

Dear customers,

Thank you for your continued support of Renesas Electronics and for your use of our products, software, and solutions.

Notice of Discontinuation of Support for This Application Note and Sample Project

Support for this application note, the accompanying sample project, and the associated Renesas-hosted dashboard service will be discontinued on December 31, 2026.

As a result, the Renesas-hosted dashboard environment and related accounts listed below will no longer be available after this date:

- Complimentary USD 10 AWS trial account provided for the Renesas Dashboard
- Renesas Dashboard (<https://renesas.cloud-ra-rx.com/>)
- Renesas Dashboard account
- AWS Grafana dashboard
- Grafana accounts for the Renesas Dashboard

Please note that this discontinuation applies only to the Renesas-hosted dashboard service and related accounts.

Customers can continue to use their own AWS accounts and cloud environment independently.

We apologize for any inconvenience this may cause and appreciate your understanding.

July 2026
Renesas Electronics Corporation

Renesas RX Family

AWS Cloud Connectivity on CK-RX65N v2 with Wi-Fi DA16600 (GCC) – Getting Started Guide

Introduction

This document describes a system that uses the CK-RX65N v2 Cloud Kit plus US159-DA16600EVZ Pmod board (part number: RTK5CK65N0S08001BE) from Renesas. This system demonstrates AWS Cloud connectivity using the CK-RX65N v2 board running Amazon FreeRTOS via a Wi-Fi connection using DA16600 Pmod. It visualizes the HS3001, ZMOD4410, ZMOD4510, OB1203, ICP20100, and ICM42605 sensor information on the dashboard and controls LEDs on the board. In addition, this application note also describes several feature options for users when using CK-RX65N v2 Cloud Kit with AWS: OTA (Over-The-Air) feature (**section 6**) and Fleet Provisioning feature (**section 7**).

The document covers the following items:

- How to create the 10 USD credit free trial account for AWS
- How to operate and install the certification information certification for Cloud
- How to see and run the sensor data on the dashboard
- How to use the OTA feature to update firmware via Cloud
- How to use Fleet Provisioning via Cloud

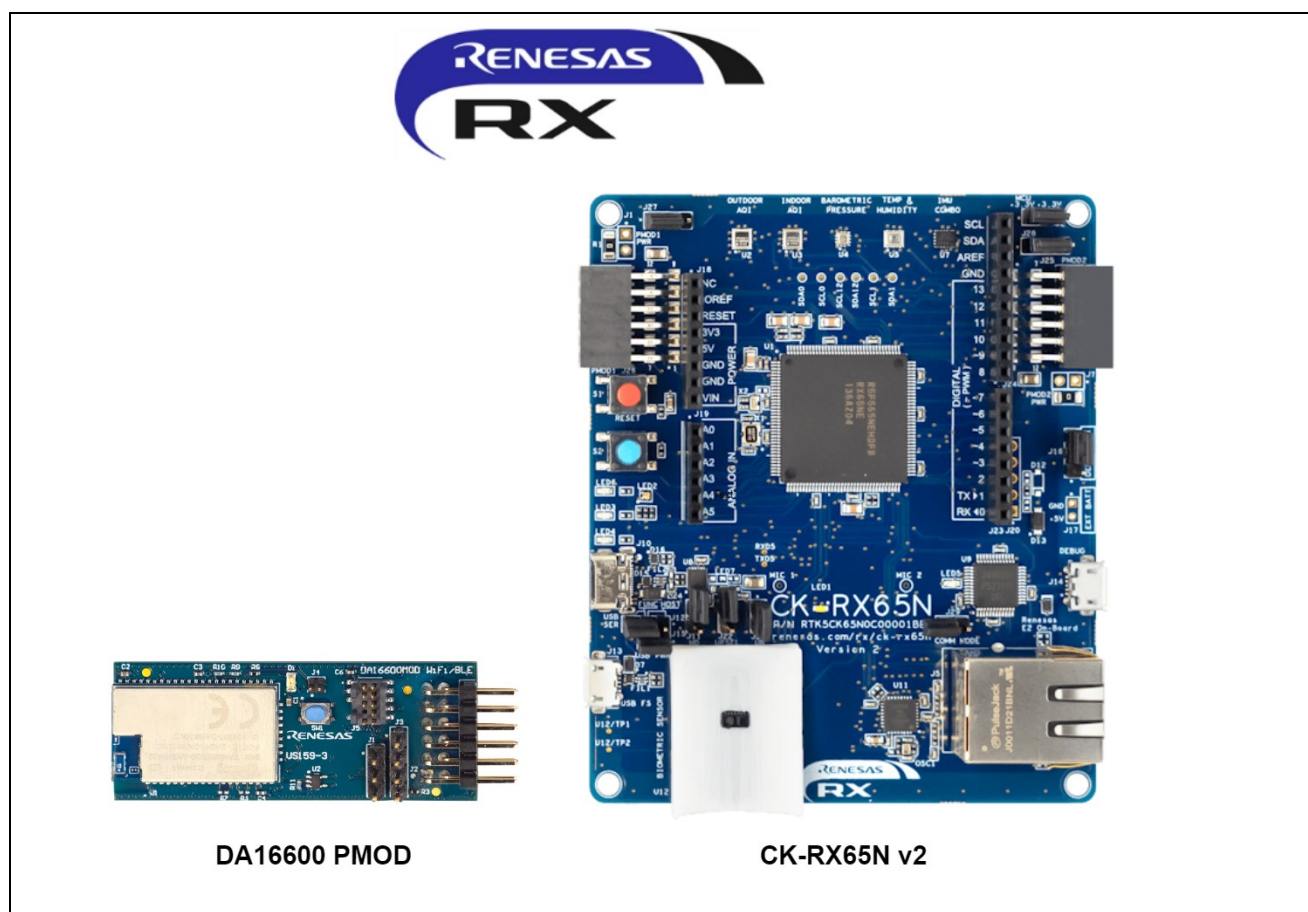


Figure 1. CK-RX65N v2 with Wi-Fi DA16600 Pmod

Contents

Introduction	1
1. Terms	5
2. Overview	5
2.1 Hardware Configuration	5
2.2 Software Configuration	6
2.3 Tera term Setting	6
3. System Diagram	7
4. Cloud Connectivity Application Example	8
4.1 Overview	8
4.2 MQTT/TLS Application Software Overview	10
5. Connection to AWS	14
5.1 Hardware Setup and Import of the Project	14
5.1.1 Hardware Preparation	14
5.1.2 Connecting the Board to the Serial port Console of the PC	15
5.1.3 Import the Project	17
5.1.4 Running the Application Project	32
5.2 For Users Using the Provided Dashboard and AWS Account of Kit	33
5.2.1 Getting the UUID Information of the Board	33
5.2.2 To Get the Account 10 USD of Trial of AWS	34
5.3 Software Preparation-Run Project from IDE	37
5.3.1 Storing the Device Certificate, Key, MQTT Broker Endpoint, and IoT Thing Name	37
5.3.2 Storing the SSID Name, Password, and Security Option	41
5.3.3 Starting the Application	44
5.4 For User Who Use Their Own AWS Account	45
5.4.1 Get an AWS Account	45
5.4.2 Log in to the AWS Management Console	46
5.4.3 Move to IoT Core Control Panel	46
5.4.4 Create a Security Policy	47
5.4.5 Register your device (thing) with AWS IoT	49
5.4.6 Check AWS IoT Endpoints	52
5.4.7 Running Application	52
5.5 Verifying the Application Project using AWS Dashboard and Renesas Dashboard	52
5.5.1 Subscribe to a Topic Message on the AWS IoT	52
5.5.2 Publish a Topic Messages on the AWS Dashboard and Renesas Dashboard	55
5.5.2.1 With AWS Dashboard	55
5.5.2.2 With Renesas Dashboard	56

6.	OTA over MQTT	57
6.1	Overview OTA	57
6.2	Prerequisites.....	57
6.2.1	Installing Python	57
6.2.2	Installing OpenSSL.....	58
6.2.3	Installing Renesas Image Generator.....	59
6.3	Setting up AWS for OTA	60
6.3.1	Register your Device in AWS	60
6.3.2	Creating an Amazon S3 bucket.....	60
6.3.3	Allocating OTA execution permission to IAM users	62
6.4	Setting up the Device	69
6.4.1	Generating Key Pairs and Certificates	69
6.4.2	Setting up the project	71
6.4.2.1	Creating initial firmware.....	71
6.4.2.2	Running OTA project.....	78
6.5	Updating the Firmware	86
6.5.1	Creating the updated firmware	86
6.5.1.1	Changing the firmware version	86
6.5.1.2	Use Renesas Image Generator to Generate the Updated Firmware	86
6.5.2	Updating the firmware	87
6.5.2.1	Creating New Job.....	87
6.5.2.2	Creating FreeRTOS OTA Job Update	87
6.5.2.3	Entering a Job Name.....	88
6.5.2.4	Updating Devices	88
6.5.2.5	Creating New Profile	89
6.5.2.6	Creating a Profile Naming	90
6.5.2.7	Creating a Profile: Importing Certificate	91
6.5.2.8	Creating a Profile: Entering Path.....	92
6.5.2.9	Confirming Profile Name	92
6.5.2.10	Updating the Firmware	93
6.5.2.11	Choosing the Role for the Job.....	93
6.5.2.12	Waiting until Firmware Reception is Complete	94
6.5.2.13	Confirming that the Firmware Version is a New Version	95
7.	Fleet Provisioning	95
7.1	Overview Fleet Provisioning.....	95
7.2	Setting up AWS for Fleet Provisioning	96
7.2.1	Policy Settings	96
7.2.2	Generating a Claim Certificate and Claim Key Pair	98
7.2.3	Creating a Fleet Provisioning Template	100
7.3	Setting up the Project	105

7.4	Running Fleet Provisioning	105
8.	Note and Trouble Shooting	111
8.1	About Stabilization Time for Sensor	111
8.2	When Build Errors Occur	111
8.3	When Run Errors Occur	112
8.4	Wi-Fi Access Point	112
8.5	Renesas AWS Dashboard account credits and quarantine	112
8.6	When Unable to Log in to the Dashboard (Grafana account)	117
8.7	How to Enable/Disable EC2 Instance	118
8.8	Grafana dashboard display is different from the one in application note	120
8.9	How to check the total amount spent in the AWS account	120
8.10	An error occurs when connecting to AWS	121
8.11	Command to create to create the initial firmware fails (OTA)	121
8.12	Initial firmware cannot be written/ does not start. (OTA)	121
8.13	Firmware does not start after starting the bootloader (OTA)	121
8.14	Firmware does not start after an OTA update (OTA)	121
	Revision History	123

1. Terms

The terms used in this document are explained below.

Table 1. Terms

Term	Meaning
AWS	Amazon Web Service
Pmod	Peripheral Module
MQTT	Message Queuing Telemetry Transport
OTA	Over-The-Air
TLS	Transport Layer Security
UUID	Unique ID for each kit

2. Overview

This section gives an overview of the hardware and software configuration of the demo project and the Tera term settings.

2.1 Hardware Configuration

The hardware configuration of the demo project is listed in the table below.

Table 2. Hardware Configuration

Item	Content	Description
CK-RX65N v2 Cloud Kit	Target board for CK-RX65N v2 Part number: RTK5CK65N0S08001BE	Please see the details at: https://www.renesas.com/rx/ck-rx65n
DA16600 Wi-Fi Pmod module	Wi-Fi connection	This Pmod is used with CK-RX65N v2 for Wi-Fi connection. DA16600 SDK version: v3.2.7.1 or later. Please see the details at: US159-DA16600EVZ - Ultra-Low-Power Wi-Fi + Bluetooth® Low Energy Combo Pmod™ Board (Renesas Quick-Connect IoT) Renesas
PC	Windows® 10 Google Chrome / Microsoft Edge	Recommended OS Web browser used.

2.2 Software Configuration

The software configuration of the demo project is listed in the following table.

Table 3. Software Configuration

Item	Content	Version
Integrated development environment	e ² studio (e² studio Renesas)	2024-04
Compiler	GCC	8.3.0.202311
Communication Software	Tera term (Tera Term - Download (softonic.com))	Version 4.99
Emulator	E2 emulator Lite (on-board)	-
RTOS	AWS FreeRTOS	V202210.01
Python	(Please see detail at: 6.2.1)	V3.11.0 or later
Keygen tool	Win64 OpenSSL (Please see detail at: 6.2.2)	V3.0 or later
Flash programming tool	Renesas Flash Programmer (Renesas Flash Programmer (Programming GUI) Renesas)	V3.12.00
Renesas Image Generator	Supplied with Firmware Update module Rev.2.03 (Please see detail at: 6.2.3)	V3.03

Note: For the CC-RX version software, refer to the Application Note “AWS Cloud Connectivity on CK-RX65N v2 with Wi-Fi DA16600 (CC-RX) – Getting Started Guide”.

2.3 Tera term Setting

Table 4. Tera term Setting

Item	Settings
Baud rate	115200
Data length	8
Parity	None
Stop bits	1
Flow Control	None

3. System Diagram

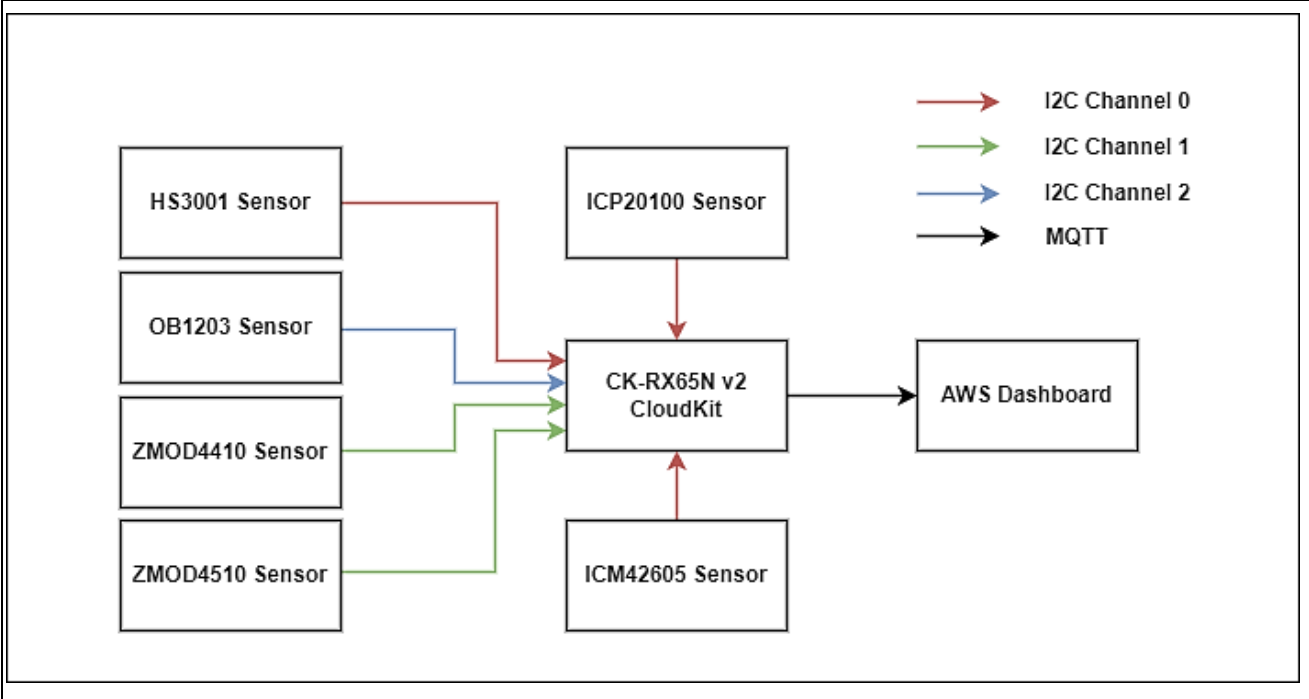


Figure 2. System Diagram

4. Cloud Connectivity Application Example

4.1 Overview

This application project demonstrates the use of Driver, Middleware and RTOS components, FIT configurator on Renesas RX65N MCU to establish AWS Cloud connectivity using Wi-Fi DA16600. It illustrates how the cloud service provider is configured and operated.

This documentation illustrates Subscribe and Publish communications between MQTT Client and MQTT Broker, on-demand publication of sensor data, and asynchronous publication of a "sensor data" event from the MCU to the Cloud.

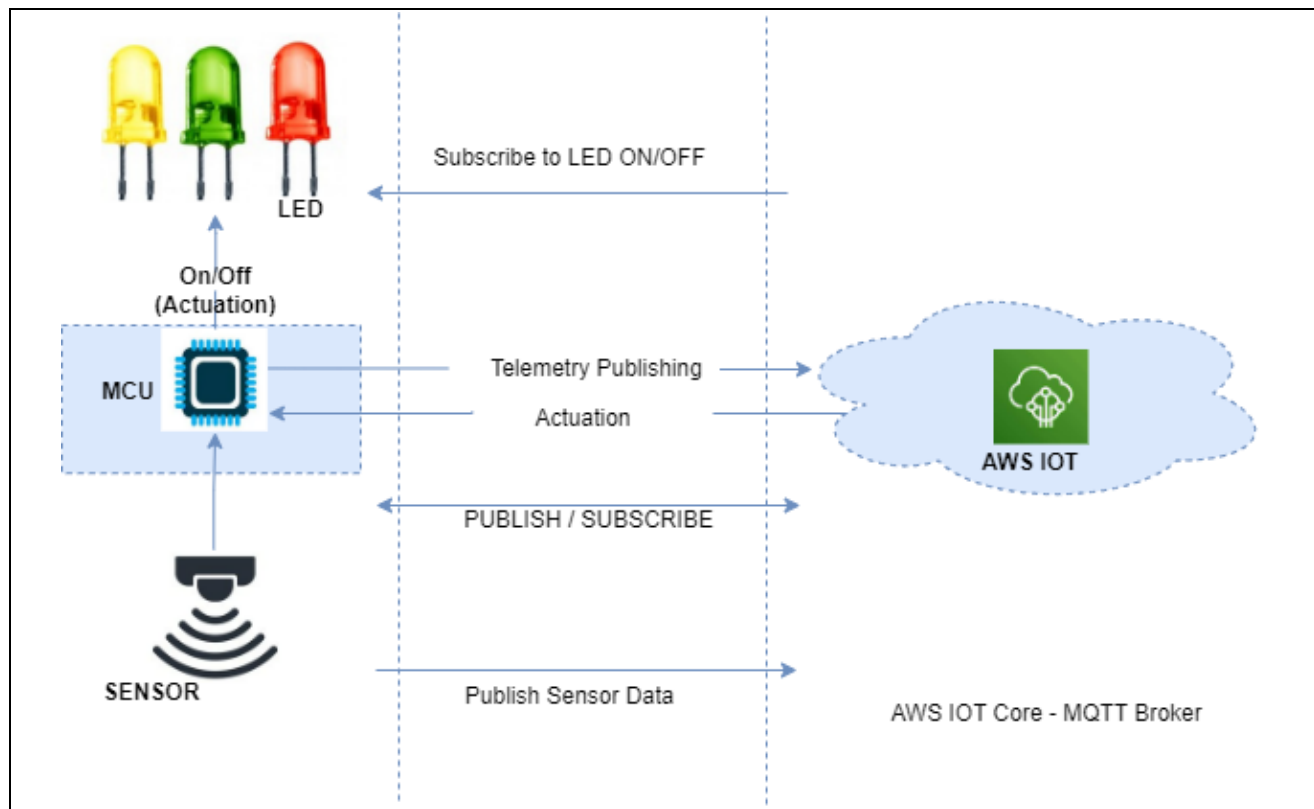


Figure 3. MQTT Publish/Subscribe to/from AWS IoT Core

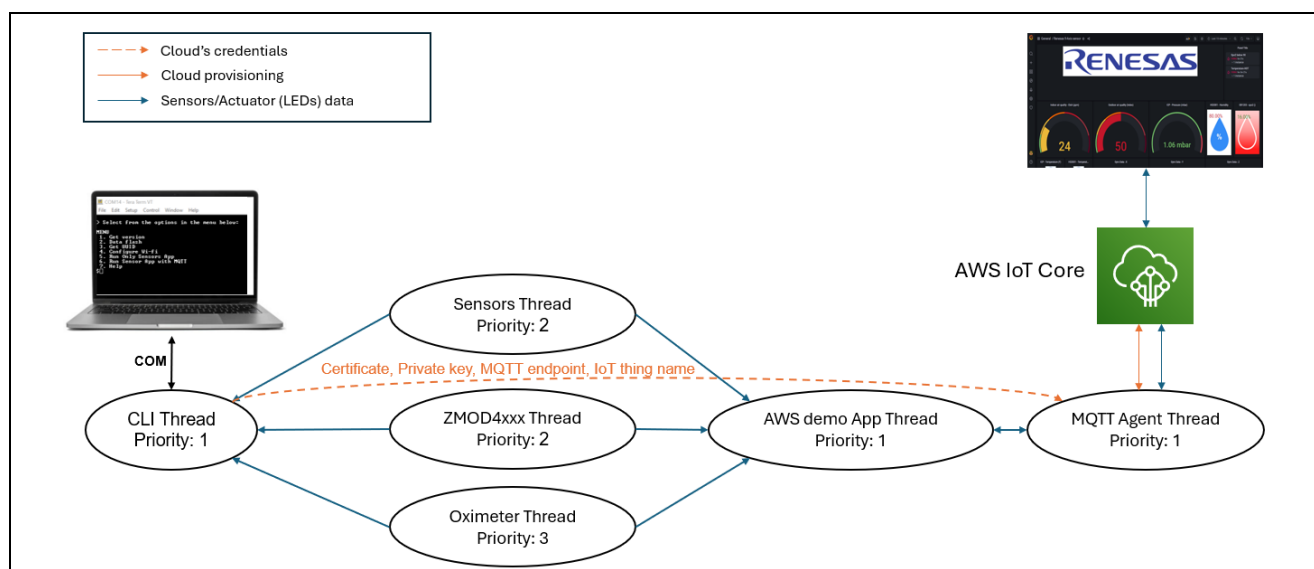
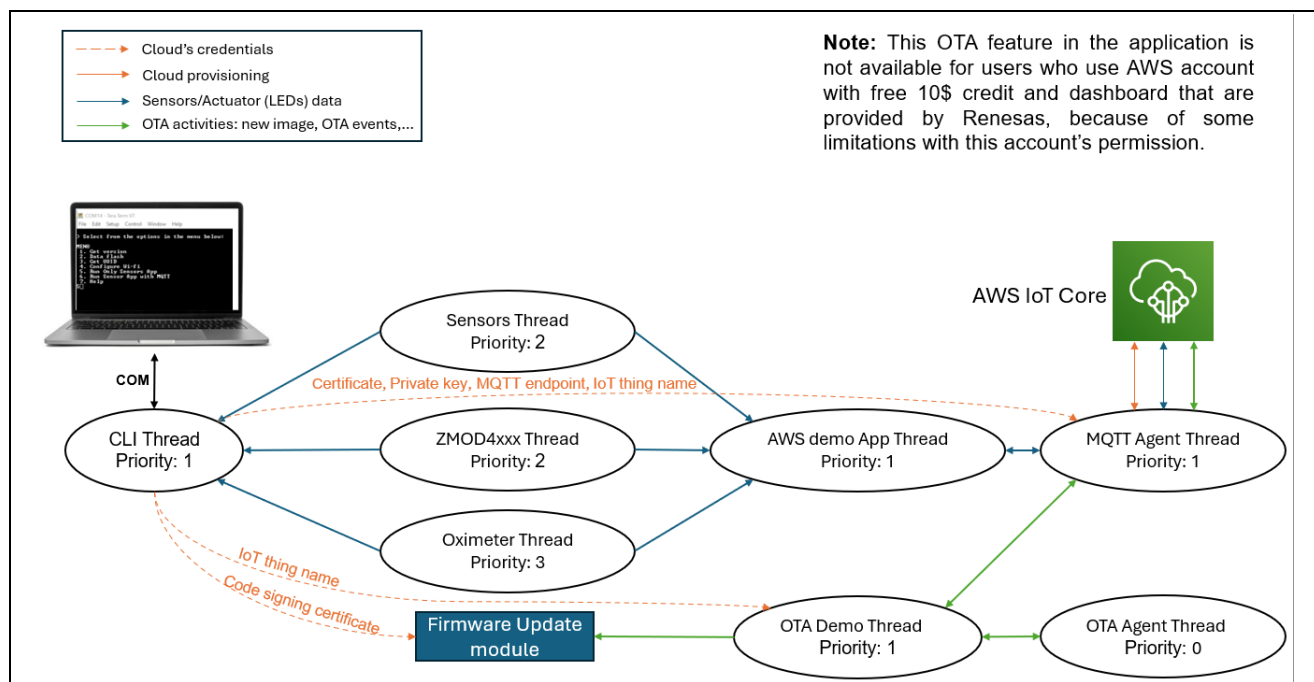


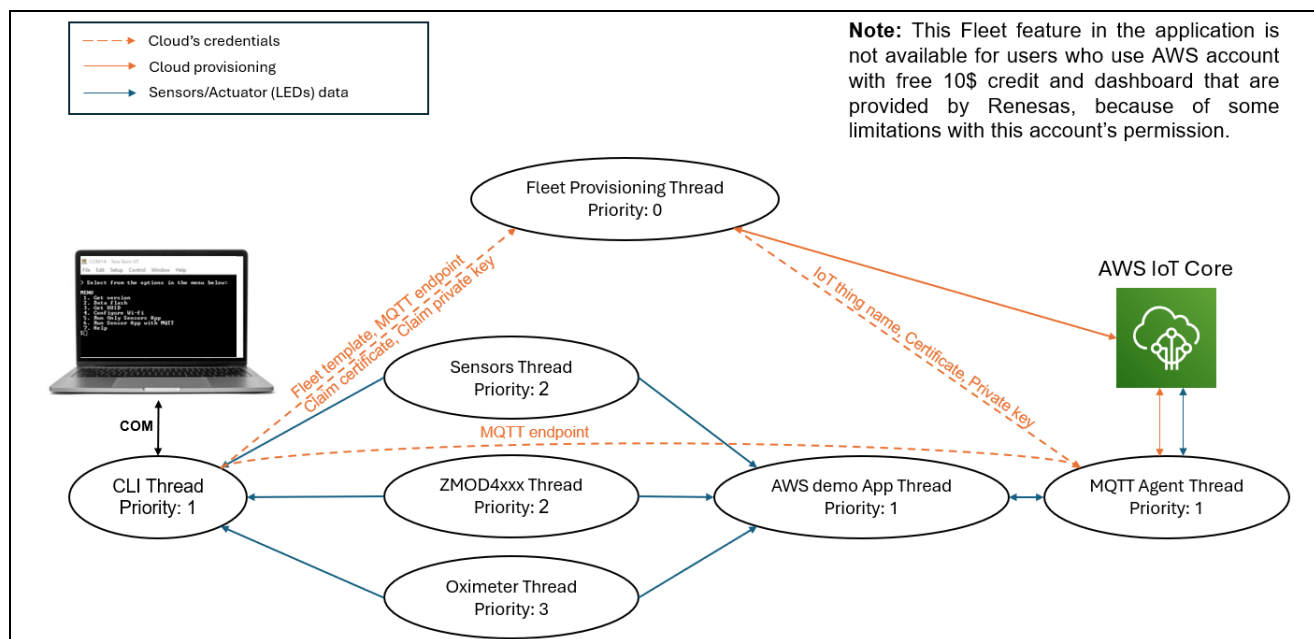
Figure 4. Thread Diagram

The application also supports:

- OTA over MQTT feature for updating new firmware (please refer to the section **6. OTA over MQTT**)

**Figure 5. Thread Diagram when enabling OTA feature**

- Fleet Provisioning (please refer to the section **Fleet Provisioning**)

**Figure 6. Thread Diagram when enabling Fleet feature**

4.2 MQTT/TLS Application Software Overview

The following files from this application project serve as a reference as shown in Table 5.

Table 5. Application Project File

No.	Filename	Purpose
1.	src/application_code/main.c	Contains the initialization code of the Wi-Fi connection, provisioning cloud credentials used in Cloud Connectivity, the main function of the application.
2.	src/application_code/usr_wifi.c	Contains Wi-Fi initialization functions and data structures.
3.	src/application_code/usr_wifi.h	Contains macros, data structures, and functions prototypes used to initialize Wi-Fi across the project.
4.	src/application_code/CommandLine/common_init.h	Contains macros, data structures, and functions prototypes used to initialize common in the project.
5.	src/application_code/CommandLine/console.c	Contains data structures and functions used to print data on the console using UART
6.	src/application_code/CommandLine/console.h	Contains the function prototypes used to print data on the console using UART
7.	src/application_code/CommandLine/menu_flash.c	Contains data structures and functions used to provide CLI flash memory-related menu
8.	src/application_code/CommandLine/menu_flash.h	Contains the function prototypes and macros used to provide CLI flash memory-related menu
9.	src/application_code/CommandLine/menu_kis.c	Contains functions to get the application's version, get UUID and help option for the main menu on the CLI
10.	src/application_code/CommandLine/menu_kis.h	Contains the function prototypes and macros used to get the application's version, get UUID, and help option for the main menu on the CLI
11.	src/application_code/CommandLine/menu_main.c	Contains data structures and functions used to provide CLI main menu options
12.	src/application_code/CommandLine/menu_main.h	Contains the function prototypes and macros used to provide CLI main menu options
13.	src/application_code/CommandLine/common_utils.h	Contains macros, data structures, and functions prototypes commonly used across the project.
14.	src/application_code/CommandLine/r_typedefs.h	Contains typedef used in the application
15.	src/application_code/sensor_thread_entry.c	Contains the code for the sensor thread (HS3001, ICP20100, and ICM42605)
16.	src/application_code/ICM42605/ICM42605.c	Contains the code for the 6-Axis MEMS Motion Tracking™ Sensor
17.	src/application_code/ICM42605/ICM42605.h	Contains the Data structure function prototypes for the 6-Axis MEMS Motion Tracking™ Sensor
18.	src/application_code/ICM42605/apex_feature.c	Contains the code for the apex feature of the 6-Axis MEMS Motion Tracking™ Sensor
19.	src/application_code/ICM42605/icm_i2c.c	Contains the I2C code to communicate with 6-Axis MEMS Motion Tracking™ Sensor

No.	Filename	Purpose
20.	src/application_code/ICM42605/icm_i2c.h	Contains the I2C function prototypes to communicate with 6-Axis MEMS Motion Tracking™ Sensor
21.	src/application_code/ICM42605/motion_sensor_icm_42605.c	Contains the code for 6-Axis MEMS Motion Tracking™ Sensor
22.	src/application_code/ICP20100/ICP_20100.c	Contains the code for Barometric Pressure and Temperature Sensor
23.	src/application_code/ICP20100/ICP_20100.h	Contains data structure and function prototypes for Barometric Pressure and Temperature Sensor
24.	src/application_code/ICP20100/ICP_I2C.c	Contains the I2C code to communicate with Barometric Pressure and Temperature Sensor
25.	src/application_code/ICP20100/ICP_I2C.h	Contains the I2C data structure and function prototypes for Barometric Pressure and Temperature Sensor
26.	src/application_code/ICP20100/pressure_sensor.c	Contains the code for Barometric Pressure and Temperature Sensor
27.	src/application_code/OB1203/RX_OB1203.c	Contains data structures and functions used for the oximeter sensor
28.	src/application_code/OB1203/ob1203_bio.c	Contains the Data structure for the oximeter sensor
29.	src/application_code/OB1203/ob1203_bio_rx.c	Contains data structures and functions used for the oximeter sensor
30.	src/application_code/OB1203/ob1203_bio.h	Contains the Data structure and function prototypes for the oximeter sensor
31.	src/application_code/OB1203/KALMAN/kalman.c	Contains algorithm for Heart Rate, Blood Oxygen Concentration, Pulse Oximetry, Proximity, Light and Color Sensor sample calculations
32.	src/application_code/OB1203/KALMAN/kalman.h	
33.	src/application_code/OB1203/SAVGOL/SAVGOL.c	
34.	src/application_code/OB1203/SAVGOL/SAVGOL.h	
35.	src/application_code/OB1203/SPO2/SPO2.c	
36.	src/application_code/OB1203/SPO2/SPO2.c	
37.	src/application_code/HS3001/RX_HS3001.c	Contains the code and function for Renesas Relative Humidity and Temperature Sensor.
38.	src/application_code/HS3001/RX_HS3001.h	Contains the common data structure's function prototypes for the Renesas Relative Humidity and Temperature sensors.
39.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_Common.c	Contains the common code for the Renesas ZMOD sensors
40.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_Common.h	Contains the common data structure's function prototypes for the Renesas ZMOD sensors
41.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_IAQ1stGen.c	Contains the common code for the Renesas ZMOD Internal Air Quality sensors
42.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_OAQ1stGen.c	Contains the common code for the Renesas ZMOD Outer Air Quality sensors
43.	src/application_code/ota_fwup_wrap_code/ota_fwup_wrap_flash.c	Contains user functions of FWUP module

No.	Filename	Purpose
44.	src/application_code/ota_fwup_wrap_code/ota_fwup_wrap_flash.h	
45.	src/application_code/ota_fwup_wrap_code/ota_fwup_wrap_verify.c	
46.	src/application_code/ota_fwup_wrap_code/ota_fwup_wrap_verify.h	
47.	src/application_code/frtos_skeleton/ob1203_thread.c	Contains the ob1203 sensor thread (for oximeter sensor)
48.	src/application_code/frtos_skeleton/sensor_thread.c	Contains the sensor's thread (for Renesas Relative Humidity and Temperature Sensor, Barometric Pressure and Temperature Sensor and the 6-Axis MEMS Motion Tracking™ Sensor)
49.	src/application_code/frtos_skeleton/zmod_thread.c	Contains the ZMOD's thread (for Renesas ZMOD Internal Air Quality sensors)
50.	src/application_code/frtos_skeleton/task_function.h	Contains the common data structure's function prototypes for thread
51.	src/application_code/frtos_startup/freertos_object_init.c	Contains the source code for FreeRTOS thread
52.	src/application_code/frtos_startup/freertos_start.c	Contains FreeRTOS user-defined functions
53.	src/application_code/frtos_startup/freertos_start.h	FreeRTOS's user-defined functions header file
54.	src/application_code/frtos_config/*.h	Contains FreeRTOS configuration header file.
55.	src/application_code/sensorsData.h	Contains the common data structure's function prototypes for sensors
56.	Demos/SimplePubSub/simple_pub_sub_task.c	Contains code and functions used in MQTT interface for Cloud Connectivity.
57.	Demos/mqtt_agent/mqtt_agent_task.c	Contains the code for running the MQTT task
58.	Demos/OtaOverMqtt/OtaOverMqttDemoExample.c	Contains function for running OTA over MQTT
59.	Demos/Fleet_Provisioning_With_CSR_Demo	Contains function for running Fleet Provisioning
60.	Demos/cli/serial.c	Contains function for serial communication.
61.	Demos/cli/serial.h	Contains the common data structure's function prototypes for serial.c
62.	Demos/include/*.h	Contains the common data structure's function prototypes for demo function.

Note: The above table only lists some important files in the application.

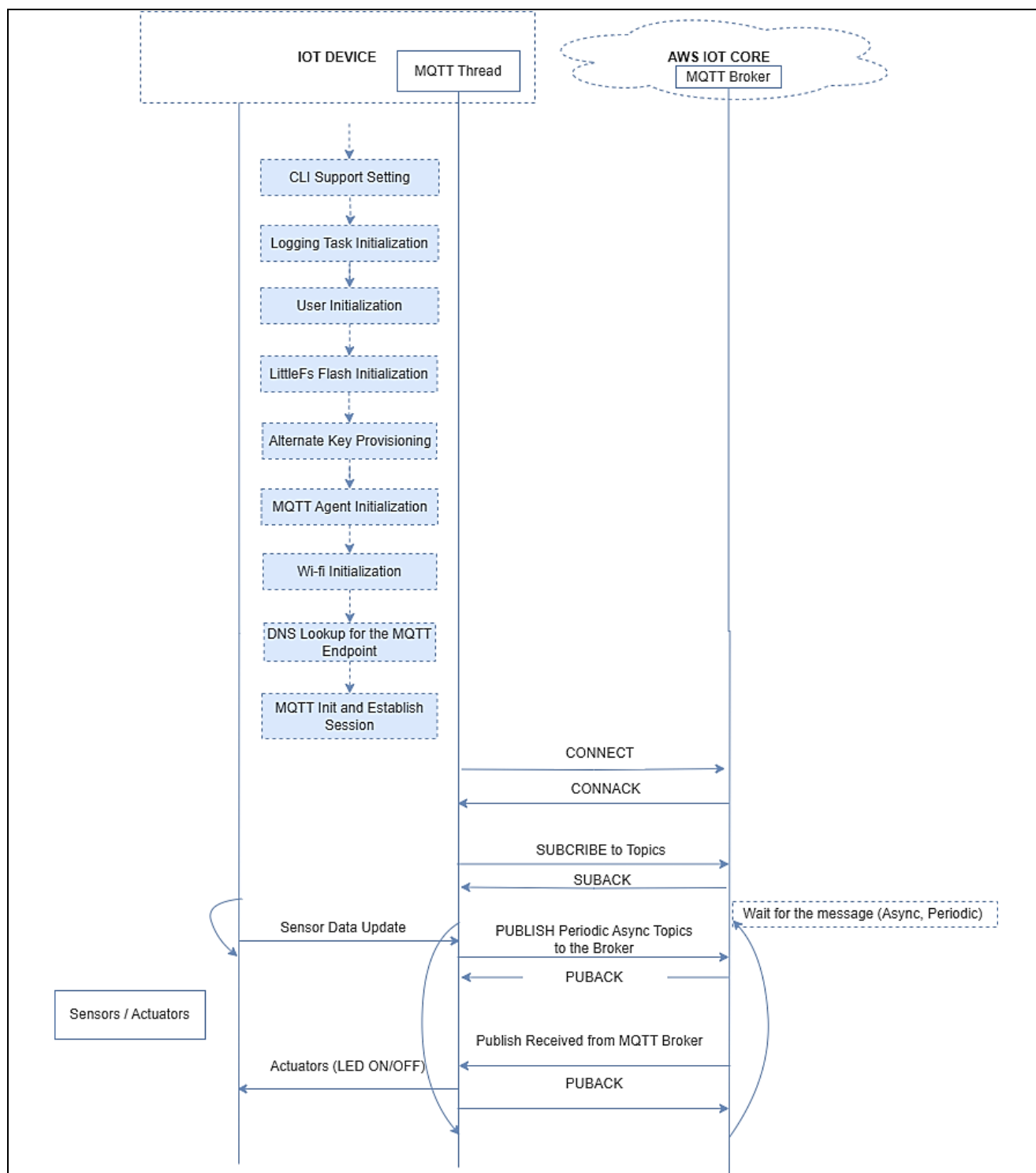


Figure 7. Application Example Implementation Details

The IoT Device (CK-RX65N v2) will perform step by step from initializations for Flash, CLI, Log Task, and Wi-Fi connection to MQTT Init and establish session. After the establish session step is completed, CK-RX65N v2 (MQTT Client) sends a CONNECT message to the MQTT broker, which responds with a CONNACK message, and the connection is established successfully.

MQTT Client takes both publishers and subscribers, so, continue to send a SUBSCRIBE message with list of desired topics (for example, LED control) and QoS (quality of service) level 1 to MQTT broker.

Note: QoS refers to the level of guarantee or assurance provided for message delivery between the MQTT broker and MQTT Clients. There are three QoS levels in MQTT:

- At most once (0)

- At least once (1)
- Exactly once (2)

The broker responds with a SUBACK message that confirms the subscription and indicates the maximum QoS level that the broker will deliver. At this time, the application sends PUBLISH messages continuously to update the value of sensors to the cloud with the user's configured time (every 2 seconds in default), which responds with a PUBACK message for publishing successfully. In addition, the MQTT broker sends messages to the IoT Device to control the status of LEDs (ON/OFF) on the device. When the network is down, the application will reset DA16600 Pmod and re-connect to the MQTT broker.

5. Connection to AWS

AWS account is necessary to connect CK-RX65N v2 Cloud Kit to AWS.

Renesas provides 10 USD of AWS account credit to users who buy the CK-RX65N v2 and this 10 USD credit cannot be used for an existing account. This document covers both cases to connect to the AWS account.

- **Case 1:** For the users who want to use a trial AWS account with 10 USD credits and Renesas Dashboard, please refer to section **5.2 For Users Using the Provided Dashboard and AWS Account of Kit** to get this AWS account.
- **Case 2:** For other users who already have an AWS account and want to use it instead of a trial account, please skip section **5.2 For Users Using the Provided Dashboard and AWS Account of Kit** and refer to section **5.4 For User Who Use Their Own AWS Account** to use a personal account with the application.

5.1 Hardware Setup and Import of the Project

5.1.1 Hardware Preparation

- Connect micro-USB cables to debug port (J14 on the CK-RX65N v2 board)
- Connect USB Type-C cables to serial port (J10 on the CK-RX65N v2 board)
- Connect the Wi-Fi DA16600 Pmod module to the **Pmod 1**
- **Set the Jumper of J16 "Debug"**

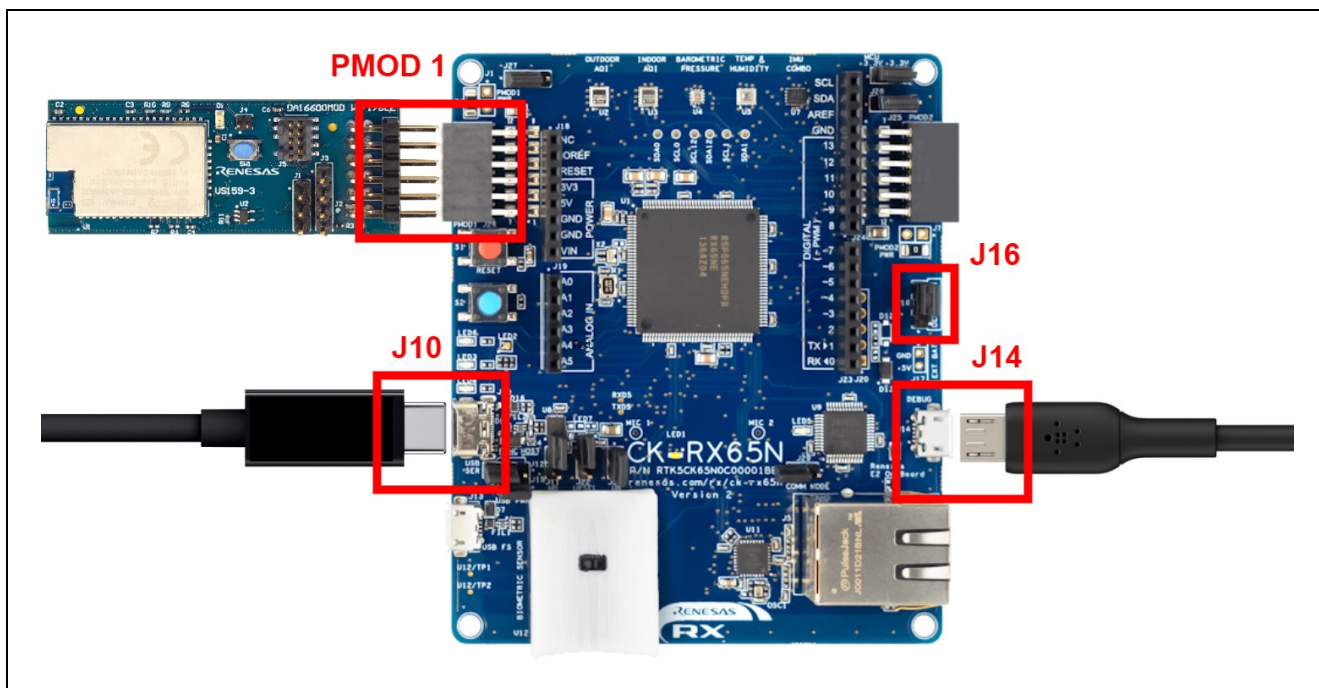


Figure 8. Connecting the USB and DA16600 Pmod

5.1.2 Connecting the Board to the Serial port Console of the PC

1. On the host PC, open Windows Device Manager. Expand **Ports (COM & LPT)**, locate **USB Serial Port (COMxx)**, and note down the COM port number to be used in the next step.

Note: USB Serial Device drivers are required to communicate between the CK-RX65N v2 board and the PC.

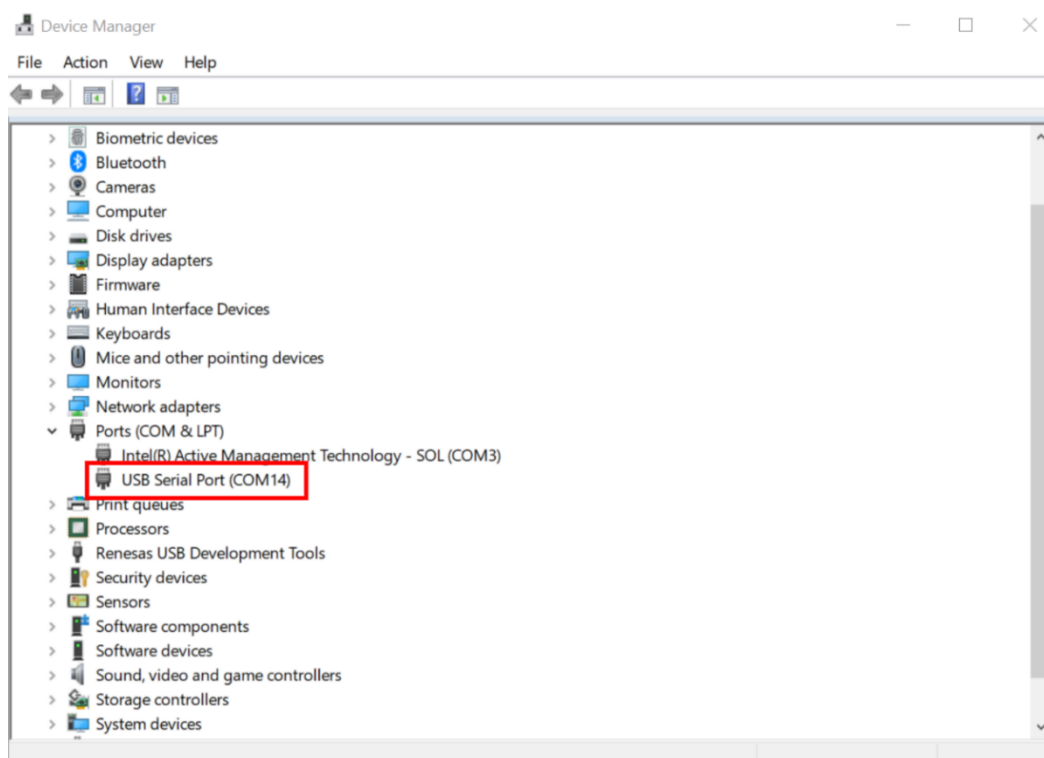


Figure 9. USB Serial Device in Windows Device Manager

Open Tera Term select **New connection** and select **Serial and COMxx: USB Serial Device (COMxx)** and click **OK**.

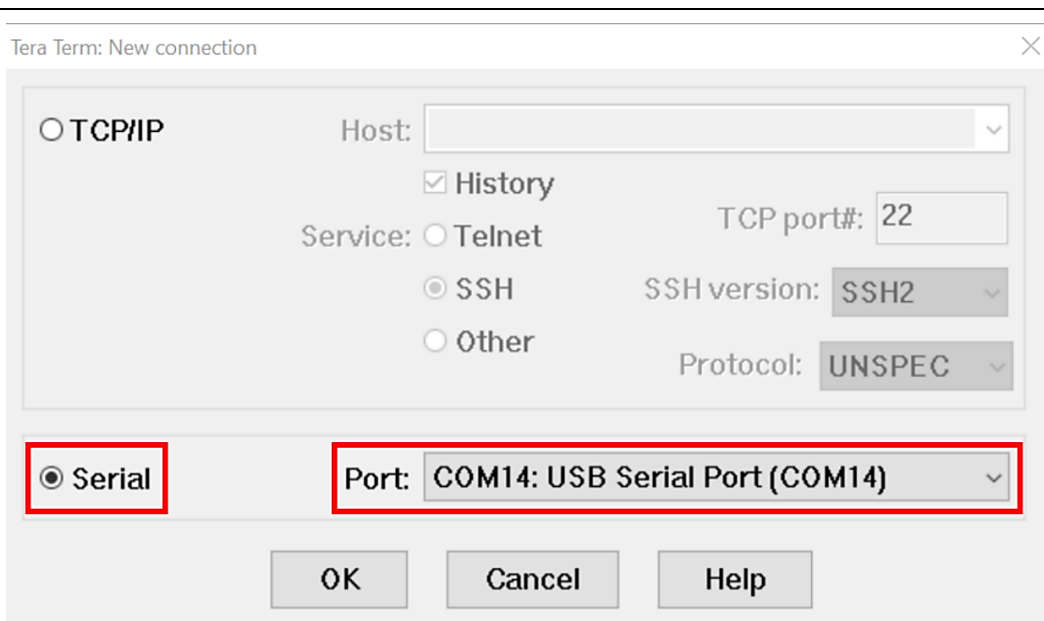


Figure 10. Selecting the Serial Port on Tera Term

- Using the **Setup** menu, select **Setup > Terminal...** and select “**AUTO**” as Receive and, select “**CR**” as Transmit, as shown in **Figure 11**.

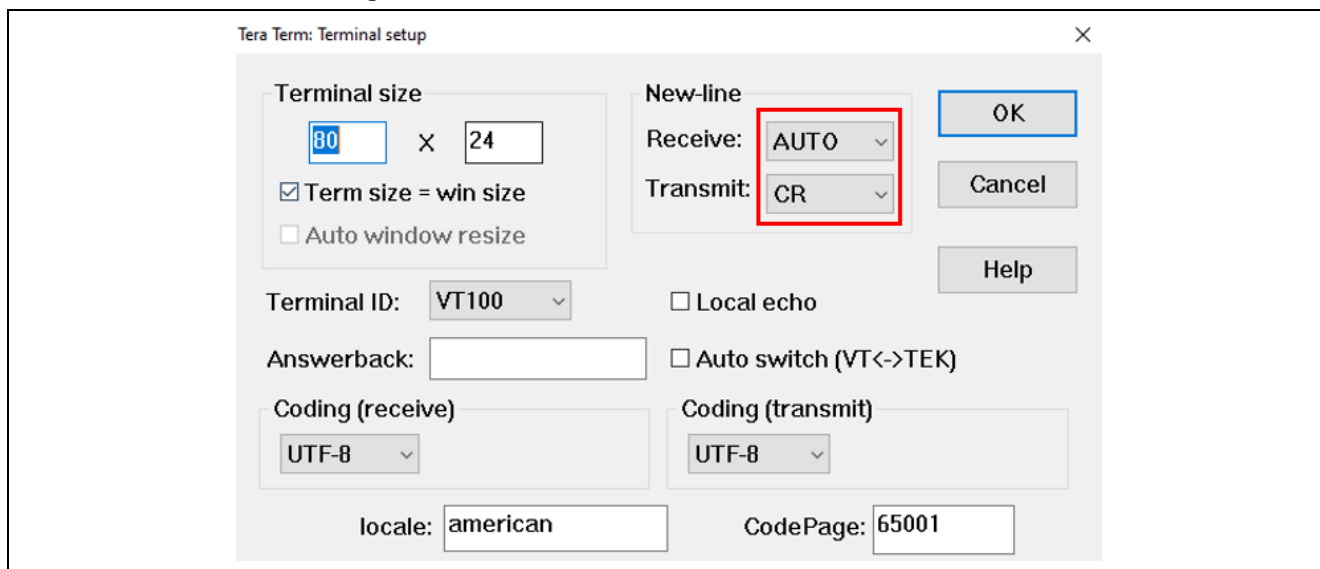


Figure 11. Select Receive: “Auto” and Transmit: “CR” on the Terminal Setting

- Using the **Setup** menu pull-down, select **Serial port...** and ensure that the speed is set to 115200, as shown in **Figure 12**.

