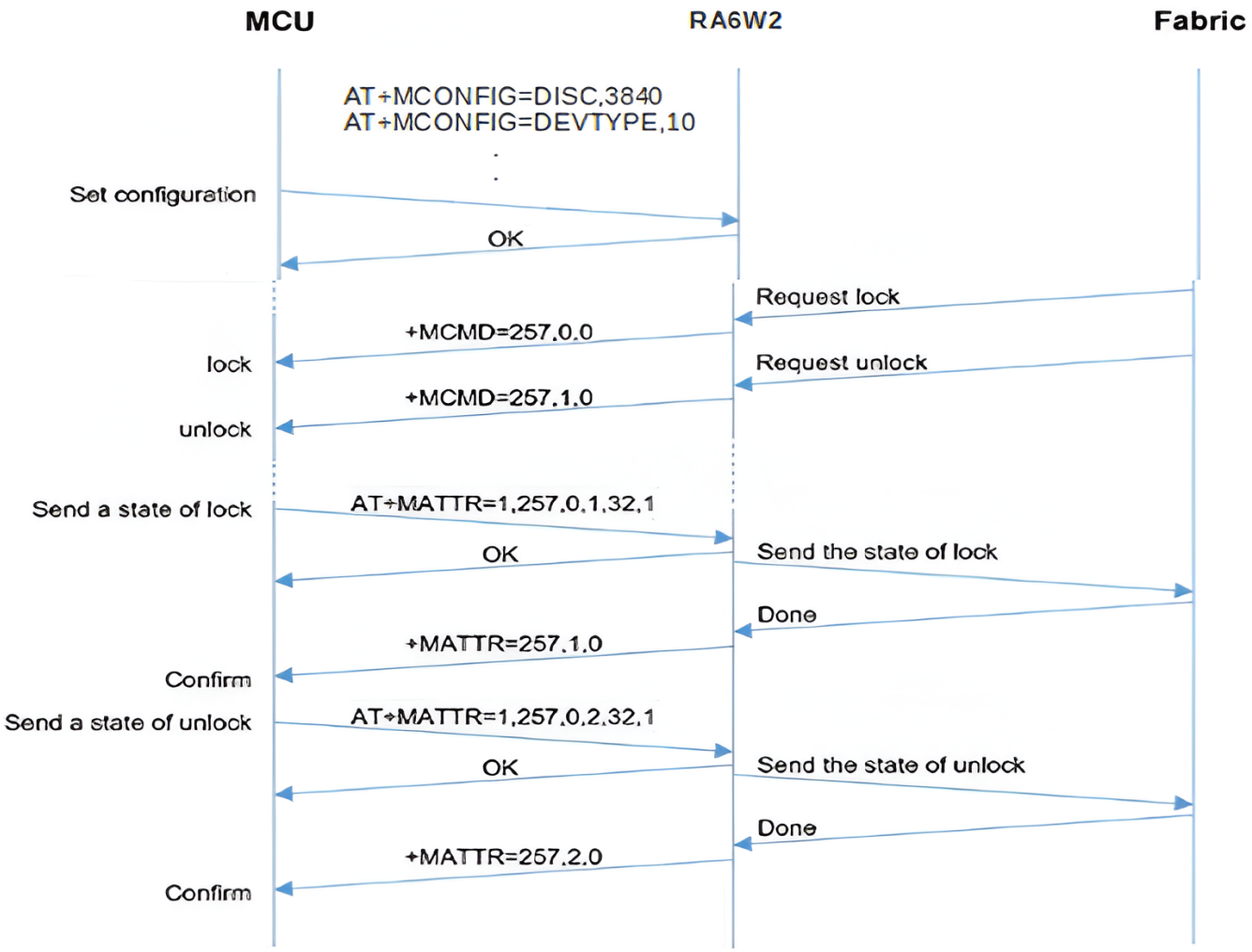

Figure 34. AT command flow for lock application

5.5.4 AT Commands for DPM Setup

DPM can be enabled using AT commands within the Matter AT application. For more information, see *AT Commands Manual*.

Connecting to an AP using the AT+WFJAP command enables DPM by default.

Further AT Commands to enable DPM is added in below table

Table 3. AT Commands for DPM Setup

Command	Parameters	Description

AT+PMGRCONSTRAINT	Example
	// Add RAM constraint
	AT+PMGRCONSTRAINT=1,1 OK
	// Query the status of constraint counter
	AT+PMGRCONSTRAINT=? +PMGRCONSTRAINT:1,1,0,0 OK
	// Add RAM constraint
		Add or Remove constraint.
		<action>
		1 (Add)
		2 (Remove)
		<constraint>
		1 (RAM)

Command	Parameters	Description

	AT+PMGRCONSTRAINT=1,1 OK
	// Query the status of constraint counter AT+PMGRCONSTRAINT=?
	+PMGRCONSTRAINT:2,1,0,0 OK
	
	// Remove RAM constraint AT+PMGRCONSTRAINT=2,1 OK
	
	// Query the status of constraint counter AT+PMGRCONSTRAINT=?
	+PMGRCONSTRAINT:1,1,0,0 OK
		2 (RETENTION)
		3 (MAC_HW)
		4 (PROHIBITED)
		Response: OK or ERROR

AT+PMGRFORCE	<sleep_mode>,<duration/twt action>,
	<neg_type/mantissa>,<min_wake_dur>,
	<flow_type>,<trig ger_en>,<neg_type>
	Example
	// Force to enter Sleep mode 2 for 10 seconds
	AT+PMGRFORCE=2,10
	OK
	// Force to enter Sleep mode 3 for 10 seconds
	AT+PMGRFORCE=3,10
	OK
	// Force to enter DPM Sleep for 10 seconds
	AT+PMGRFORCE=4,10
	OK
	// TWT setup with an 8000 TU interval, exponent 10, 3 TU minimum wake duration, flow type 0, trigger disabled, negotiation type 1
	AT+PMGRFORCE=5,1,8000,10,3,0,0,1
	OK
		Force to enter Sleep mode for the specified time.
		<sleep_mode>
		2 (Sleep mode 2)
		3 (Sleep mode 3)
		4 (DPM Sleep)
		5 (TWT)

6. Revision History

Revision	Date	Description
1.04	Apr 28, 2026	Updated the Zap tool installation details and a section for development of other matter application from a reference application.
1.03	Nov 30, 2025	Updated the new matter module structure in sections about software setup and customizing application development.
1.02	Aug 31, 2025	Updated Device name, M R19US0027EK0102 atter version and added Bluetooth LE commissioning option.
1.01	Mar 13, 2025	Added the section of Matter DPM.
1.00	Dec 30, 2024	First release.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Renesas Electronics' suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.