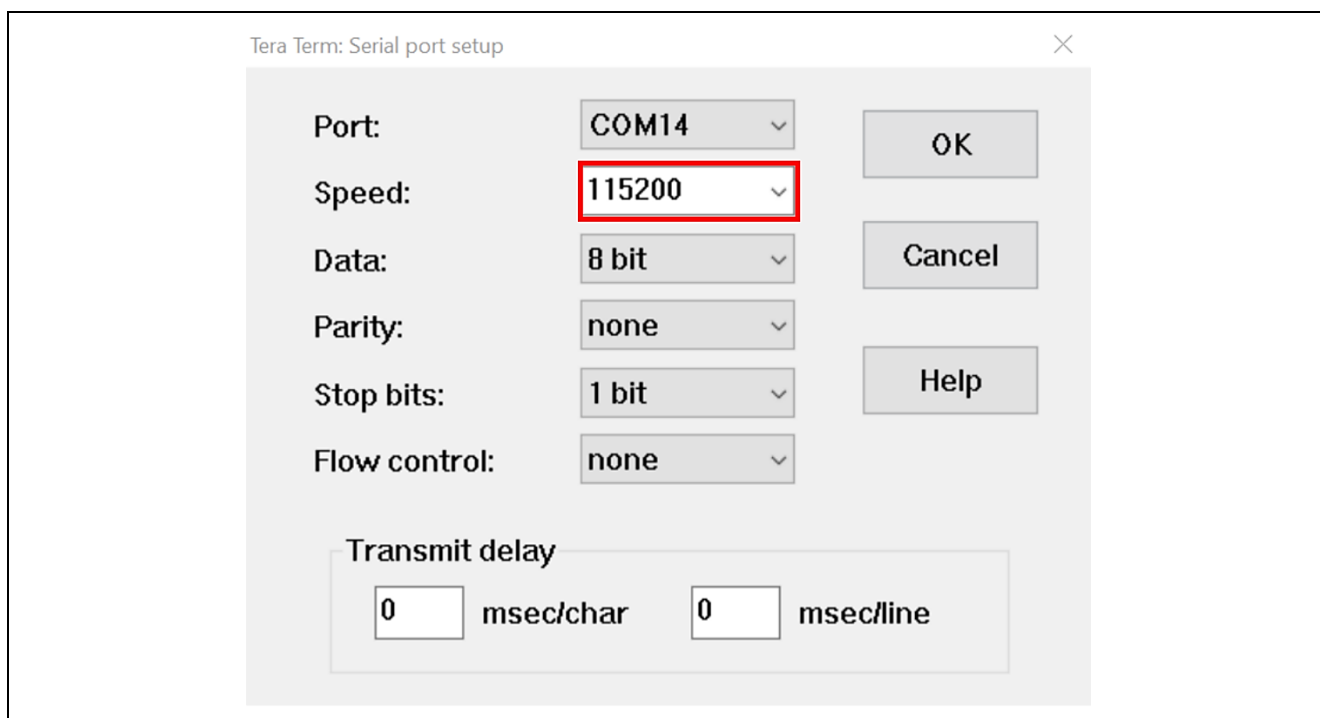


Figure 12. Select 115200 on the Speed Pulldown

5.1.3 Import the Project

Use the following steps to prepare the software for the demo program:

1. Extract the project files from the archive and **unzip the project file to the shortest path of your PC.**
If the path is deep, a build error may occur due to the file path length issue.
2. Launch e² studio and specify a workspace directory and click **Launch**.

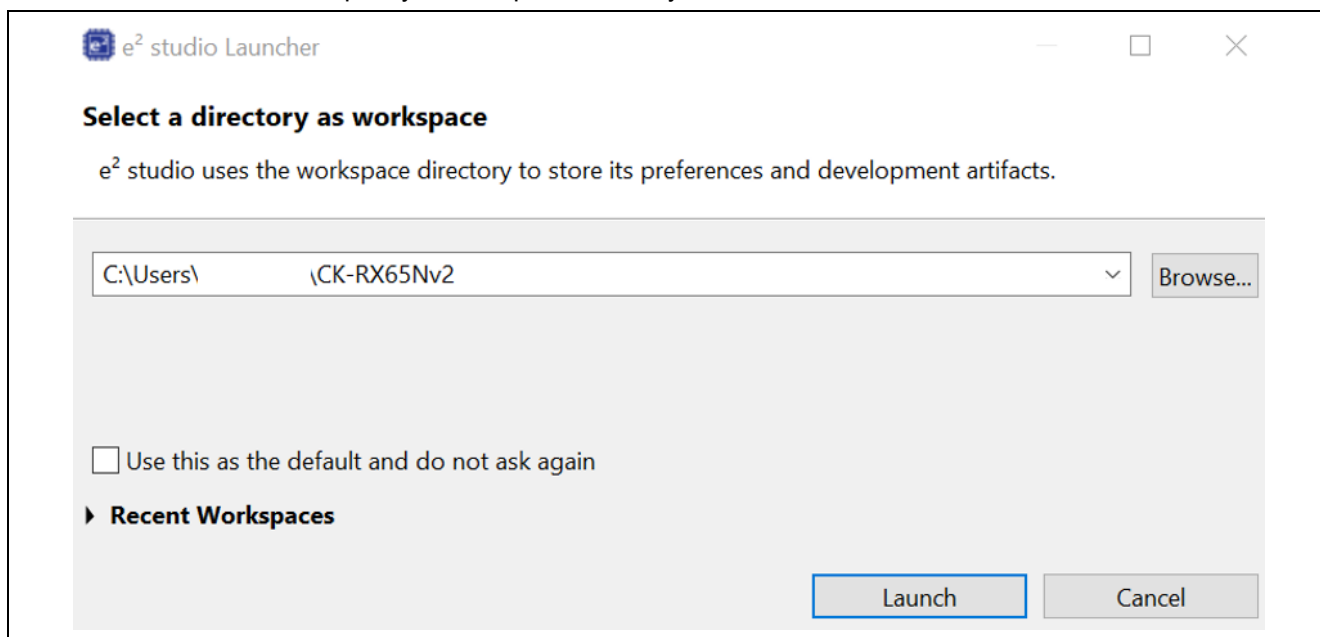


Figure 13. Launch e² studio

3. Select **File > Import...**

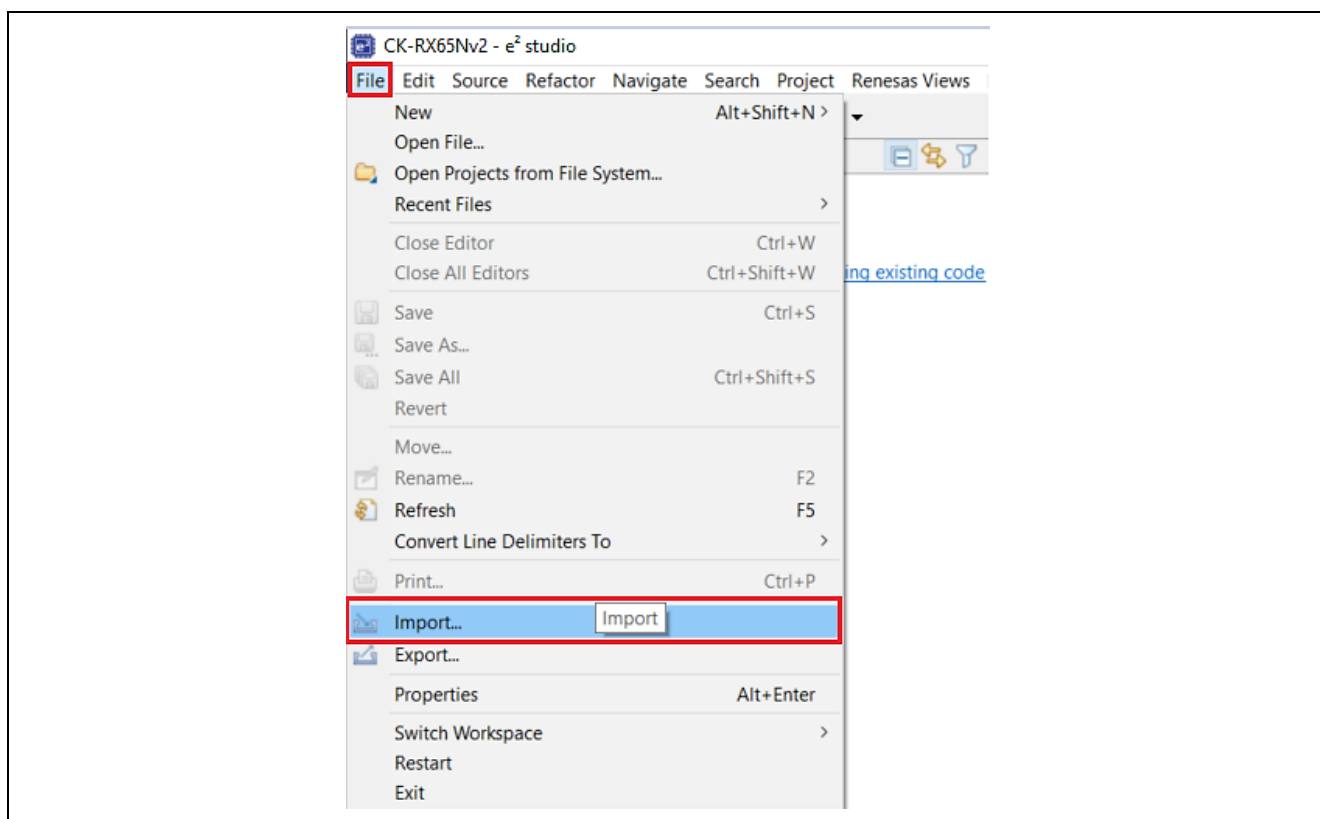


Figure 14. Select Import

4. Click **General** > **Existing Projects into Workspace** > **Next**.

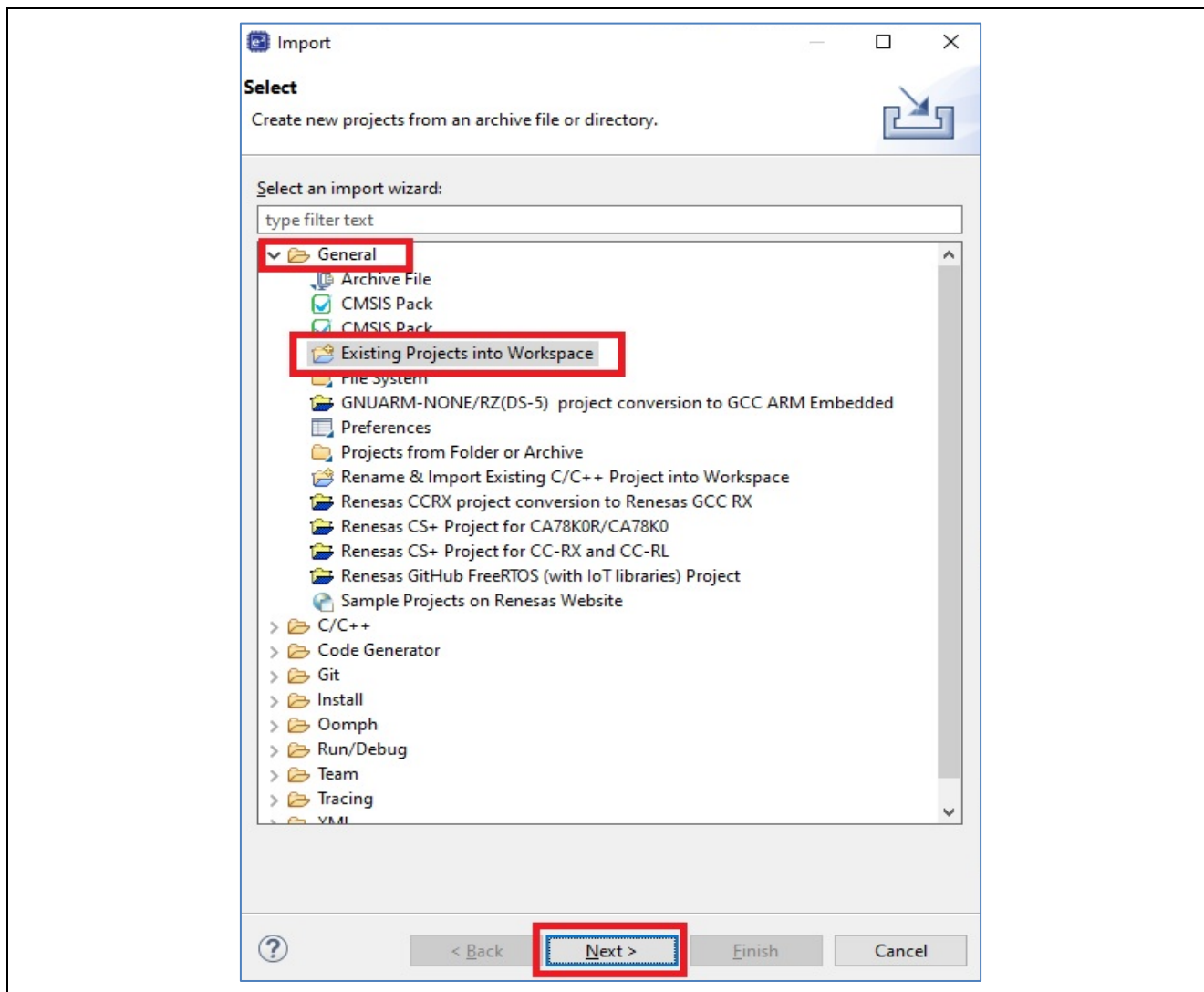


Figure 15. Select Existing Projects into Workspace

5. Click **Browse...**, then specify the root directory as follows.

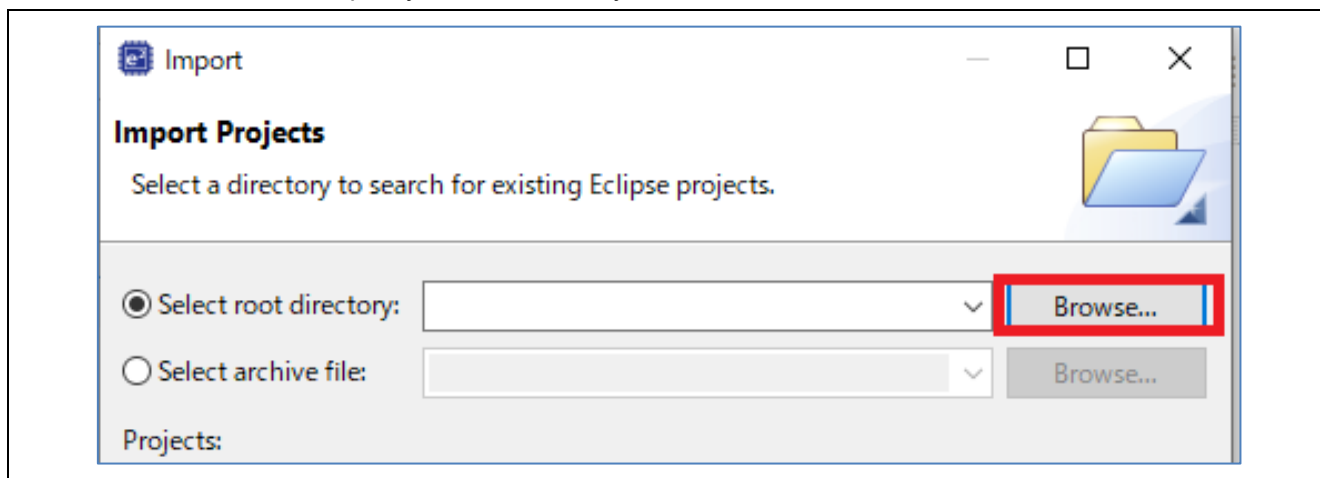


Figure 16. Find the Project

Please go to the “[Project Root folder]\Projects\aws_da16600_ck_rx65n” folder

Project Details

Project Name	Compiler	Connectivity
aws_da16600_ck_rx65n\e2studio_gcc	GCC	Wi-Fi DA16600

Open the “[Project Root folder]\Projects\aws_da16600_ck_rx65n\e2studio_gcc” folder.

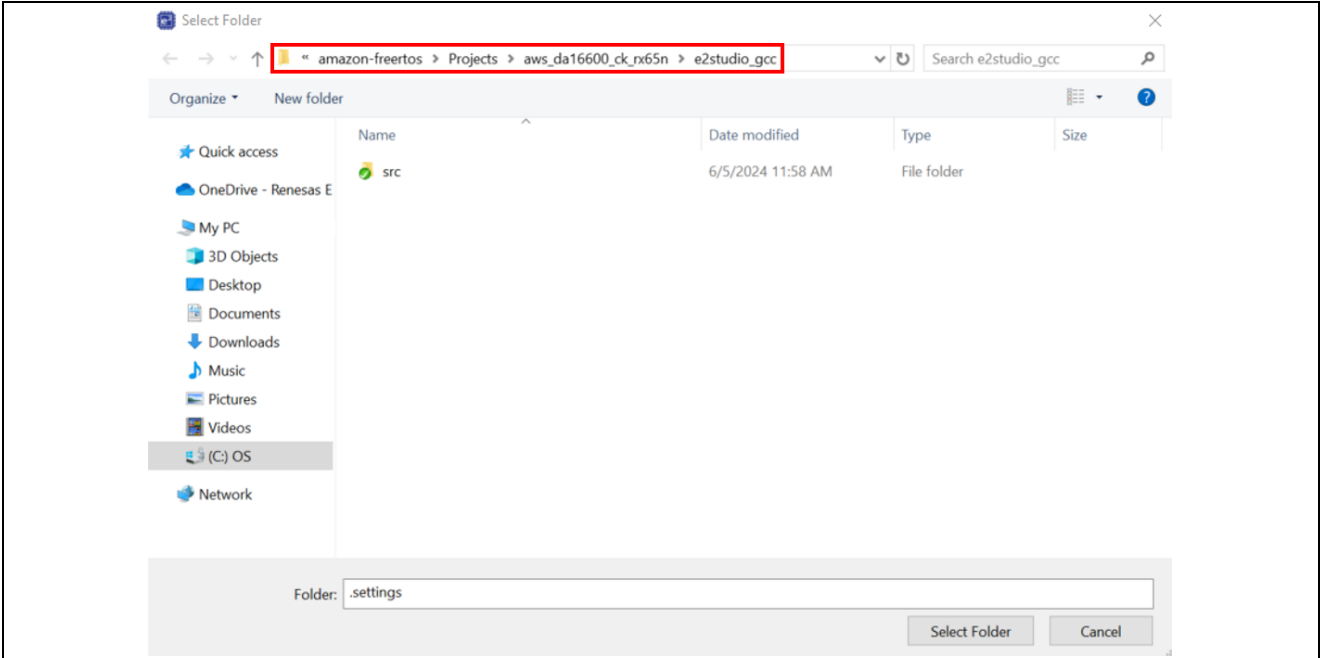


Figure 17. Select the Project Folder

Finally, click **Finish**.

Note: Make sure “Copy projects into workspace” is **unchecked**.

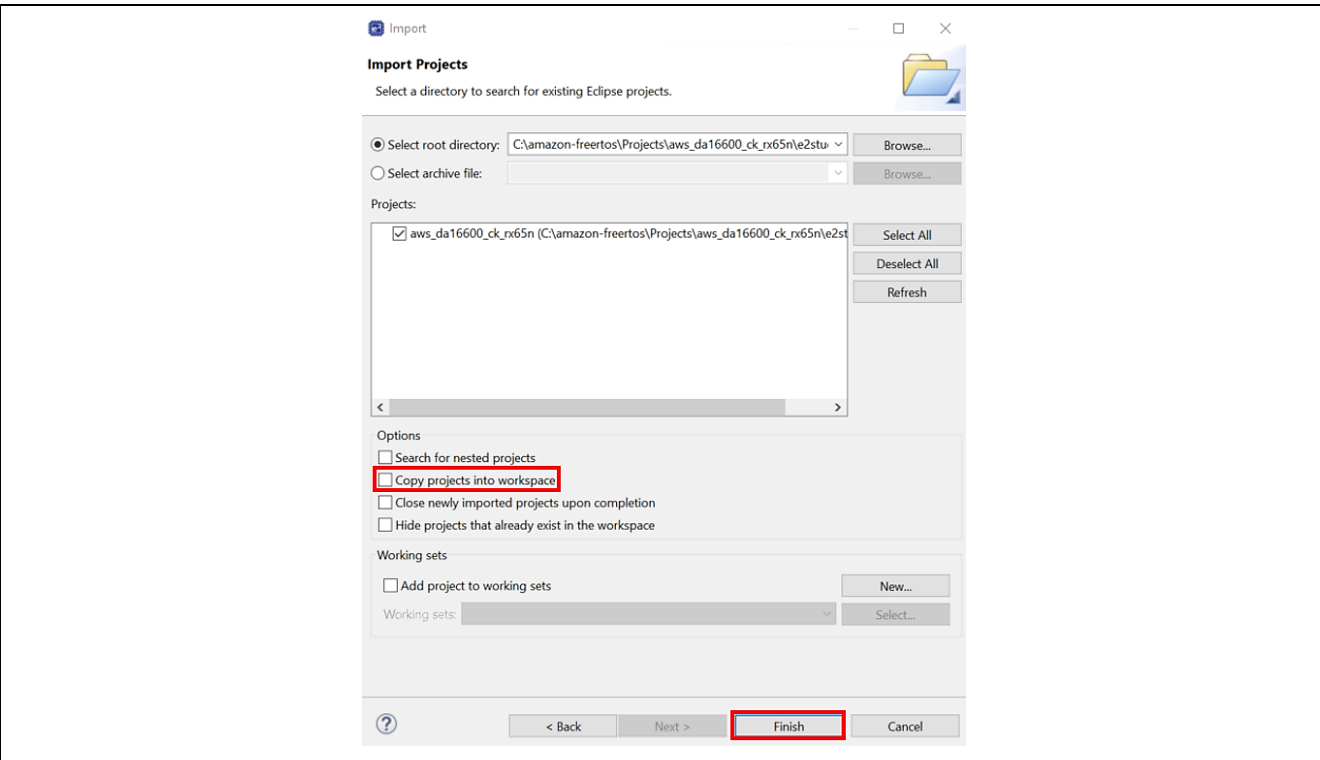


Figure 18. Import the Project

6. Execute code generation.

This application uses FIT; the user can configure them in the “aws_da16600_ck_rx65n.scfg” file.

If you have changed the Smart Configurator settings, click **Generate Code**.

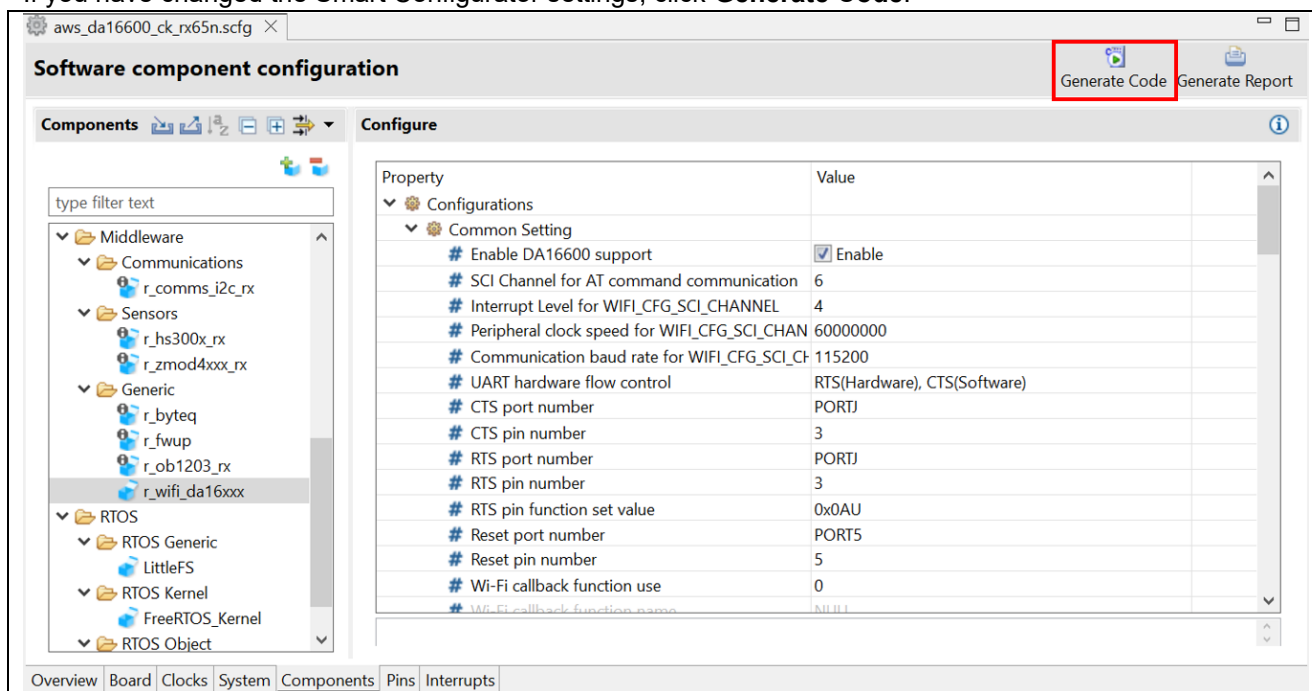


Figure 19. Generate Code

Note: If the user's environment does not have the FIT component's version match with the application, please download them. Users can choose one of the following two ways:

1. Choosing aws_da16600_ck_rx65n.scfg > Components > downloading it

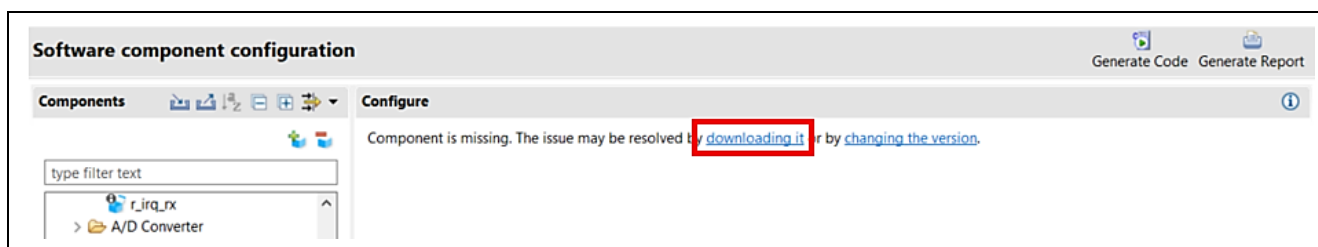


Figure 20. Downloading missing components (1/4)

2. Choosing aws_da16600_ck_rx65n.scfg > Components > Choose “Add component” (Plus symbol)

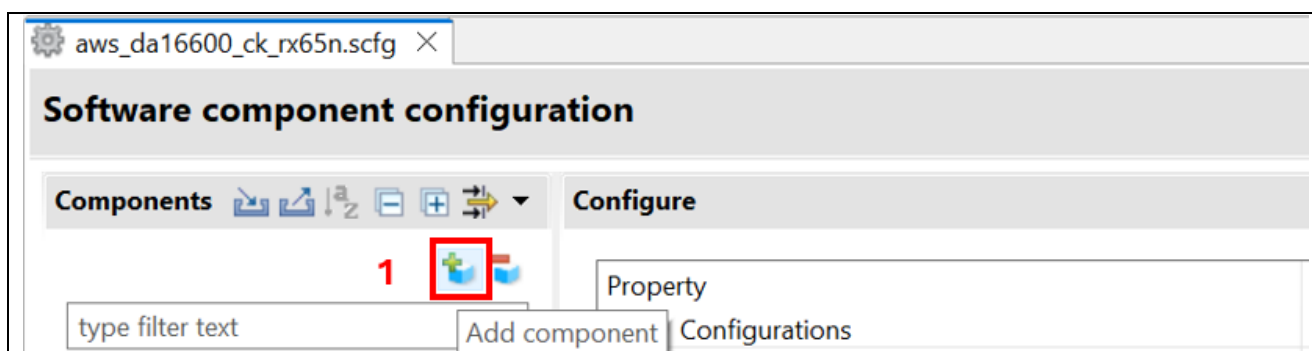


Figure 21. Download missing components (2/4)

Then choose “Download the latest FIT driver and middleware”.

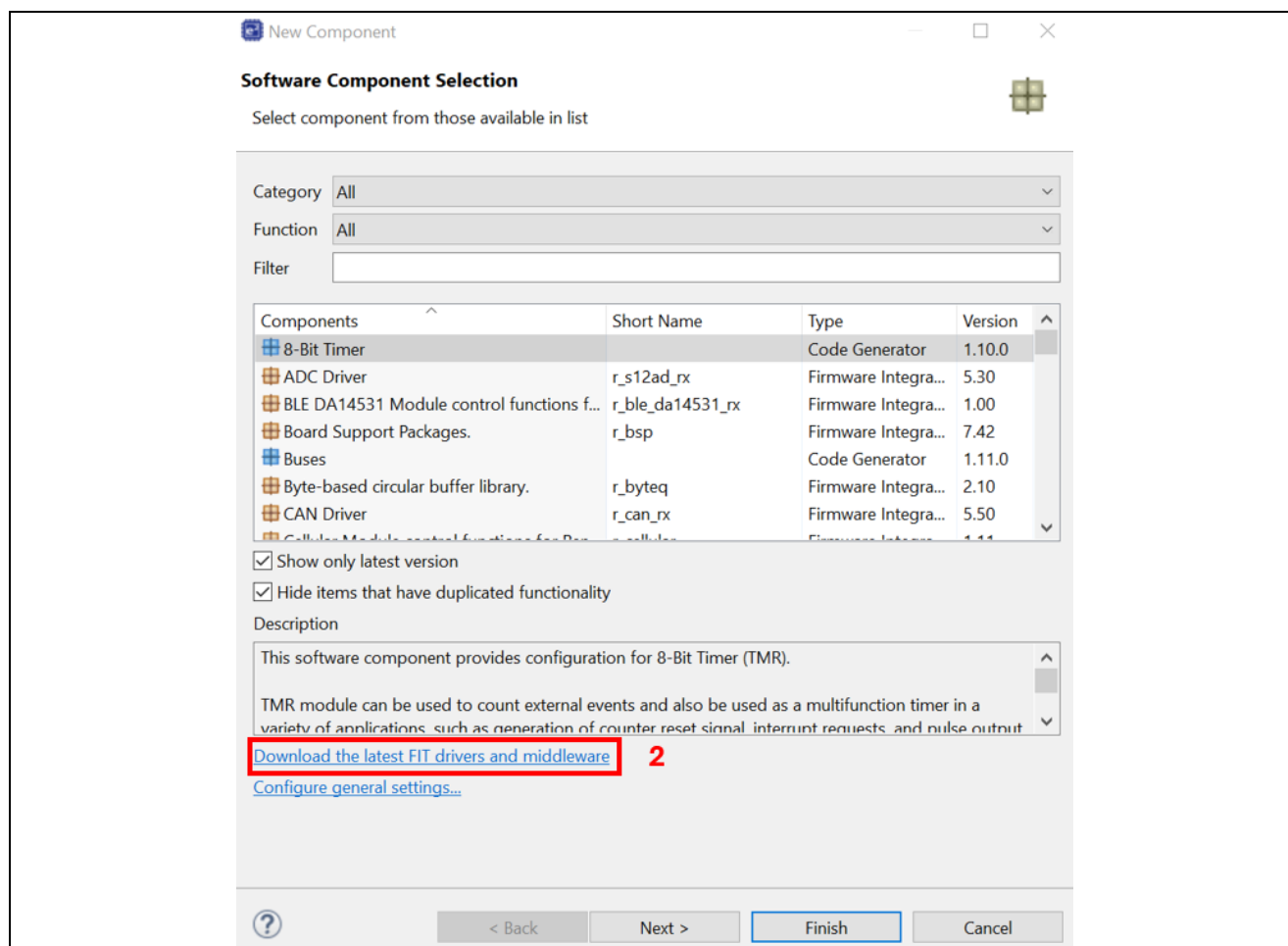


Figure 22. Download missing components (3/4)

Clear the filter check box “**Show RX Driver Package only**”, then choose missing components for the project and click “**Download**”.

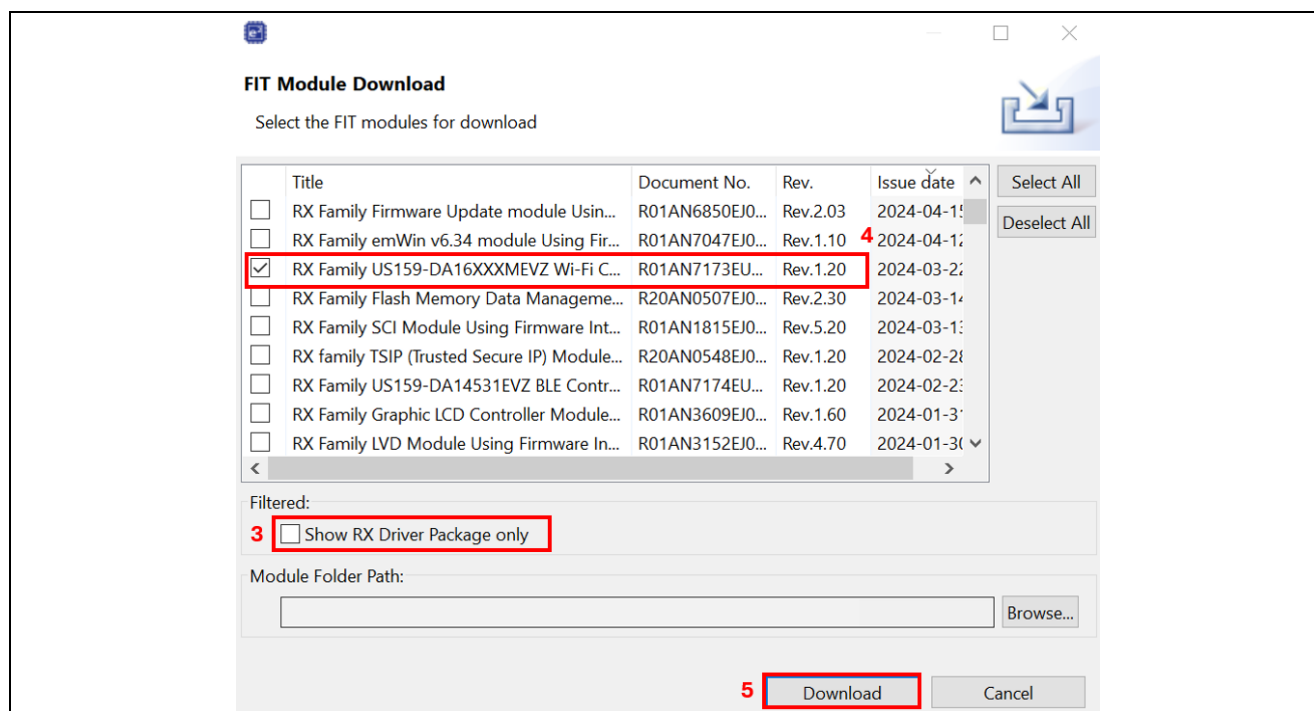


Figure 23. Download missing components (4/4)

Note: If the user's environment does not have CK-RX65N v2 board description file (*.bdf), please download it by choosing **aws_da16600_ck_rx65n.scfg** > **Board** > **Download more boards...** > **New Cloud Kit V2 for RX65N Board Description File** > **Download**

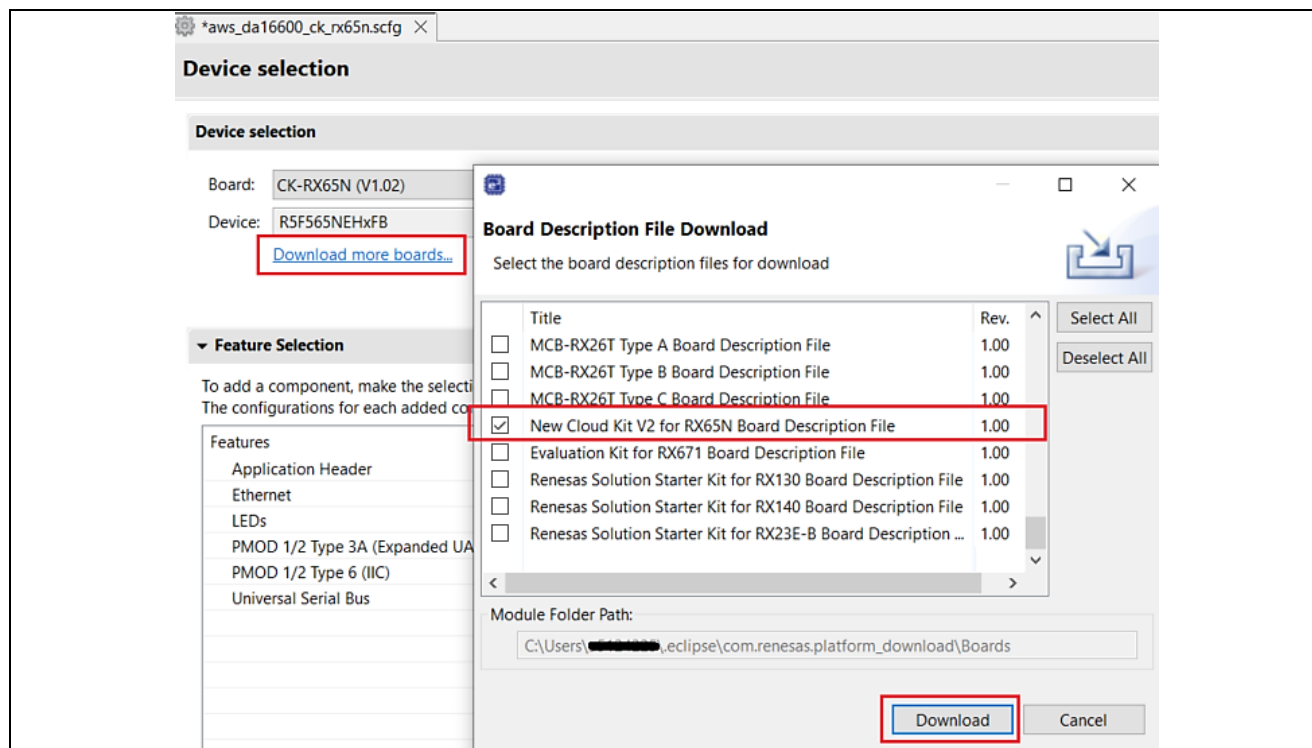


Figure 24. Download CK-RX65N v2 Board Description File on e² studio

Table 6 shows the configuration of each component in this Project.

Table 6. Components Configuration

No	Component	Configuration
1	Startup→Generic→r_bsp (v7.42)	User stack setting: 2 stacks
		User stack size: 0x2000
		Interrupt stack size: 0x400
		Heap size: 0x1000
		Initializes C input and output library functions: Enable
		Enable user stdio charget function: Use BSP charget() function
		Enable user stdio charput function: Use BSP charput() function
		Software Interrupt Unit1 (SWINT1): Unused
		Software Interrupt Unit2 (SWINT2): Unused
		Serial terminals select: Enable
		Channel for serial terminal: Channel 5
		Bitrate for serial terminal: 115200
		Interrupt priority for serial terminal: Priority level 15 (highest)
2	Drivers→Interrupt→r_irq_rx (v3.80)	Select whether to enable bus priority initialization: Disable
		Select whether it is bootloader project: Not bootloader project
		Locking function for IRQ APIs: Enable

No	Component	Configuration
		Resources → ICU: IRQ0 Pin: ✓ IRQ1 Pin: ✓ IRQ2 Pin: ✓ IRQ3 Pin: IRQ4 Pin: IRQ5 Pin: ✓ IRQ6 Pin: ✓ IRQ7 Pin: ✓ IRQ8 Pin: IRQ9 Pin: IRQ10 Pin: IRQ11 Pin: ✓ IRQ12 Pin: IRQ13 Pin: ✓ IRQ14 Pin: ✓ IRQ15 Pin: ✓
3	Drivers→A/D Converter→r_s12ad_rx (v5.30)	Resources→S12AD→S12AD1: ✓ →AN115 Pin: ✓ →AN117 Pin: ✓
4	Drivers→I/O Ports→r_gpio_rx (v5.00)	Configurations → Parameter checking: System Default
5	Drivers→Memory→r_flash_rx (v5.11)	Enable code flash programming: Includes code to program ROM area Enable BGO/Non-blocking data flash operations: Enable BGO (background operations/interrupt) mode Enable BGO/Non-blocking code flash operations: Enable BGO (background operations/interrupt) mode Enable code flash self-programming: Programming code flash while executing from another segment in ROM
6	Drivers→Security→r_tsip_rx (v1.20.l)	-
7	Drivers→Communications→r_riic_rx (v2.49)	MCU supported channels for CH0: Supported MCU supported channels for CH1: Supported CH0 RIIC bps(kbps): 400 CH1 RIIC bps(kbps): 400 Resources→RIIC RIIC0: ✓ - SCL0 Pin: ✓ Used - SDA0 Pin: ✓ Used RIIC1: ✓ - SCL1 Pin: ✓ Used - SDA1 Pin: ✓ Used
8	Drivers→Communications→r_sci_iic_rx (v2.49)	MCU supported channels for CH12: Supported Resource→SCI SCI12: ✓ →SSCL12 Pin: ✓ Used →SSDA12 Pin: ✓ Used
9	Drivers→Communications→r_sci_rx (v5.00)	Include software support for channel 5: Include Include software support for channel 6: Include ASYNC mode TX queue buffer size for channel 5: 80 ASYNC mode TX queue buffer size for channel 6: 2180

No	Component	Configuration
		ASYNC mode RX queue buffer size for channel 5: 80 ASYNC mode RX queue buffer size for channel 6: 8192 Transmit end interrupt: Enable GROUPBL0 (ERI, TEI) interrupt priority: 3 Resources → SCI →SCI5: ✓ - RXD5/SMISO5/SSCL5 Pin: ✓ Used - TXD5/SMOSI5/SSDA5 Pin: ✓ Used →SCI6: ✓ - RXD6/SMISO6/SSCL6 Pin: ✓ Used - TXD6/SMOSI6/SSDA6 Pin: ✓ Used - CTS6#/RTS6#/SS6# Pin: ✓ Used
10	Middleware → Communications → r_comms_i2c_rx (v1.22)	Number of I2C Share Buses: 3 Number of I2C Communication Devices: 7 I2C Driver Type for I2C Shared Bus0: RIIC Channel No. for I2C Shared Bus0: 0 I2C Driver Type for I2C Shared Bus1: RIIC Channel No. for I2C Shared Bus1: 1 I2C Driver Type for I2C Shared Bus2: SCI IIC Channel No. for I2C Shared Bus2: 12 I2C Shared Bus No. for I2C Communication Device0: I2C Shared Bus0 Slave address for I2C Communication Device0: 0x44 Callback function for I2C Communication Device0: rm_hs300x_callback0 I2C Shared Bus No. for I2C Communication Device1: I2C Shared Bus1 Slave address for I2C Communication Device1: 0x32 Callback function for I2C Communication Device1: rm_zmod4xxx_callback0 I2C Shared Bus No. for I2C Communication Device2: I2C Shared Bus1 Slave address for I2C Communication Device2: 0x33 Callback function for I2C Communication Device2: rm_zmod4xxx_callback1 I2C Shared Bus No. for I2C Communication Device3: I2C Shared Bus2 Slave address for I2C Communication Device3: 0x53 Callback function for I2C Communication Device3: rm_ob1203_callback0 I2C Shared Bus No. for I2C Communication Device4: I2C Shared Bus2 Slave address for I2C Communication Device4: 0x53 Callback function for I2C Communication Device4: rm_ob1203_callback1 I2C Shared Bus No. for I2C Communication Device5: I2C Shared Bus0 Slave address for I2C Communication Device5: 0x63 Callback function for I2C Communication Device5: comms_i2c_callback_icp

No	Component	Configuration
		I2C Shared Bus No. for I2C Communication Device6: I2C Shared Bus0 Slave address for I2C Communication Device6: 0x68 Callback function for I2C Communication Device6: comms_i2c_callback_icm
11	Middleware→Sensors→r_hs300x_rx (v1.23)	Number of HS300x Sensors: 1
		Data types from HS300x Sensor: Humidity and Temperature
		Programming mode for HS300x sensor: ON
		I2C Communication device No. for HS300x sensor device0: I2C Communication Device0
		Callback function for HS300x sensor device0: hs300x_callback
12	Middleware→Sensors→r_zmod4xxx_rx (v1.31)	Number of ZMOD4xxx Sensors: 2
		Operation mode of ZMOD4XXX Sensor0: IAQ 1st Gen. (Continuous)
		I2C Communication device No. for ZMOD4XXX sensor device0: I2C Communication Device1
		I2C callback function for ZMOD4XXX sensor device0: zmod4xxx_user_i2c_callback0
		Enable IRQ from ZMOD4XXX sensor device0: Enabled
		IRQ Callback function for ZMOD4XXX sensor device0: zmod4xxx_user_irq_callback0
		IRQ number for ZMOD4XXX sensor device0: IRQ14
		IRQ trigger for ZMOD4XXX sensor device0: Falling
		IRQ interrupt priority for ZMOD4XXX sensor device0: Priority 10
		Operation mode of ZMOD4XXX Sensor1: OAQ 1st Gen.
		I2C Communication device No. for ZMOD4XXX sensor device1: I2C Communication Device2
		I2C callback function for ZMOD4XXX sensor device1: zmod4xxx_user_i2c_callback1
		Enable IRQ from ZMOD4XXX sensor device 1: Enabled
		IRQ Callback function for ZMOD4XXX sensor device1: zmod4xxx_user_irq_callback1
		IRQ number for ZMOD4XXX sensor device1: IRQ13
		IRQ trigger for ZMOD4XXX sensor device1: Falling
		IRQ interrupt priority for ZMOD4XXX sensor device1: Priority 5
13	Middleware→Generic→r_byteq (v2.10)	Memory allocation for queue control blocks: Static memory allocation
		Number of static queue control block: 32
14	Middleware→Generic→r_fwup (v2.03)	Select the update mode: Dual bank
		Select the function mode: user for User program
		Main area start address: 0xFFFF0000
		Buffer area start address: 0xFFE00000
		Install area size: 0xF0000
		Select the algorithm of signature verification: ECDsa + SHA256
		Enable user disable interrupt function: Use FWUP r_fwup_wrap_disable_interrupt() function

No	Component	Configuration
		Enable user enable interrupt function: Use FWUP r_fwup_wrap_enable_interrupt() function
		Enable user software delay function: Use FWUP r_fwup_wrap_software_delay() function
		Enable user software reset function: Use FWUP r_fwup_wrap_software_reset() function
		Enable user sha256 init function: Use user r_fwup_wrap_sha256_init() function
		User sha256 init function name: ota_sha256_init_function
		Enable user sha256 update function: Use user r_fwup_wrap_sha256_update() function
		User sha256 update function name: ota_sha256_update_function
		Enable user sha256 final function: Use user r_fwup_wrap_sha256_final() function
		User sha256 final function name: ota_sha256_final_function
		Enable user verify ecdsa function: Use user r_fwup_wrap_verify_ecdsa() function
		User verify ecdsa function name: ota_verify_edcsa_function
		Enable user get crypt context function: Use user r_fwup_wrap_get_crypt_context() function
		User get crypt context function name: ota_get_crypt_context_function
		Enable user flash open function: Use user r_fwup_wrap_flash_open() function
		User flash open function name: ota_flash_open_function
		Enable user flash close function: Use user r_fwup_wrap_flash_close() function
		User flash close function name: ota_flash_close_function
		Enable user flash erase function: Use user r_fwup_wrap_flash_erase() function
		User flash erase function name: ota_flash_erase_function
		Enable user flash write function: Use user r_fwup_wrap_flash_write() function
		User flash write function name: ota_flash_write_function
		Enable user flash read function: Use user r_fwup_wrap_flash_read() function
		User flash read function name: ota_flash_read_function
		Enable user bank swap function: Use user r_fwup_wrap_bank_swap() function
		User bank swap function name: ota_bank_swap_function
15	Middleware→Generic→r_ob1203_rx (v1.01)	Number of OB1203 Sensors: 2
		Sensor mode of OB1203 Sensor device0: Proximity sensor mode
		I2C Communication device No. for OB1203 sensor device0: I2C Communication Device 3
		I2C callback function for OB1203 sensor device0: ob1203_comms_i2c_callback
		Enable IRQ from OB1203 sensor device0: Enabled
		IRQ callback function for OB1203 sensor device0: ob1203_irq_callback

No	Component	Configuration
		IRQ number for OB1203 sensor device0: IRQ15
		IRQ trigger for OB1203 sensor device0: Falling
		IRQ interrupt priority for OB1203 sensor device0: Priority 14
		Sensor mode of OB1203 Sensor device1: PPG sensor mode
		I2C Communication device No. for OB1203 sensor device1: I2C Communication Device 4
		I2C callback function for OB1203 sensor device1: ob1203_comms_i2c_callback
		Enable IRQ from OB1203 sensor device1: Enabled
		IRQ callback function for OB1203 sensor device1: ob1203_irq_callback
		IRQ number for OB1203 sensor device1: IRQ15
		IRQ trigger for OB1203 sensor device1: Falling
		IRQ interrupt priority for OB1203 sensor device1: Priority 14
16	Middleware→Generic→r_wifi_da16xx (v1.20)	Enable DA16600 support: ✓ Enable SCI Channel for AT command communication: 6 Interrupt Level for WIFI_CFG_SCI_CHANNEL: 4 Communication baud rate for WIFI_CFG_CHANNEL: 115200 UART hardware flow control: RTS(Hardware), CTS(Software) CTS port number: PORTJ CTS pin number: 3 RTS port number: PORTJ RTS pin number: 3 RTS pin function set value: 0x0AU Reset port number: PORT 5 Reset pin number: 5 AT command transfer buffer size: 1024 AT command receive buffer size: 2048 Use SNTP client service: ✓ Enable Timezone offset in hours (-12 ~ 12): 7 Note: Depending on the user's time zone. Please refer to the Settings of Country code and GMT timezone (Only using Wi-Fi) Country code: "VN" Note: Depending on the user's country. Please refer to the Settings of Country code and GMT timezone (Only using Wi-Fi) Use FreeRTOS logging functionality: ✓ Enable Select Wi-Fi protocol → TCP protocol support: ✓ Enable Creatable sockets number: 4 Socket Receive buffer size: 8192
17	RTOS→RTOS Generic → LittleFS (V202210.01)	.block_count: 70 Starting block of Data Flash allocated to littleFS: FLASH_DF_BLOCK_0_MACRO
18	RTOS→RTOS Kernel→FreeRTOS_Kernel (V202210.01)	RTOS scheduler: Preemptive Maximum number of priorities to the application task: 7 The frequency of the RTOS tick interrupt: (TickType_t) 1000 The size of the stack used by the idle task: 768 The total amount of RAM available in the FreeRTOS heap: 256

No	Component	Configuration
		The maximum permissible length of name: 12
		Use trace facility: ✓ Enable
		Idle should yield: ✓ Enable
		Mutex functionality: ✓ Enable
		Recursive mutex functionality: ✓ Enable
		Counting semaphore functionality: ✓ Enable
		Thread local storage pointers: 3
		Record stack high address: ✓ Enable
		Daemon task startup hook: ✓ Enable
		Queue set functionality: ✓ Enable
		Tick hook: ✓ Enable
		Idle hook: ✓ Enable
		The malloc() failed function: ✓ Enable
		Check for stack overflow: Check by tick value and stack pointer
		Software timer functionality: ✓ Enable
		The priority of the software timer task: 6
		The length of the software timer command queue: 5
		Kernel interrupt priority: 1
		Maximum syscall interrupt priority: 4
		Tick vector: _CMT0_CM10
		Event groups: ✓ Enable
		Maximum number of priorities to the application co-routines: 2
		Dynamic allocation: ✓ Enable
		Static allocation: ✓ Enable
		vTaskPrioritySet: ✓ Enable
		vTaskPriorityGet: ✓ Enable
		vTaskDelete: ✓ Enable
		vTaskSuspend: ✓ Enable
		vTaskDelayUntil: ✓ Enable
		vTaskDelay: ✓ Enable
		xTaskGetSchedulerState: ✓ Enable
		xEventGroupSetBitsFromISR: ✓ Enable
		xTimerPendFunctionCall: ✓ Enable
		xTaskGetCurrentTaskHandle: ✓ Enable
		xTaskAbortDelay: ✓ Enable
		Max message length: 192
		Include time and task name: ✓ Enable
		Include demo debug stats: ✓ Enable
		Max output size: 850
		Platform name: "RenesasRX65N"
		Include platform.h instead of iodef.h: ✓ Enable
		Enable time-slicing for equal priority tasks: ✓ Enable

No	Component	Configuration			
		Initialize	Task Code	Priority	Stack Size
19	RTOS→RTOS Object→FreeRTOS_Object (V202210.01)	kernel start	ob1203_thread	3	1024
		kernel start	zmod_thread	2	1024
		kernel start	sensor_thread	2	2048

7. Data Publishing Interval Settings (Optional)

Data publish interval can be set by the user. Default publishes an interval time of 2 seconds.

“Demos/SimplePubSub/simple_pub_sub_task.c” file has the macro to change the publish time interval.

```
#define mqttexampleDELAY_BETWEEN_PUBLISH_OPERATIONS_MS (2000U)
```

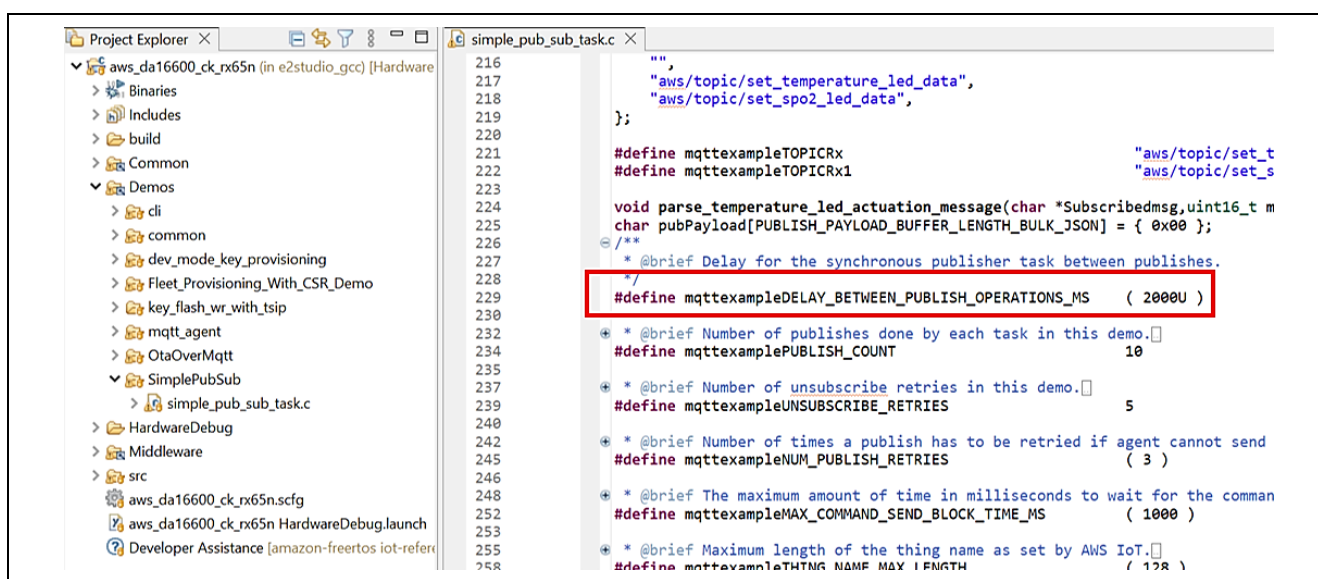


Figure 25. Data Publishing Interval Settings (Optional)

8. Select **Project** → **Build All** and confirm that 0 errors are reported.

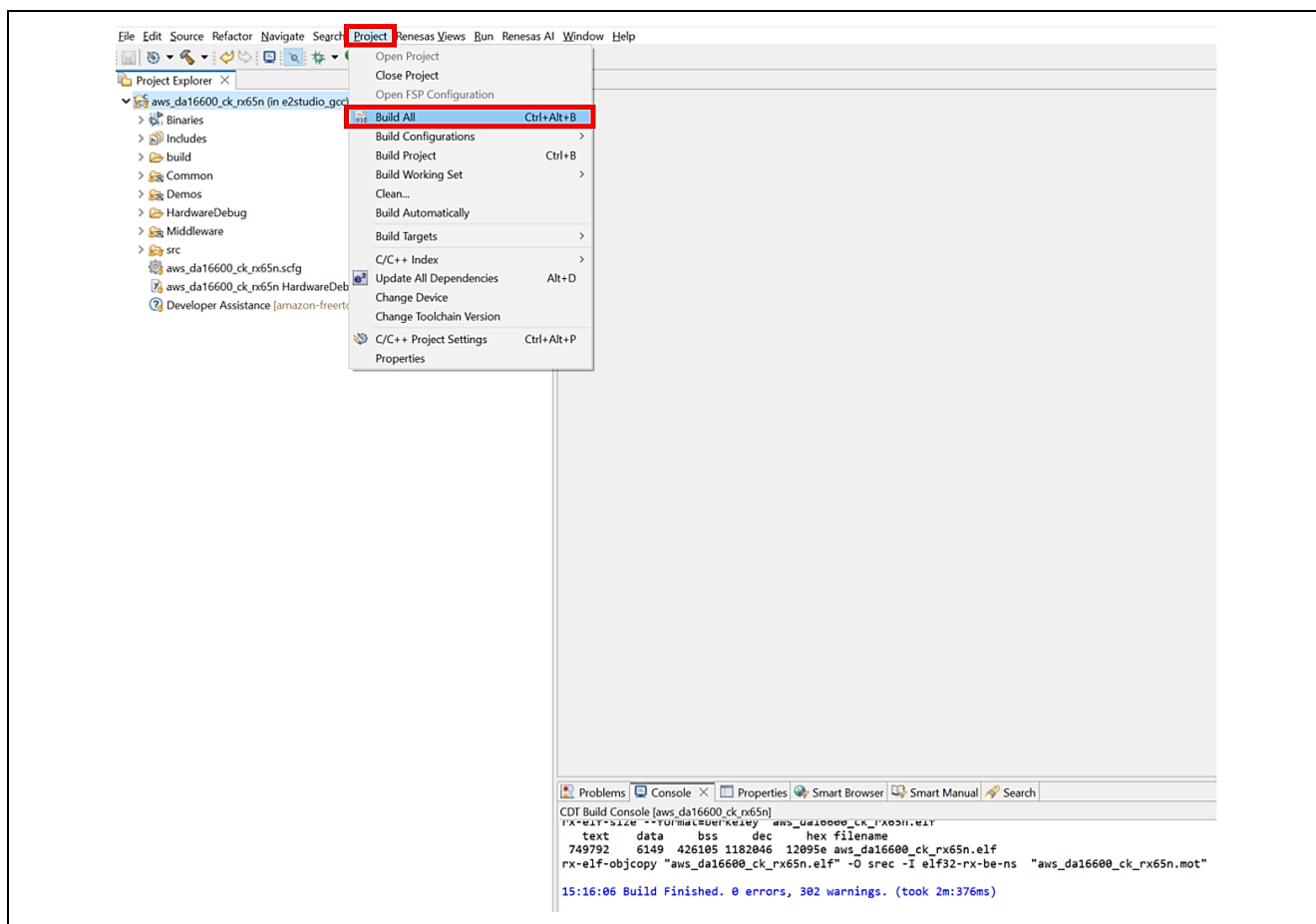


Figure 26. Build the Project

9. Debug configuration and load image to the board:

Open **Debug Configurations**: Right-click on project > **Debug As** > **Debug Configurations...**

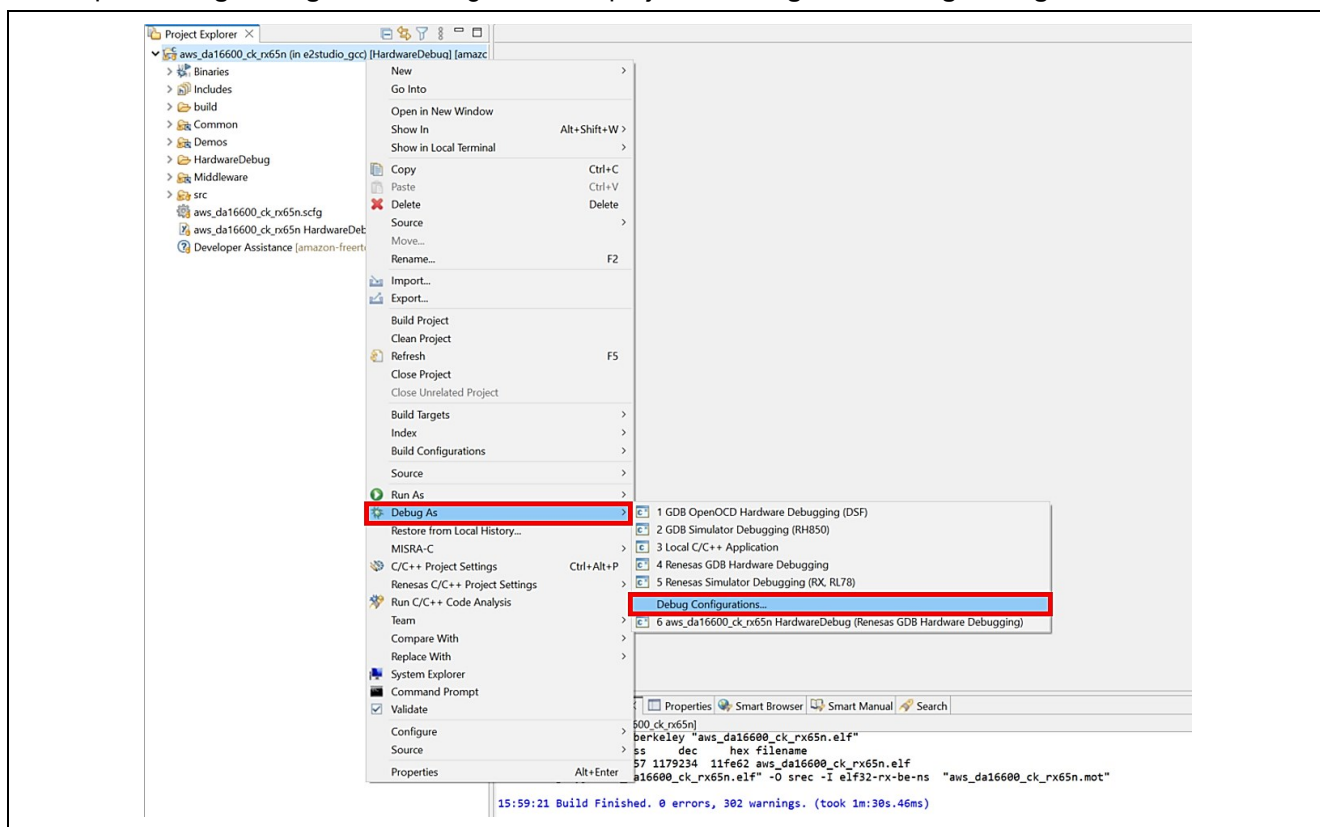


Figure 27. Configuration Debugger (1/2)

Go to **Renesas GDB Hardware Debugging** > **aws_da16600_ck_rx65n HardwareDebug** > **Debugger** tab > **Connection Setting** tab, then configure for “Main Clock Source: **EXTAL**” and “Connection Type: **Fine**”.

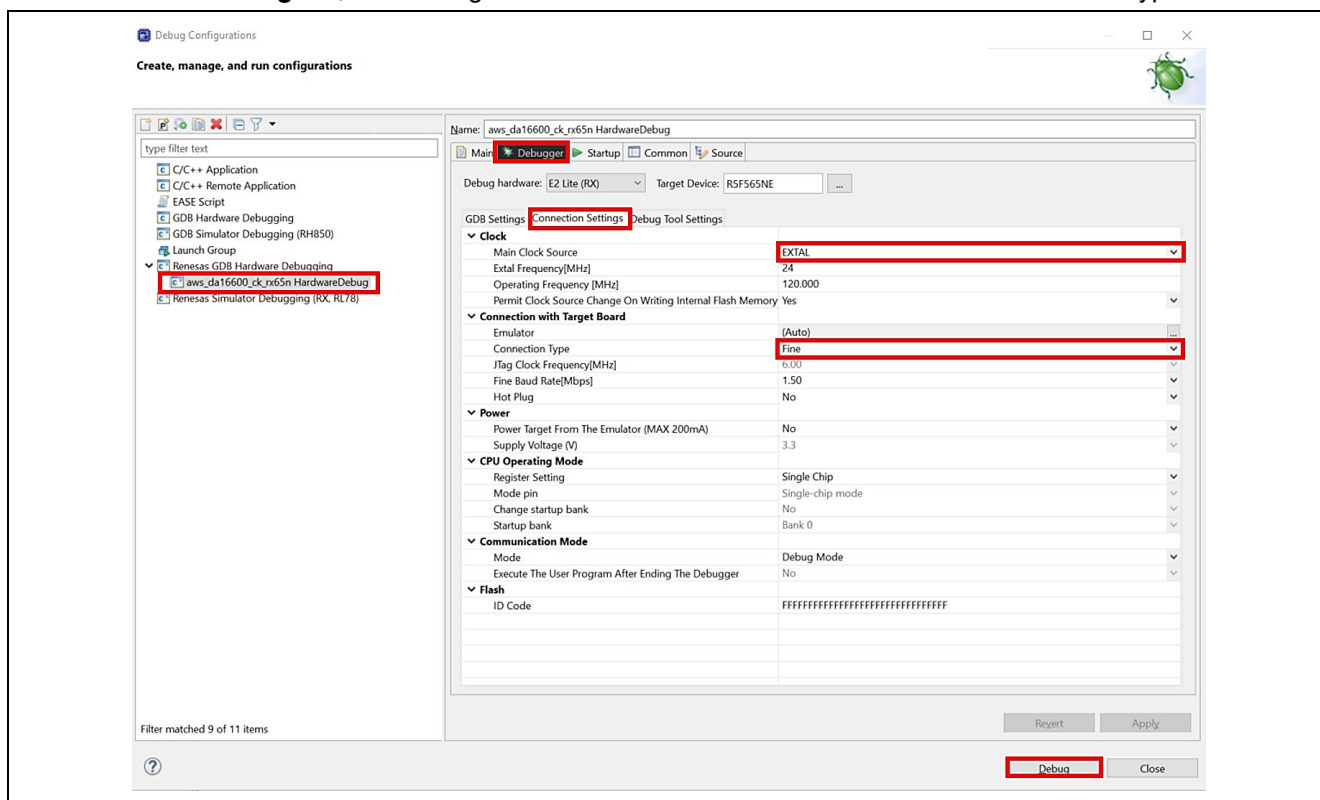


Figure 28. Configuration Debugger (2/2)

5.1.4 Running the Application Project

To run the Application project, use the following instructions.

The serial port console of the PC is already setup in the section 5.1.2, Tera Term, the console will display as below.

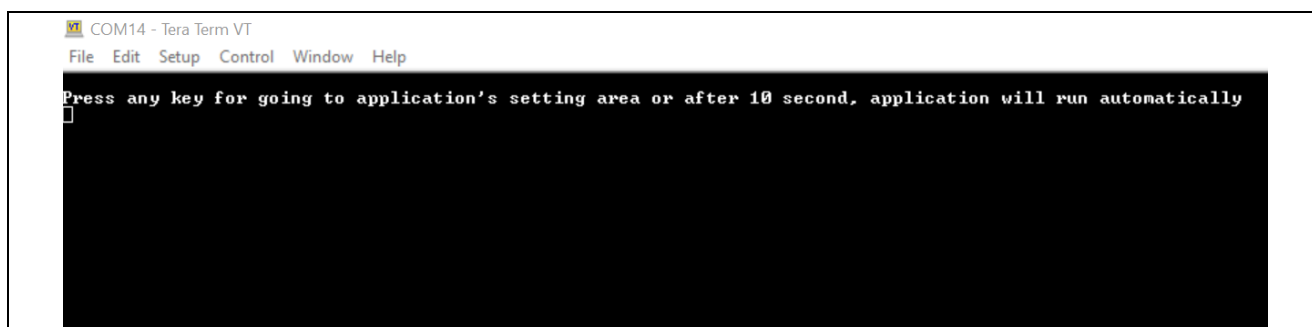


Figure 29. Start the Application

In the first-time users run the application (or users want to change the configuration), please press any key to set necessary credentials for application.

Note: After 10s, the application will run automatically with the option “6. Run Sensor App with MQTT” in the application’s menu. The user can modify this time’s value by changing the value of **WAIT_USER_TIME** macro in the file: **Projects\aws_da16600_ck_rx65n\le2studio_gcc\src\application_code\CommandLine\menu_main.h** before building the project:

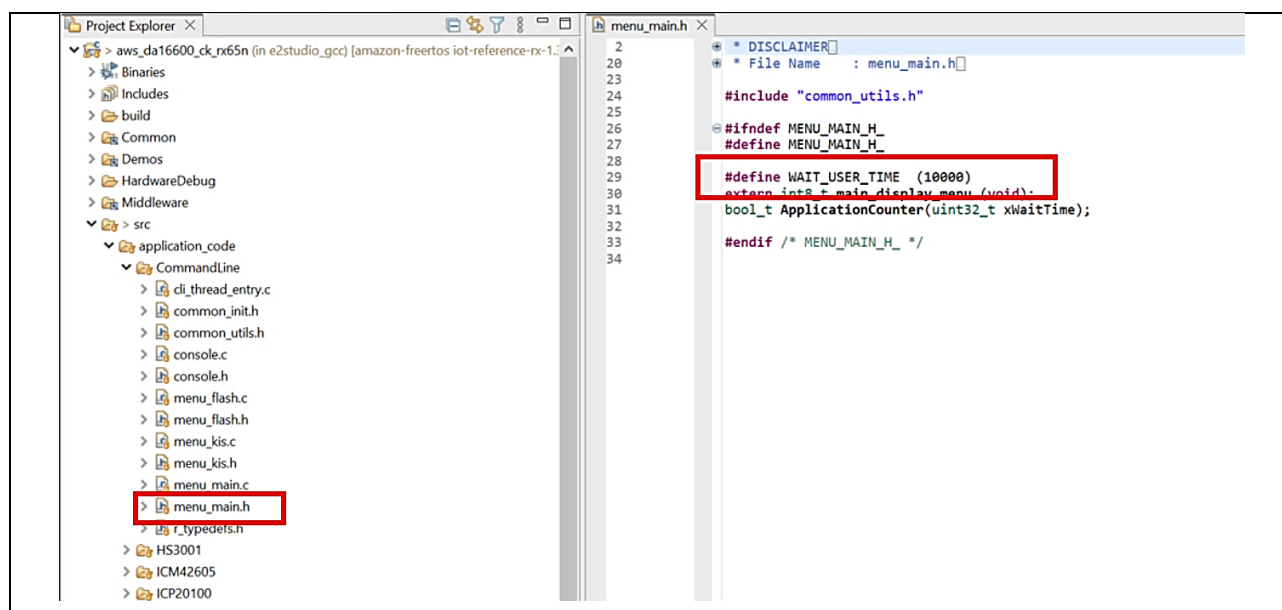


Figure 30. Modify macro to Wait for User Input

After pressing any key, the settings are as shown below.

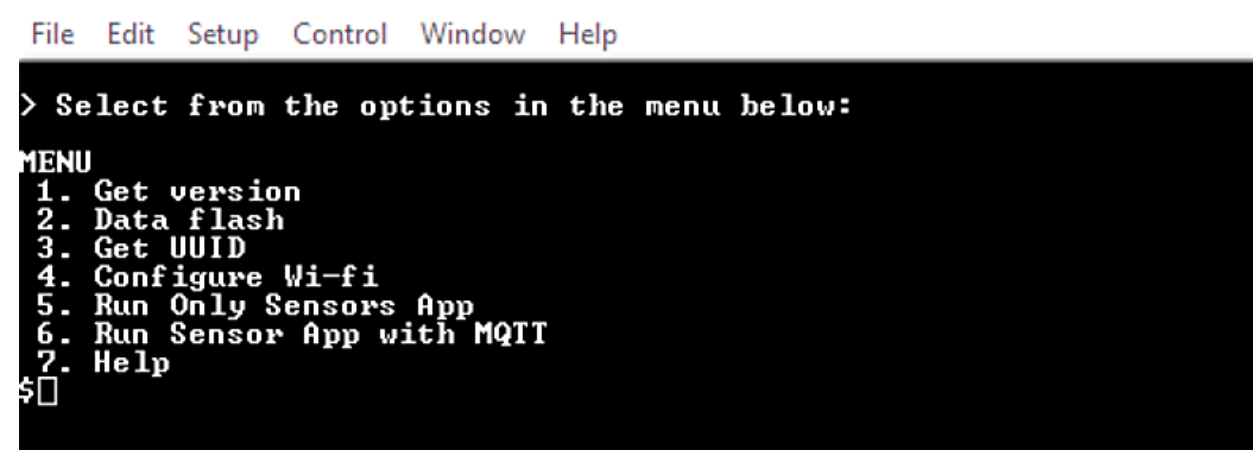
A terminal window with a menu. At the top, there is a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. Below it, a prompt '>' is followed by the text 'Select from the options in the menu below:'. Then, the word 'MENU' is displayed. Below 'MENU' is a list of seven options: '1. Get version', '2. Data flash', '3. Get UUID', '4. Configure Wi-fi', '5. Run Only Sensors App', '6. Run Sensor App with MQTT', and '7. Help'. At the bottom, there is a prompt '\$' followed by a cursor.

Figure 31. Main Menu

Choose the number to select the commands. For example, when you press '1'. The firmware version of the application is displayed as shown below. To return to the main menu, press the "space bar" key.

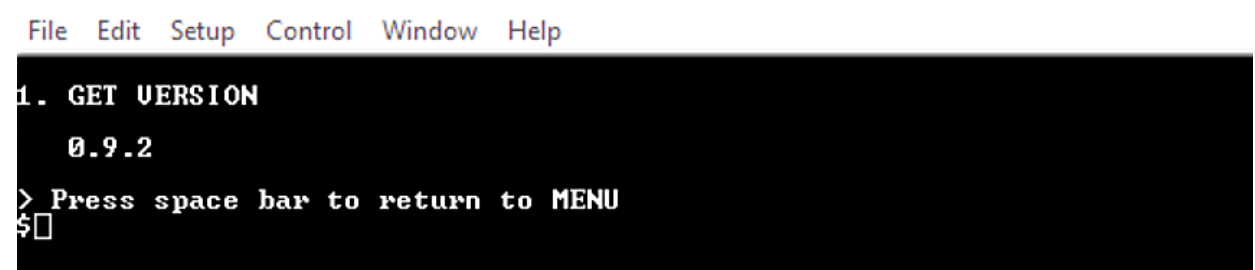
A terminal window showing the result of selecting option 1. The menu bar is the same. Below the prompt '>', it says '1. GET VERSION'. Then, the version '0.9.2' is displayed. Below that, a prompt '>' is followed by the text 'Press space bar to return to MENU'. At the bottom, there is a prompt '\$' followed by a cursor.

Figure 32. Get Version Information

The next section introduces steps to setup for the AWS account.

5.2 For Users Using the Provided Dashboard and AWS Account of Kit

This section explains the registration account and access dashboard. It needs to get the "UUID" of the kit.

Note: If users want to use their own AWS account, please skip this section and refer to 5.4.

5.2.1 Getting the UUID Information of the Board

Press '3' from the **Main Menu** to display **RX MCU UUID**. This command obtains the UUID information of the kit and displays it on the console like the snapshot shown below. You will need this information to register on the Cloud Dashboard.

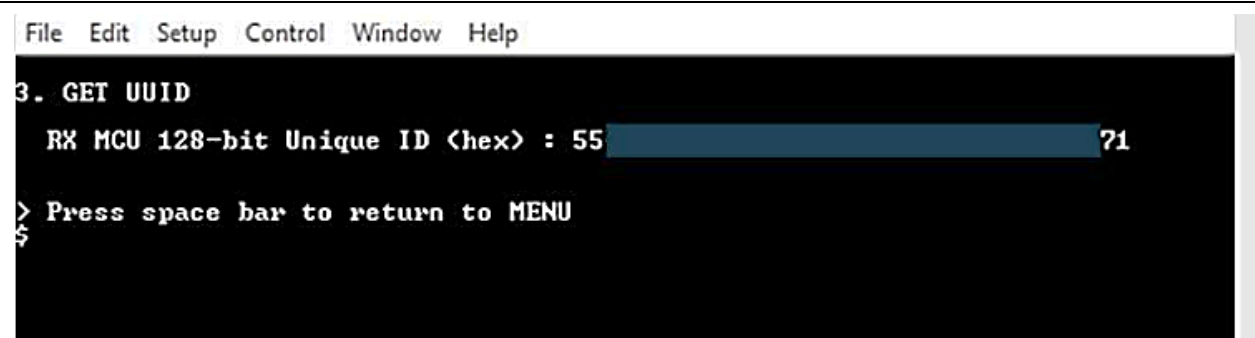
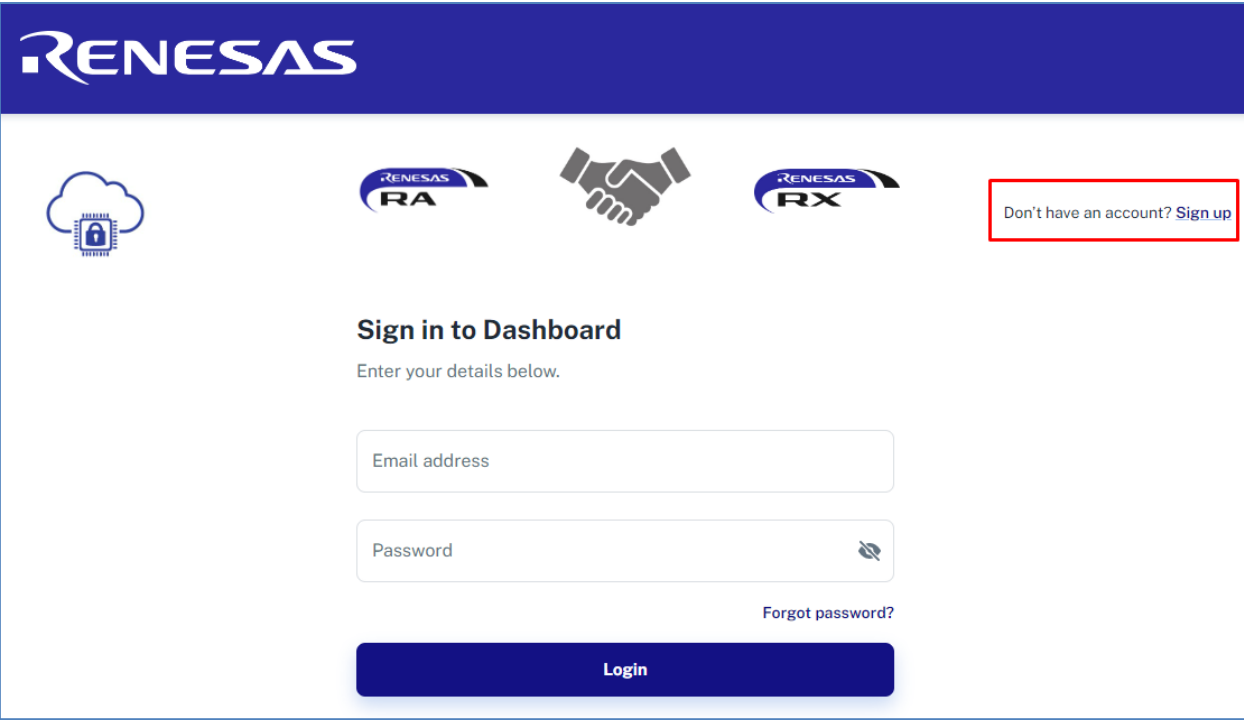
A terminal window showing the result of selecting option 3. The menu bar is the same. Below the prompt '>', it says '3. GET UUID'. Then, the text 'RX MCU 128-bit Unique ID <hex> : 55' is displayed, followed by a long blue bar, and then '71'. Below that, a prompt '>' is followed by the text 'Press space bar to return to MENU'. At the bottom, there is a prompt '\$' followed by a cursor.

Figure 33. Getting Board UUID Information

5.2.2 To Get the Account 10 USD of Trial of AWS

1. Register/sign up at "<https://renesas.cloud-ra-rx.com/>" with **an email account that was not used previously for signing up to an AWS account.**

Note: The provided free credit starts consuming when users register their email and UUID on this system. Renesas recommended disabling the AWS EC2 service when users don't use this system. Please refer to the How to Enable/Disable EC2



RENEASAS

Don't have an account? [Sign up](#)

Sign in to Dashboard
Enter your details below.

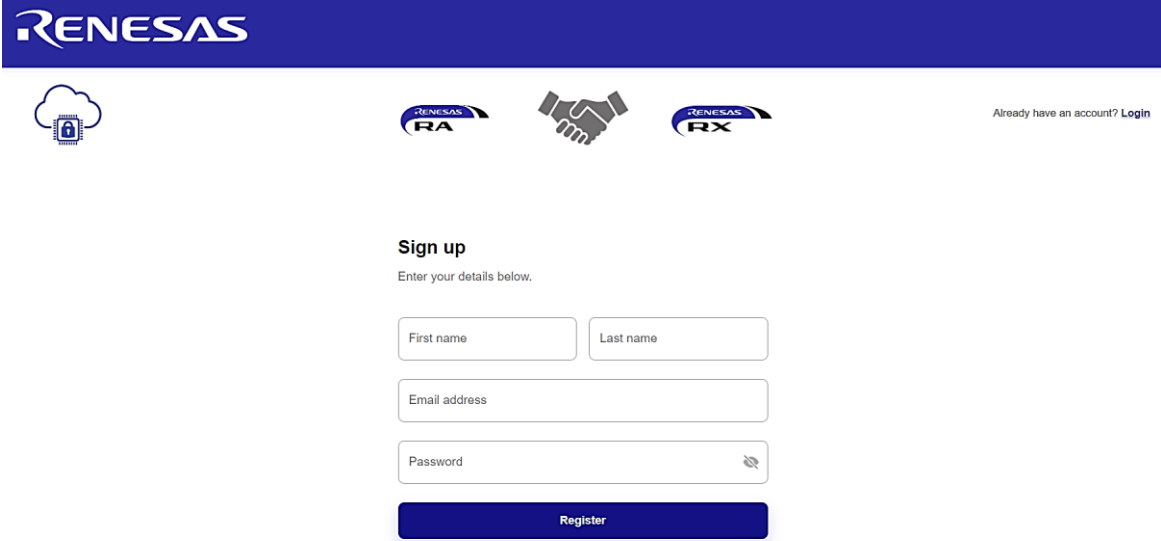
Email address

Password

[Forgot password?](#)

Login

Figure 34. Creating Account



RENEASAS

Already have an account? [Login](#)

Sign up
Enter your details below.

First name

Last name

Email address

Password

Register

Figure 35. Registering Information

The rules for a valid first name and last name:

- Length Constraints: Minimum length of 2. Maximum length of 24.
- Information must be entered in English or another Latin character-based language.

The rules for a valid email address:

- The address must be a minimum of 6 and a maximum of 64 characters long.
- All characters must be 7-bit ASCII characters.
- There must be one and only one @ symbol, which separates the local name from the domain name.
- The local name cannot contain any of the following characters: whitespace, " ' () < > [] : ; , \ | % &
- The local name cannot begin with a dot (.)
- The local name cannot contain double Plus, for example [account+rnss+alpha@domain.com](#)
- The domain name can consist of only the characters [a-z],[A-Z],[0-9], hyphen (-), or dot (.)
- The domain name cannot begin or end with a hyphen (-) or dot (.)
- The domain name must contain at least one dot.

The rules for a valid password:

- The password must be a minimum of 8 and a maximum of 64 characters long.

Password must contain at least one uppercase character, one lowercase character, one number, and one special character: ! # \$ % & * ? @.

2. Verification code will be sent to your email (~10 min). Enter the code and press the Send button. You are redirected to the Register Device page.

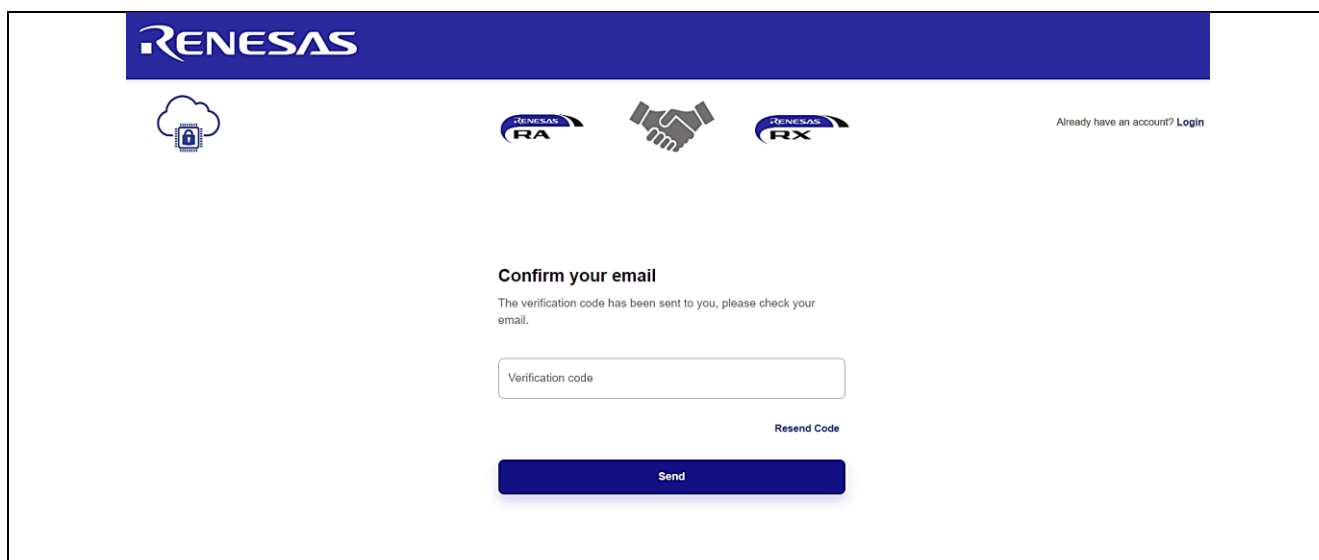


Figure 36 Confirming Email

If you do not receive an email with the code, please click on **Resend Code**.

3. Then put on the email and UUID to register the kit in the window below. You can get the UUID from section **5.2.1 Getting the UUID Information of the Board**.

Note: Only 1 device will be assigned to an account.

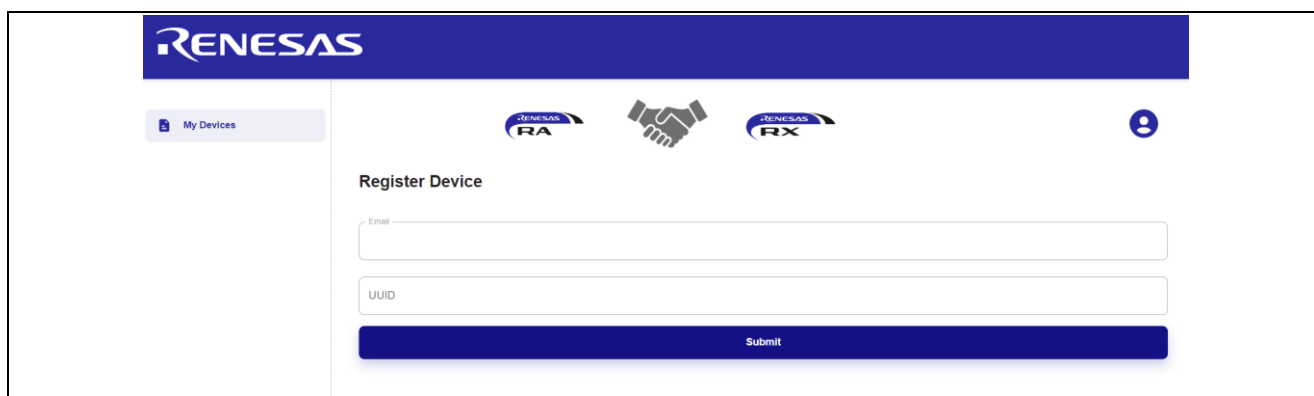


Figure 37. Register Device

- Wait for the status change on the registration page/ wait for provisioning to complete. Please refresh the page in case the Registration in progress screen still shows up.

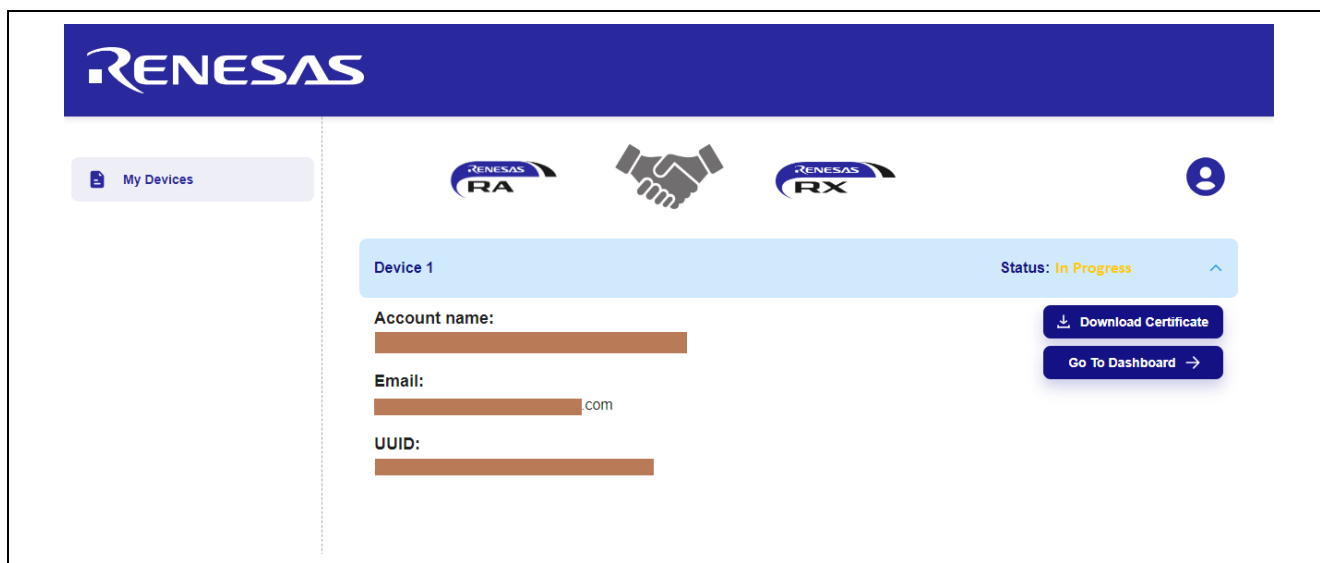


Figure 38. Device Registration In Progress

- Refresh the page and the status of the registration changes to 'Online'.

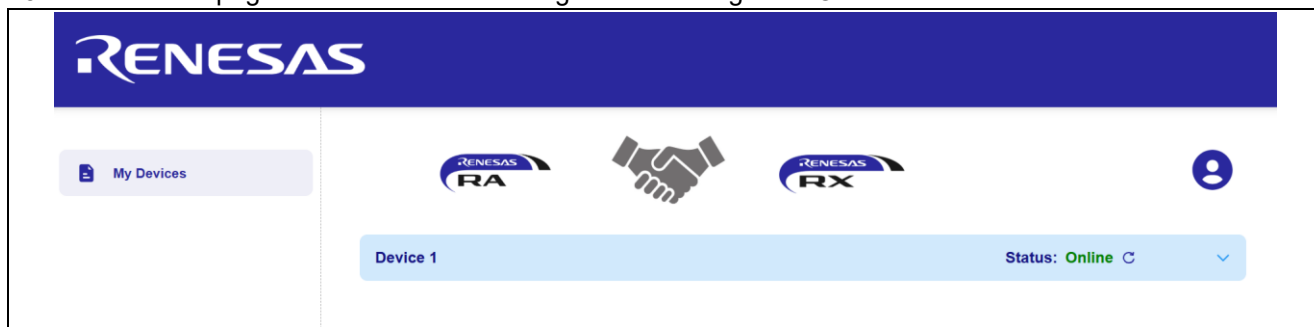


Figure 39. Active Device

- After finishing the progress, you can get the file of certification to connect the dashboard from the "Download Certificate" button. It is used for installation on the application demo of kits at 5.3.

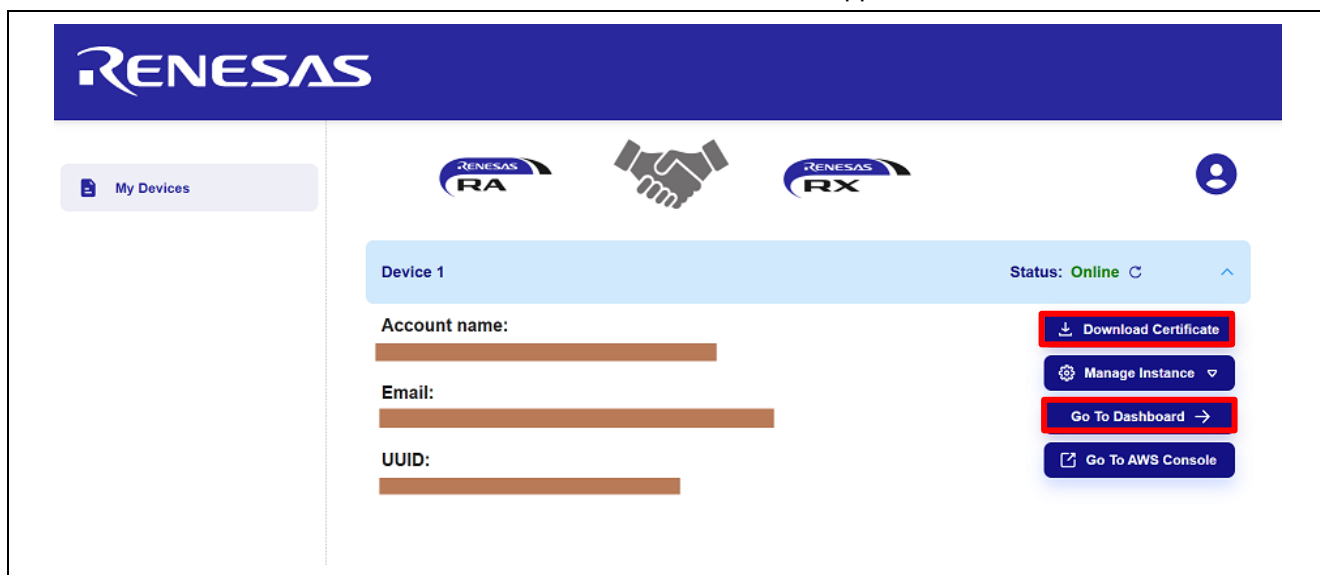


Figure 40. Completing Device Provisioning

- Click **"Go to Dashboard"** to access the dashboard. First time users will access the dashboard with default 'username' is **admin** and default 'password' is **admin1234**. Once completed, users can access the dashboard.

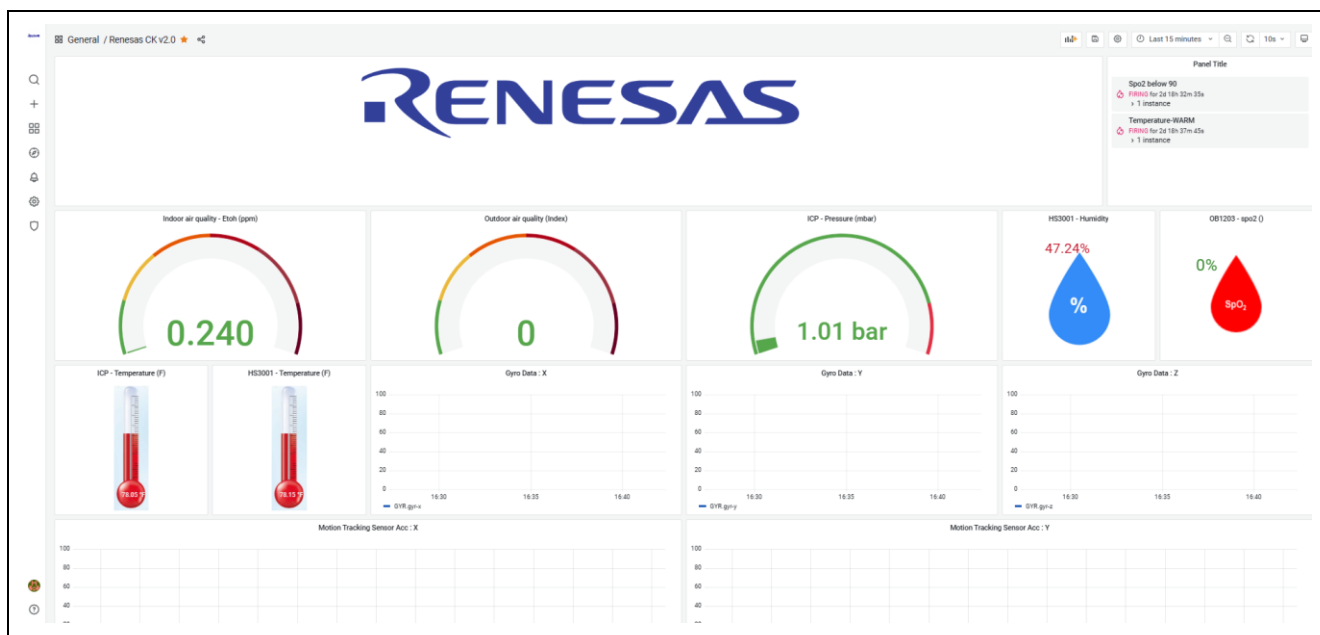


Figure 41. Dashboard for this Application

5.3 Software Preparation-Run Project from IDE

5.3.1 Storing the Device Certificate, Key, MQTT Broker Endpoint, and IoT Thing Name

Device Certificate, Device Private Key, MQTT Broker Endpoint, and IOT Thing name need to be stored in the data flash for the application to work. These are obtained after registering to the Cloud Dashboard.

- Press **'2'** on the **Main Menu** to display **Data Flash-related** commands as shown in the following snapshots. This sub-menu has commands to store, read, and validate the data.

```

COM14 - Tera Term VT
File Edit Setup Control Window Help

> Select from the options in the menu below:
2. DATA FLASH

a) Info
b) Write Certificate
c) Write Private Key
d) Write MQTT Broker end point
e) Write IOT Thing name
f) Write code signing certificate (for OTA)
g) Write template name (for Fleet)
h) Write claim cert ID (for Fleet)
i) Write claim private key ID (for Fleet)
j) Read Flash
k) Check credentials stored in flash memory
l) Format Flash data
m) Help

> Press space bar to return to MENU
$ 

```

Figure 42. Data Flash Related Menu and Commands

- Please unzip the `cert.zip` from the dashboard (downloaded at Figure 40. **Completing Device Provisioning**)

3. To store the **Device Certificate**, press option 'b' Click the **File** tab of the Tera Term and **Send File** option and choose the downloaded Device certificate file from the dashboard "xxxxxcertificate.pem.crt". The details of downloading the certificates are explained in the Dashboard document linked as part of this Application Note.

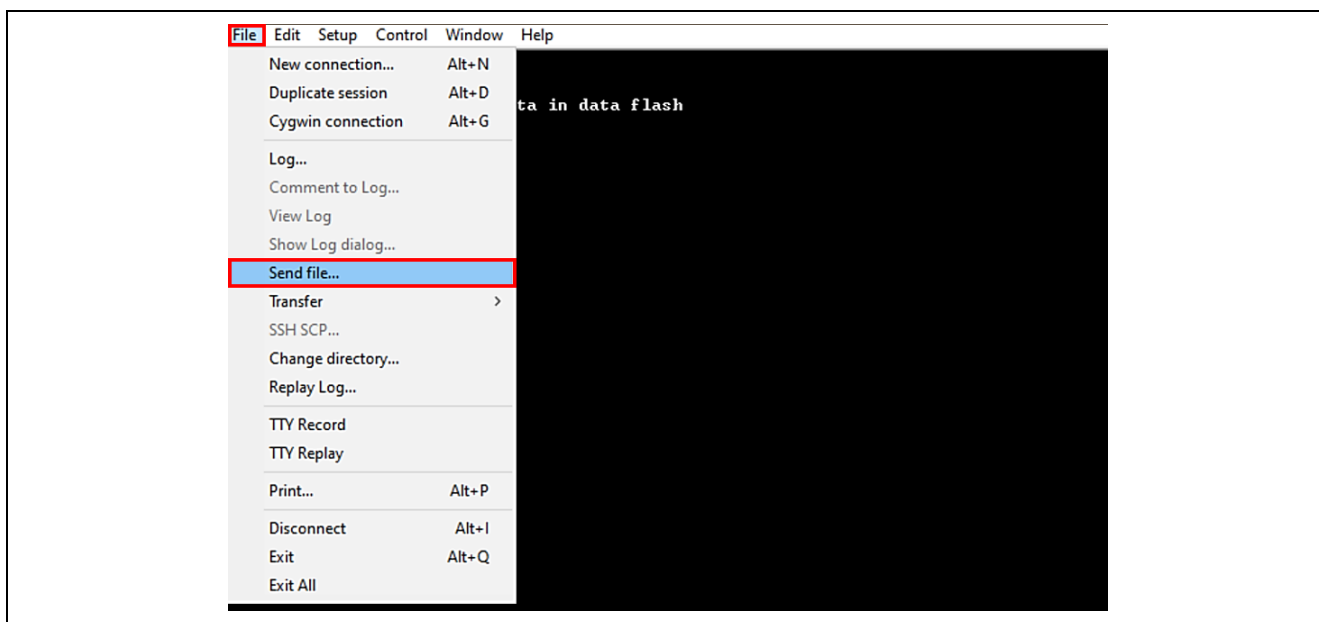


Figure 43. Accessing the Device Certificate

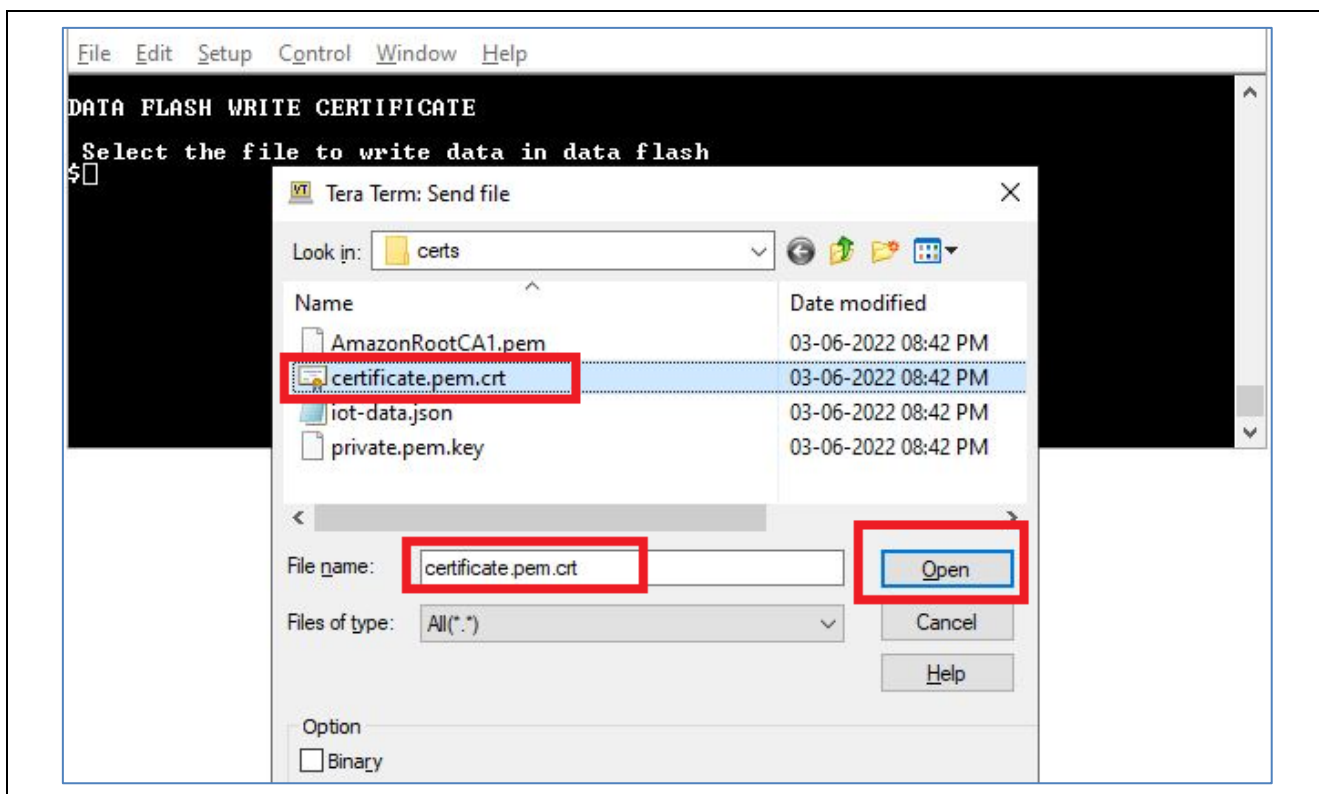
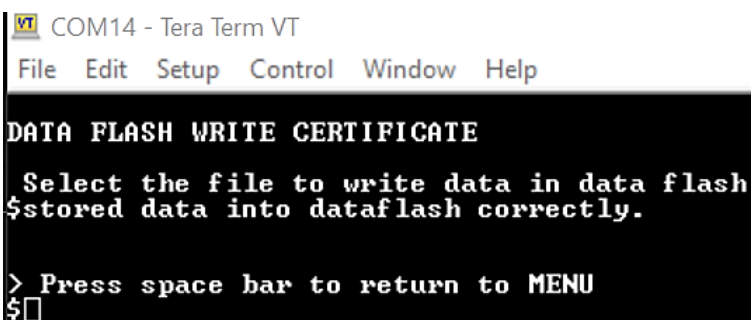


Figure 44. Downloading the Device Certificate into the Data Flash



```

COM14 - Tera Term VT
File Edit Setup Control Window Help

DATA FLASH WRITE CERTIFICATE

Select the file to write data in data flash
$stored data into dataflash correctly.

> Press space bar to return to MENU
$

```

Figure 45. Status of the Downloaded Device Certificate into the Data Flash

4. To store the **Device Private Key** press the option 'c' and click the **File** tab of the Tera Term and **Send File** option. Choose the downloaded Device Private Key "xxxxxxxprivate.pem.key" which is downloaded from the Dashboard download link.
5. **Open the "iot-data.json" file**
This file has information about IoT things name and IoT Endpoint.

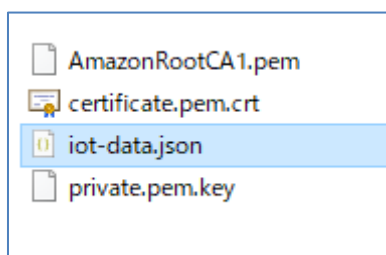
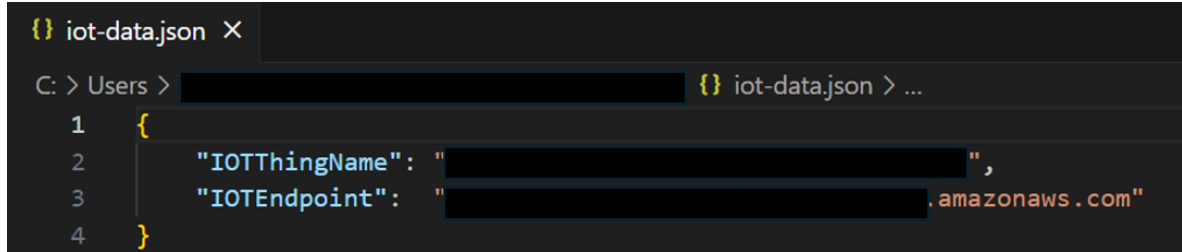


Figure 46. Getting the IoT Things Name and IoT Endpoint Information

6. To store the **MQTT Broker end point**, copy the end point string between the quotes **xxxxxxxxxx.iot.us-east-1.amazonaws.com** from the downloaded certificate link, press the option 'd' and click the **Edit** tab of the Tera Term and **Paste<CR>** and verify and confirm the valid string and press **OK**.

Note: Please copy the IOTEndpoint without ".".



```

{} iot-data.json X
C: > Users > [redacted] {} iot-data.json > ...
1 {
2   "IOTThingName": "[redacted]",
3   "IOTEndpoint": "[redacted].amazonaws.com"
4 }

```

Figure 47. Copy IoT Endpoint Information

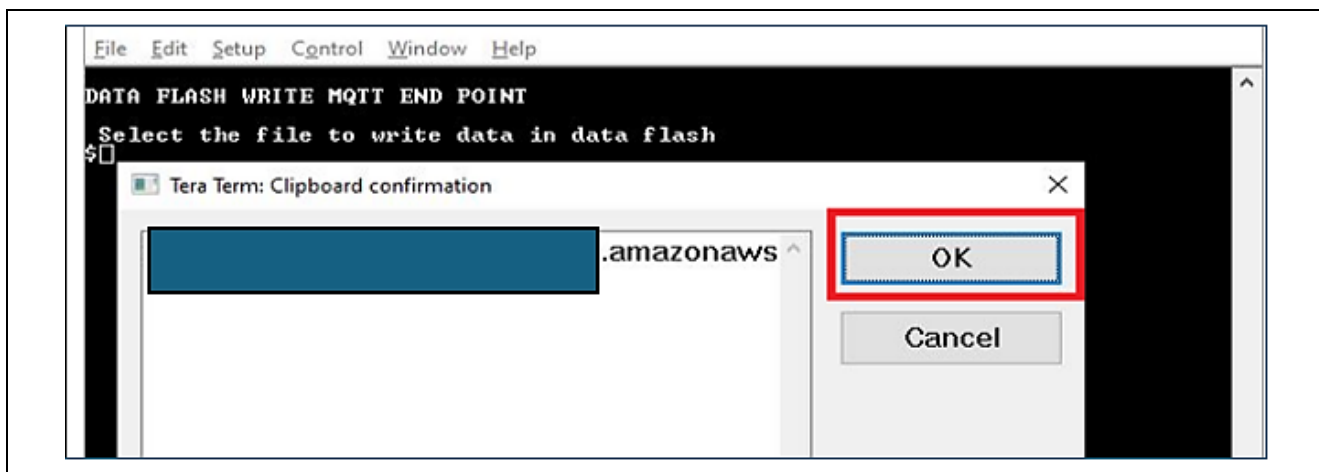


Figure 48. Storing the MQTT IoT Endpoint into the Data Flash

7. To store the IOT Thing Name, copy the Thing Name string between the quotes xxxxxxxx-xxxx-xxxxxxx-xxxx of IoT thing Name from the downloaded certificate link, press the option 'e' and click the **Edit** tab of the Tera Term and "**Paste<CR>**" and verify and confirm the valid string and press OK.

Note: Please copy the **IOTThingName** without **"**.

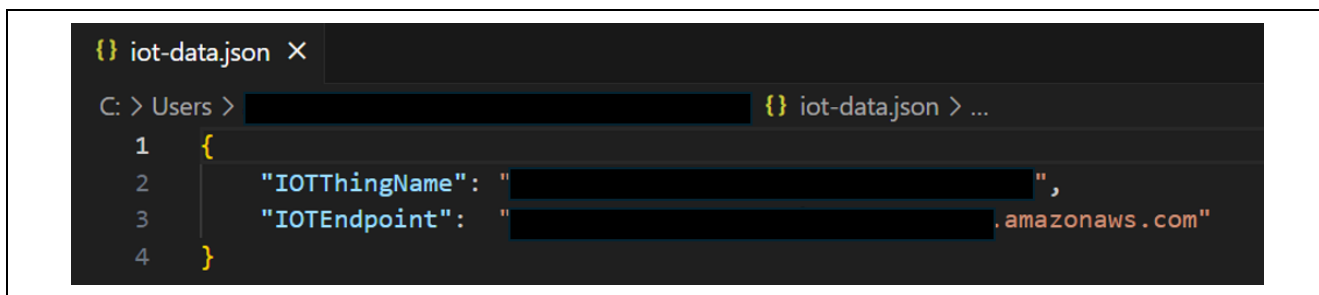


Figure 49. Copy the IOTThingName

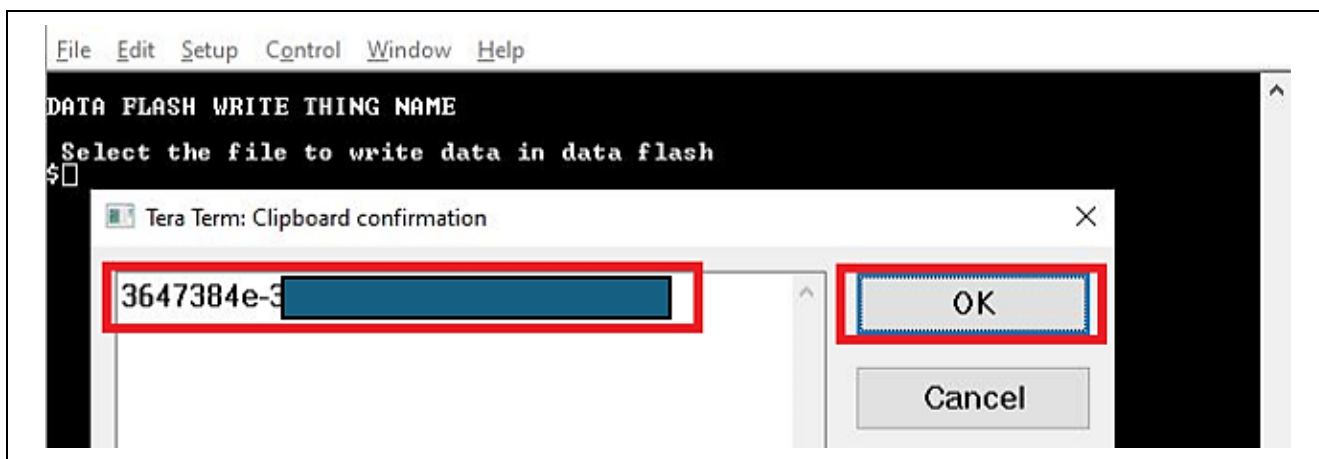


Figure 50. Storing the Thing Name into the Data Flash

8. Press option 'j' and 'k' to read and validate the stored information in the data flash.

Note: Validation of the stored data is very limited and validates a minimum set of data points. Users are required to input the valid data to the flash obtained from the Dashboard for the proper working of the Application.

Note: Option '**I) Format Flash data**' will erase all saved values in data flash. Please be careful when using this option in the application.

5.3.2 Storing the SSID Name, Password, and Security Option

SSID Name, Password, and Security option need to be stored in the data flash for the Wi-Fi connection of the application.

1. Press '4' on the Main Menu to display Configure Wi-Fi-related commands as shown in the following snapshot. This sub menu has commands to store and read the data.

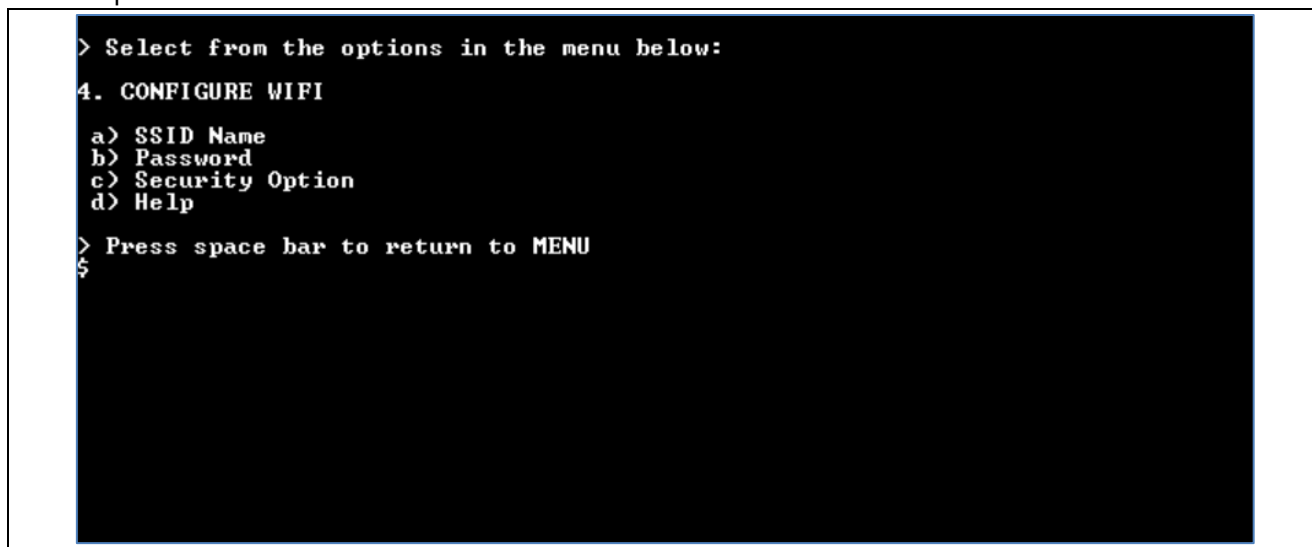


Figure 51. Configure Wi-Fi related Menu and Commands

2. To store the SSID Name, in the Wi-Fi menu, press the option 'a'.

Note: The maximum length of SSID is 32 characters and the minimum length is 2 characters.

- If users have copied the SSID before, please click the “Edit” tab of the Tera Term and “Paste<CR>” and verify and confirm the valid string and press OK as shown below.

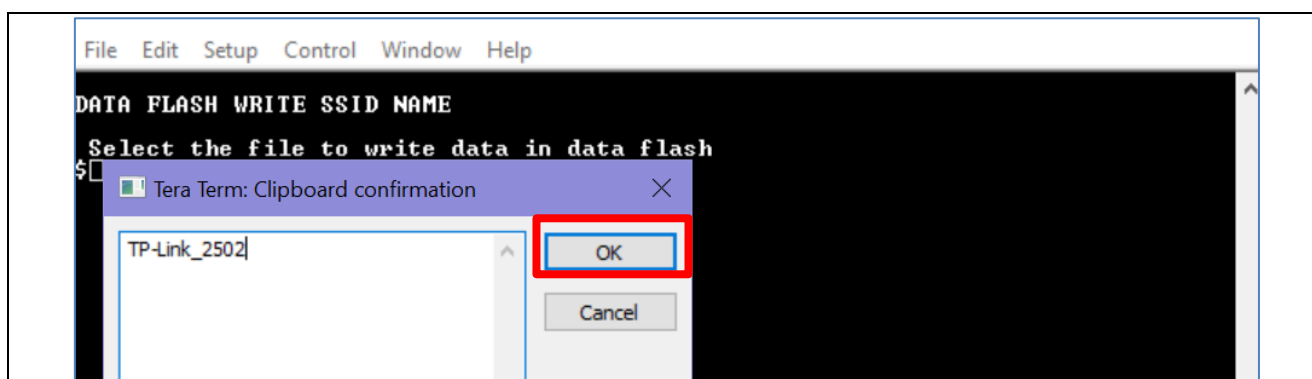


Figure 52. Storing the SSID Name into the Data Flash (1/3)

- If the user did not copy the SSID before this step, they can input directly to the Tera Term. To check the SSID, click the **Setup > Terminal > Local echo** of the Tera Term as below.

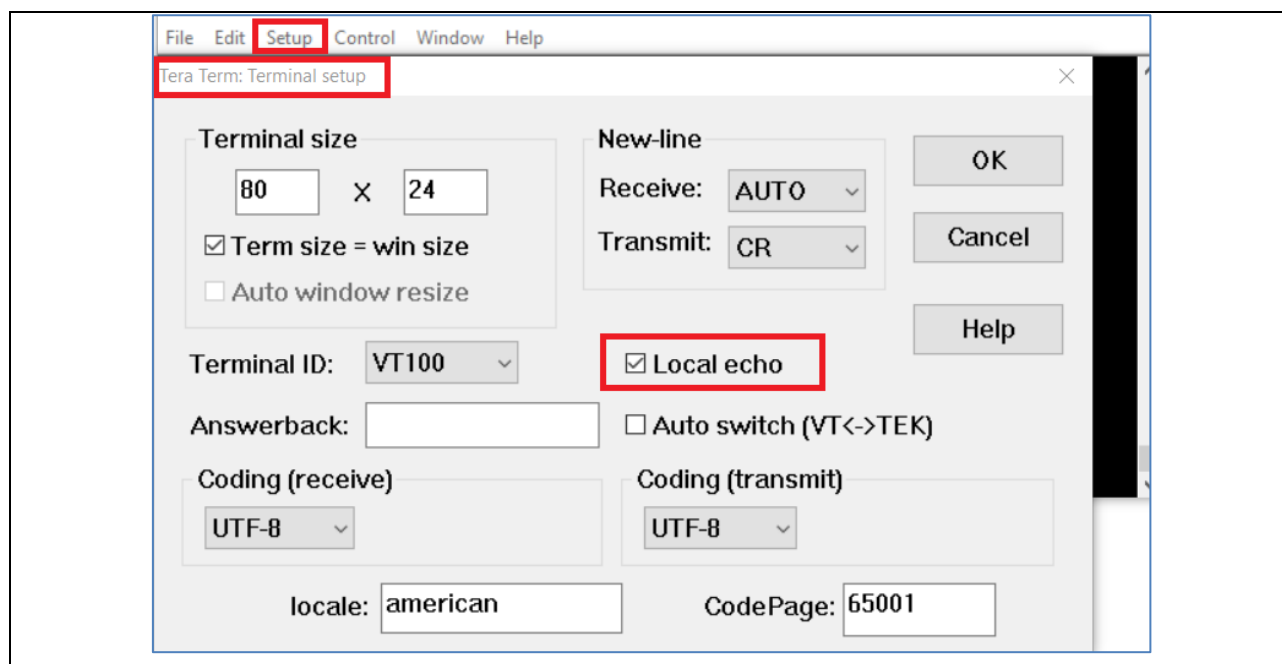


Figure 53. Storing the SSID Name into the Data Flash (2/3)

— Input SSID and press **Enter**.

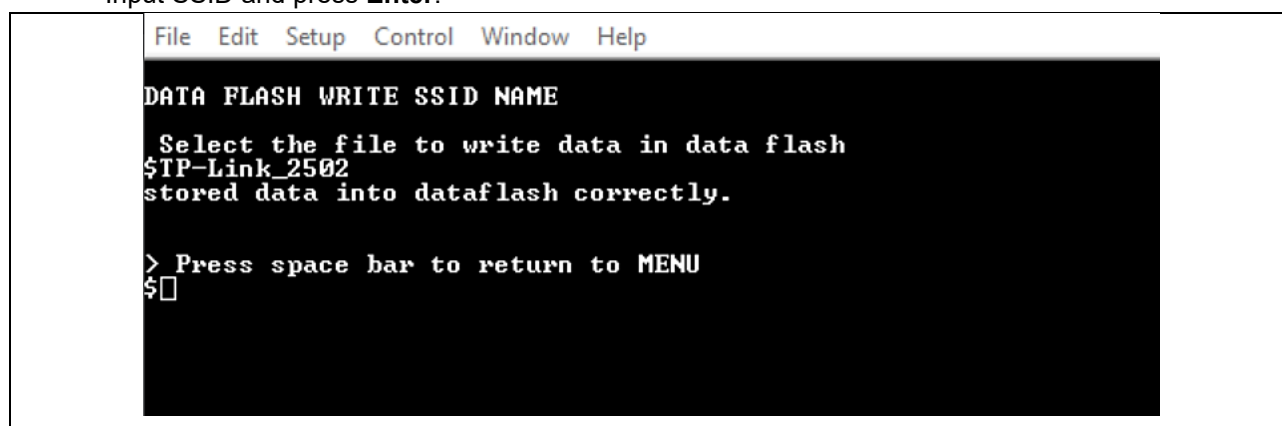


Figure 54. Storing the SSID Name into the Data Flash (3/3)

- To store the password, in the Wi-Fi menu, press the option 'b', input the password, confirm the valid string then press Enter. The maximum length of the password is 32 characters, and the minimum length is 1 character.

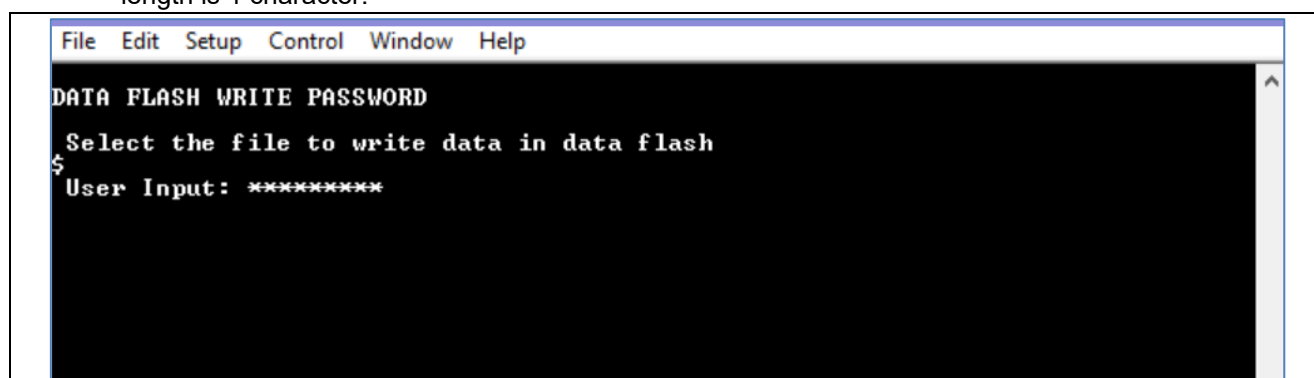


Figure 55. Storing the Wi-Fi Password into the Data Flash

4. To store the security type, in the Wi-Fi menu, press the option 'c', then press the option 'a' for **Open Wi-Fi**, 'b' for **WPA security**, and 'c' for **WPA2 security** to choose the correct security for Wi-Fi configuration.

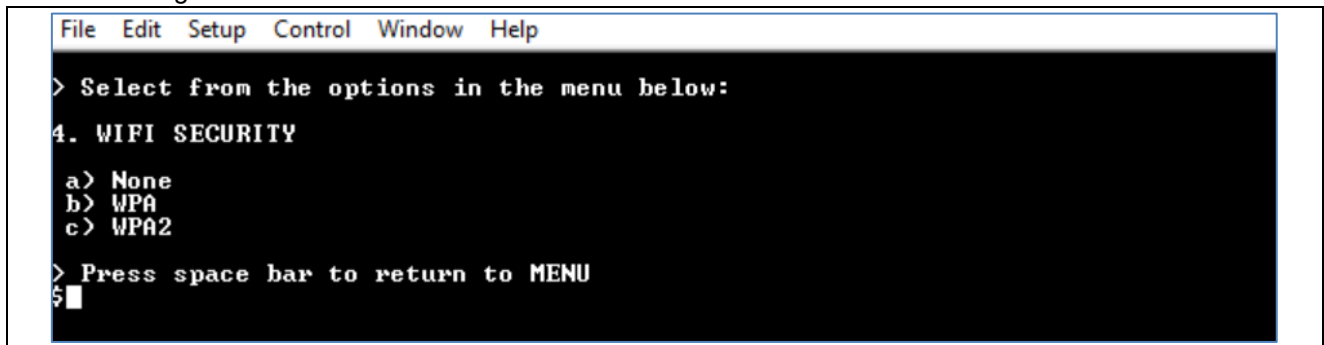


Figure 56. Storing the Security into the Data Flash

5.3.3 Starting the Application

After registering to the Dashboard and configuring the required Cloud credentials through the CLI, the application is ready to run. Press the option '6' to start the application. The application prints the Welcome screen along with the status of validating the Cloud credentials data present in the data flash as shown below. When the connection is successful, the data is shown.

```

COM14 - Tera Term VT
File Edit Setup Control Window Help

CHECK CREDENTIALS STORED IN DATA FLASH
Fleet is disabled, do not need Claim private key ID
Fleet is disabled, do not need Claim cert ID
Fleet is disabled, do not need template name
OTA is disabled, do not need code sign certificate
Wi-Fi's Security saved in data flash is verified and successful
Wi-Fi's Password saved in data flash is verified and successful
Wi-Fi's SSID saved in data flash is verified and successful
IOT thing name saved in data flash is verified and successful
MQTT Endpoint saved in data flash is verified and successful
Private Key saved in data flash is verified and successful
Certificate saved in data flash is verified and successful
All credentials in data flash is verified and successful
0 12087 [CLI] Write certificate...

** Alternate Key Provisioning successfully **
!!! Wi-Fi Init Successful !!!**
SSID:
Connecting to
Wi-Fi connected to SSID
Device IP address: 192.168.122.227
Device network mask: 255.255.255.0
Device gateway address: 192.168.122.243
MQTT End point IP address = 54.197.68.34
1 24400 [CLI] -----STARTING DEMO-----
2 24400 [MQTT] [INFO] -----Start MQTT Agent Task-----
3 24400 [MQTT] [INFO] Creating a TLS connection to .amazonaws.com:8883.
4 24400 [MQTT] [INFO] Created new ICP socket.
5 24755 [MQTT] [INFO] Established ICP connection with .amazonaws.com.
6 28021 [MQTT] [INFO] (Network connection 0x8587b0) TLS handshake successful.
7 28021 [MQTT] [INFO] (Network connection 0x8587b0) Connection to .amazonaws.com established.
8 28021 [MQTT] [INFO] Creating an MQTT connection to the broker.
9 28987 [MQTT] [INFO] MQTT connection established with the broker.
10 28987 [MQTT] [INFO] Successfully connected to MQTT broker.
11 28987 [obi203_thre1] I2C bus 2 setup success
12 28987 [obi203_thre1]
13 28987 [sensor_thre1] I2C bus 0 setup success
14 28987 [sensor_thre1] HS3001 open sensor instance successful: 0
15 28997 [sensor_thre1] ICP20100 open sensor instance successful: 0
17 29007 [zmod_thread] I2C bus 1 setup success

```

Figure 57. Welcome Screen on the Console

```

COM14 - Tera Term VT
File Edit Setup Control Window Help

16 28997 [AWS_DA16600] [INFO] -----Start AWS Wi-Fi DA16600 - MQTT Demo Task -----
18 29442 [zmod_thread] ZMOD4410 open sensor instance successful: 0
19 29873 [zmod_thread] ZMOD4510 open sensor instance successful: 0
20 29874 [zmod_thread] Task zmod4410 measurement Success:0
21 30003 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
22 30009 [sensor_thre] ICM42605 open sensor instance successful: 0
23 31618 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
24 31618 [AWS_DA16600] [Send Data] ZMOD4410-IAQ TVOC: 000.000
25 31618 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ETOH: 000.000
26 31618 [AWS_DA16600] [Send Data] ZMOD4410-IAQ EC02 : 000.000
27 31903 [zmod_thread] ZMOD4410 in stabilization:196609
28 32319 [MQTT] [INFO] Publishing message to aws/topic/iaq_sensor_data.
29 33138 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
30 33138 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
31 33907 [zmod_thread] ZMOD4410 in stabilization:196609
32 35231 [AWS_DA16600] [Send Data] ZMOD4510-OAQ : 000.000
33 35503 [MQTT] [INFO] Publishing message to aws/topic/oaq_sensor_data.
34 35917 [zmod_thread] ZMOD4410 in stabilization:196609
35 36357 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
36 36357 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
37 37922 [zmod_thread] ZMOD4410 in stabilization:196609
38 39127 [AWS_DA16600] [Send Data] HS3001-Humidity : 041.069
39 39127 [AWS_DA16600] [Send Data] HS3001-Temperature: 086.630
40 39510 [MQTT] [INFO] Publishing message to aws/topic/hs3001_sensor_data.
41 39927 [zmod_thread] ZMOD4410 in stabilization:196609
42 40505 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
43 40505 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
44 41989 [zmod_thread] ZMOD4410 in stabilization:196609
45 43528 [AWS_DA16600] [Send Data] ICM42605 - AX = -000.000 g
46 43528 [AWS_DA16600] [Send Data] ICM42605 - AY = -000.002 g
47 43528 [AWS_DA16600] [Send Data] ICM42605 - AZ = 000.999 g
48 43528 [AWS_DA16600] [Send Data] ICM42605 - MX = 000.000
49 43528 [AWS_DA16600] [Send Data] ICM42605 - MY = 000.000
50 43528 [AWS_DA16600] [Send Data] ICM42605 - MZ = 000.000
51 43528 [AWS_DA16600] [Send Data] ICM42605 - GX = 000.109 dps
52 43528 [AWS_DA16600] [Send Data] ICM42605 - GY = -000.038 dps
53 43528 [AWS_DA16600] [Send Data] ICM42605 - GZ = -000.072 dps
54 43655 [MQTT] [INFO] Publishing message to aws/topic/icm_sensor_data.

```

Figure 58. Application with MQTT

Note: Sensor data will be able to read correctly after having stabilization time. You can also check the sensor's operation by choosing option **"5. Run Only Sensors App"**.

Note: With the OB1203 sensor, besides the stabilization time, OB1203 sensor data which is sent to the MQTT (is showed in the terminal) is affected by the "Data Publishing Interval Settings" (refer to **Data Publishing Interval Settings (Optional)** to set this value). So, please keep your finger on the sensor until the terminal displays the correct data. It can be longer than the stabilization time a little bit.

To find out more about the details of stabilization time, please see **Table 9. Sensor Stabilization Time**.

5.4 For User Who Use Their Own AWS Account

Note: Complete the steps up to section 5.4.6 Check AWS IoT endpoints.

5.4.1 Get an AWS Account

[Get an AWS account](#) > Click the **Sign into the Console** button.

5.4.2 Log in to the AWS Management Console

[Amazon Web Services](#) > [My Account](#) > [AWS Management Console](#)

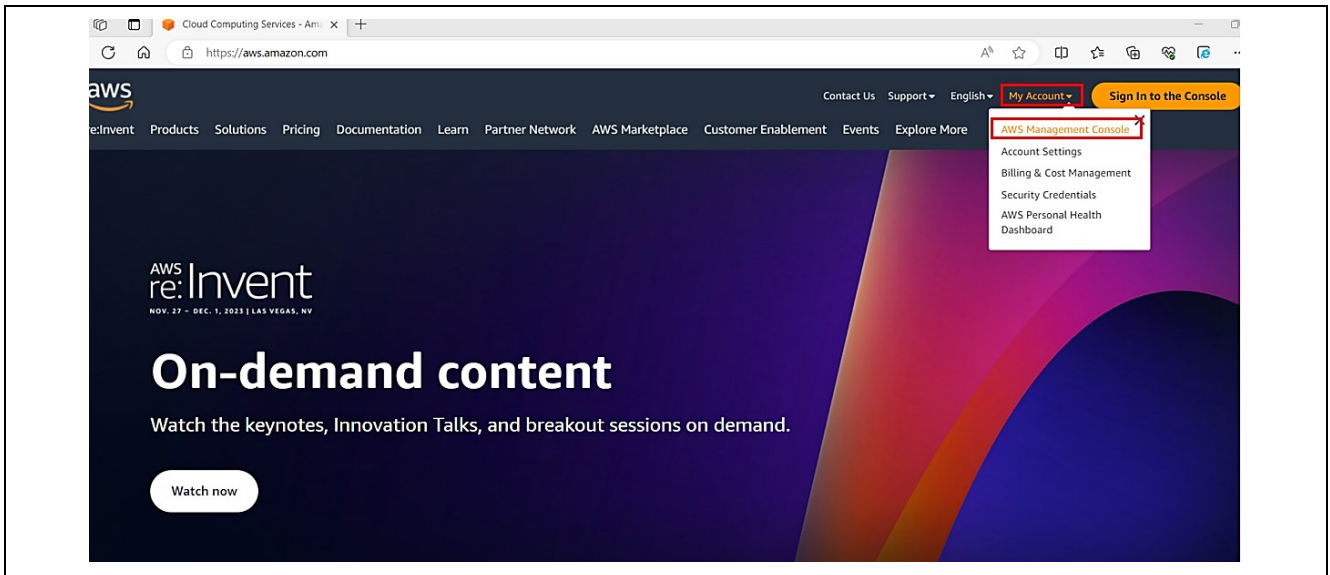


Figure 59. Login the AWS

5.4.3 Move to IoT Core Control Panel

[AWS services](#) > [All services](#) > [IoT Core](#)

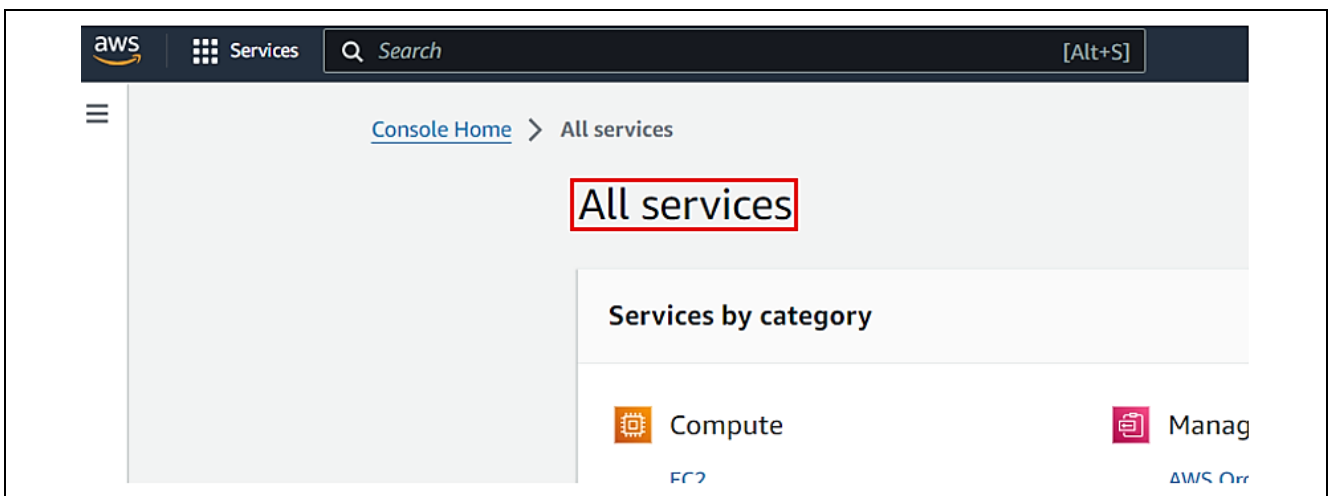


Figure 60. Search the IoT Core (1/2)

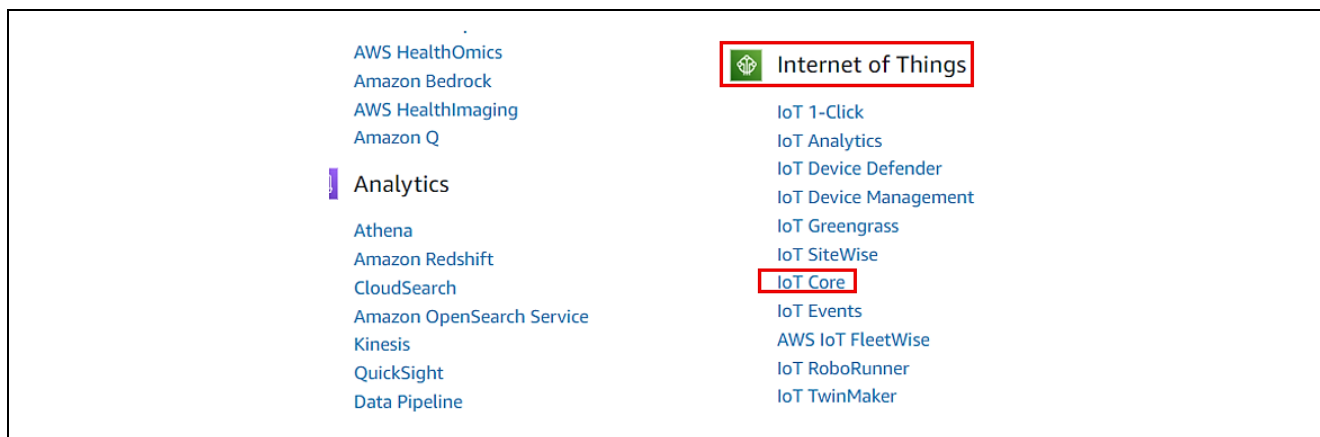


Figure 61. Search the IoT Core (2/2)

5.4.4 Create a Security Policy

Secure > Policies > Create a policy.



Figure 62. Create the Policy (1/3)

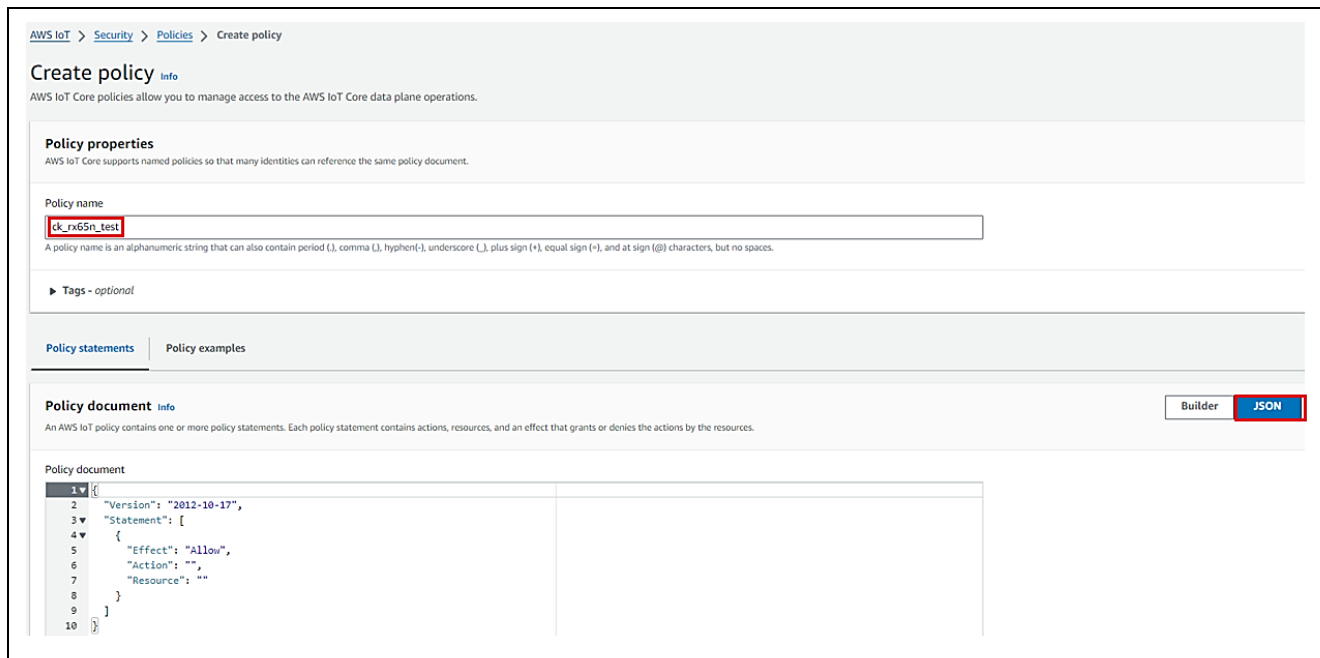


Figure 63. Create the Policy (2/3)

Copy the following code:

```

{
  "Version": "2012-10-17",
  "Statement": [

```

```
{
  "Effect": "Allow",
  "Action": "iot:Connect",
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": "iot:Publish",
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": "iot:Receive",
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": "iot:Subscribe",
  "Resource": "*"
}
]
```

Paste the copied code into the policy syntax > Create.

Policy name
ck_rx65n_test
A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

Tags - optional

Policy statements | Policy examples

Policy document **Info** Builder JSON
An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "*"
    }
  ]
}
```

JSON Line 27, Column 1 Errors: 0 Warnings: 0

Cancel Create

Figure 64. Create the Policy (3/3)

5.4.5 Register your device (thing) with AWS IoT

- **Manage > Things > Create things.**

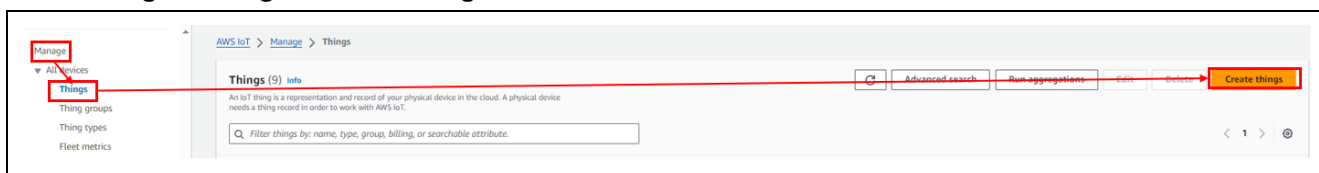


Figure 65. Creating the Things (1/5)

Creating **AWS IoT things > Create a single thing.**

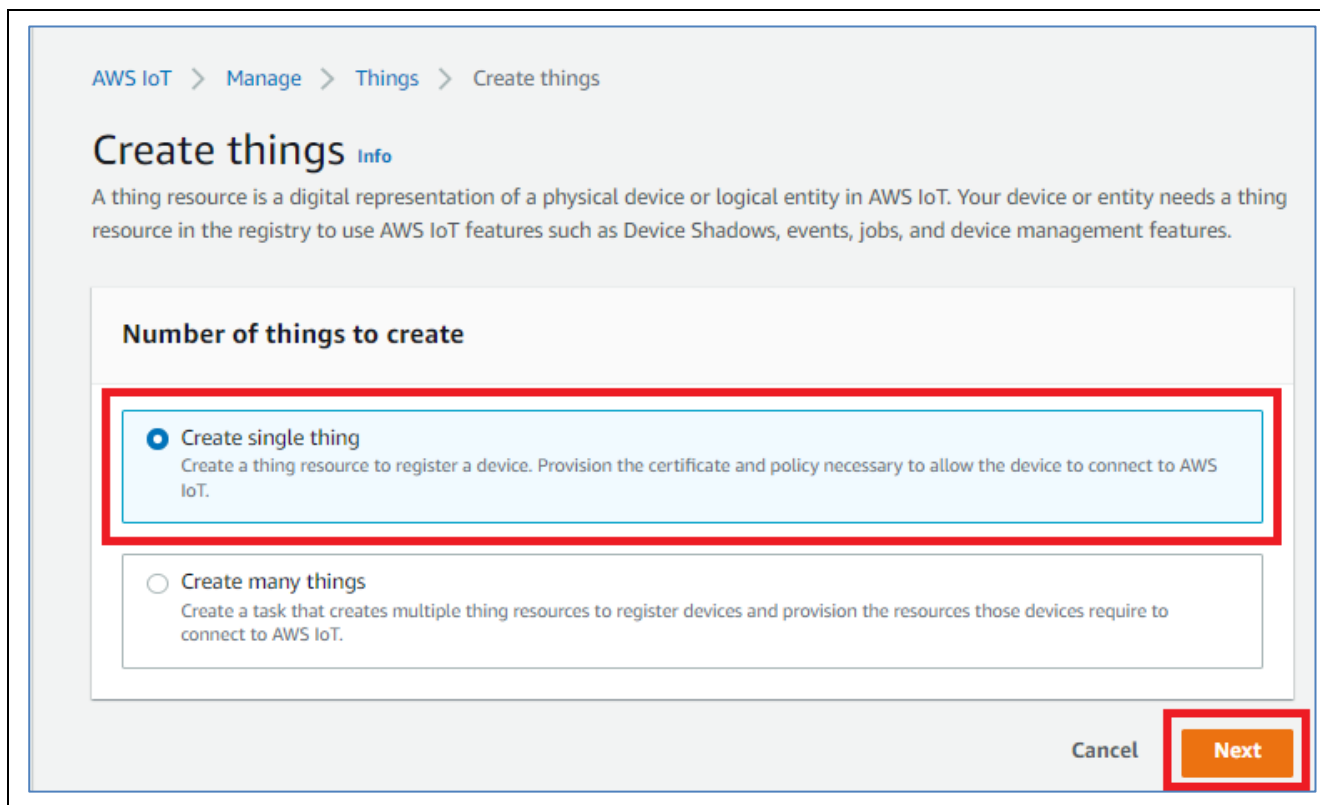


Figure 66. Creating the Things (2/5)

- Add your device to the thing name > **Next**.
- **Make a note of the name with a text editor (will be used later).**

The screenshot shows the 'Specify thing properties' step in the AWS IoT console. The breadcrumb trail is 'AWS IoT > Manage > Things > Create things > Create single thing'. The left sidebar shows 'Step 1 Specify thing properties' as the active step, with 'Step 2 - optional Configure device certificate' and 'Step 3 - optional Attach policies to certificate' listed below. The main content area is titled 'Specify thing properties' with an 'Info' icon. It explains that a thing resource is a digital representation of a physical device or logical entity in AWS IoT. Below this, the 'Thing properties' section has a 'Thing name' field containing 'ck_rx_65n_test', which is highlighted with a red box. Below the field is a note: 'Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.' The 'Additional configurations' section includes expandable options for 'Thing type - optional', 'Searchable thing attributes - optional', 'Thing groups - optional', and 'Billing group - optional'. The 'Device Shadow' section has three radio button options: 'No shadow' (selected), 'Named shadow' (with a sub-note: 'Create multiple shadows with different names to manage access to properties, and logically group your devices properties.'), and 'Unnamed shadow (classic)' (with a sub-note: 'A thing can have only one unnamed shadow.'). At the bottom right, there are 'Cancel' and 'Next' buttons, with the 'Next' button highlighted by a red box.

Figure 67. Creating the Things (3/5)

Auto-generate a new certificate.

The screenshot shows the 'Configure device certificate - optional' step in the AWS IoT console. The breadcrumb trail is 'AWS IoT > Manage > Things > Create things > Create single thing'. The left sidebar shows 'Step 1 Specify thing properties' and 'Step 2 - optional Configure device certificate' as the active step, with 'Step 3 - optional Attach policies to certificate' listed below. The main content area is titled 'Configure device certificate - optional' with an 'Info' icon. It explains that a device requires a certificate to connect to AWS IoT and offers three options for certificate registration. The 'Device certificate' section has three radio button options: 'Auto-generate a new certificate (recommended)' (selected and highlighted with a red box, with a sub-note: 'Generate a certificate, public key, and private key using AWS IoT's certificate authority.'), 'Use my certificate' (with a sub-note: 'Use a certificate signed by your own certificate authority.'), and 'Upload CSR' (with a sub-note: 'Register your CA and use your own certificates on one or many devices.'). Below these is a 'Skip creating a certificate at this time' option with a sub-note: 'You can create a certificate for this thing and attach a policy to the certificate at a later time.' At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons, with the 'Next' button highlighted by a red box.

Figure 68. Creating the Things (4/5)

- Add a policy for your thing.

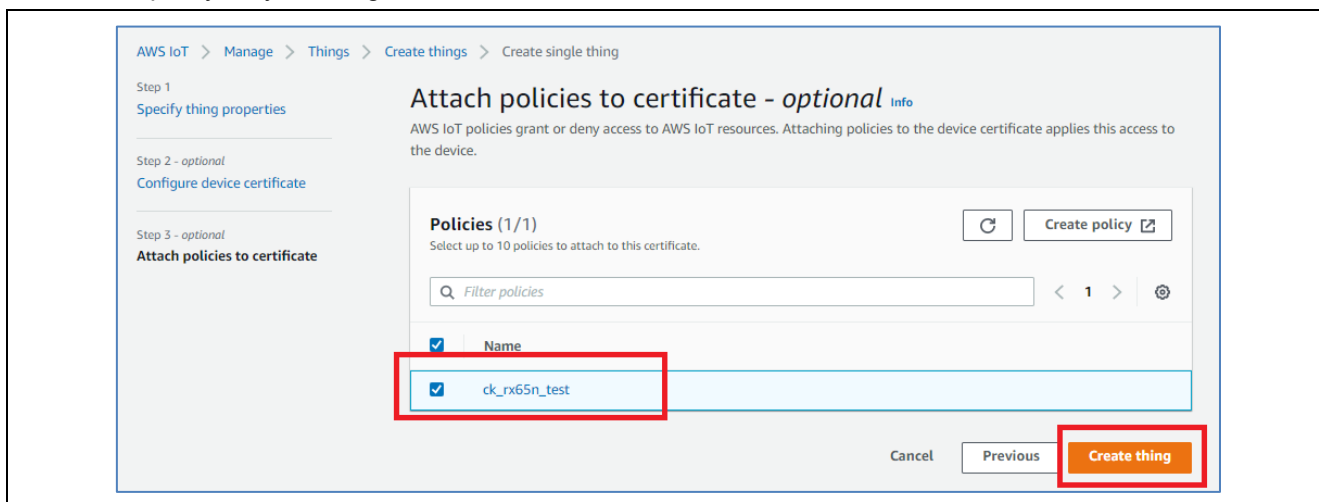


Figure 69. Creating the Things (5/5)

Download a Certificate for this Thing/A Public Key/A Private Key

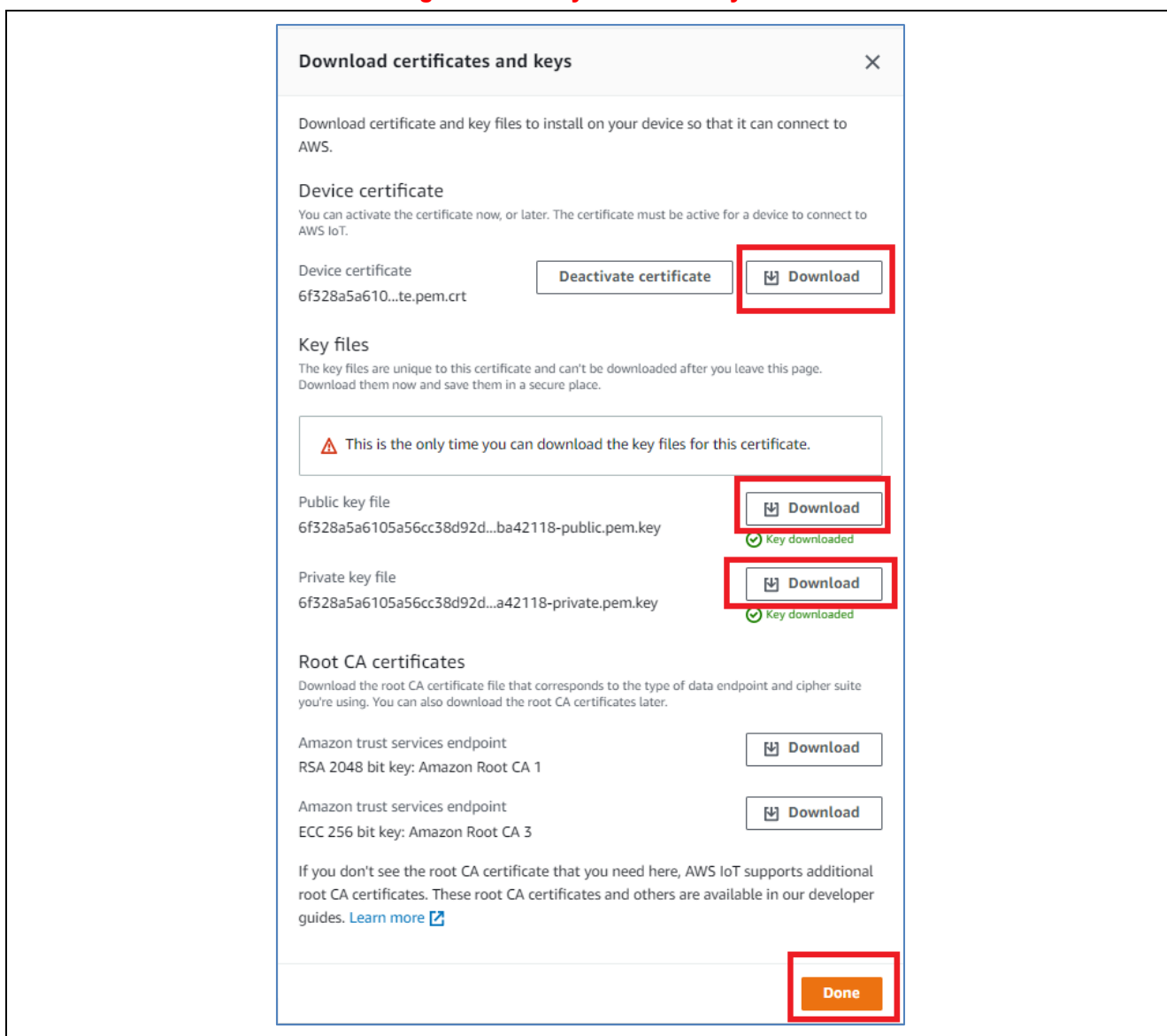


Figure 70. Download A Certificate for this Thing/A Public Key/A Private Key

5.4.6 Check AWS IoT Endpoints

- Make a note of the Endpoint in a text editor and so forth (will be used later)

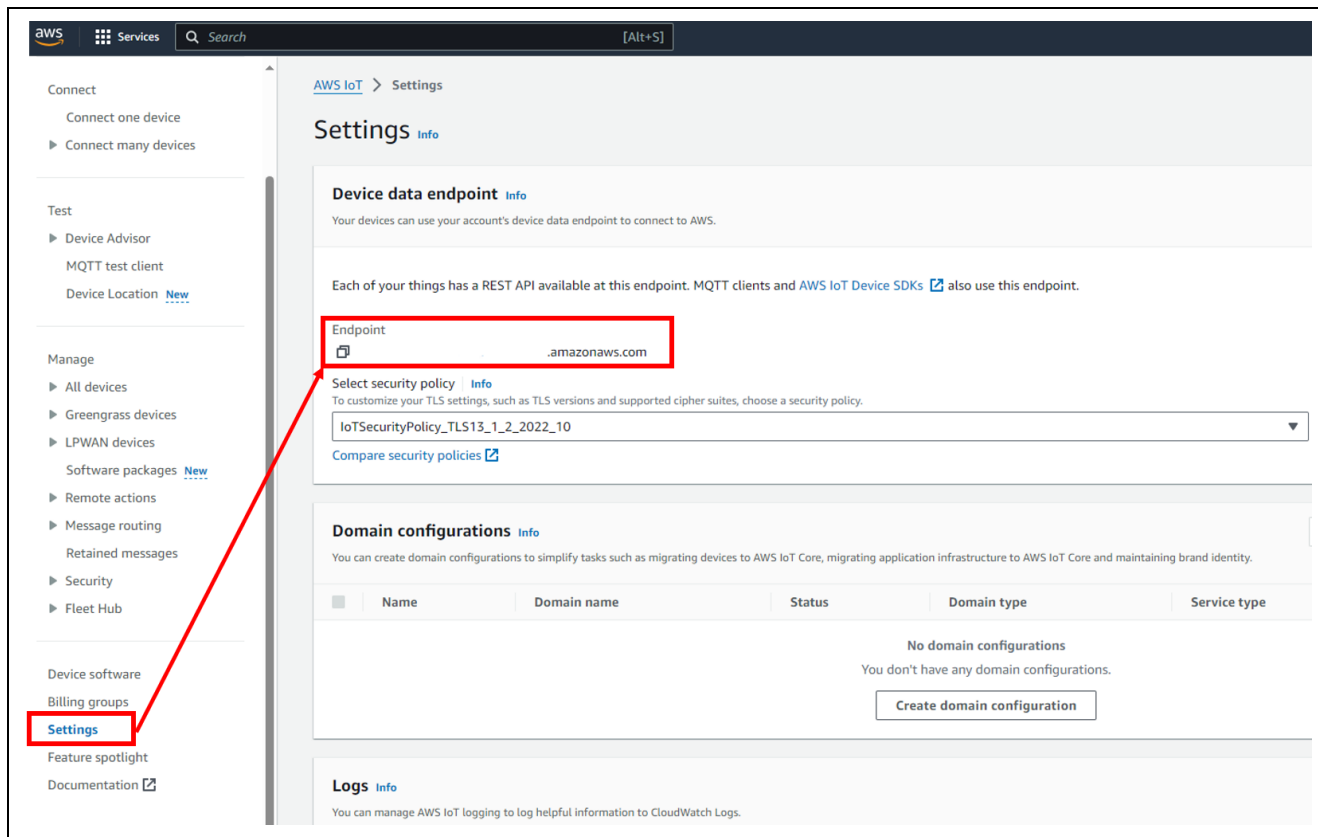


Figure 71. Check AWS IoT Endpoints

Reference: Register the device to AWS IoT tutorial also available GitHub using <https://github.com/renesas/amazon-freertos/wiki/Register-device-to-AWS-IoT>

Install the credential in the application as instructed at section 5.3. **Software Preparation-Run Project from IDE.**

5.4.7 Running Application

After collecting Cloud Credentials, it includes:

- Device Certificate, Key, IoT Thing name: after registering the device (see section 5.4.5)
- MQTT Broker endpoint: (see section 5.4.6)

Please refer to the section 5.3. **Software Preparation-Run Project from IDE** for storing Cloud credentials and running applications.

Note: Instead of getting cloud credentials from the `cert.zip` file, users collected them directly from the AWS cloud.

5.5 Verifying the Application Project using AWS Dashboard and Renesas Dashboard

5.5.1 Subscribe to a Topic Message on the AWS IoT

This section describes the steps for verifying this application example's functions.

Note: Wait for the board to get the IP address from the service provider upon successful Wi-Fi initialization, and for the board to resolve the DNS lookup for the endpoint. After the successful MQTT connection

message on the console ***“An MQTT connection is established with***

<MQTT_BROKER_ENDPOINT>”, the device is ready for Publishing and Subscription of messages.

Note: This application involves AWS MQTT IoT Core, the user has an option to use the AWS IoT Dashboard for validation purposes, in addition to using the Renesas GUI-based Dashboard for a customized view of all the Sensor Data.

For verification purposes, the user can use the AWS IoT core Dashboard for configuring and controlling the subscription and publishing of the topics as described in the following sections.

On the AWS cloud Dashboard side, go to IoT Core and select **Test**, then choose **MQTT test client**. Subscribe to a topic listed below one at a time. A sample snapshot of subscribing to the topics is shown below.

Note: The messages shown below are **case-sensitive**; users need to take care of this by entering the publish or subscribe messages.

Only enter one message at a time. Copy the message ‘as-is’ between the quotes and do not include any extra spaces.

```
“aws/topic/iaq_sensor_data”
“aws/topic/oaq_sensor_data”
“aws/topic/hs3001_sensor_data”
“aws/topic/icm_sensor_data”
“aws/topic/icp_sensor_data”
“aws/topic/ob1203_sensor_data”
```

Note: After the subscription to the Topics, the Dashboard is ready to receive the messages being published from the device.

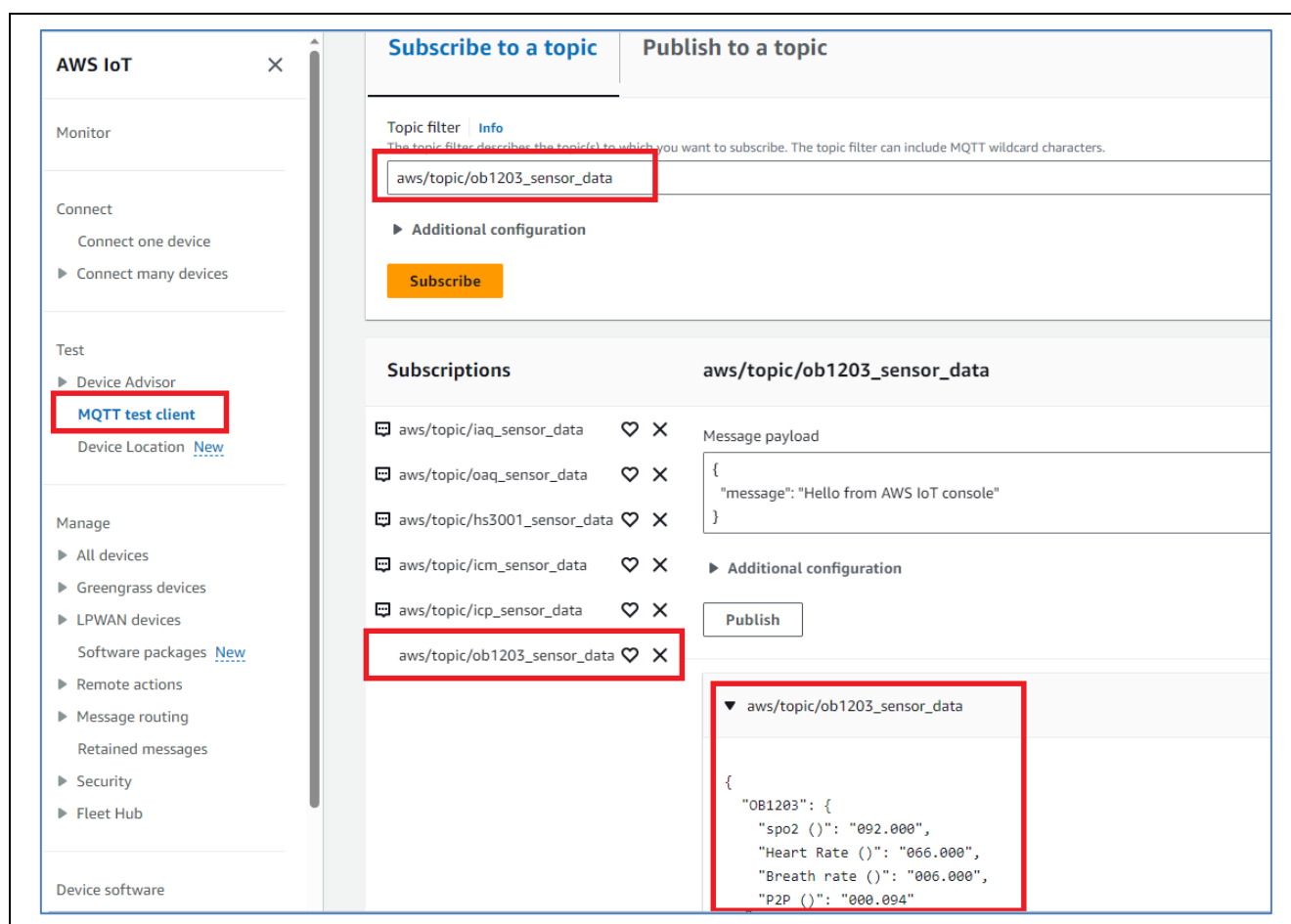


Figure 72. Subscribe to a Topic Messages on the AWS IoT Screen

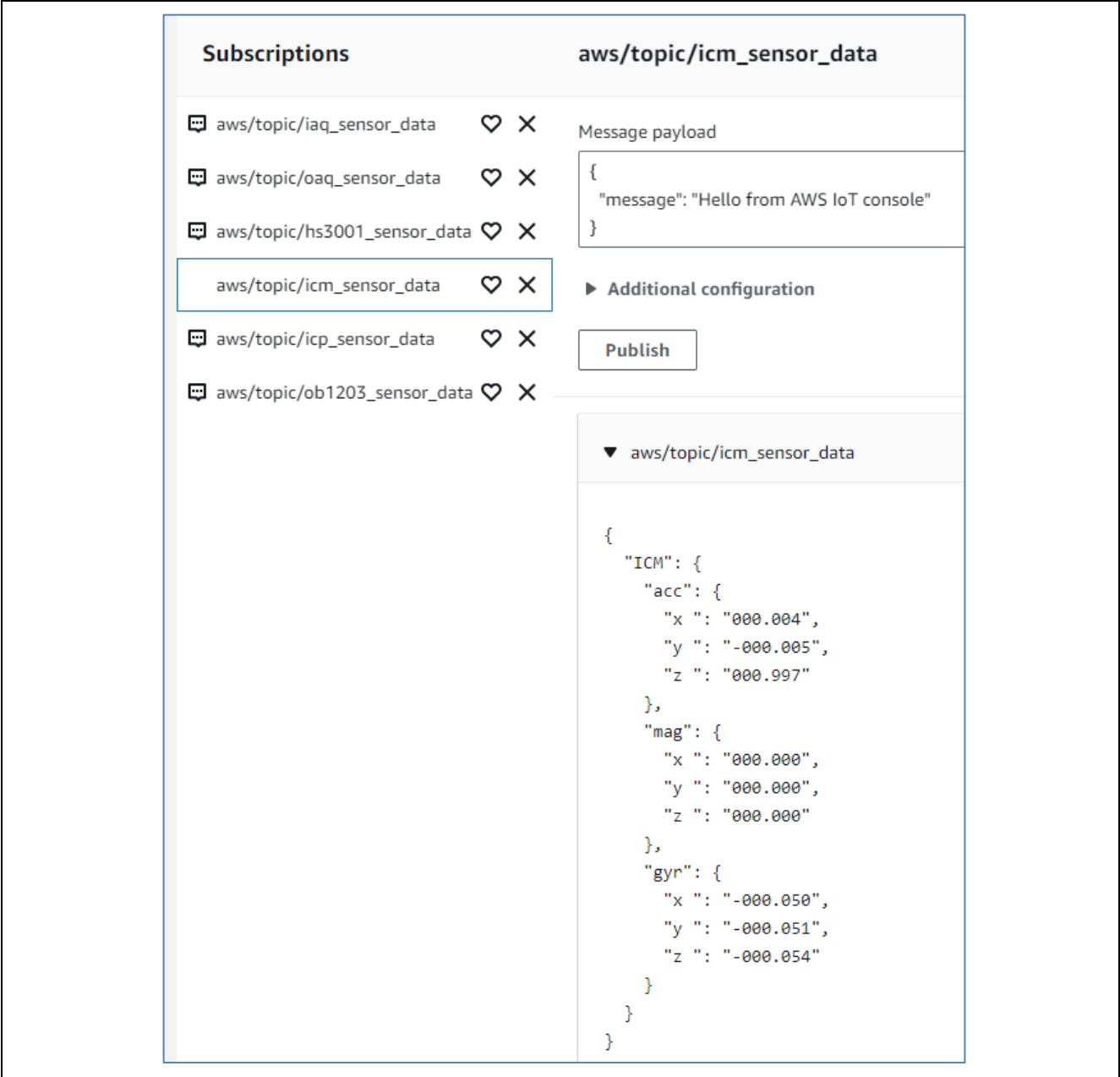


Figure 73. Subscribed Messages on the AWS IoT Screen

5.5.2 Publish a Topic Messages on the AWS Dashboard and Renesas Dashboard

5.5.2.1 With AWS Dashboard

The board subscribed to the topic: **aws/topic/<topicRx>**

If we publish the below data from AWS console

HS3001 temperature alerts:

Based on temperature dashboard will send the alert messages to CK-RX65N v2 kit via below topic

Topic: aws/topic/set_temperature_led_data

Message: {"Temperature_LED": "HOT"}	Will turn on RED in Tri-Color LED
Message: {"Temperature_LED": "WARM"}	Will turn on GREEN in Tri-Color LED
Message: {"Temperature_LED": "COLD"}	Will turn on BLUE in Tri-Color LED

Example:

Click **Test > MQTT test client**

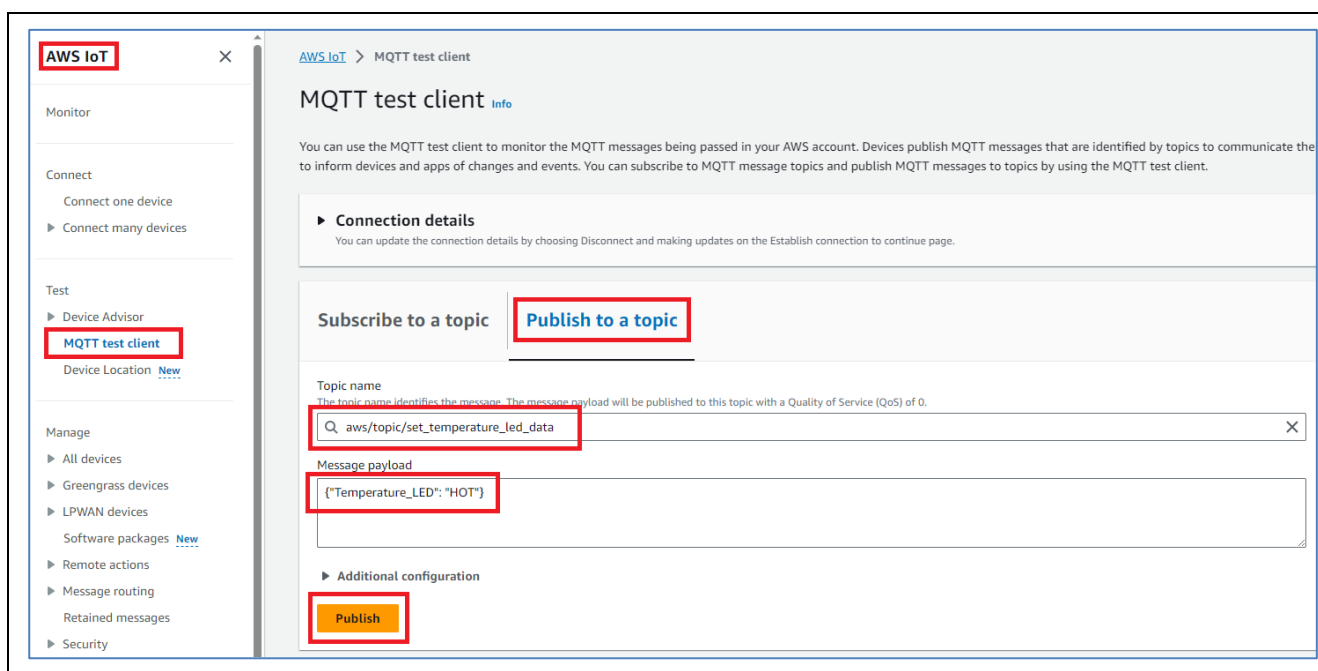


Figure 74. Publish the MQTT Message (1/2)

OB1203 SPO2 alerts:

Based on SPO2 value dashboard will send the alert messages to CK-RX65N v2 kit via below topic

Topic: aws/topic/set_spo2_led_data

Message: {"Spo_LED": "ON"}	Will turn on BLUE LED in CK-RX65N v2
Message: {"Spo_LED": "OFF"}	Will turn off BLUE LED in CK-RX65N v2

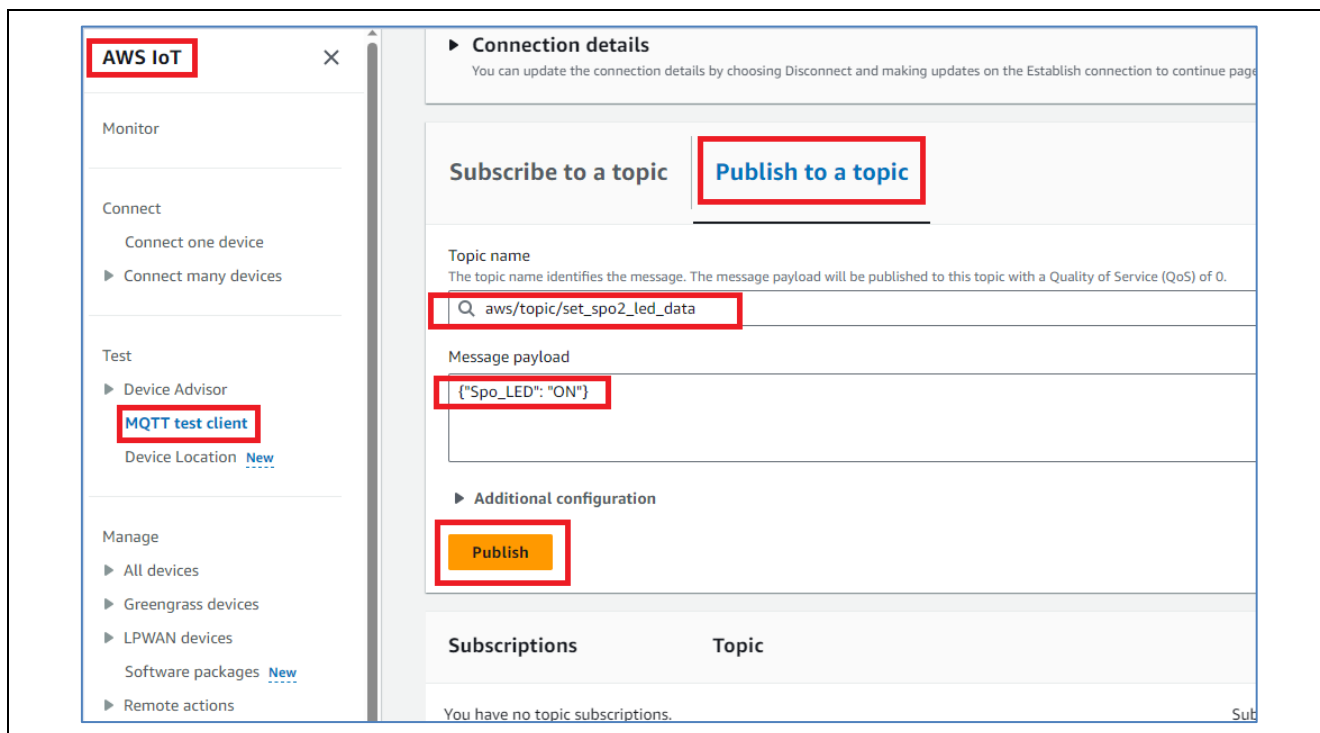


Figure 75. Publish the MQTT Message (2/2)

5.5.2.2 With Renesas Dashboard

Grafana alerts are a way to send notifications when a metric crosses a threshold that has been configured. By default, the dashboard has thresholds for the following sensors:

- OB1203-SPO2: SPO2 above 90, SPO2 below 90
- HS3001 – Temperature, F:
 - Temperature – Cold: below 65
 - Temperature – Warm: within a range from 65 to 85
 - Temperature – Hot: above 85

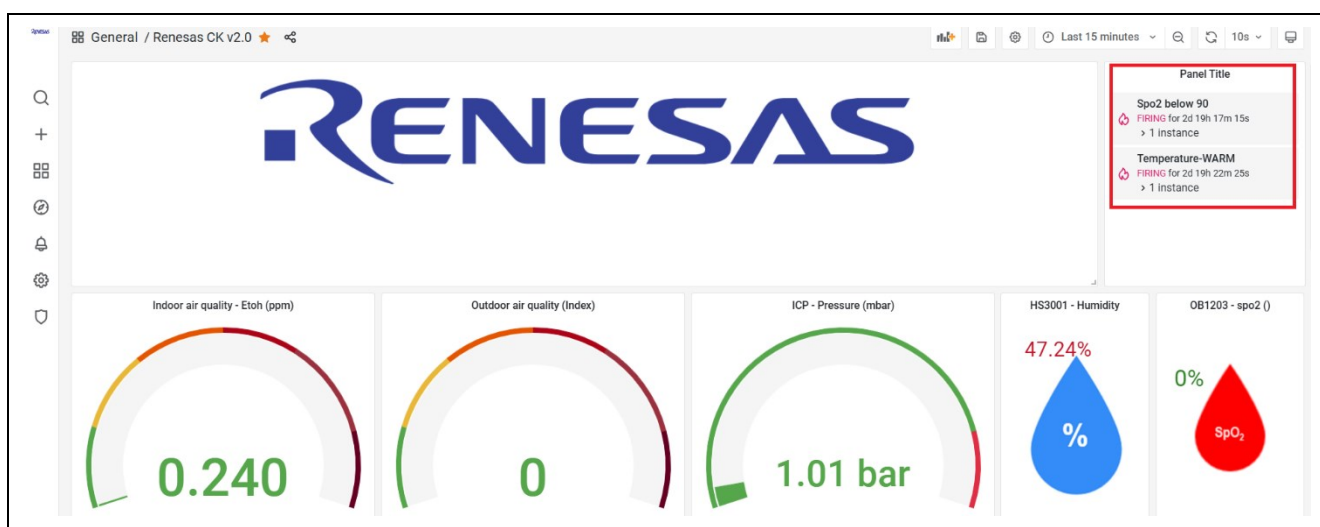


Figure 76. Sensor Status Feedback

Sensor status feedback is sent to the device which is indicated by the LEDs.

6. OTA over MQTT

6.1 Overview OTA

This section describes the steps for using OTA in this application.

OTA (Over-The-Air) updates are crucial in maintaining the functionality, security, and performance of IoT devices. This feature allows the user to efficiently deploy updates, patches, or new versions of software to connected IoT devices without requiring physical access to each device.

Please refer to the document about OTA: [OTA-using-Amazon-Web-Services-in-RX65N-FreeRTOS-for-v202210.01-LTS-rx-1.1.0](#)

Note: This OTA feature in the application is not available for users who use AWS accounts with free 10\$ credit and a dashboard that are provided by Renesas, because of some limitations with this account's permission.

6.2 Prerequisites

6.2.1 Installing Python

1. Access the Python download website: [Python downloaded website](#) and Download the Python 3.11.0 installer: Click the **Download** link for Python 3.11.0

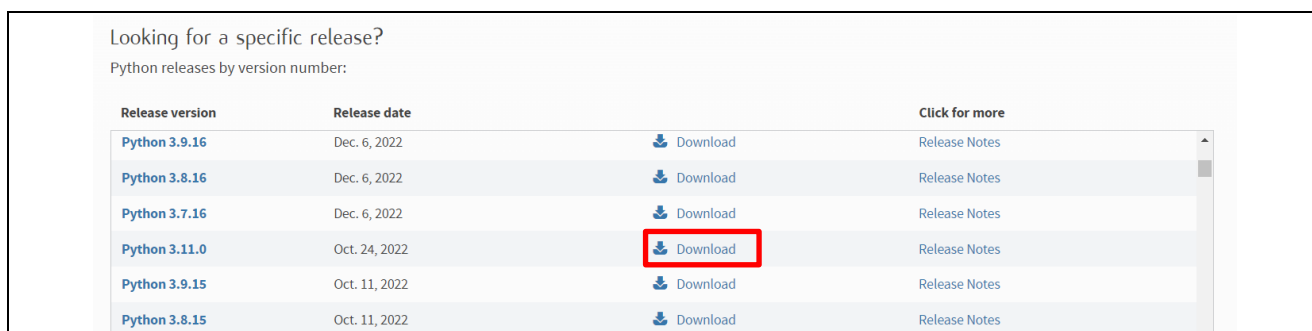


Figure 77. Python Download Website

2. Run the installer and follow the prompts to install Python.

On the installation screen, select the **Add python.exe to PATH** check box.

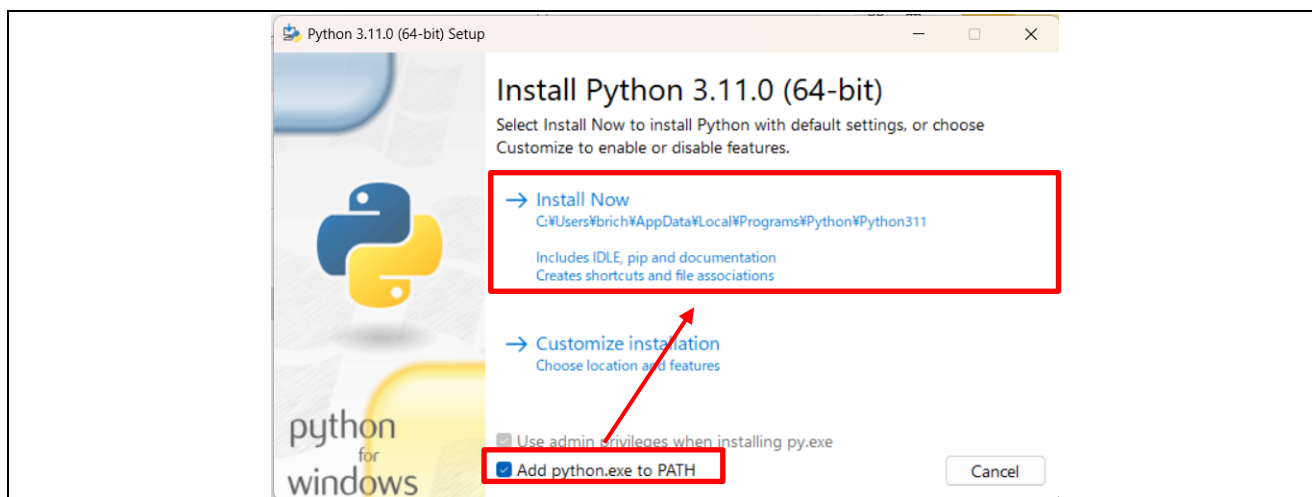


Figure 78. Python 3.11.0 installer

3. Open a command prompt and confirm that Python 3.11.0 is installed.

Execute the following command and confirm that information appears: `$python -V`

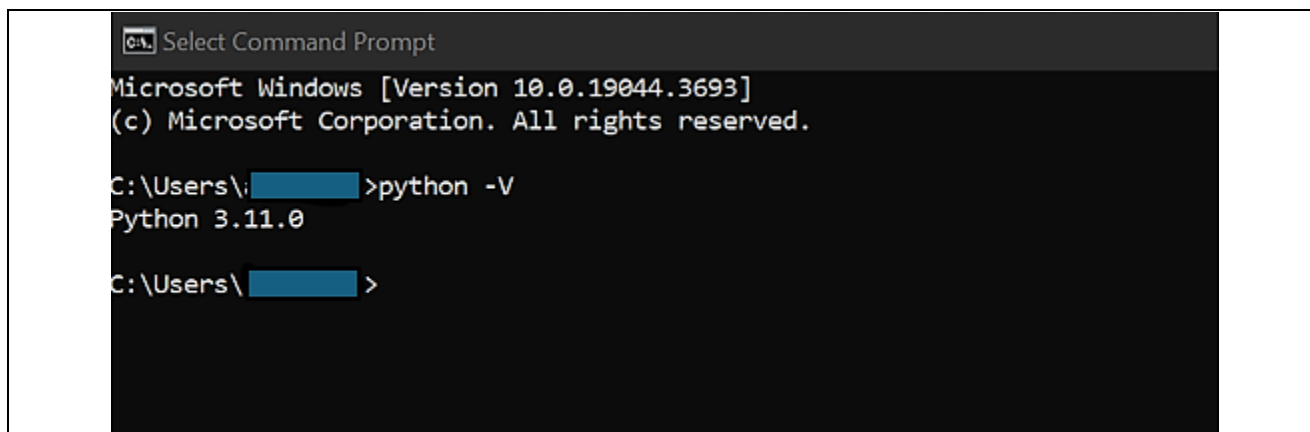


Figure 79. Checking version of Python

4. Install the Python encryption library (pycryptodome)

Install the encryption library by executing the following command: `$ pip install pycryptodome`

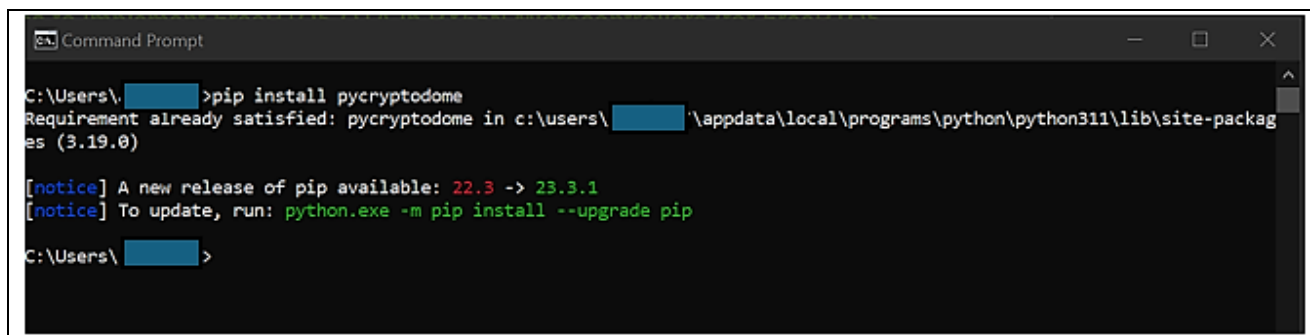


Figure 80. Installing Python encryption library

6.2.2 Installing OpenSSL

1. Access the Win32/Win64 download web site for OpenSSL: [OpenSSL Download Website](#) and download the installer for the operating system you are using.

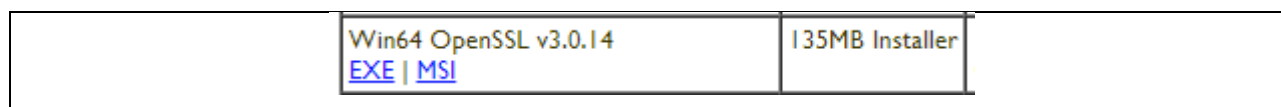


Figure 81. OpenSSL Download Website

2. Run the installer and follow the prompts to install OpenSSL.
3. From the Start Menu, open the Win64 OpenSSL Command Prompt and confirm that OpenSSL is installed.

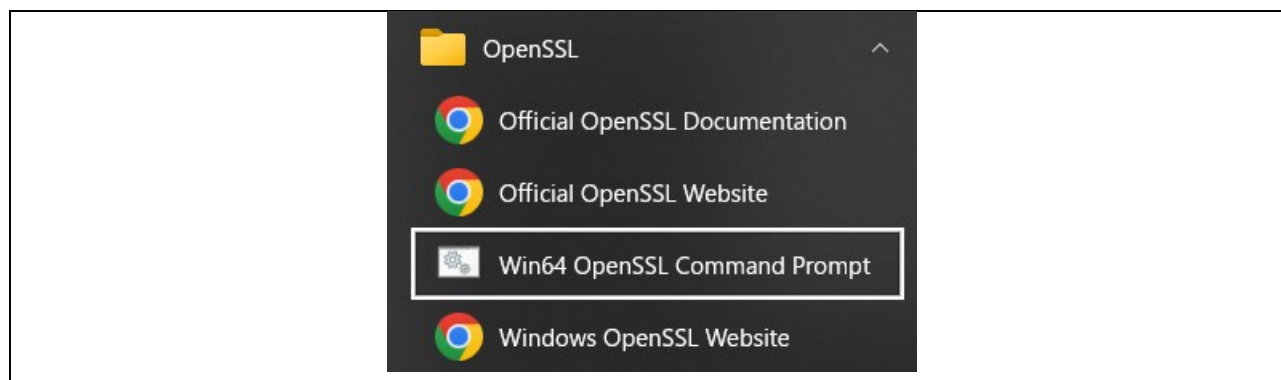


Figure 82. Open the Win64 OpenSSL Command Prompt

Execute the following command and confirm that information appears: `$openssl version`

6.2.3 Installing Renesas Image Generator

Renesas Image Generator is a tool that generates the firmware images used by the firmware update module. Renesas Image Generator can generate the following images for use by the firmware update module:

- Initial image: An image file containing the bootloader and application program written by flash writer during initial system configuration (extension: mot)
- Update image: An image file containing the updated firmware (extension: rsu)

Renesas Image Generator is provided as part of the Firmware Update FIT module.

Note: Version Rev.2.00 and later of the Firmware Update module only support firmware generation using Python scripts.

1. Access the link [RX Family Firmware Update module Using Firmware Integration Technology Application Notes Rev.2.03 - Sample Code | Renesas](#) and download the firmware update module.

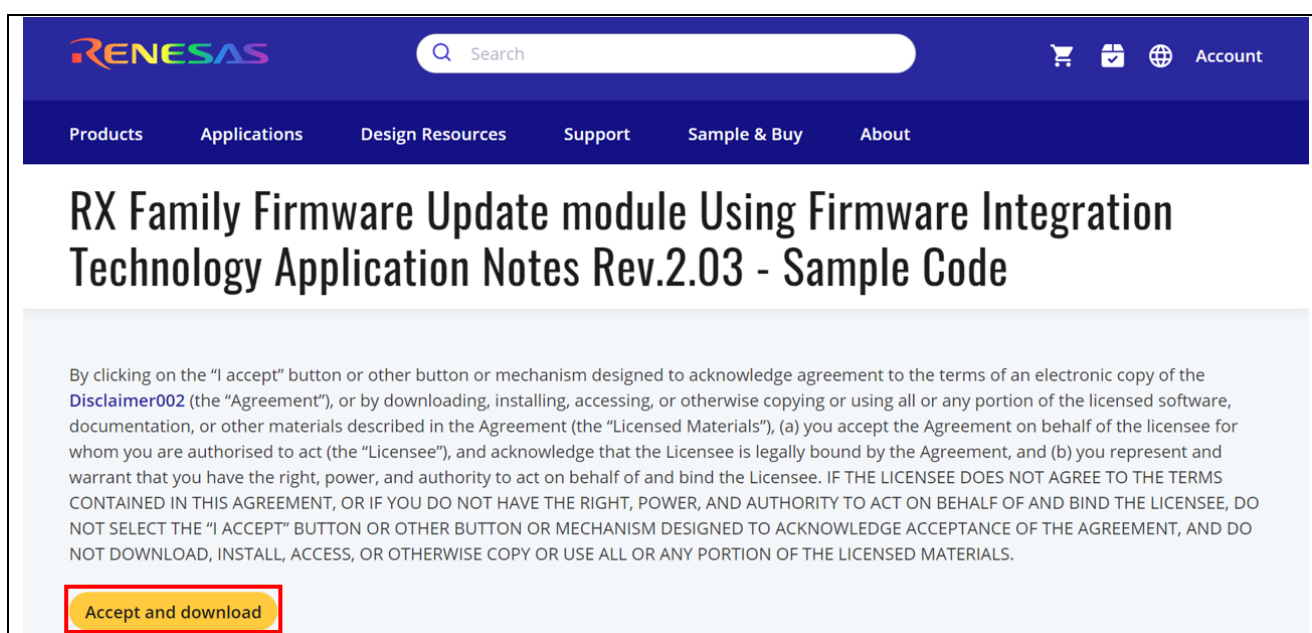


Figure 83. Renesas Image Generator Downloading (1/2)

2. Extract the downloaded firmware update module.

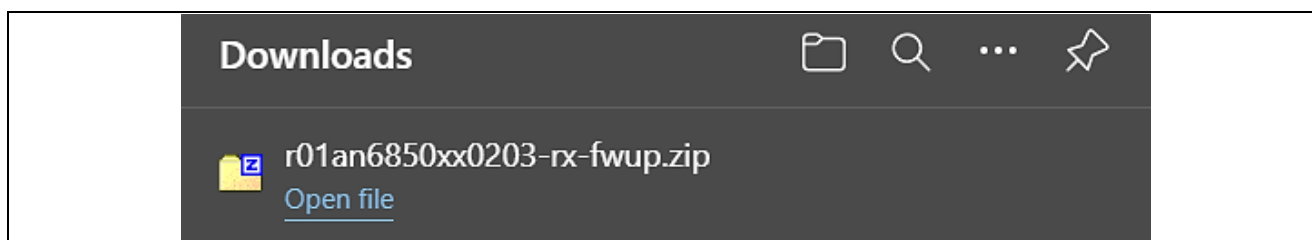


Figure 84. Renesas Image Generator Downloading (2/2)

3. Extract Renesas Image Generator.

Extract the file RenesasImageGenerator.zip in the firmware update module. The RenesasImageGenerator folder contains the Renesas Image Generator script file (image-gen.py) and the parameter files for various devices (*_ImageGenerator_PRM.csv).

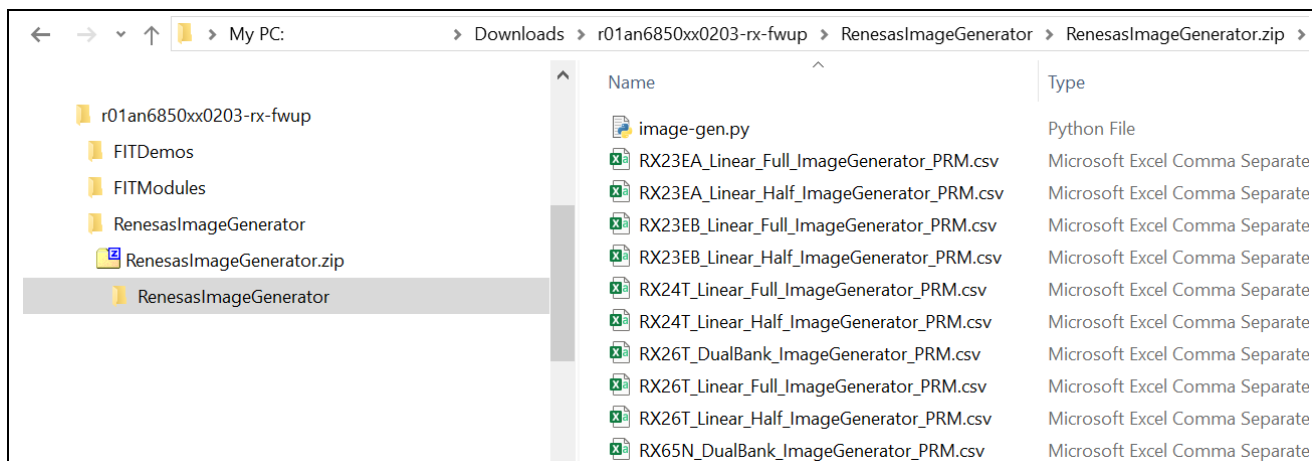


Figure 85. Renesas Image Generator package

6.3 Setting up AWS for OTA

6.3.1 Register your Device in AWS

See chapter 5.4 for details on how to sign up for an AWS account.

6.3.2 Creating an Amazon S3 bucket

Amazon S3 is an online storage web service used to store the firmware with which the device will be updated.

1. From the **Services** menu, select **Storage** and then choose **S3**.

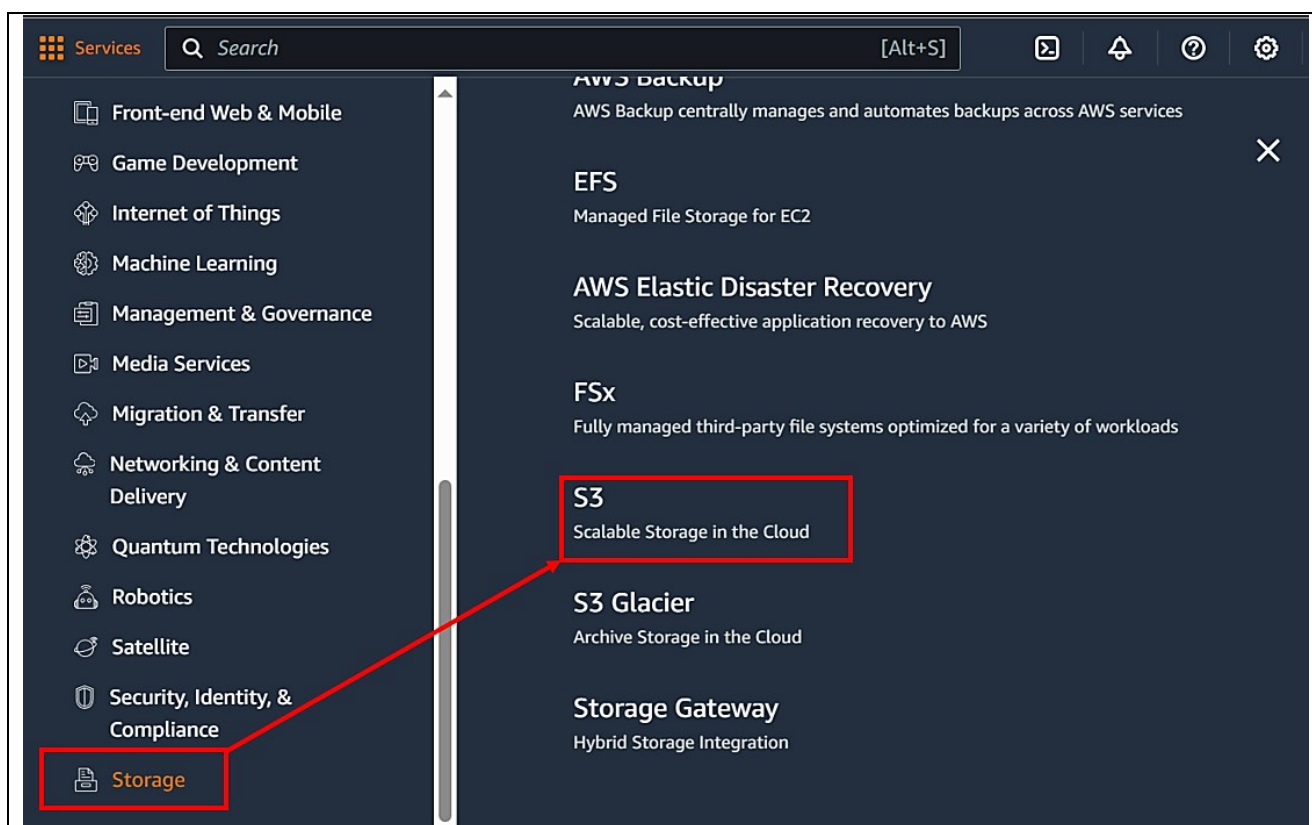


Figure 86. Create AWS S3 bucket (1/2)

2. On the **Buckets** page, click the **Create bucket** button.

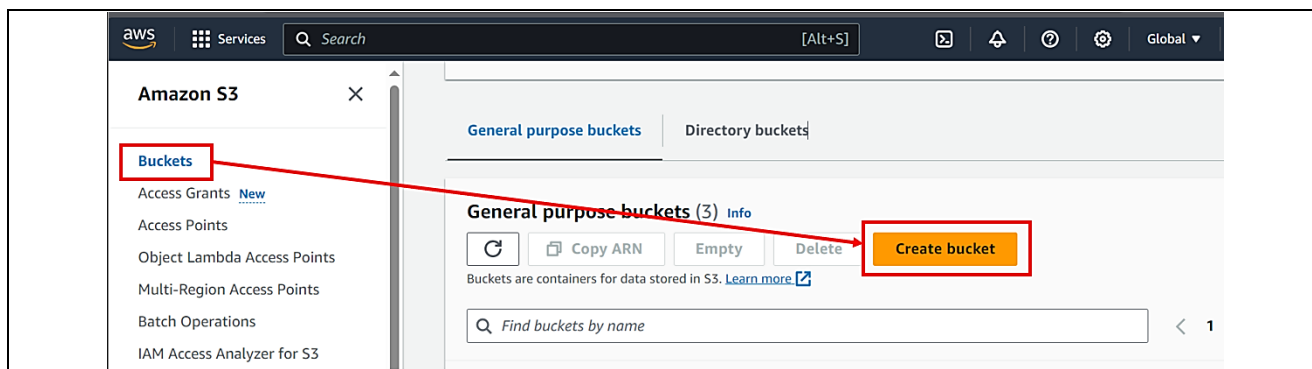


Figure 87. Create AWS S3 bucket (2/2)

3. Enter a bucket name (example: s3test-rx65nv2)

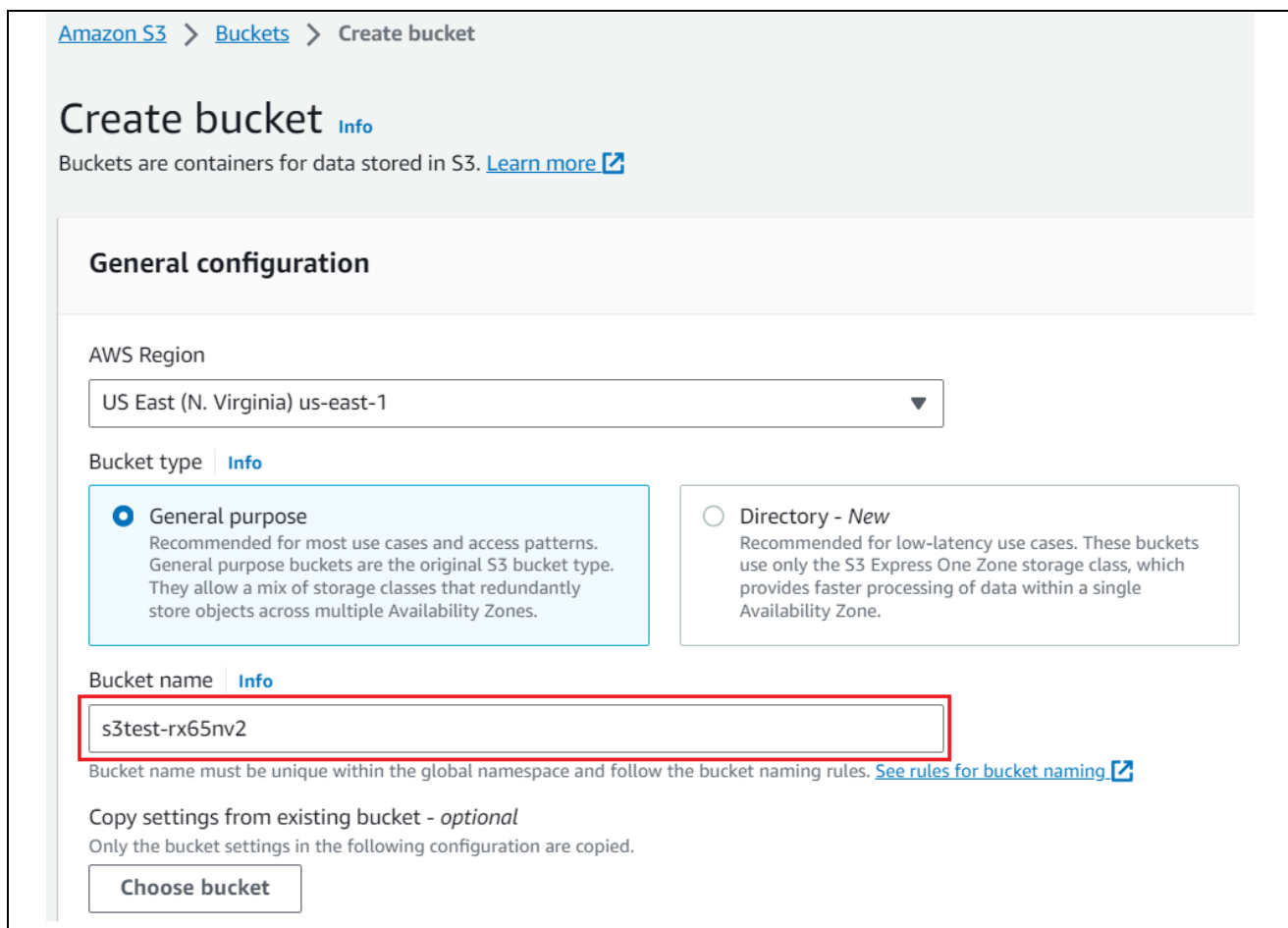


Figure 88. Naming for bucket

Note: The bucket name must be globally unique. The following error message: “Bucket with the same name already exists” appears if the bucket name is already in use. In this case, use another name.

4. Create the bucket.

Enter the settings as follows, and then click the **Create bucket** button.

- Block the Public Access setting for this bucket: **Block all public access.**
- Bucket Versioning: **Enable**

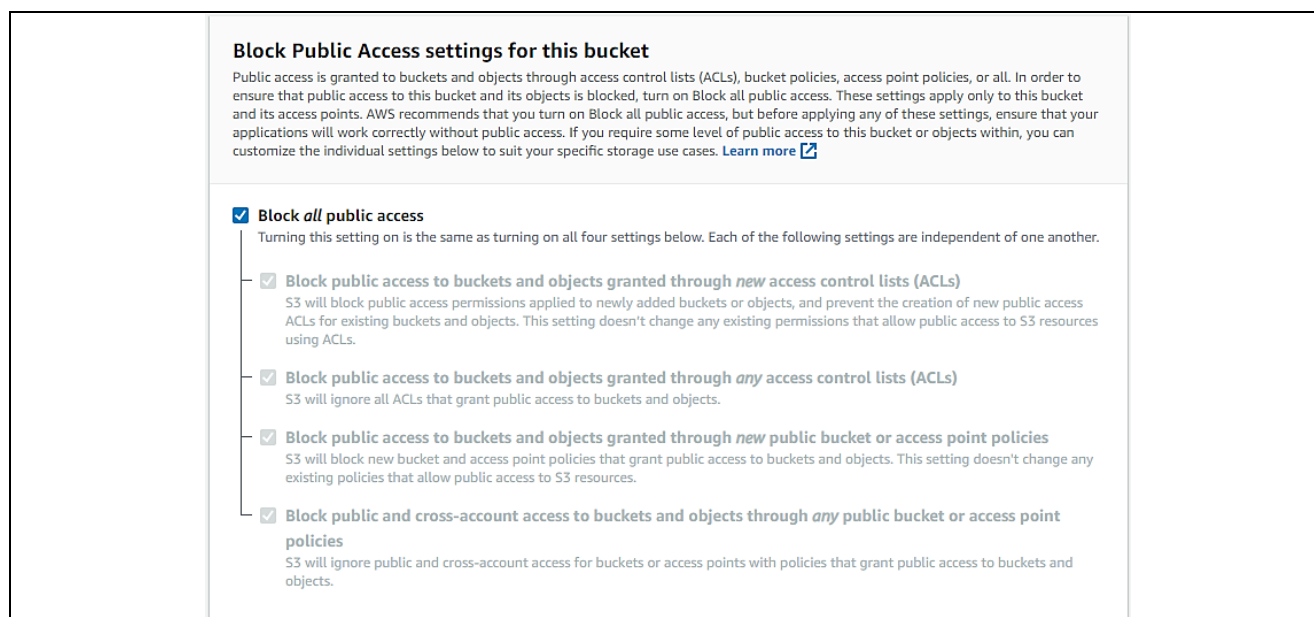


Figure 89. Bucket setting (1/2)

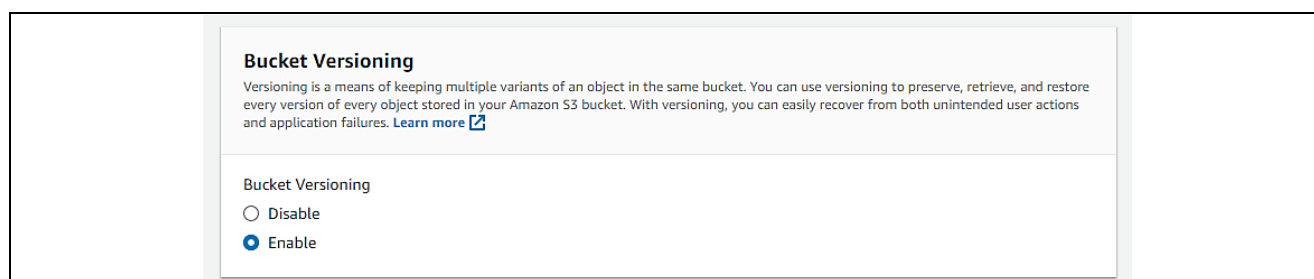


Figure 90. Bucket setting (2/2)

6.3.3 Allocating OTA execution permission to IAM users

Create a role with the appropriate access permissions to create OTA update jobs.

1. Enter "IAM" in the search box at the top of the screen and click **IAM** in the search results.

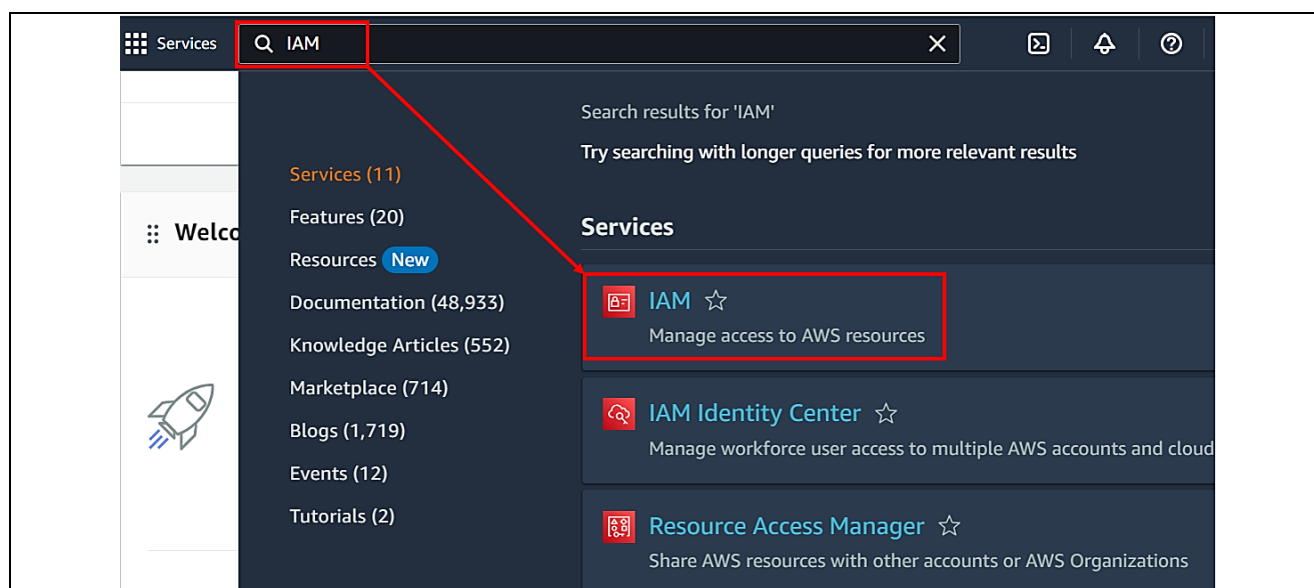


Figure 91. IAM search box

2. In the menu, click **Roles** and then click the **Create role** button.

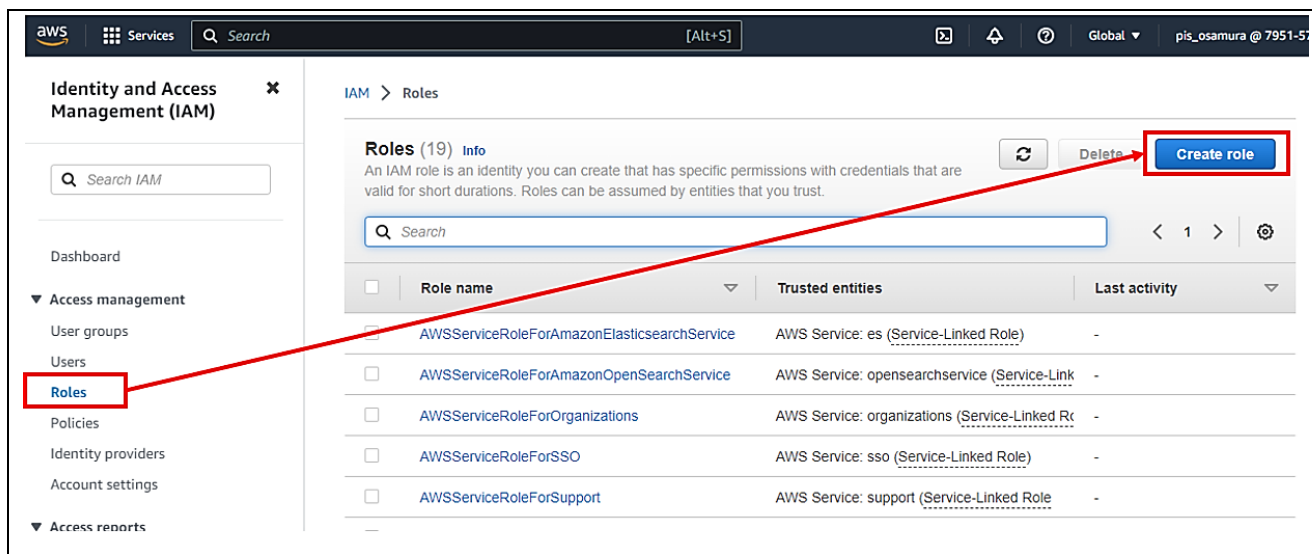


Figure 92. Creating a role

3. Under **Select trusted entity**, enter the following settings, and then click **Next**:

- Under **Trusted entity type**, select **AWS service**
- Under **Use cases for other AWS services**, select **IoT**
- Select the **IoT** option button

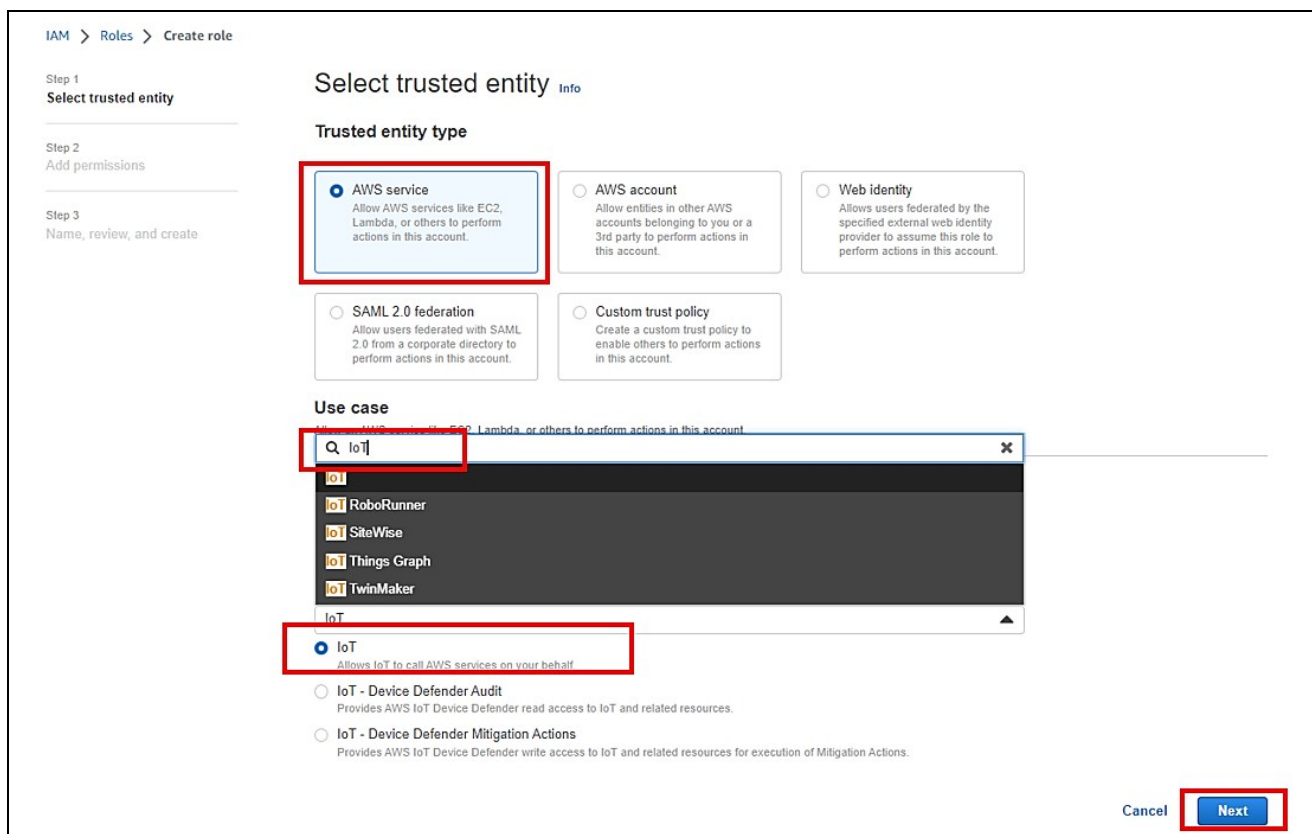


Figure 93. Selecting trusted entity

4. Click **Next** on the **Add permissions** page without making any changes.

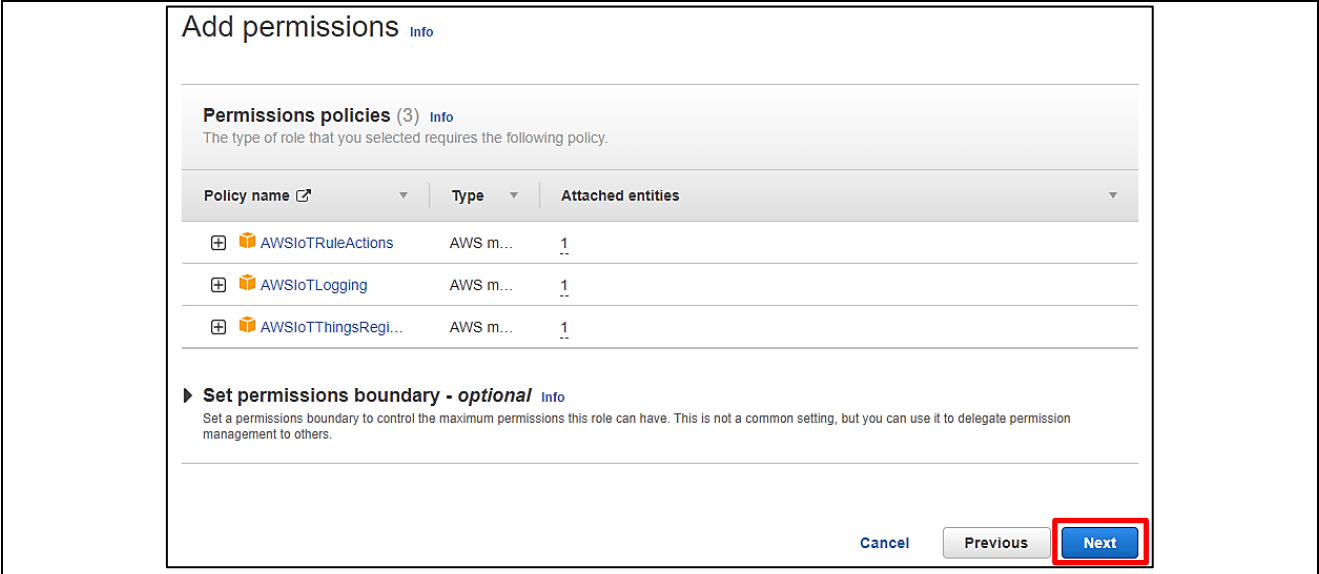


Figure 94. Add Permissions for Role

5. Enter a role name (example: ota_role_rx65n), and then click the **Create role** button

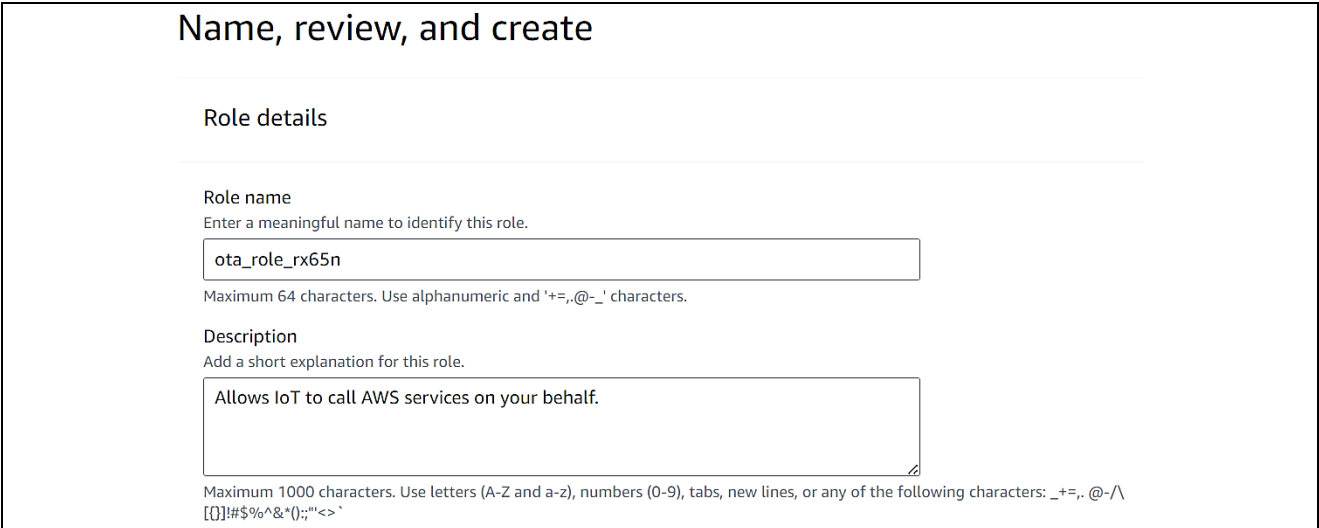


Figure 95. Naming for Created Role

6. Click on the role you created.

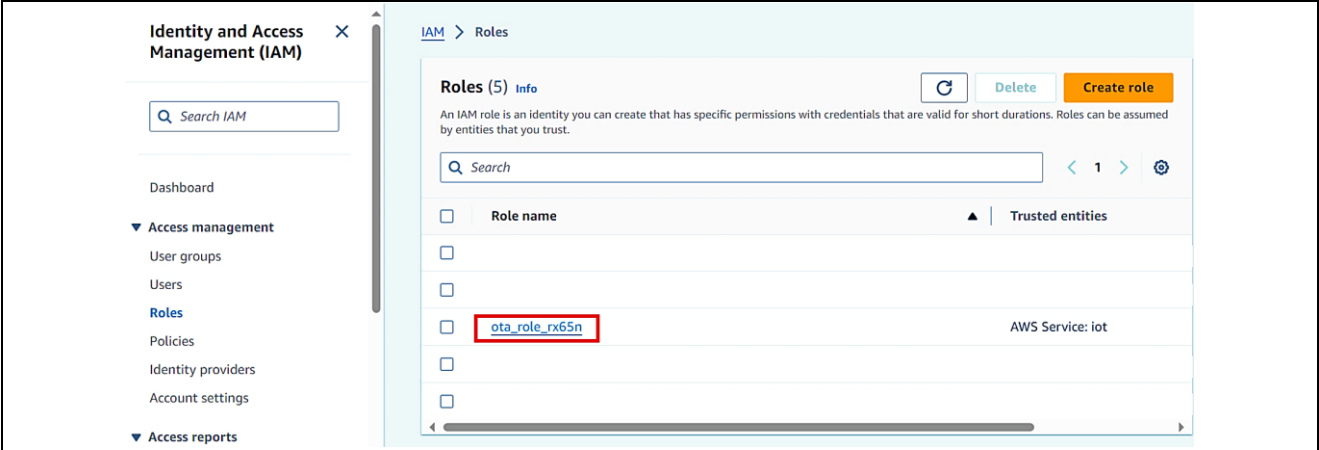


Figure 96. Created Role

7. Select Attach policies.

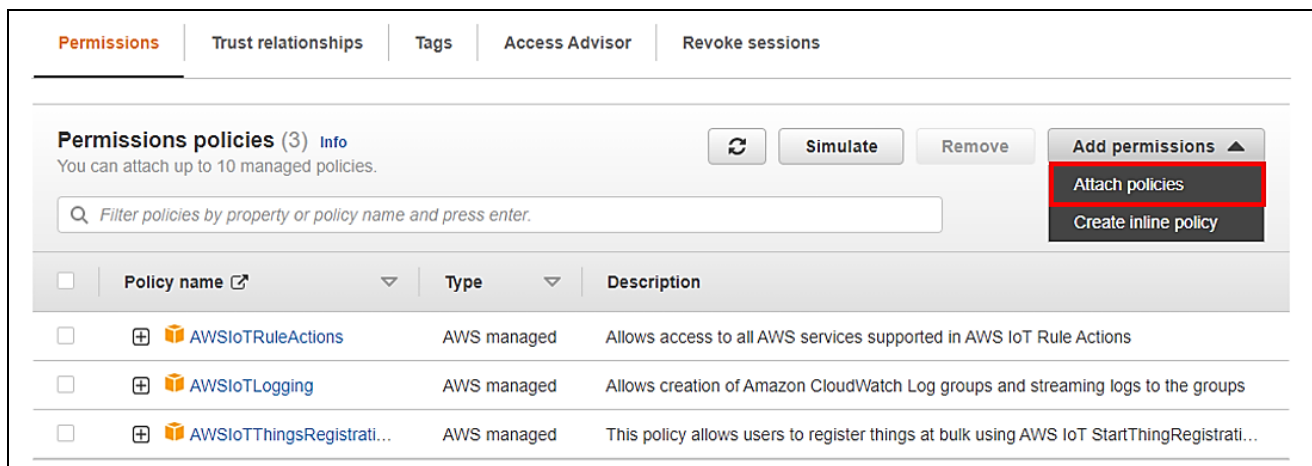


Figure 97. Add permission (1/3)

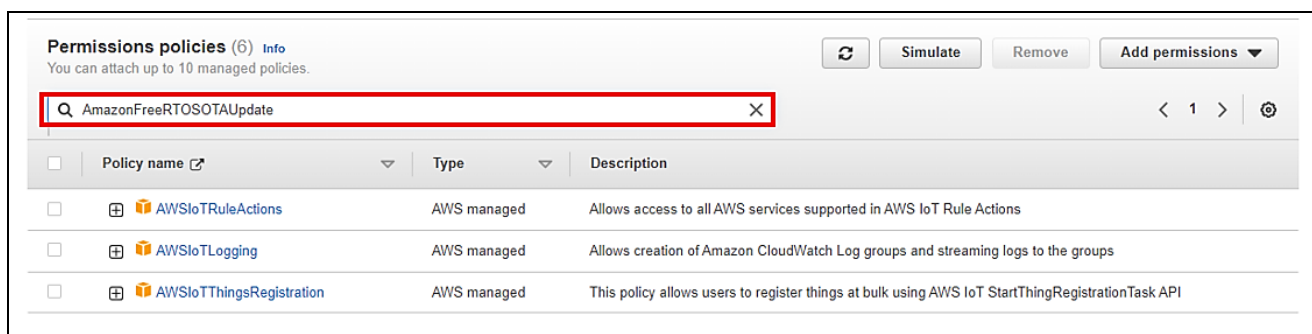
8. Enter AmazonFreeRTOSOTAUpdate in the **Permissions policies** search box, and then press the **Enter** key.

Figure 98. Add permission (2/3)

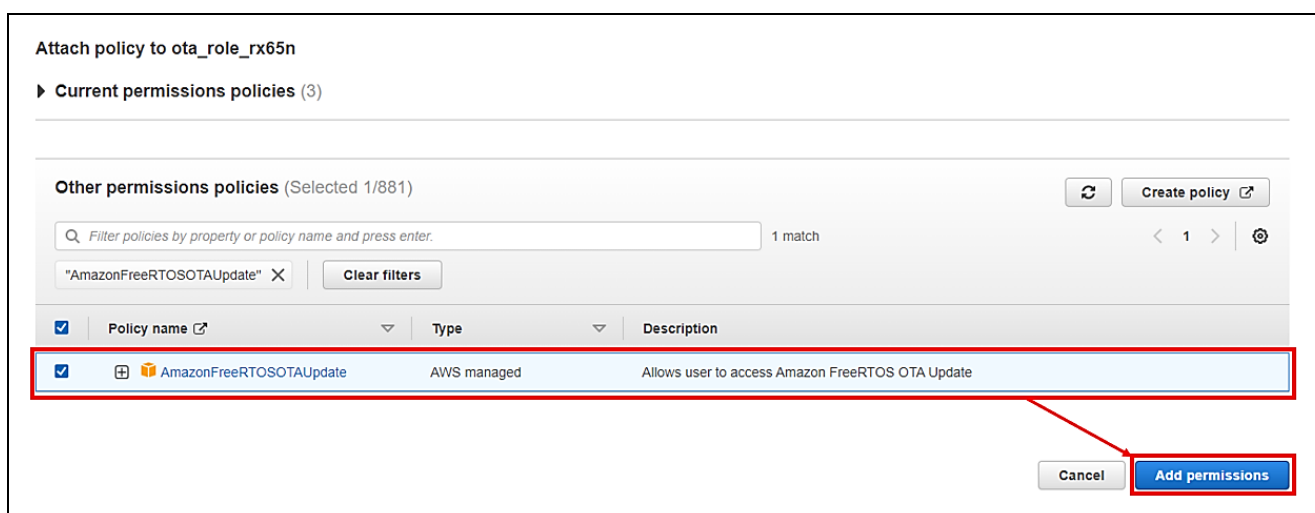
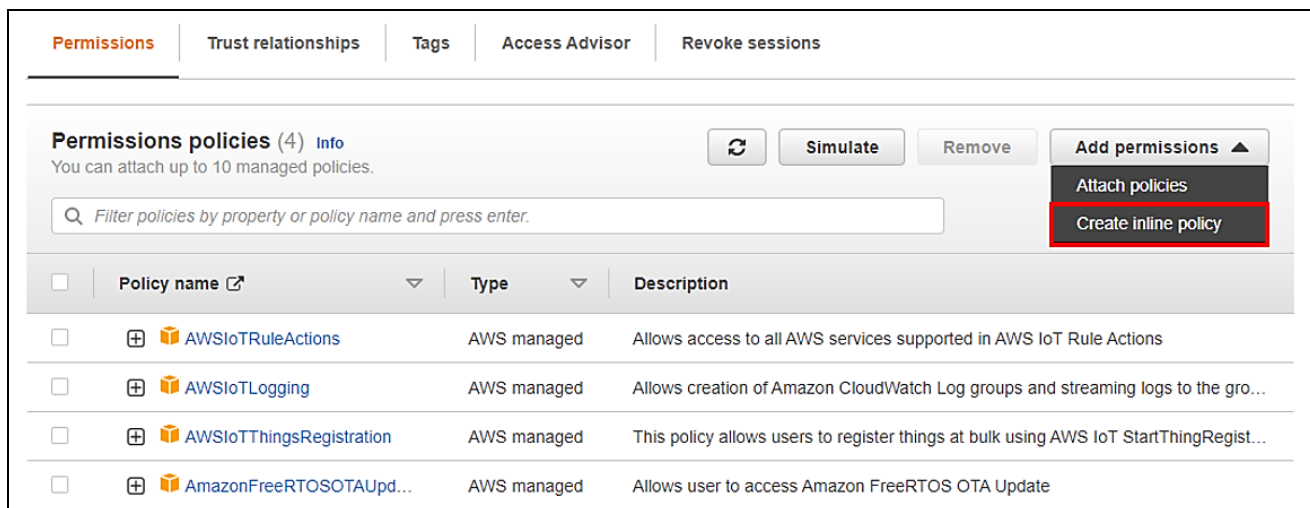
9. Select the check box beside the AmazonFreeRTOSOTAUpdate policy, and then click the **Add permissions** button.

Figure 99. Add permission (3/3)

10. From the Add permissions drop-down list, select Create inline policy.



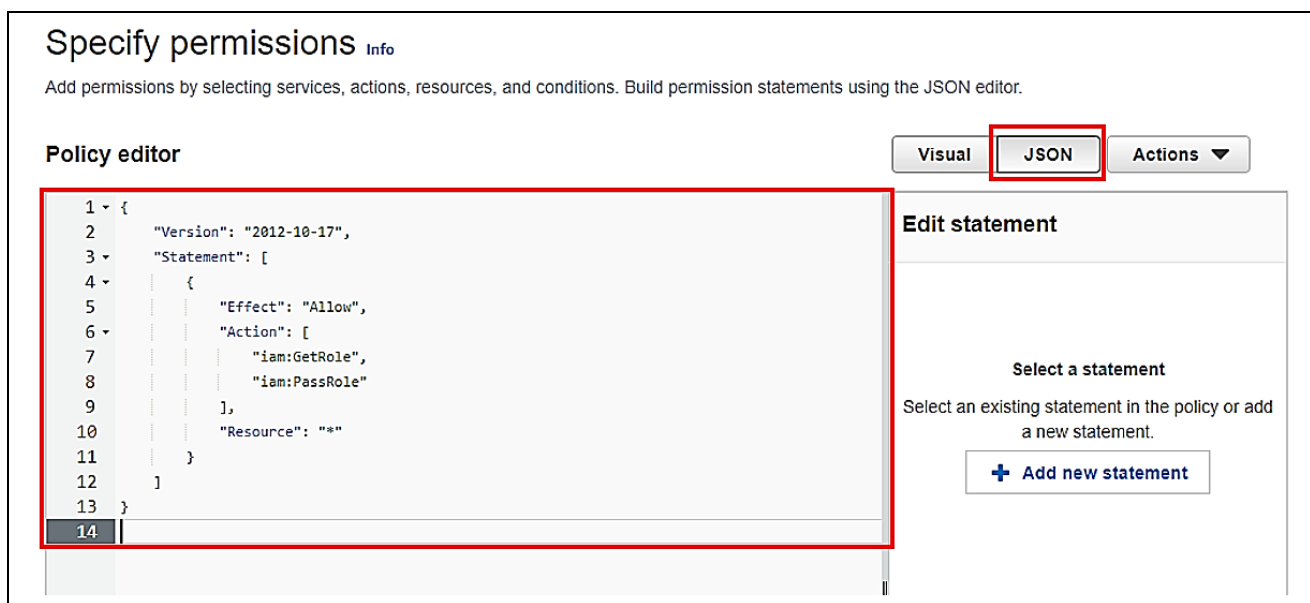
The screenshot shows the AWS IAM console 'Permissions policies' page. The 'Add permissions' dropdown menu is open, and 'Create inline policy' is highlighted with a red box. The page also shows a list of existing policies, including AWS managed policies like AWSIoTRuleActions, AWSIoTLogging, AWSIoTThingsRegistration, and AmazonFreeRTOSOTAUpdate.

Figure 100. Create inline policy (1/3)

11. Click **JSON**, paste the following code, and then click **Next**.

This code grants permission to pass the IAM role to AWS services.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```



The screenshot shows the AWS IAM console 'Specify permissions' page. The 'JSON' tab is selected, and the JSON code is pasted into the editor. The 'Add new statement' button is highlighted with a red box. The page also shows a list of existing statements and a 'Select a statement' dropdown menu.

Figure 101. Create inline policy (2/3)

12. Enter a policy name (example: rx65n_ota_demo_iam_policy), and then click the **Create policy** button.

Review and create
Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and *,_,@,- characters.

Permissions defined in this policy [Info](#) [Edit](#)
Permissions in the policy document specify which actions are allowed or denied.

Allow (1 of 384 services) ☐ Show remaining 383 services

Service	Access level	Resource	Request condition
IAM	Limited: Read, Write	All resources	None

[Cancel](#) [Previous](#) [Create policy](#)

Figure 102. Create inline policy (3/3)

13. Again, from the **Add permissions** drop-down list, select **Create inline policy**.

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (4) [Info](#) [Refresh](#) [Simulate](#) [Remove](#) [Add permissions](#)

You can attach up to 10 managed policies.

☐	Policy name	Type	Description
☐	AWSIoTRuleActions	AWS managed	Allows access to all AWS services supported in AWS IoT Rule Actions
☐	AWSIoTLogging	AWS managed	Allows creation of Amazon CloudWatch Log groups and streaming logs to the gro...
☐	AWSIoTTThingsRegistration	AWS managed	This policy allows users to register things at bulk using AWS IoT StartThingRegistr...
☐	AmazonFreeRTOSOTAUpd...	AWS managed	Allows user to access Amazon FreeRTOS OTA Update

Figure 103. Create inline policy (1/3)

14. Click **JSON**, paste the following code, and then click **Next**.

This code allows access to Amazon S3 where the updated firmware is stored.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

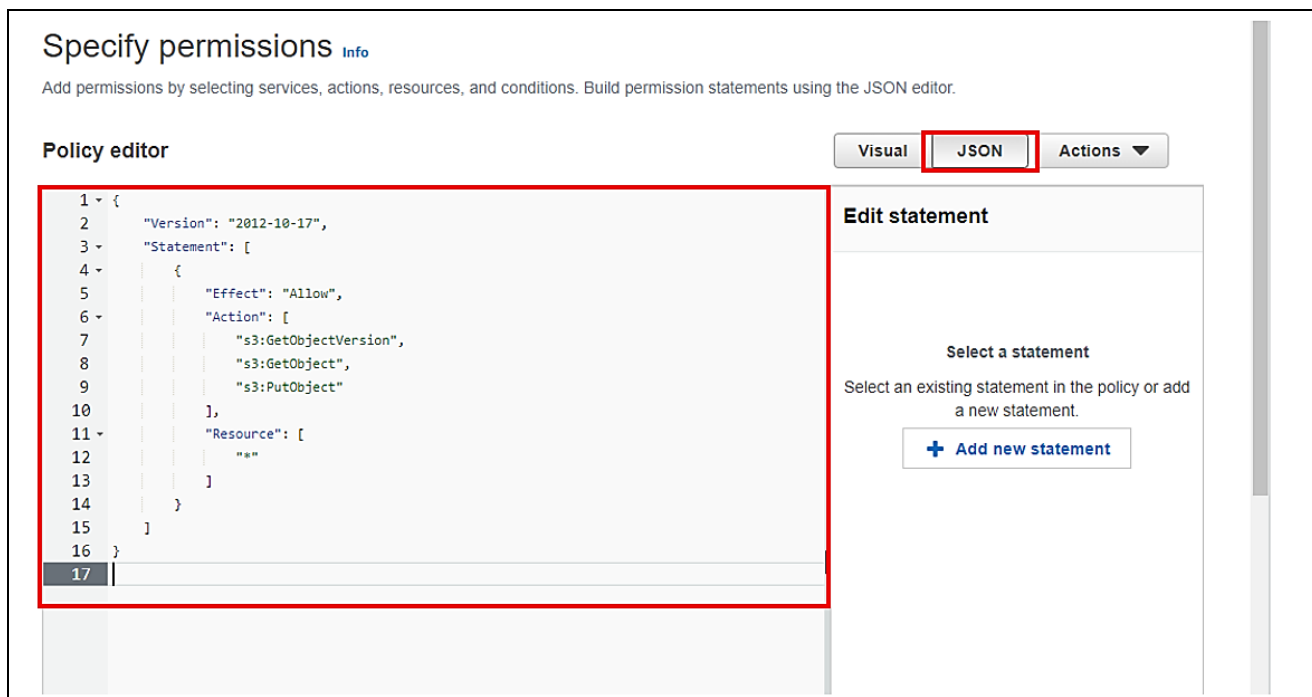


Figure 104. Create inline policy (2/3)

15. Enter a policy name (example: rx65n_ota_demo_s3_policy), and then click the **Create policy** button.

Review and create
Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and "+, @, _" characters.

Permissions defined in this policy [Info](#) [Edit](#)

Permissions in the policy document specify which actions are allowed or denied.

Allow (1 of 384 services) ☐ Show remaining 383 services

Service	Access level	Resource	Request condition
S3	Limited: Read, Write	All resources	None

[Cancel](#) [Previous](#) [Create policy](#)

Figure 105. Create inline policy (3/3)

6.4 Setting up the Device

6.4.1 Generating Key Pairs and Certificates

1. Open the **Win64 OpenSSL Command Prompt**
2. Create a CA private key using ECDSA.

Execute the following command: `"openssl ecparam -genkey -name secp256r1 -out ca.key"`

```
C:\Users\...\Create_key>openssl ecparam -genkey -name secp256r1 -out ca.key
using curve name prime256v1 instead of secp256r1
```

Figure 106. Creating CA private key

3. Execute the command to create a CA certificate from the CA private key you created

Execute the following command: `"openssl req -x509 -sha256 -new -nodes -key ca.key -days 3650 -out ca.crt"`

```
C:\Users\...\Create_key>openssl req -x509 -sha256 -new -nodes -key ca.key -days 3650 -out ca.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Enter any character string for these attributes.
```

Figure 107. Creating CA certificate

4. Create an ECDSA key pair.

Execute the following command: “openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair”

```
C:\Users\...\Create_key>openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1
```

Figure 108. Creating ECDSA key pairs

5. Create a certificate signing request from the created ECDSA key pair.

Execute the following command: “openssl req -new -sha256 -key secp256r1.keypair > secp256r1.csr”

You can enter any character string for **Country Name** onward. For the last two lines, press **Enter** without entering anything.

```
C:\Users\...\Create_key>openssl req -new -sha256 -key secp256r1.keypair > secp256r1.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\Users\...\Create_key>
```

Enter any character string for these attributes.

Press **Enter** without entering anything.

Figure 109. Creating certificate signing request

6. Create a certificate from the certificate signing request, CA certificate, and CA private key.

Execute the following command:

“openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out secp256r1.crt”

```
C:\Users\...\Create_key>openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out secp256r1.crt
Certificate request self-signature ok
subject=C = , ST = , L = , O = , OU = , CN = , emailAddress =
```

Figure 110. Creating code signing certificate

7. Extract the private key from the ECDSA key pair.

Execute the following command:

“openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey”

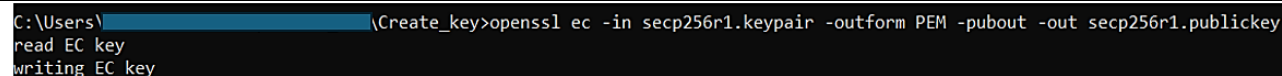
```
C:\Users\...\Create_key>openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
read EC key
writing EC key
```

Figure 111. Create a private key (secp256r1.privatekey)

8. Extract the public key from the ECDSA key pair.

Execute the following command:

`"openssl ec -in secp256r1.keypair -outform PEM -pubout -out secp256r1.publickey"`



```
C:\Users\...\Create_key>openssl ec -in secp256r1.keypair -outform PEM -pubout -out secp256r1.publickey
read EC key
writing EC key
```

Figure 112. Create a public key (secp256r1.publickey)

9. After creating public and private keys, please check the results:

Name	Date modified	Type	Size
ca.crt	12/10/2023 10:06 AM	Security Certificate	1 KB
ca.key	12/10/2023 10:05 AM	KEY File	1 KB
ca.srl	12/10/2023 10:07 AM	SRL File	1 KB
secp256r1.crt	12/10/2023 10:07 AM	Security Certificate	1 KB
secp256r1.csr	12/10/2023 10:07 AM	CSR File	1 KB
secp256r1.keypair	12/10/2023 10:06 AM	KEYPAIR File	1 KB
secp256r1.privatekey	12/10/2023 10:18 AM	PRIVATEKEY File	1 KB
secp256r1.publickey	12/10/2023 10:14 AM	PUBLICKEY File	1 KB

Figure 113. Total created files

6.4.2 Setting up the project

6.4.2.1 Creating initial firmware

The following explains how to create the initial firmware that combines the boot loader (boot_loader_ck_rx65n_v2) and the firmware (aws_da16600_ck_rx65n).

1. Additional import "[Project Root folder]\Projects\boot_loader_ck_rx65n_v2\le2studio_gcc project:

Make sure that configuration in **boot_loader_ck_rx65n_v2.scfg** file of boot_loader_ck_rx65n_v2 project as below:

Table 7. Components Configuration of boot loader project

No	Component	Configuration
1	Startup→Generic→r_bsp (v7.42)	Enable user stdio charput function: Use user charput() function
		User stdio charput function name: my_sw_charput_function
		Select whether it is bootloader project: bootloader project
2	Drivers→Memory→r_flash_rx (v5.11)	Enable code flash programming: Includes code to program ROM area
		Enable BGO/Non-blocking data flash operations: Forces data flash API function to block until completed.
		Enable BGO/Non-blocking code flash operations: Forces ROM API function to block until completed.
		Enable code flash self-programming: Programming code flash while executing from another segment in ROM

3	Drivers→Communications→r_sci_rx (v5.00)	Include software support for channel 5: Include
4	Middleware→Generic→r_byteq (v2.10)	Memory allocation for queue control blocks: Static memory allocation
		Number of static queue control block: 32
5	Middleware→Generic→r_fwup (v2.03)	Select the update mode: Dual bank
		Select the function mode: user for Boot Loader
		Main area start address: 0xFFFF0000
		Buffer area start address: 0xFFE00000
		Install area size: 0xF0000
		Select the algorithm of signature verification: ECDSA + SHA256
		Enable user disable interrupt function: Use FWUP r_fwup_wrap_disable_interrupt() function
		Enable user enable interrupt function: Use FWUP r_fwup_wrap_enable_interrupt() function
		Enable user software delay function: Use FWUP r_fwup_wrap_software_delay() function
		Enable user software reset function: Use FWUP r_fwup_wrap_software_reset() function
		Enable user sha256 init function: Use FWUP r_fwup_wrap_sha256_init() function
		Enable user sha256 update function: Use FWUP r_fwup_wrap_sha256_update() function
		Enable user sha256 final function: Use FWUP r_fwup_wrap_sha256_final() function
		Enable user verify ecdsa function: Use FWUP r_fwup_wrap_verify_ecdsa() function
		Enable user get crypt context function: Use FWUP r_fwup_wrap_get_crypt_context() function
		Enable user flash open function: Use user r_fwup_wrap_flash_open() function
		User flash open function name: my_flash_open_function
		Enable user flash close function: Use user r_fwup_wrap_flash_close() function
		User flash close function name: my_flash_close_function
		Enable user flash erase function: Use user r_fwup_wrap_flash_erase() function
		User flash erase function name: my_flash_erase_function
		Enable user flash write function: Use user r_fwup_wrap_flash_write() function
		User flash write function name: my_flash_write_function

	Enable user flash read function: Use user r_fwup_wrap_flash_read() function
	User flash read function name: my_flash_read_function
	Enable user bank swap function: Use user r_fwup_wrap_bank_swap() function
	User bank swap function name: my_bank_swap_function

Check the boot_loader device.

In the **boot_loader_ck_rx65n_v2.scfg** file, click the **Board** tab. Confirm that **R5F565NEHxFB_DUAL** appears in the **Device** field, and **CK-RX65N-V2** in the **Board** field (Figure 114), and choose “Generate Code”.

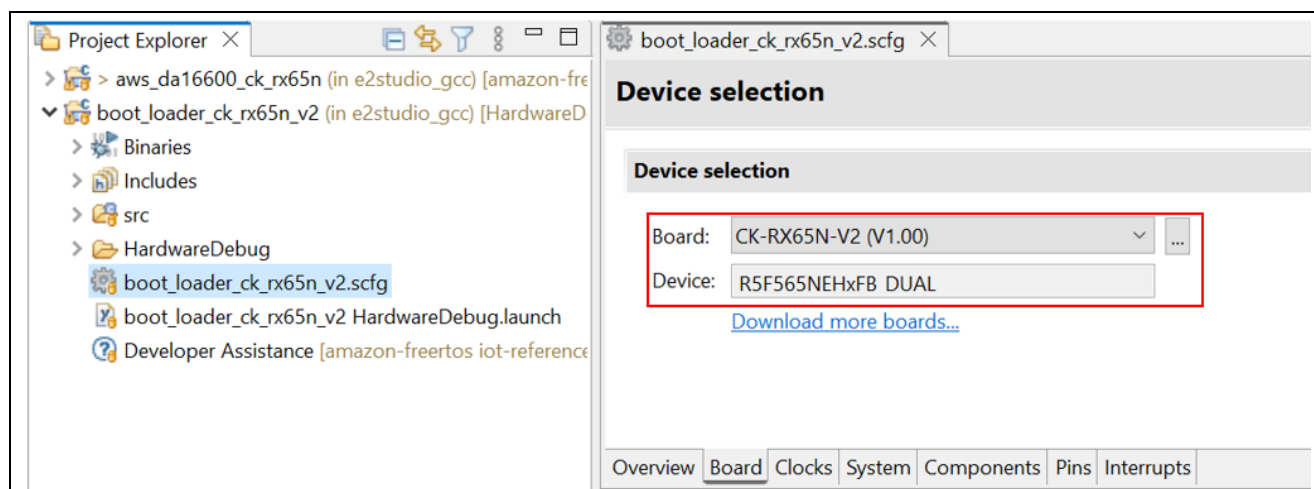


Figure 114. bootloader's device selection

Assign public key: Copy the contents of the secp256r1.publickey file you created in 6.4, and paste the contents into **CODE_SIGNER_PUBLIC_KEY_PEM** defined in the following files:
boot_loader_ck_rx65n_v2\src\key\code_signer_public_key.h

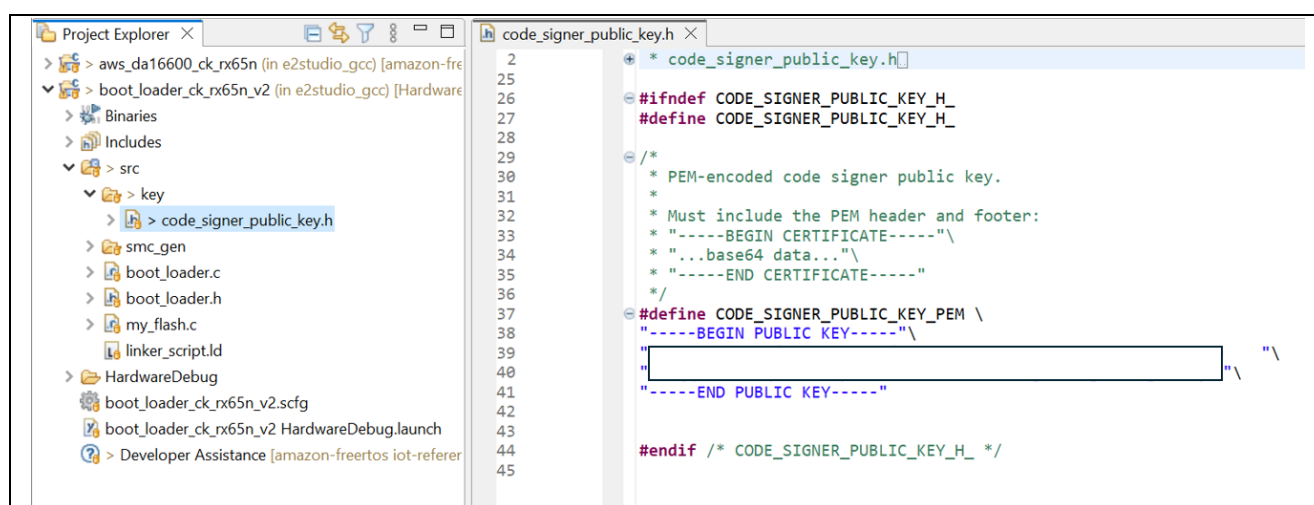


Figure 115. Add Code Signing Public Key to Boot Loader Project

2. Set **ENABLE_OTA_UPDATE_DEMO** to **1** (Enable) in `aws_da16600_ck_rx65n\src\rtos_config\demo_config.h`. (The default is 0)

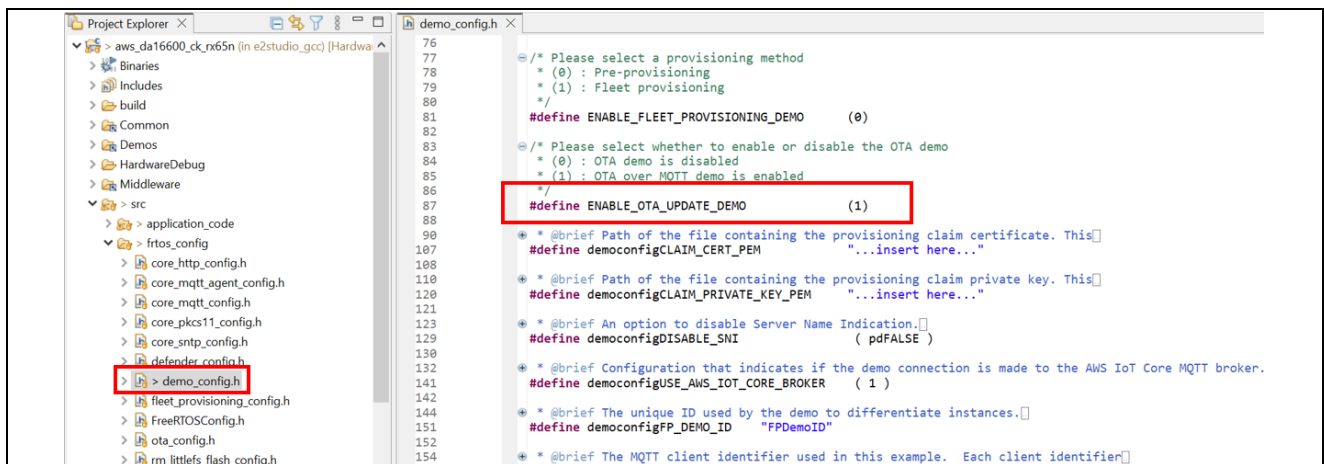


Figure 116. Enable OTA Demo

3. Confirm the initial project version:

Confirm the version definitions in `aws_da16600_ck_rx65n\src\rtos_config\demo_config.h`:

- APP_VERSION_MAJOR
- APP_VERSION_MINOR
- APP_VERSION_BUILD

Example: The initial project version is 0.9.2

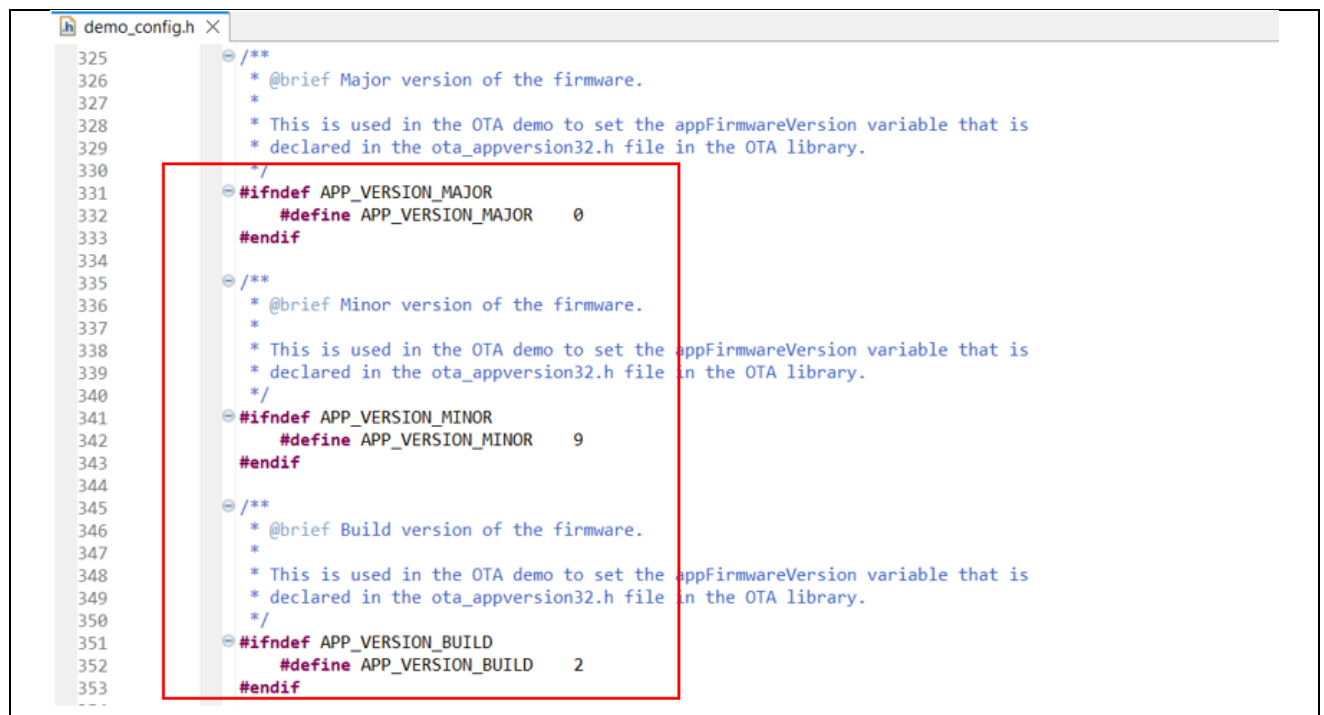


Figure 117 The initial project version 0.9.2

4. Checking the project environment settings:

For both projects, from the **Projects** menu, select **Properties**, expand the **C/C++ Build** menu, and click **Settings**.

On the **Tool Settings** tab, expand the **Objcopy** menu and select **General**. Confirm that the **Motorola S-record (srec)** is chosen for **OutFormat**.

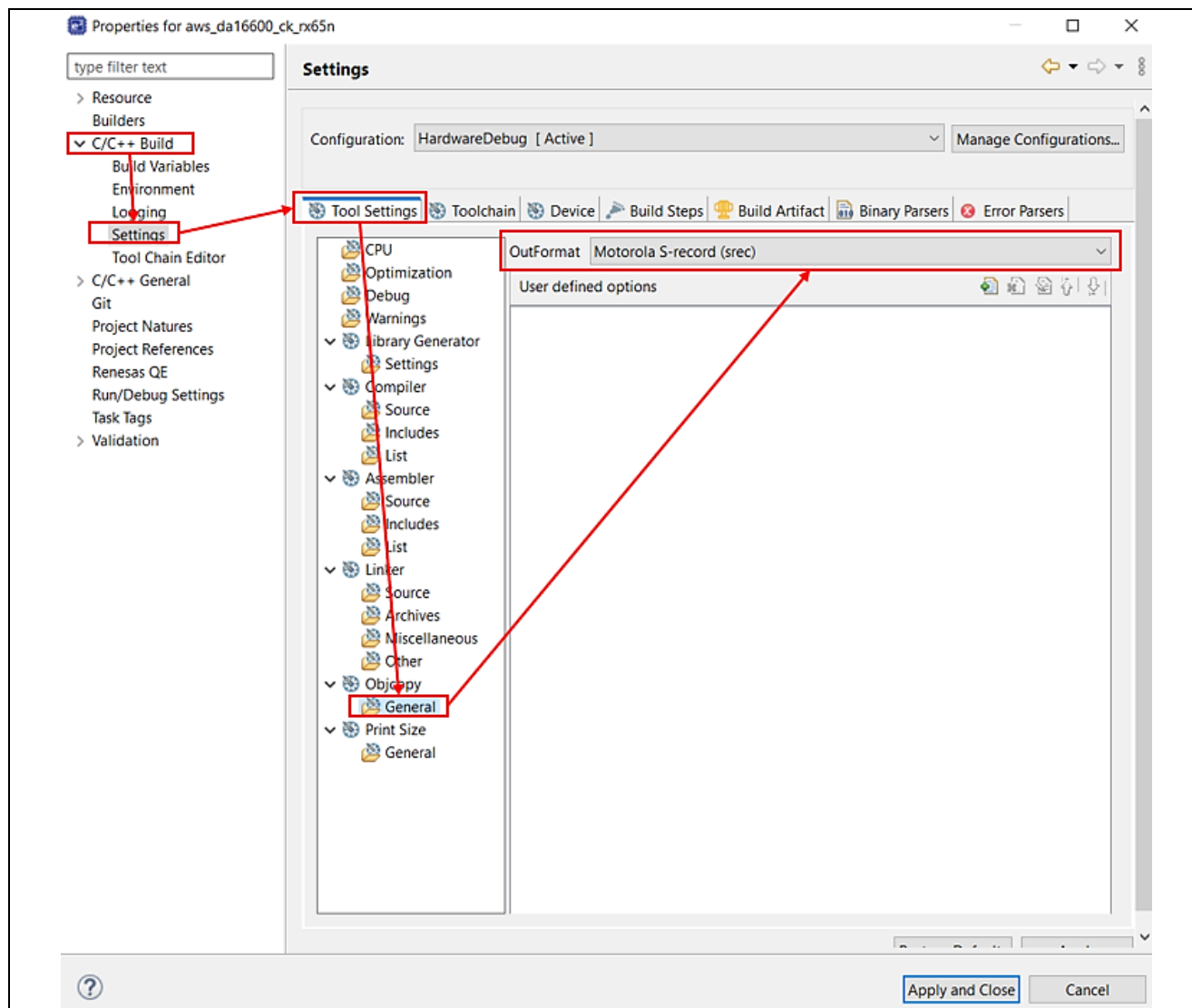


Figure 118. Output format

On the **Toolchain** tab, confirm that the toolchain is **GCC for Renesas RX**.

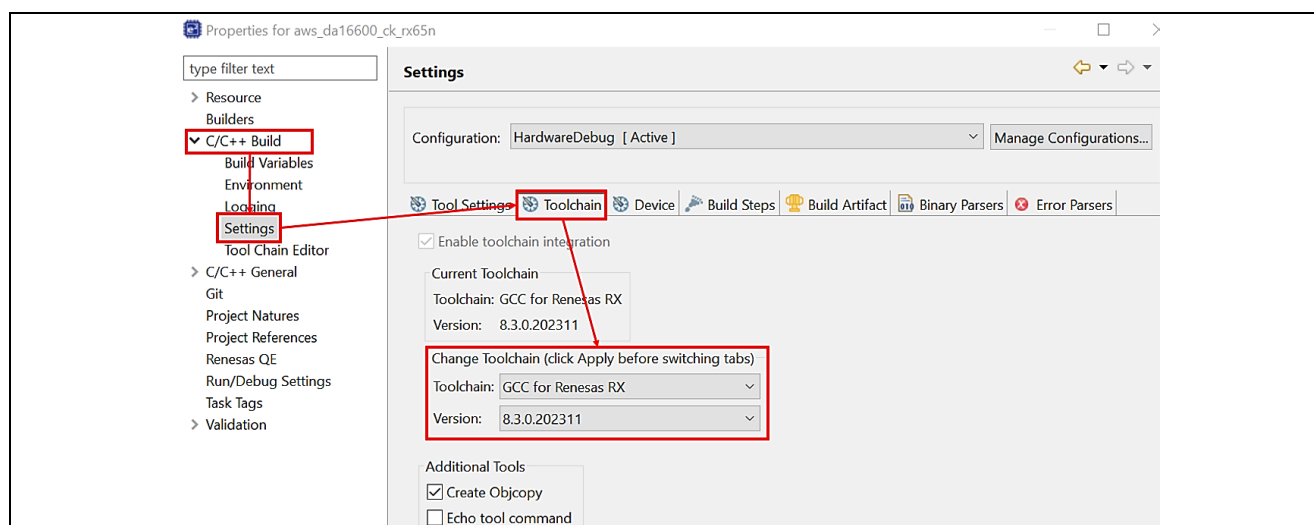


Figure 119. Project toolchain

5. Device selection setting

Open the file `aws_da16600_ck_rx65n.scfg` and click the **Board** tab. Click the ellipsis (...) beside the **Board** field in the **Device selection** area.

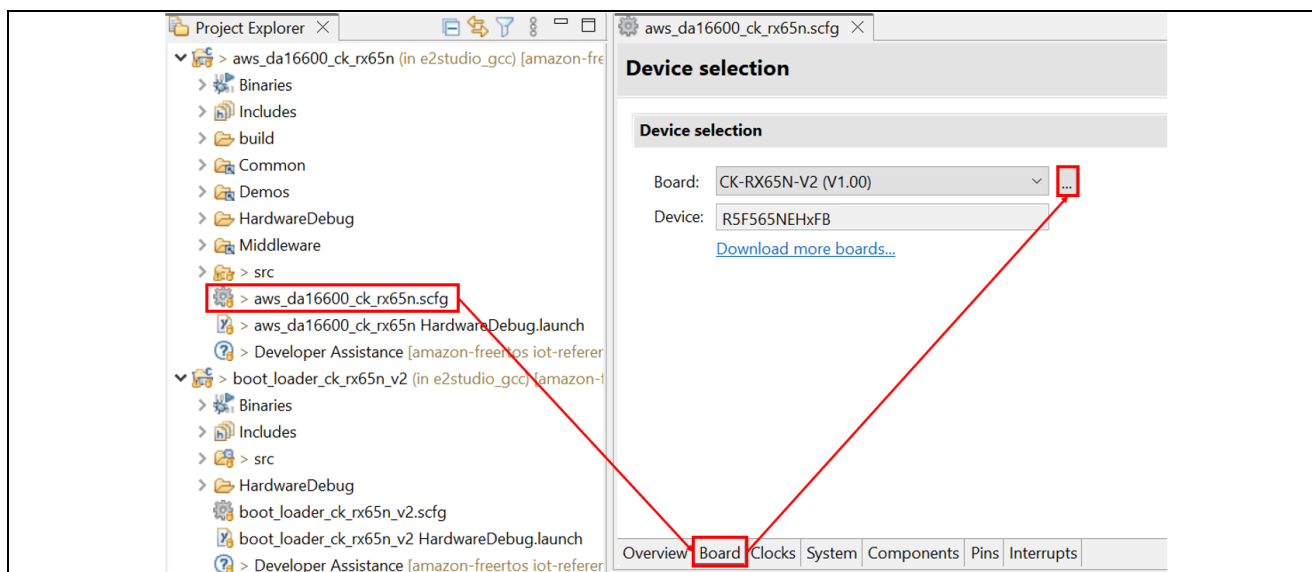


Figure 120. Device selection (1/3)

Select **“Target Board: CK-RX65N-V2”** and **“Bank Mode: Dual Bank”** then click **Next**

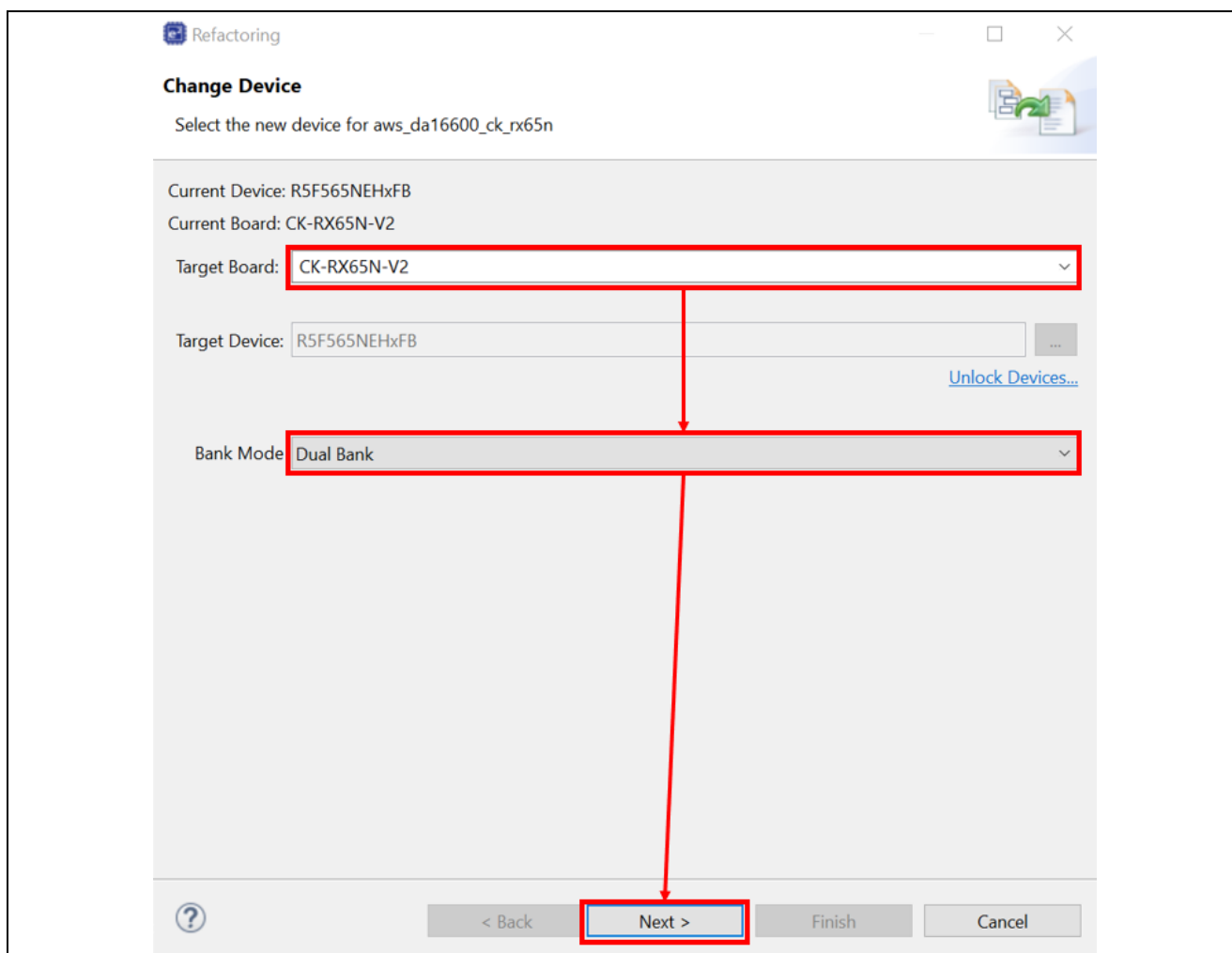


Figure 121. Device selection (2/3)

Under **Change Device for aws_da16600_ck_rx65n** > **Project Files**, clear the **src/linker_script.ld** check the box and then click **Finish**.

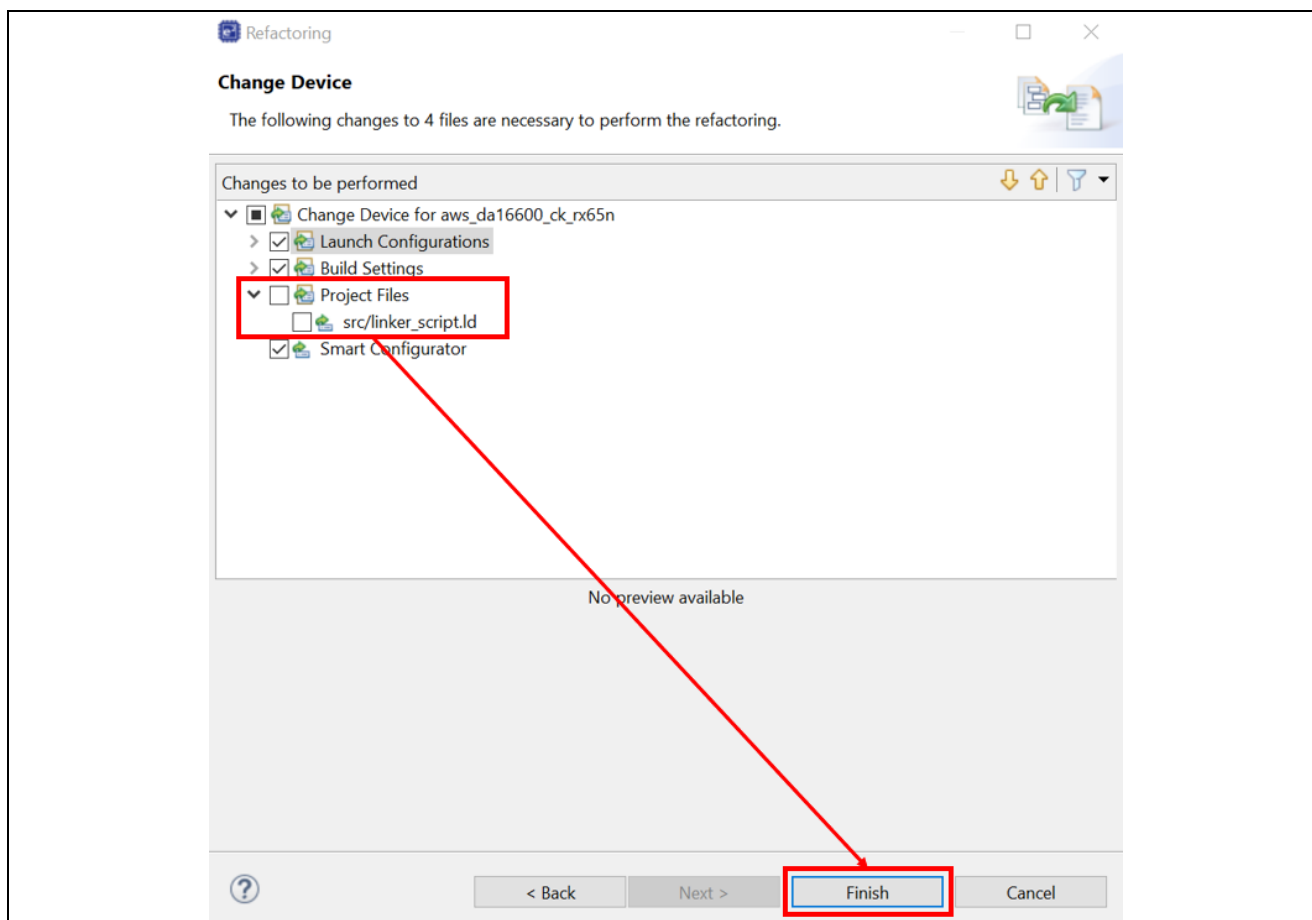


Figure 122. Device selection (3/3)

6. Change the firmware (aws_da16600_ck_rx65n) vector

Open the **aws_da16600_ck_rx65n\src\linker_script.ld** file then allocate **.exvectors** to **0xFFFFF80: AT(0xFFFFF80)** and **.fvectors** to **0xFFFFF80: AT(0xFFFFF80)**

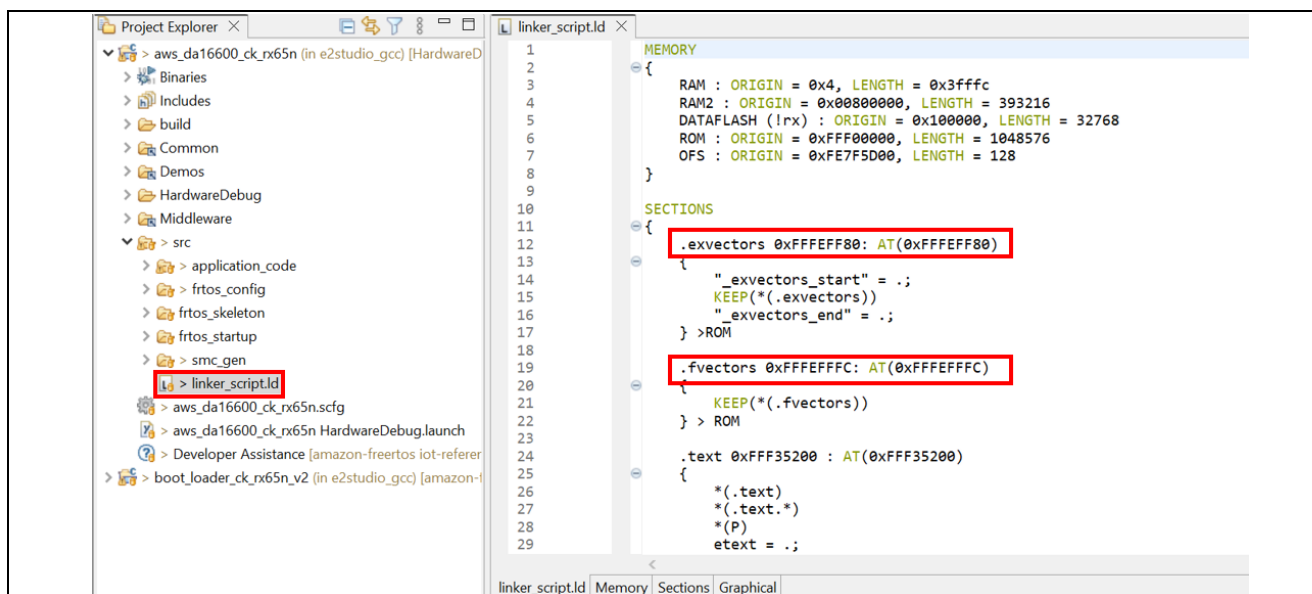


Figure 123. Vector setting

Build the project by choosing **Project** → **Build All** as instructed in **Figure 26**. **Build the Project**.

7. Generate the initial firmware

After users built projects successfully, please place the following files in the Renesas Image Generator folder:

- The results of building the firmware: **aws_da16600_ck_rx65n.mot**
- The results of building the boot loader: **boot_loader_ck_rx65n_v2.mot**
- The private key created in 6.4.1: **secp256r1.privatekey**

Open a command prompt, navigate to the Renesas Image Generator folder, and execute the following command to generate the file **userprog.mot**.

```
$ python image-gen.py -iup aws_da16600_ck_rx65n.mot -ip RX65N_DualBank_ImageGenerator_PRM.csv -o userprog -ibp boot_loader_ck_rx65n_v2.mot -key secp256r1.privatekey -vt ecdsa -ff RTOS
```

6.4.2.2 Running OTA project

1. Start Renesas Flash Programmer and create new project:

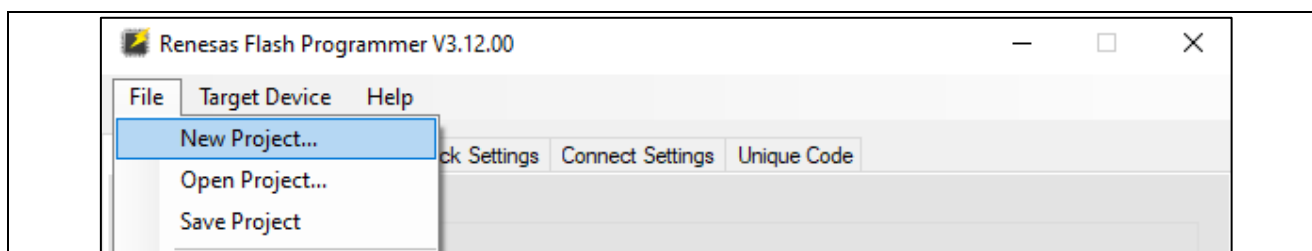


Figure 124. Create New Flash Project (1/4)

After that:

- Choose Microcontroller: **RX65x**
- Input project name.
- Browse “Project Folder”
- Communication: Tool: **E2 emulator Lite**
- Communication: Interface: **FINE**
- Choose “Connect”

Note: Jumper of J16 is in “Debug” mode.

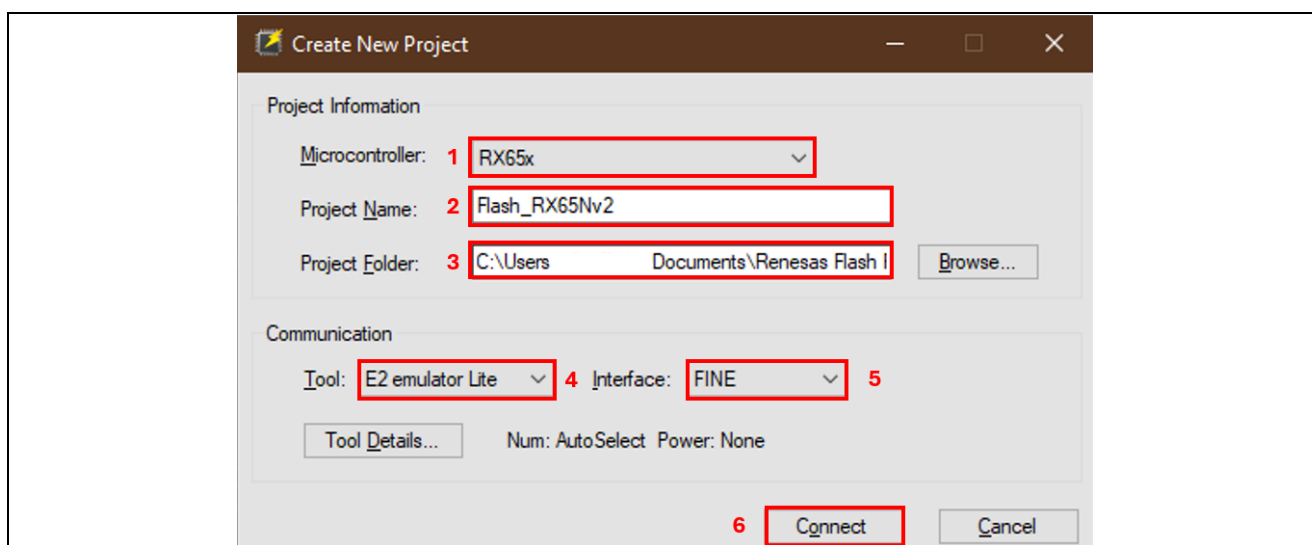


Figure 125. Create New Flash Project (2/4)

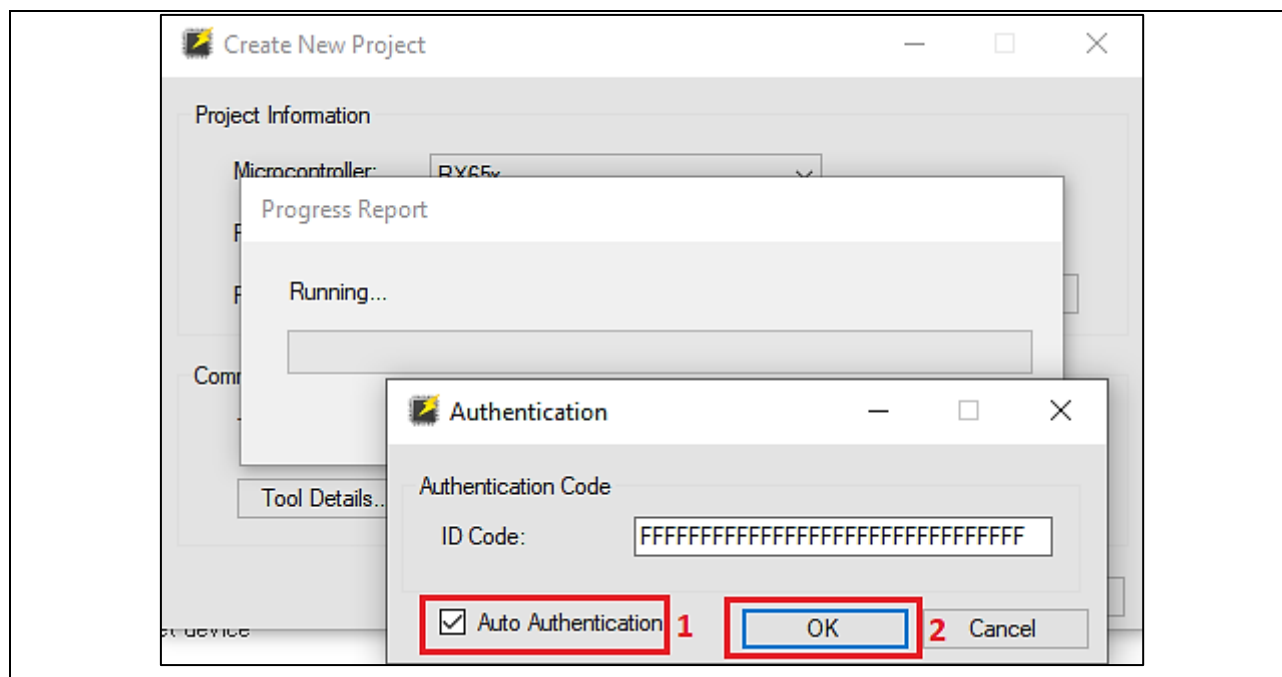


Figure 126. Create New Flash Project (3/4)

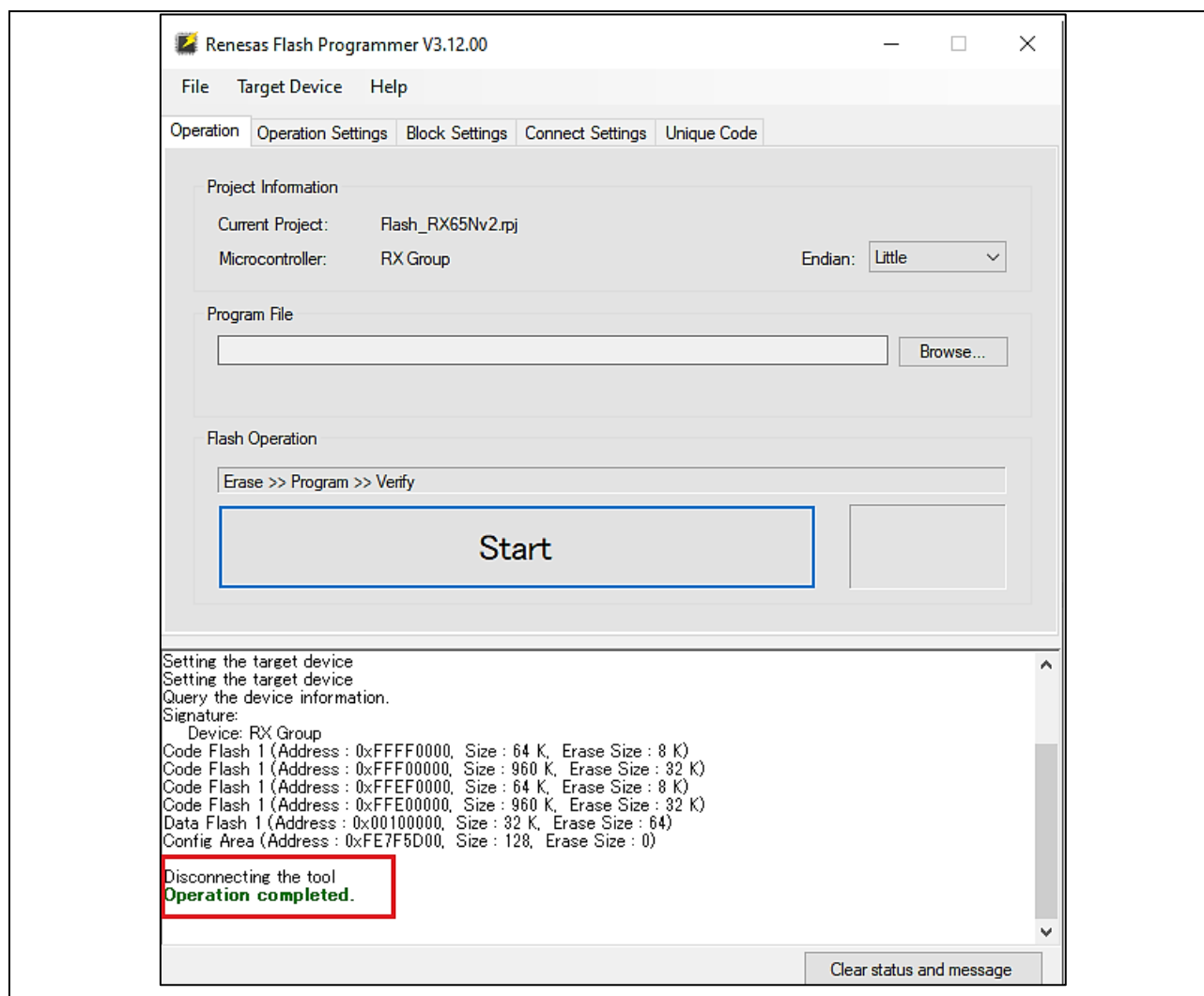


Figure 127. Create New Flash Project (4/4)

2. Choose code flash area in configuration:

Only tick the option “**Select**” for “**Code Flash 1**”.

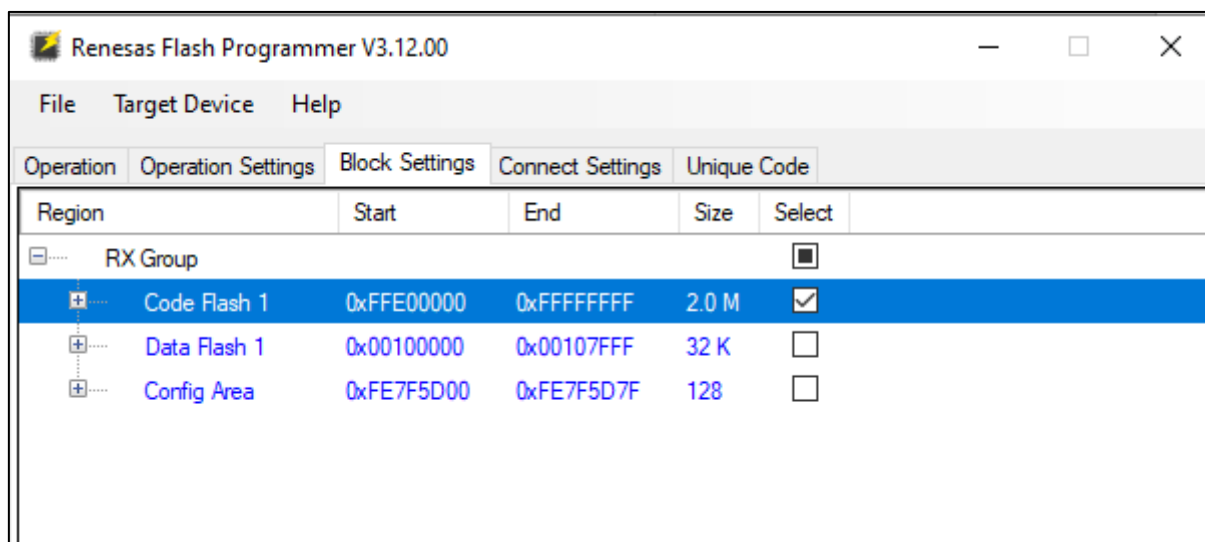


Figure 128. Choose Code Flash Area

3. Erase “Code Flash” before loading a new image:

Choose “**Operation Settings**” > “**Erase**”.

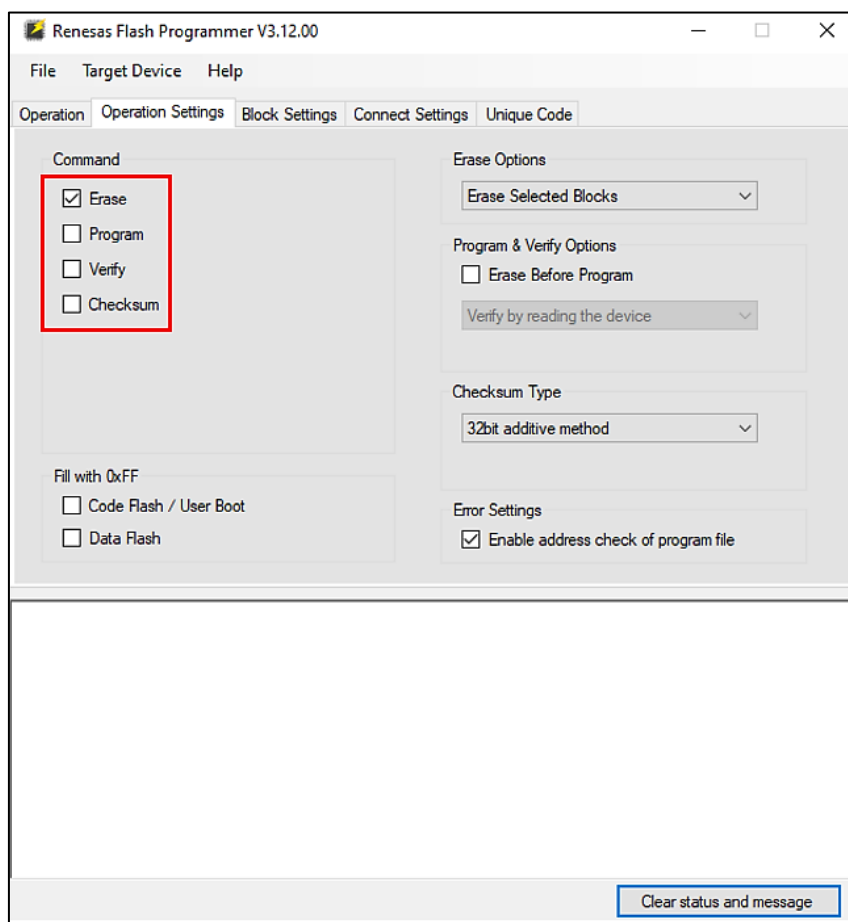


Figure 129. Erase Code Flash (1/2)

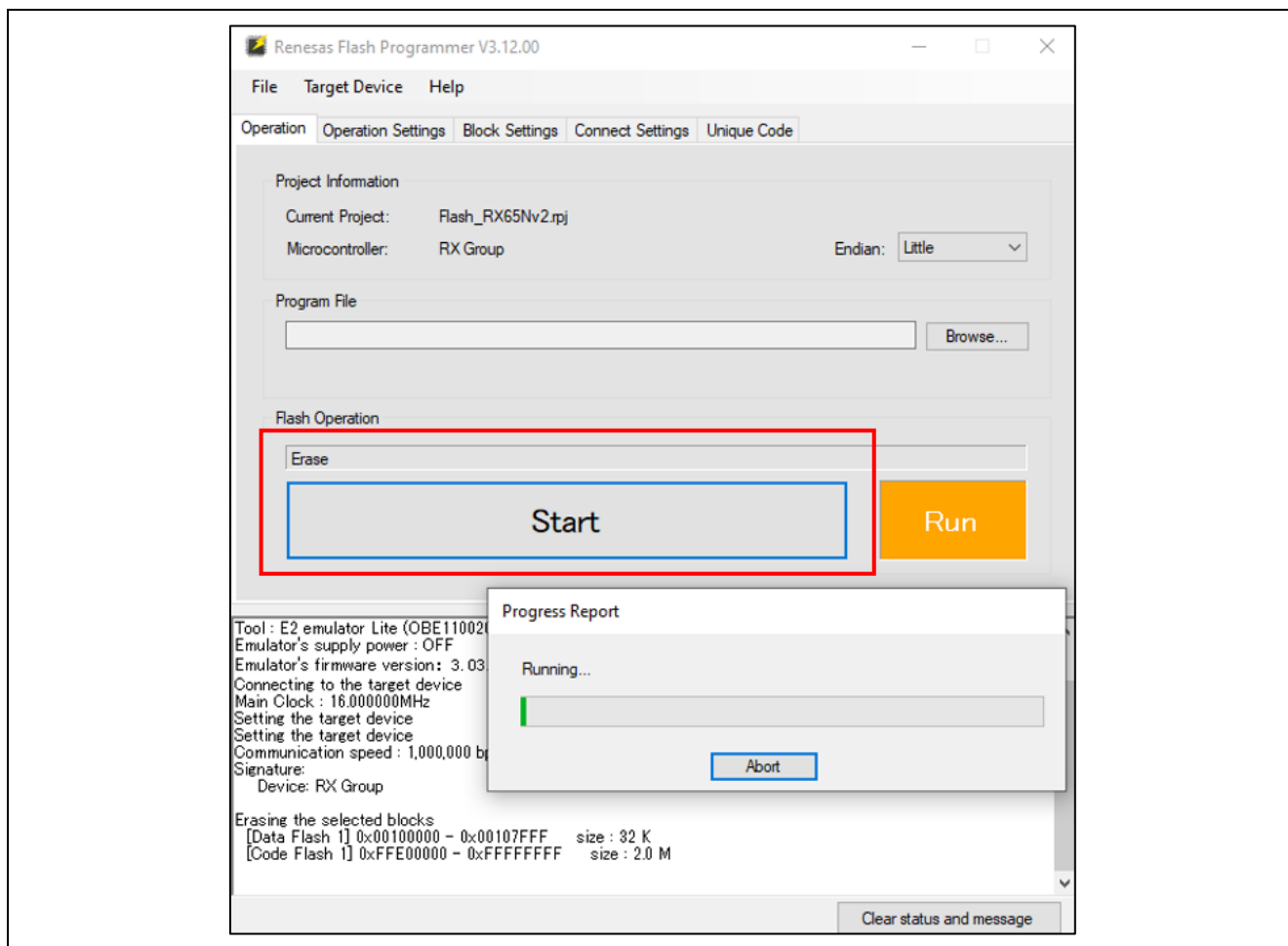


Figure 130. Erase Code Flash (2/2)

4. Write the initial firmware (userprog.mot)

This flash project will use commands: Erase, Program, and Verify.

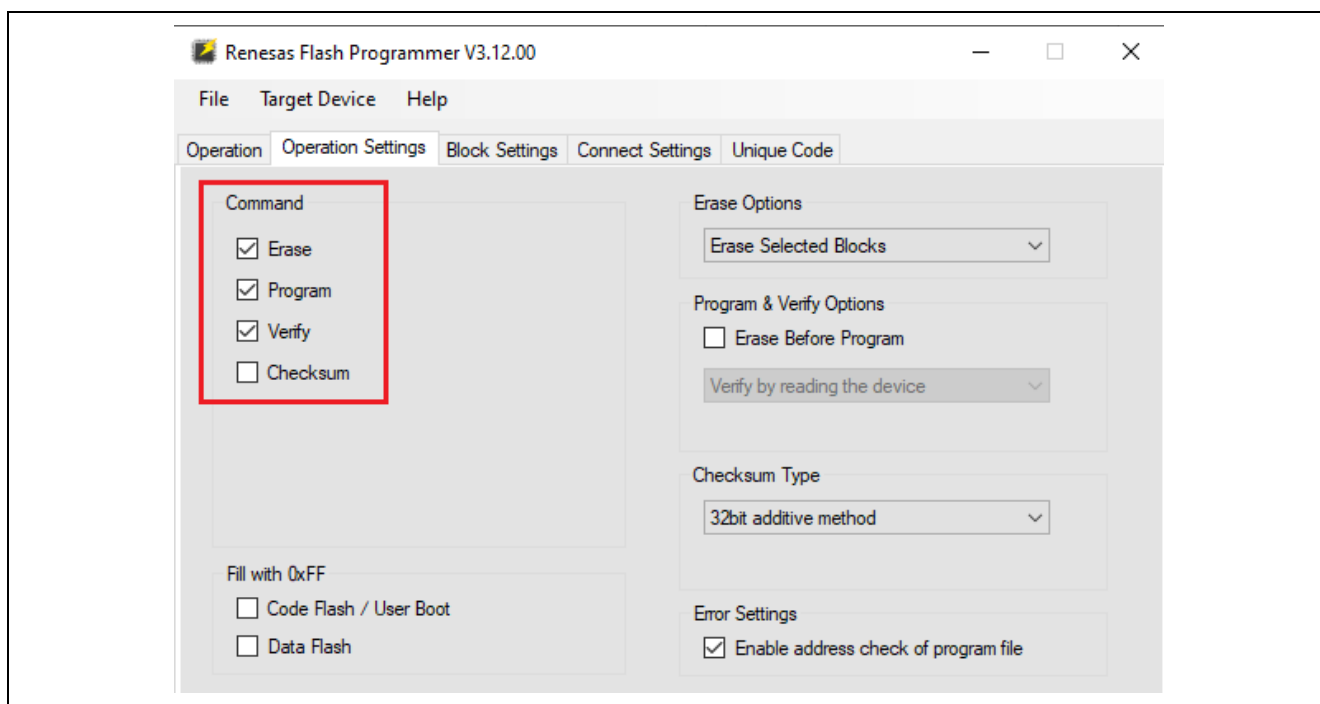


Figure 131. Flash the Firmware (1/3)

Add firmware's path (users have created in **Generate the initial firmware**) to "Program File" and click "Start":

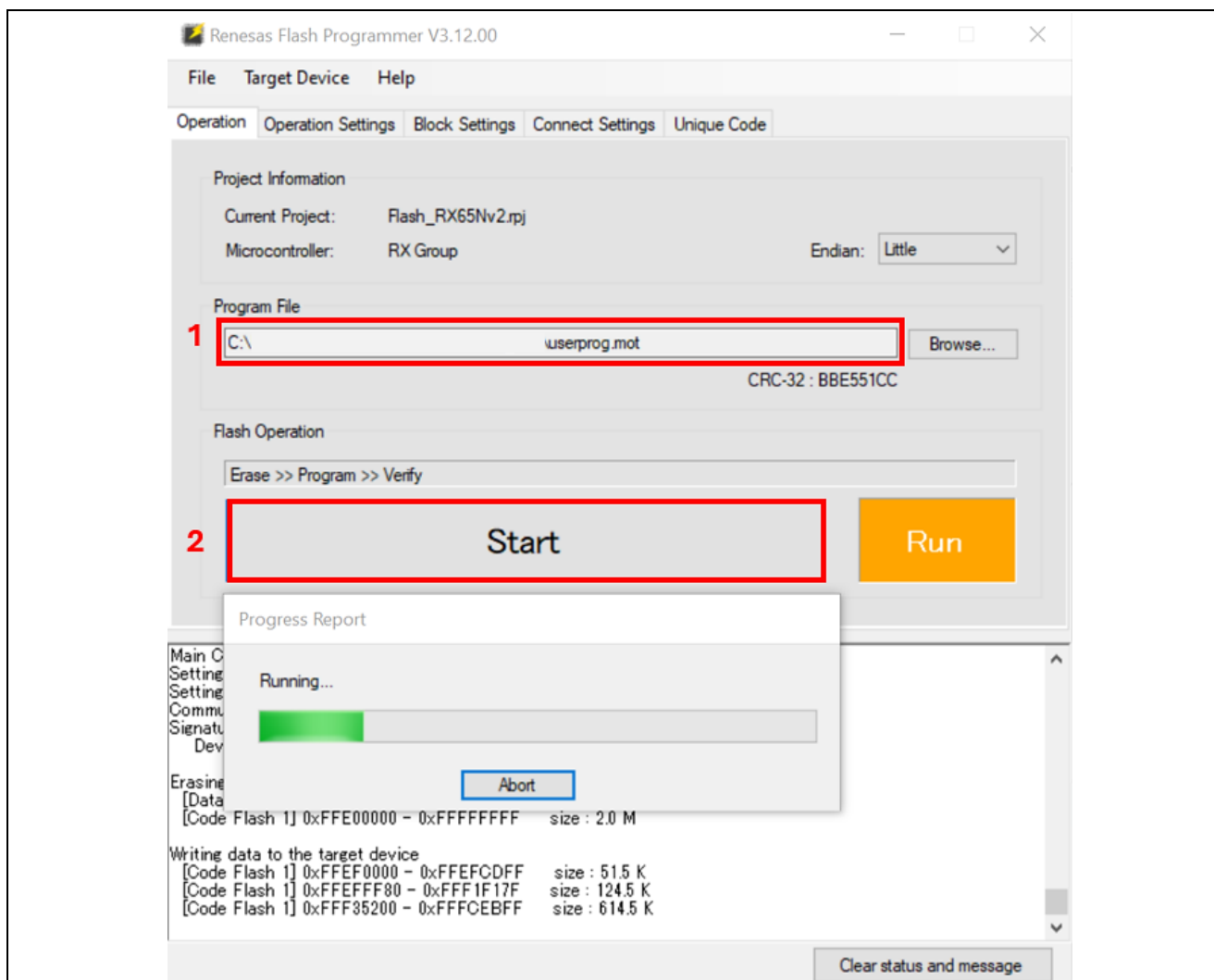


Figure 132. Flash the Firmware (2/3)

If it is successful, it will display "Operation completed":

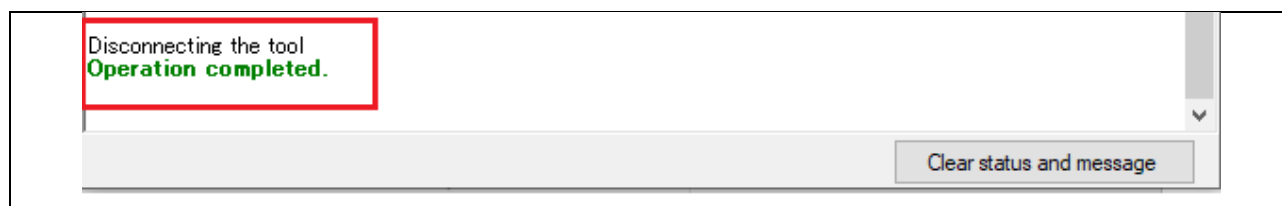


Figure 133. Flash the Firmware (3/3)

5. Running the application:

Set jumper of J16 is in "Run" mode. The application will run as below:

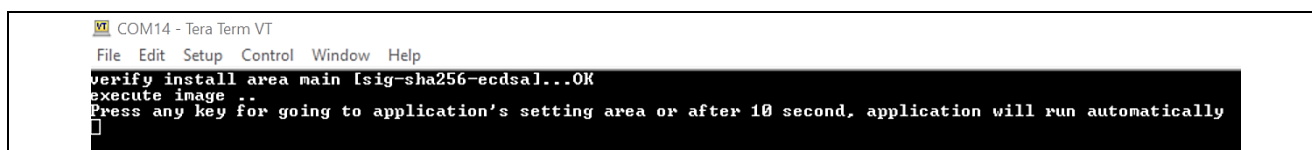
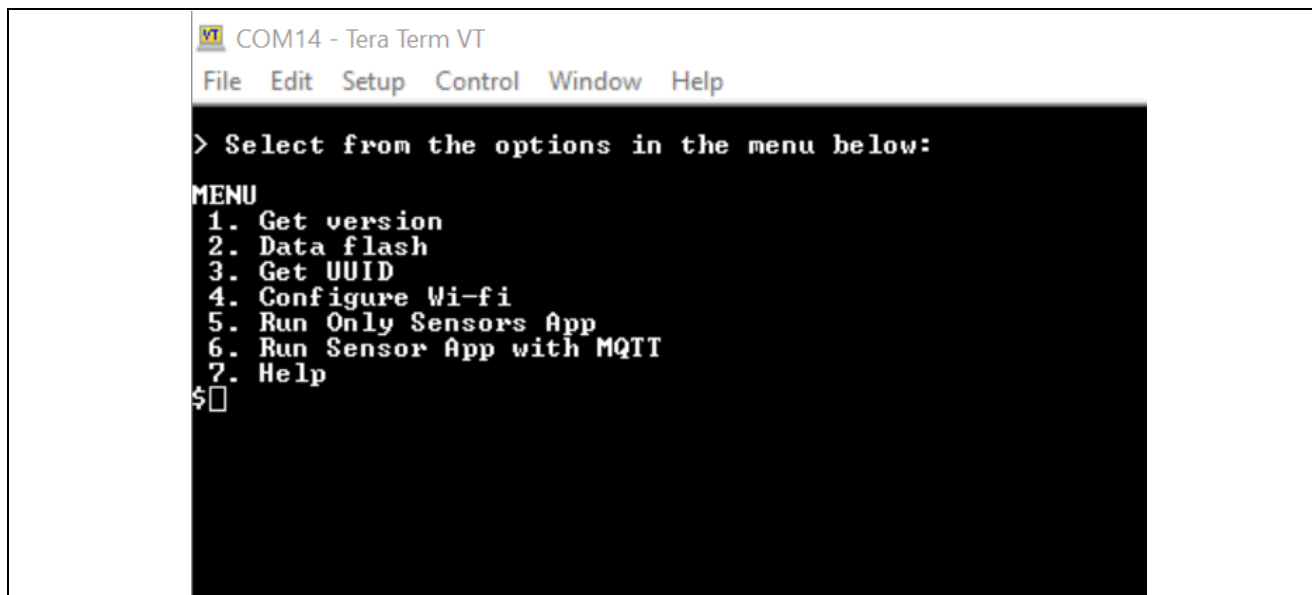


Figure 134. Start Application with OTA

Press any key to configure the application. For setting Cloud's credentials for IoT Devices, please refer to the **Starting the Application**

Besides that, for OTA, please store the code signing certificate to device (user created “secp256r1.crt” file at step: **6.4.1***Error! Reference source not found.*).

Press ‘2’ on the Main Menu to display Data Flash.

A screenshot of a Tera Term VT terminal window titled 'COM14 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The terminal displays a prompt '> Select from the options in the menu below:' followed by a 'MENU' section. The menu lists seven options: 1. Get version, 2. Data flash, 3. Get UUID, 4. Configure Wi-fi, 5. Run Only Sensors App, 6. Run Sensor App with MQTT, and 7. Help. A dollar sign prompt '\$' is visible at the bottom left of the terminal area.

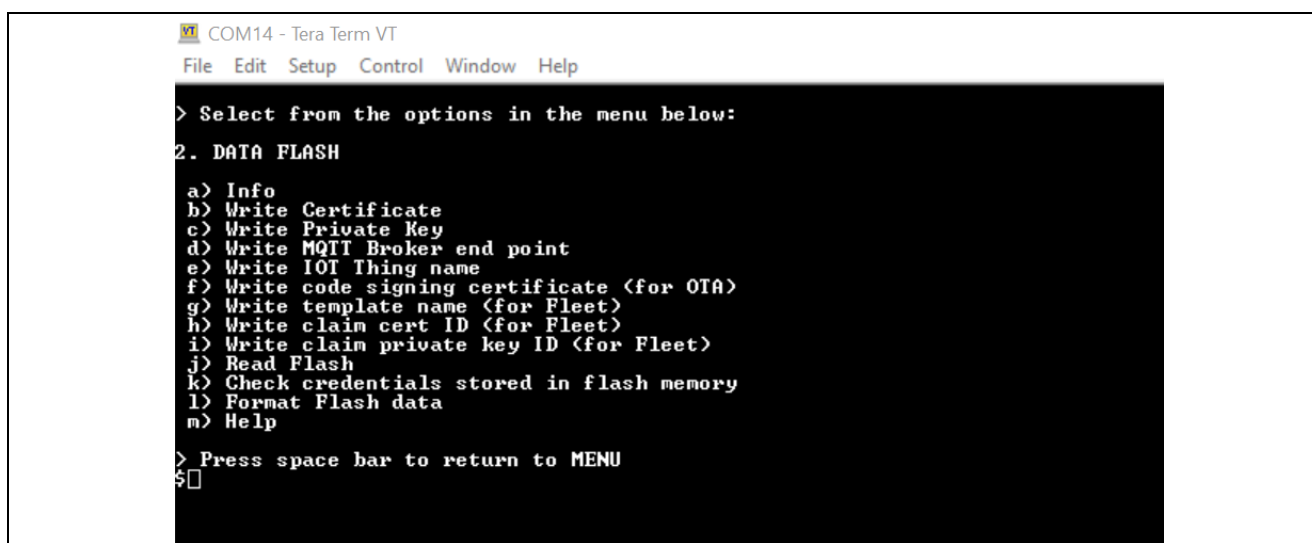
```
COM14 - Tera Term VT
File Edit Setup Control Window Help

> Select from the options in the menu below:

MENU
1. Get version
2. Data flash
3. Get UUID
4. Configure Wi-fi
5. Run Only Sensors App
6. Run Sensor App with MQTT
7. Help
$
```

Figure 135. Main Menu

Press ‘f’ for storing code signing certificate:

A screenshot of a Tera Term VT terminal window titled 'COM14 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The terminal displays a prompt '> Select from the options in the menu below:' followed by '2. DATA FLASH'. Below this, a list of options is shown, each preceded by a letter in parentheses: a) Info, b) Write Certificate, c) Write Private Key, d) Write MQTT Broker end point, e) Write IOT Thing name, f) Write code signing certificate <for OTA>, g) Write template name <for Fleet>, h) Write claim cert ID <for Fleet>, i) Write claim private key ID <for Fleet>, j) Read Flash, k) Check credentials stored in flash memory, l) Format Flash data, and m) Help. A prompt '> Press space bar to return to MENU' is shown at the bottom, followed by a dollar sign prompt '\$' and a cursor.

```
COM14 - Tera Term VT
File Edit Setup Control Window Help

> Select from the options in the menu below:

2. DATA FLASH

a) Info
b) Write Certificate
c) Write Private Key
d) Write MQTT Broker end point
e) Write IOT Thing name
f) Write code signing certificate <for OTA>
g) Write template name <for Fleet>
h) Write claim cert ID <for Fleet>
i) Write claim private key ID <for Fleet>
j) Read Flash
k) Check credentials stored in flash memory
l) Format Flash data
m) Help

> Press space bar to return to MENU
$
```

Figure 136. Data Flash Menu

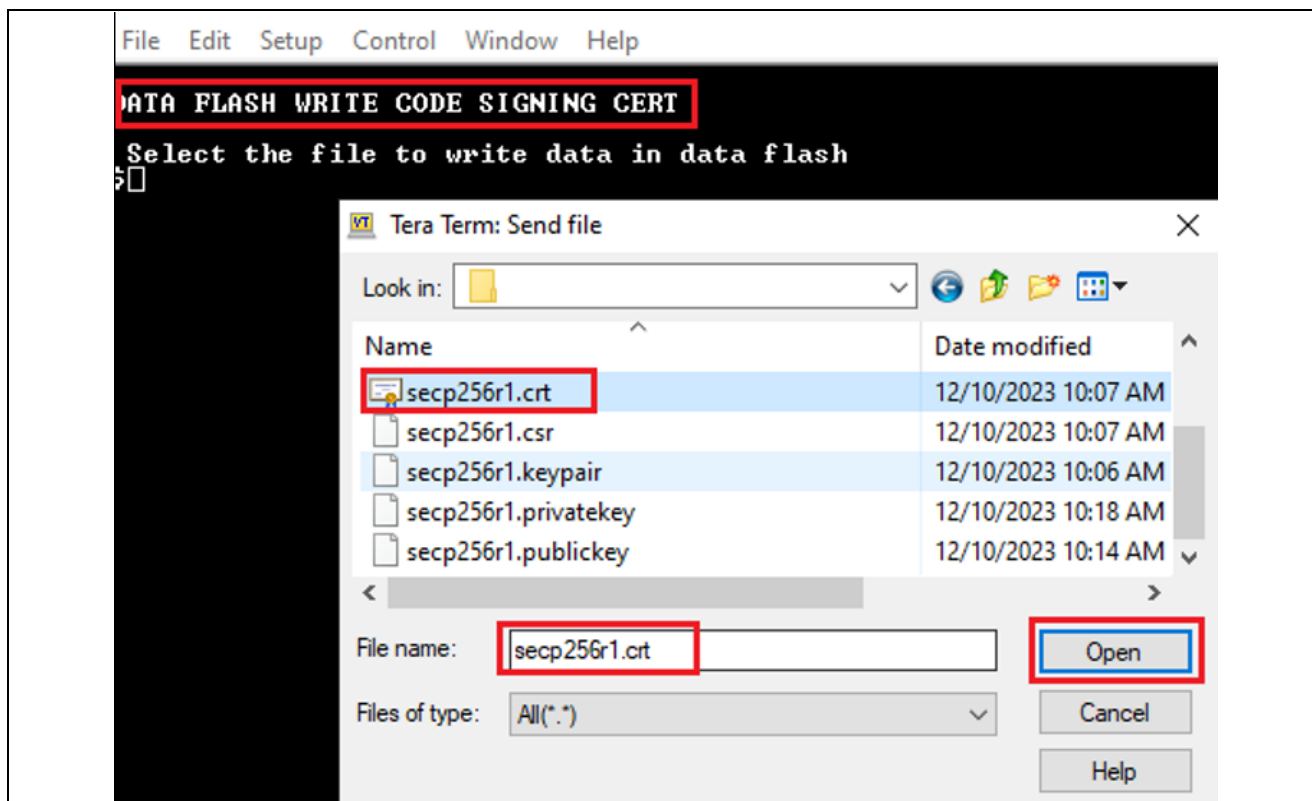


Figure 137. Store code signing certificate into flash (1/2)

Note: please check the EOL of secp256r1.crt and convert it to LF before saving it into data flash.

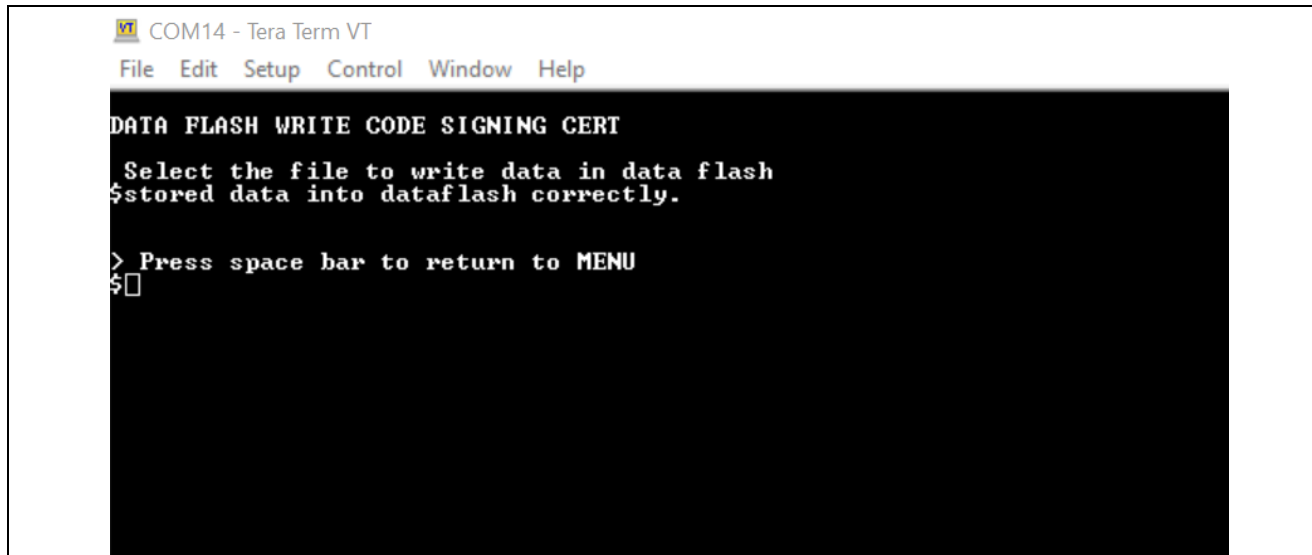


Figure 138. Store code signing certificate into flash (2/2)

After saving all Cloud credentials, and Wi-Fi credentials, the user starts the application by choosing option '6' on the Main Menu:

```

CHECK CREDENTIALS STORED IN DATA FLASH
Fleet is disabled, do not need Claim private key ID
Fleet is disabled, do not need Claim cert ID
Fleet is disabled, do not need template name
Code signing certificate saved in data flash is verified and successful
Wi-Fi's Security saved in data flash is verified and successful
Wi-Fi's Password saved in data flash is verified and successful
Wi-Fi's SSID saved in data flash is verified and successful
IOT thing name saved in data flash is verified and successful
MQTT Endpoint saved in data flash is verified and successful
Private Key saved in data flash is verified and successful
Certificate saved in data flash is verified and successful
All credentials in data flash is verified and successful
0 7178 [CLI] Write certificate...

** Alternate Key Provisioning successfully **
!!! Wi-Fi Init Successful !!!**

SSID: TP-Link_2502
Connecting to TP-Link_2502
Wi-Fi connected to SSID TP-Link_2502.
Device IP address: 192.168.50.27
Device network mask: 255.255.255.0
Device gateway address: 192.168.50.65
1 24898 [CLI] -----STARTING DEMO-----
2 24898 [MQTT] [INFO] -----Start MQTT Agent Task-----
3 24898 [MQTT] [INFO] Creating a TLS connection to .....us-east-1.amazonaws.com:8883.
4 24898 [MQTT] [INFO] Created new TCP socket.
5 27213 [MQTT] [INFO] Established TCP connection with .....us-east-1.amazonaws.com.
6 30342 [MQTT] [INFO] <Network connection 0x85879c> TLS handshake successful.
7 30342 [MQTT] [INFO] <Network connection 0x85879c> Connection to .....t.us-east-1.amazonaws.com established.
8 30342 [MQTT] [INFO] Creating an MQTT connection to the broker.
9 30999 [MQTT] [INFO] MQTT connection established with the broker.
10 30999 [MQTT] [INFO] Successfully connected to MQTT broker.
11 30999 [oh1203_thread] I2C bus 2 setup success
12 30999 [oh1203_thread]
OB1203 Device open success
13 30999 [OTA Demo Tal] [INFO] -----Start OTA Task-----
14 30999 [OTA Demo Tal] [INFO] OTA over MQTT demo, Application version 0.9.2
15 31009 [sensor_thread] I2C bus 0 setup success

```

Figure 139. Application with OTA (1/2)

While sub/pub message to MQTT (sensor's data, control LEDs, and so forth), the application will also check the event of OTA from Cloud to process with it.

```

12 30999 [oh1203_thread]
OB1203 Device open success
13 30999 [OTA Demo Tal] [INFO] -----Start OTA Task-----
14 30999 [OTA Demo Tal] [INFO] OTA over MQTT demo, Application version 0.9.2
15 31009 [sensor_thread] I2C bus 0 setup success
16 31010 [sensor_thread] HS3001 open sensor instance successful: 0
17 31010 [sensor_thread] ICP20100 open sensor instance successful: 0
18 31019 [zmod_thread] I2C bus 1 setup success
19 31021 [OTA Demo Tal] [INFO] Received: 0 Queued: 0 Processed: 0 Dropped: 0
20 31021 [AWS_DA16600] [INFO] -----Start AWS Wi-Fi DA16600 - MQTT Demo Task -----
21 31454 [zmod_thread] ZMOD4410 open sensor instance successful: 0
22 31884 [zmod_thread] ZMOD4510 open sensor instance successful: 0
23 31885 [zmod_thread] Task zmod4410 measurement Success:0
24 31913 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
25 32015 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
26 32019 [sensor_thread] ICM42605 open sensor instance successful: 0
27 33061 [OTA Demo Tal] [INFO] Received: 0 Queued: 0 Processed: 0 Dropped: 0
28 33113 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
29 33113 [AWS_DA16600] [Send Data] ZMOD4410-IAQ IUOC: 000.000
30 33113 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ETOH: 000.000
31 33113 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ECO2 : 000.000
32 33852 [zmod_thread] ZMOD4410 in stabilization:196609
33 34522 [OTA Agent T] [INFO] Subscribed to topic $aws/things/C.....TA/jobs/notify-next.

```

Figure 140. Application with OTA (2/2)

6.5 Updating the Firmware

6.5.1 Creating the updated firmware

6.5.1.1 Changing the firmware version

Change the firmware version to a higher version. (Example: Because the previous version is 0.9.2 (**Figure 117**), so the new version that we can choose is 0.9.3 or higher)

Repeat the build process, this time with 3 specified for the APP_VERSION_BUILD definition in `aws_da16600_ck_rx65n\e2studio_gcc\src\frtos_config\demo_config.h`.

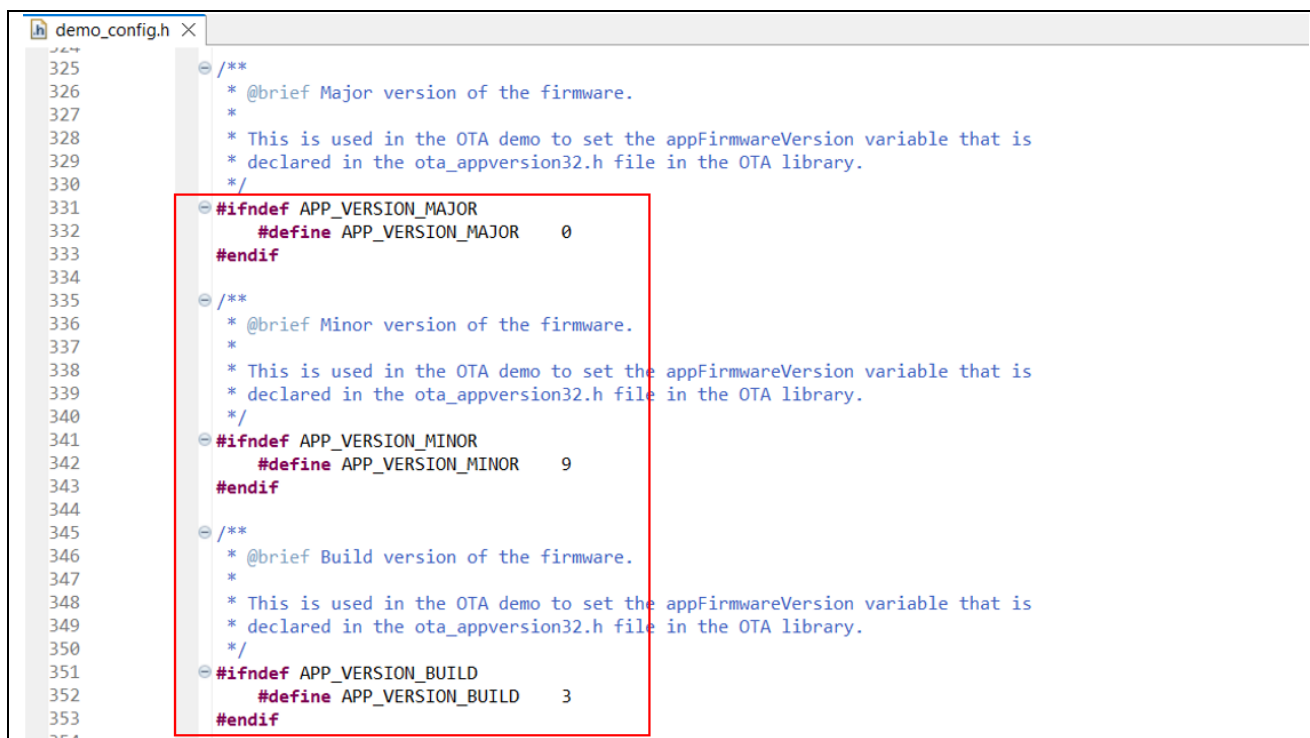


Figure 141. Setting New Version for Firmware

6.5.1.2 Use Renesas Image Generator to Generate the Updated Firmware

Overwrite the file in the Renesas Image Generator folder with the firmware you rebuilt in 6.5.1.1 (`aws_da16600_ck_rx65n.mot`), and then execute the following command at the command prompt:

```
$ python image-gen.py -iup aws_da16600_ck_rx65n.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o user_093 -key secp256r1.privatekey -vt
ecdsha -ff RTOS
```

This command generates a file named `user_093.rsu`.

6.5.2 Updating the firmware

In AWS, create an OTA update job that will update the firmware.

6.5.2.1 Creating New Job

In the IoT Core menu, select **Manage > Remote actions > Jobs**, and then click the **Create job** button.

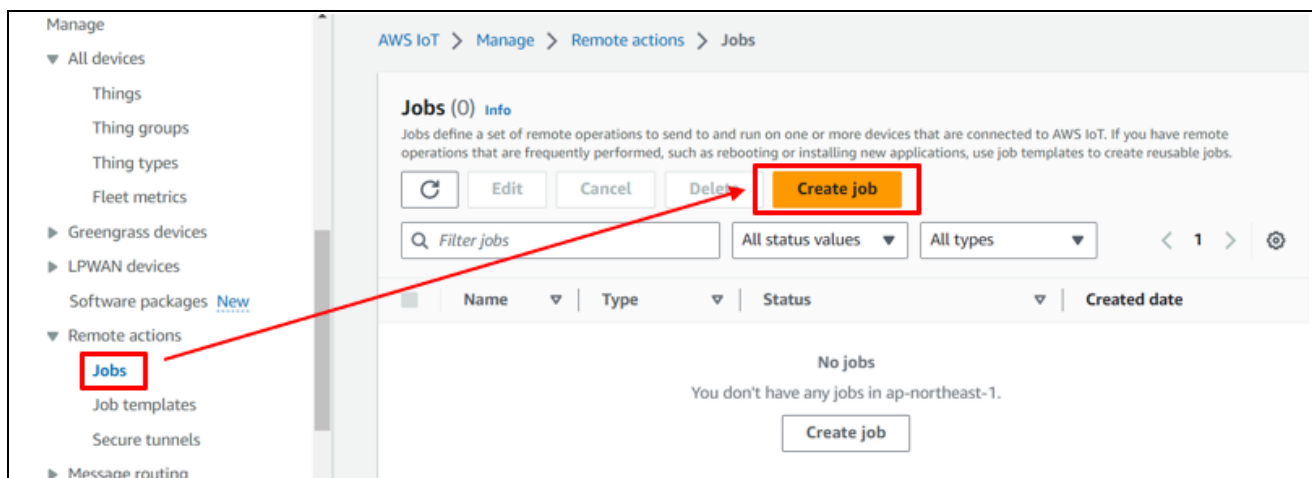


Figure 142. Create New Job for OTA (1/2)

6.5.2.2 Creating FreeRTOS OTA Job Update

Select Create FreeRTOS OTA update job and then click Next.

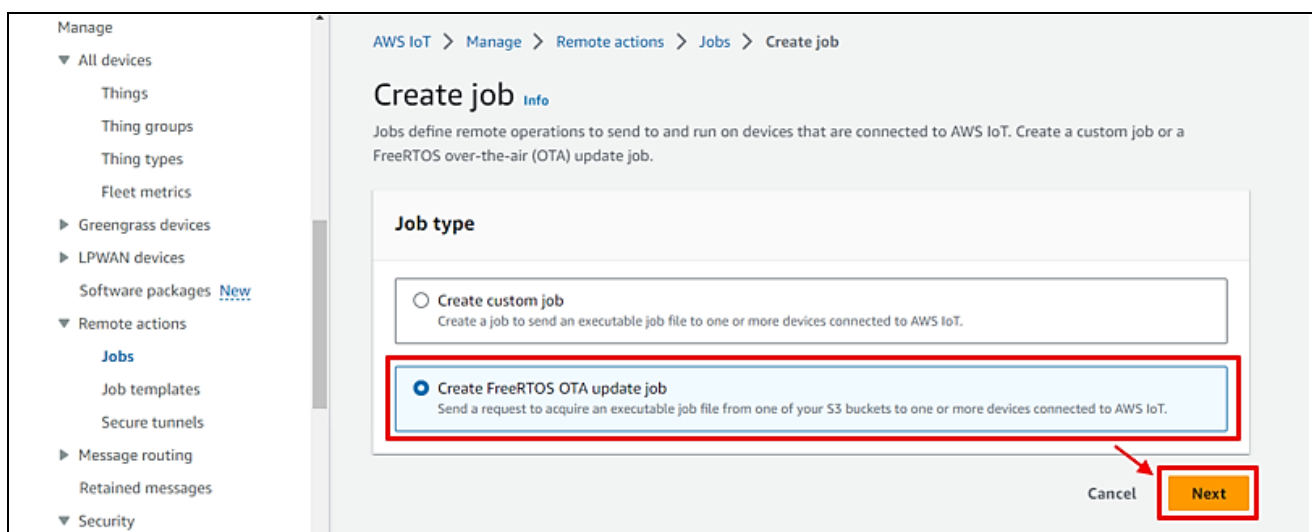


Figure 143. Create New Job for OTA (2/2)

6.5.2.3 Entering a Job Name

Enter a job name (example: rx65n_ota_demo_job) and then click **Next**.

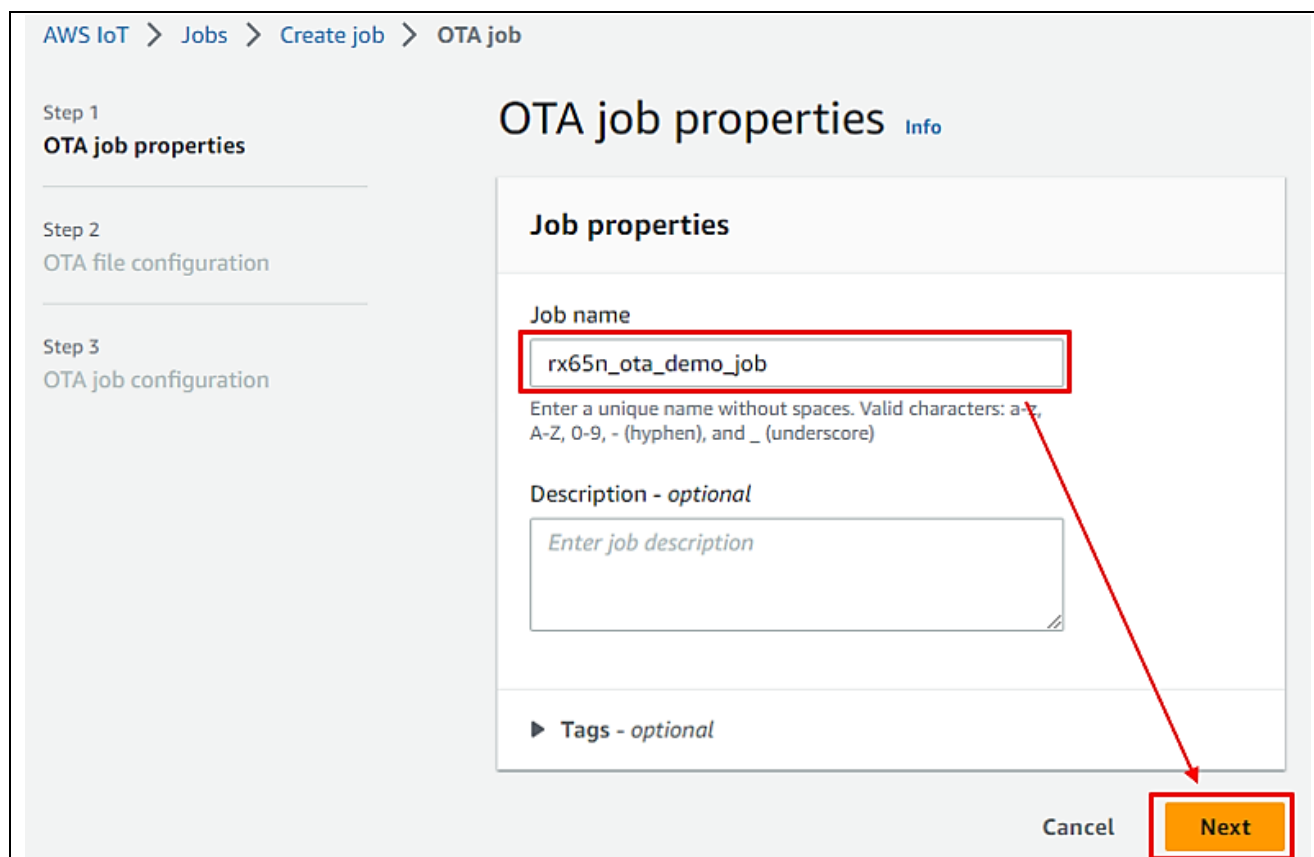


Figure 144. Enter OTA Job Name

6.5.2.4 Updating Devices

Click the Devices to update drop-down list and select the device to update.

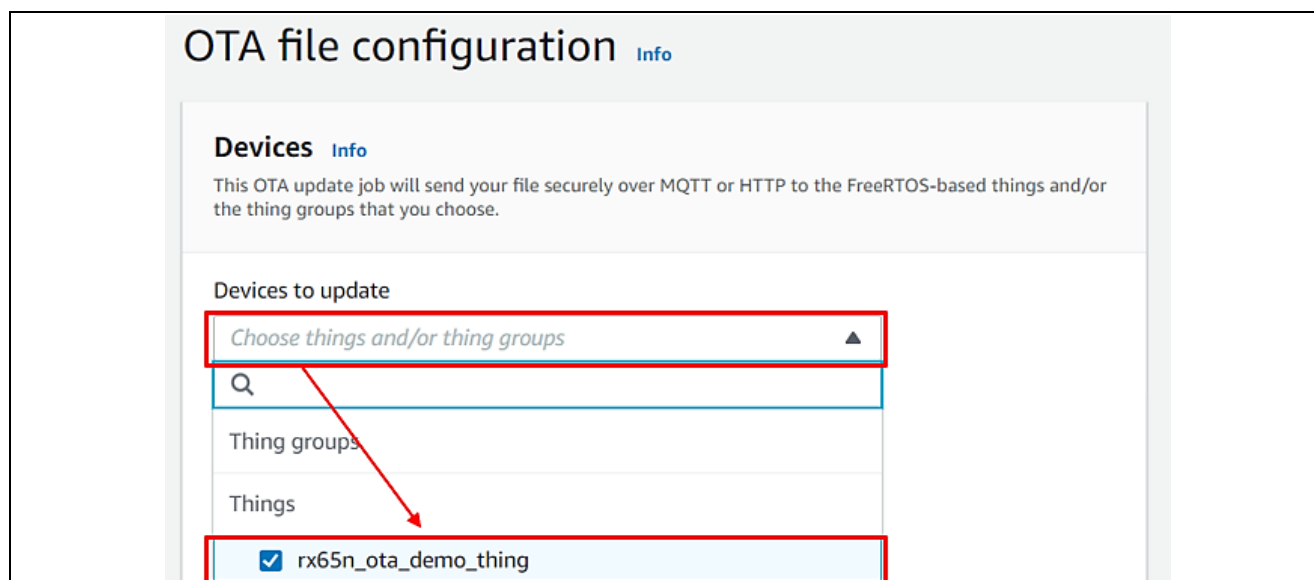
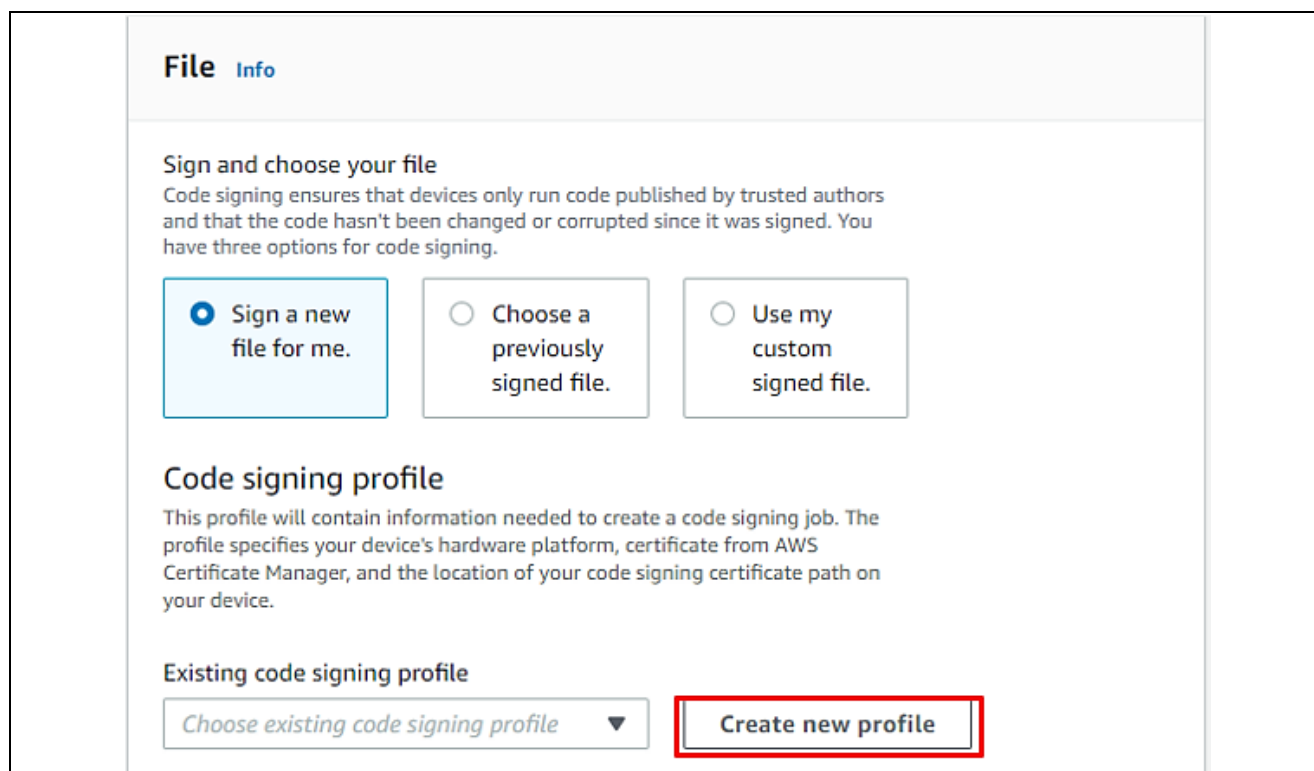


Figure 145. Choose Device to Update

6.5.2.5 Creating New Profile

Click **Create new profile**.



File Info

Sign and choose your file
Code signing ensures that devices only run code published by trusted authors and that the code hasn't been changed or corrupted since it was signed. You have three options for code signing.

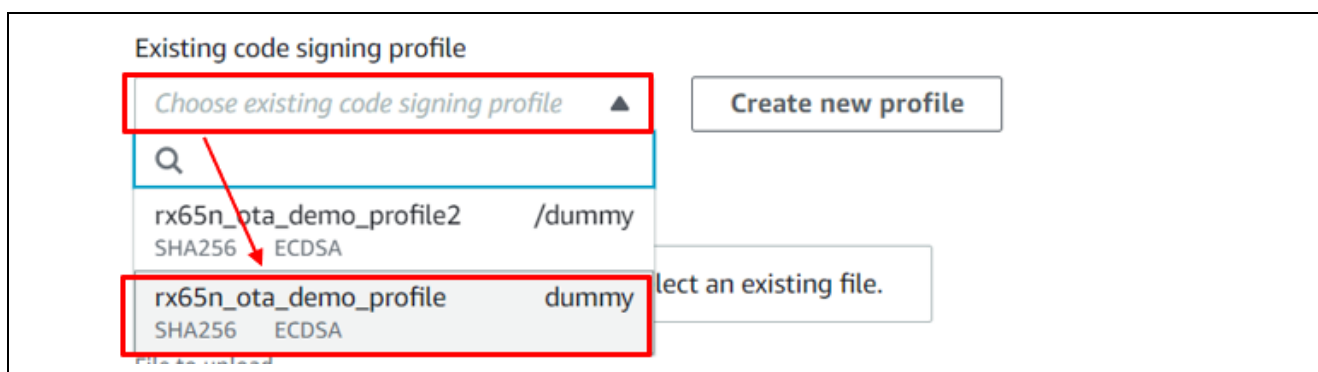
☒ Sign a new file for me. ☐ Choose a previously signed file. ☐ Use my custom signed file.

Code signing profile
This profile will contain information needed to create a code signing job. The profile specifies your device's hardware platform, certificate from AWS Certificate Manager, and the location of your code signing certificate path on your device.

Existing code signing profile
Choose existing code signing profile ▼ **Create new profile**

Figure 146. Create New Code Signing Profile

You can skip steps 6.5.2.5 to 6.5.2.9 if you have already created a profile. Click **Choose existing code signing profile** and select the profile you created from the drop-down list.



Existing code signing profile

Choose existing code signing profile ▲

Q

rx65n_ota_demo_profile2 /dummy
SHA256 ECDSA

rx65n_ota_demo_profile dummy
SHA256 ECDSA

Create new profile

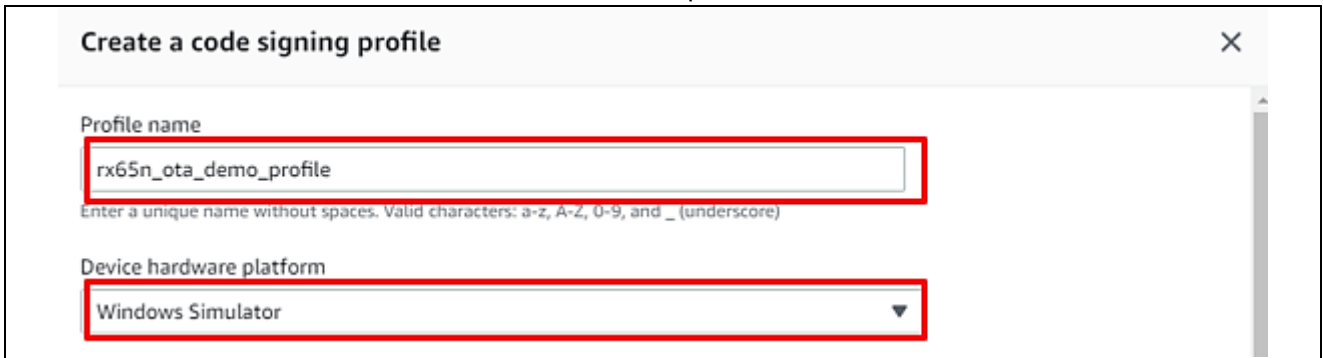
Select an existing file.

Figure 147. Choose Existing Code Signing Profile

6.5.2.6 Creating a Profile Naming

Create a profile (1): Profile name and device hardware platform:

- Enter the profile name (example: rx65n_ota_demo_profile)
- Select **Windows Simulator** as the device hardware platform!



The screenshot shows a dialog box titled "Create a code signing profile" with a close button (X) in the top right corner. Inside the dialog, there are two main sections. The first section is labeled "Profile name" and contains a text input field with the value "rx65n_ota_demo_profile". Below the input field, there is a small note: "Enter a unique name without spaces. Valid characters: a-z, A-Z, 0-9, and _ (underscore)". The second section is labeled "Device hardware platform" and contains a dropdown menu with "Windows Simulator" selected. Both the text input field and the dropdown menu are highlighted with red rectangles.

Figure 148. Create New Code Signing (1/2)

6.5.2.7 Creating a Profile: Importing Certificate

Create a profile (2): Import a certificate.

- In the **Code signing certificate** area, click **Import new code signing certificate**.
- In **Certificate body**, select the file secp256r1.crt.
- In **Certificate private key**, select the file secp256r1.privatekey
- In **Certificate chain**, select the file ca.crt.

Note: You have created the above files in 6.4.

- Click **Import**

Create a code signing profile

Code signing certificate
AWS Certificate Manager (ACM) handles the complexity of creating, managing, or importing SSL/TLS certificates. You can use ACM to create an ACM Certificate or import a third-party certificate that you use for signing. You must have a certificate to sign code.

☒ Import new code signing certificate ☐ Select an existing certificate

Certificates

Field	File Name	Size	Status
Certificate body	secp256r1.crt	753 bytes	Uploaded
Certificate private key	secp256r1.privatekey	232 bytes	Uploaded
Certificate chain - optional	ca.crt	890 bytes	Uploaded

Import

Path name of code signing certificate on device
This is the name and location of the certificate that your FreeRTOS device firmware uses to perform OTA image signature verification.

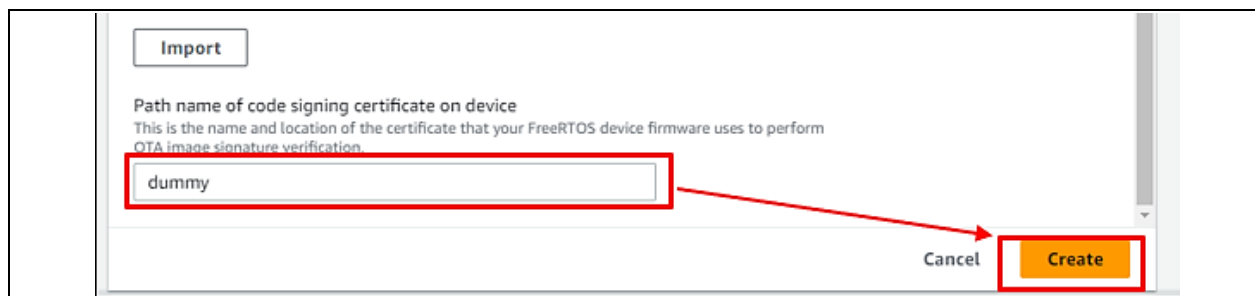
Cancel **Create**

Figure 149. Create New Code Signing (2/2)

6.5.2.8 Creating a Profile: Entering Path

Create a profile (3): Enter the path of the code signing certificate of the device and then click **Create**.

You can enter any path. (Example: dummy)



Import

Path name of code signing certificate on device
This is the name and location of the certificate that your FreeRTOS device firmware uses to perform OTA image signature verification.

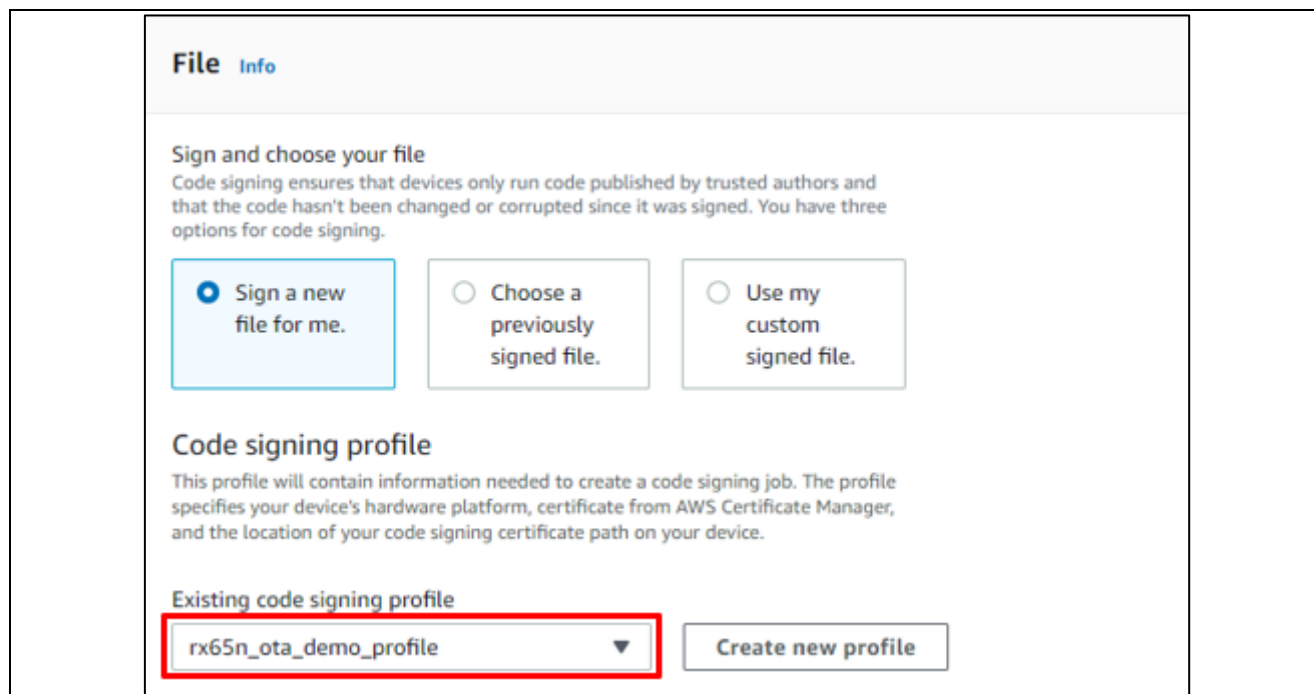
dummy

Cancel Create

Figure 150. Enter the Path of the Code Signing Certificate of the Device

6.5.2.9 Confirming Profile Name

Confirm that the name of the profile you created earlier is selected in the Existing code signing profile drop-down list.



File Info

Sign and choose your file
Code signing ensures that devices only run code published by trusted authors and that the code hasn't been changed or corrupted since it was signed. You have three options for code signing.

☒ Sign a new file for me. ☐ Choose a previously signed file. ☐ Use my custom signed file.

Code signing profile
This profile will contain information needed to create a code signing job. The profile specifies your device's hardware platform, certificate from AWS Certificate Manager, and the location of your code signing certificate path on your device.

Existing code signing profile
rx65n_ota_demo_profile ▼

Create new profile

Figure 151. Choose Existing Code Signing Profile

6.5.2.10 Updating the Firmware

- Select **Upload a new file**.
- In **File to upload**, select the file `user_093.rsu` you created in 6.5.1.2.
- Click **Browse S3** and select the S3 bucket you created in 6.3.
- Enter a path name in **Path name of file on device** (You can enter any path name. Example: `/device/updates`).

File

☒ Upload a new file. ☐ Select an existing file.

File to upload

Choose file

`user_093.rsu`
427776 bytes

File upload location in S3

This is the location in S3 where your file will be stored.

S3 URL

Format: `s3://bucket/prefix/object`.

Path name of file on device

This is the name and location where the file will be stored on the FreeRTOS device.

► File type - optional

Figure 152. Choose Firmware to Update

6.5.2.11 Choosing the Role for the Job

In the role drop-down list, select the role you created in 6.3 and then click **Next**.

IAM role [Info](#)

Role

Choose a role that grants AWS IoT access to S3, AWS IoT jobs, and AWS Code signing resources.

Figure 153. Choose Role for Creating Job

Click **Create job**.

OTA job configuration [Info](#)

Job run type
Choose how to run this job.

☒ Your job will complete after deploying to the devices and groups that you chose (snapshot)

☐ Your job will continue to deploy to any devices added to the groups that you chose (continuous)

► **Job start rollout configuration - optional**
Specify how quickly devices will be notified when a pending job starts.

► **Job stop configuration - optional**
These configurations define when to automatically stop the job. The job stops if a percentage of devices fail the deployment after a minimum number have deployed. The job cancels if any of the criteria are met after the job starts.

► **Job run timeout configuration - optional**
Specify how long the job will run.

Cancel Back **Create job**

Figure 154. Create Job

6.5.2.12 Waiting until Firmware Reception is Complete

Wait until firmware reception is complete.

When the job starts, the job receives and writes the firmware.

The Received counter is incremented when the reception starts.

```
3770 878989 [OTA Agent T] [INFO] Current State=[WaitingForFileBlock], Event=[RequestFileBlock], New
3771 880087 [OTA Demo Ta] [INFO] Received: 181 Queued: 181 Processed: 181 Dropped: 0
3772 880262 [MQTT] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
3773 880262 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
3774 880262 [MQTT] [INFO] Received data message callback, size 4121.
```

Figure 155. OTA Job is Processed

When the update process is complete, the device resets and the initial menu appears.

```
4054 983994 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
4055 983994 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
4056 984450 [OTA Demo Ta] [INFO] Received: 185 Queued: 185 Processed: 185 Dropped: 0
verify install area main [sig-sha256-ecdsa]...OK
execute image ..
Press any key for going to application's setting area or after 10 second, application will run automatically
Overtime for setting, will run application now
Running sensor app with MQTT...
```

Figure 156. Device Reset and Update New Firmware

6.5.2.13 Confirming that the Firmware Version is a New Version

Example: 0.9.3 (updated at 6.5.1.1)

The application will run with new firmware and run self-test mode with it.

```

13 36788 [OTA Demo Tal] [INFO] -----Start OTA Task-----
14 36788 [OTA Demo Tal] [INFO] OTA over MQTT demo, Application version 0.9.3
15 36798 [sensor_thre] I2C bus 0 setup success
16 36799 [sensor_thre] HS3001 open sensor instance successful: 0
17 36799 [sensor_thre] ICP20100 open sensor instance successful: 0
18 36799 [OTA Demo Tal] [INFO] Received: 0   Queued: 0   Processed: 0   Dropped: 0
19 36808 [zmod_thread] I2C bus 1 setup success
20 36810 [AWS_DA16600] [INFO] -----Start AWS Wi-Fi DA16600 - MQTT Demo Task -----
21 36824 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
22 37251 [zmod_thread] ZMOD4410 open sensor instance successful: 0

```

Figure 157. Firmware with New Version is Updated

```

60 43557 [OTA Agent T] [INFO] In self test mode.
61 43557 [OTA Agent T] [INFO] New image has a higher version number than the current image: New image version=0.9.3, Previous image version=0.9.2
62 43557 [OTA Agent T] [INFO] Image version is valid: Begin testing file: File ID=0
63 43557 [OTA Agent T] [INFO] Testing.
64 43693 [MQTT] [INFO] Publishing message to $aws/things/CanFit-Update-OTA/jobs/APR-OTA-RR-12-Jan-093/update.
65 44303 [MQTT] [INFO] Ack packet deserialized with result: MQTISuccess.
66 44303 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
67 44313 [MQTT] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTISuccess.
68 44313 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
69 44313 [MQTT] [WARN] Received an unsolicited publish from topic $aws/things/(.....)/update/accepted
70 44317 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/(.....)/update to broker.
71 44317 [OTA Agent T] [INFO] Job parsing success: OtaJobParseErr_t=OtaJobParseErrNone, Job name=APR-OTA-.....
72 44317 [OTA Agent T] [WARN] Received an unhandled callback event from OTA Agent. event = 6
73 44317 [OTA Agent T] [INFO] Current State=[CreatingFile], Event=[ReceivedJobDocument], New state=[CreatingFile]
74 44317 [OTA Agent T] [INFO] Beginning self-test.
75 44317 [OTA Agent T] [INFO] Received OtaJobEventStartTest callback from OTA Agent.
76 44419 [OTA Agent T] [INFO] Accepted and committed final image.

```

Figure 158 Test and accept new firmware

When OTA updates the firmware successfully, the status will be “Succeeded” as below:

The screenshot shows the AWS IoT console interface. On the left, there is a sidebar with navigation options: 'MQTT test client', 'Device Location', 'Manage', 'All devices', 'Things', 'Thing groups', 'Thing types', 'Fleet metrics', 'Greengrass devices', 'LPWAN devices', 'Software packages', 'Remote actions', 'Jobs', 'Job templates', and 'Secure tunnel'. The 'Jobs' option is selected. The main panel displays the 'Job executions' tab for a specific job. Under 'Execution overview', there is a table showing the status of the job: 'Succeeded' (1), 'Failed' (0), 'Canceled' (0), and 'Rejected' (0). Below this, there is a section for 'Job executions (1)' with a search bar and a dropdown menu.

Figure 159. Job's Status in AWS Portal

7. Fleet Provisioning

7.1 Overview Fleet Provisioning

This section describes the steps of using Fleet Provisioning in this application.

Fleet provisioning is a procedure in which provisioning takes place when each IoT device is started for the first time.

Generally speaking, it can be implemented in either of the following two ways.

1. Provisioning by claim (approach using provisioning claim certificates)
2. Provisioning by a trusted user (mobile or web app user, etc.)

In addition, either of the following two procedures can be used to obtain the individual certificates and private keys used for fleet provisioning.

A) Having the AWS certification authority generate a new individual certificate and private key and send it to the device (CreateKeysAndCertificate).

B) Generating a key pair on the device internally and sending a certificate signature request (CSR) to AWS to have them generate only an individual certificate and send it to the device (CreateCertificateFromCsr).

This document describes the implementation of a fleet provisioning that combines 1. And B).

Advantages

- The device's private key never leaves the device.
- There is no need to establish a connection between the manufacturing plant and AWS IoT.
- There is no need to put in place a structure for issuing individual certificates or registering devices.

On the other hand, it also has the following disadvantages. It is necessary to be aware of both the advantages and the disadvantages when using this provisioning method.

Disadvantages

- It is necessary to take into account the possibility that the provisioning claim certificate could leak to an unauthorized party.
- It is necessary to implement functionality on the device to issue a provisioning request and receive a response.

For detail about Fleet Provisioning, please refer to chapters 3, and 4 of the AN: [RX Family Provisioning Procedure for IoT Devices Rev.1.00 \(renesas.com\)](#) (Demo application for Cellular + Ethernet).

7.2 Setting up AWS for Fleet Provisioning

It is necessary to configure AWS settings to run the fleet provisioning demo.

1. Policy settings
2. Generating a claim certificate and claim key pair
3. Creating a fleet provisioning template

7.2.1 Policy Settings

Follow the steps below to create AWS IoT Core policies. The first policy you create will be used when fleet provisioning is run.

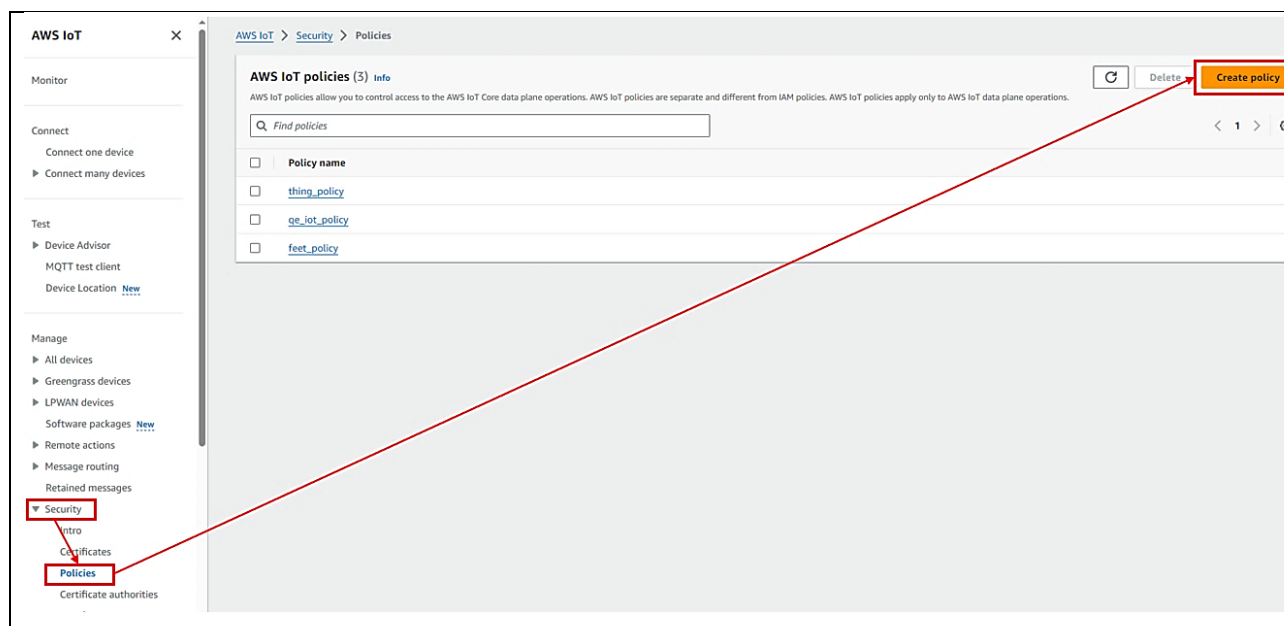


Figure 160. Creating an AWS IoT Policy (1/2)

In the Policy name field, enter the policy name of your choice.

Create policy Info

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

Policy properties

Policy name

Policy/Name

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore(_), plus sign(+), equal sign(=), and at sign(@) characters, but no spaces.

Tags - optional

Policy document Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Builder JSON

Policy document

Copy and paste the text of the policy document here.

JSON Line 1, Column 1 Errors: 0 Warnings: 0

Cancel Create

Figure 161. Creating an AWS IoT Policy (2/2)

Click the JSON button to display the policy document input field, then copy and paste the policy document shown in **Table 8. Policy** into the input field. When copying and pasting the policy document in **Table 8. Policy**, make the following changes:

- Change “us-east-1” to match the region used.
- Change <account id> to your account ID (account ID is the 12-digit number after @ that is displayed by clicking on the account name in the upper right corner, excluding the hyphen)

Table 8. Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive",
        "iot:RetainPublish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:<account id>:topic/$aws/certificates/create-from-csr/*",
        "arn:aws:iot:us-east-1:<account id>:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:<account id>:topicfilter/$aws/certificates/create-from-csr/*",
        "arn:aws:iot:us-east-1:<account id>:*"
      ]
    }
  ]
}
```

7.2.2 Generating a Claim Certificate and Claim Key Pair

Generate a provisioning claim certificate and provisioning claim key pair for use in fleet provisioning.

Select **Security** → **Certificates** and then click **Add certificate** → **Create certificate**.

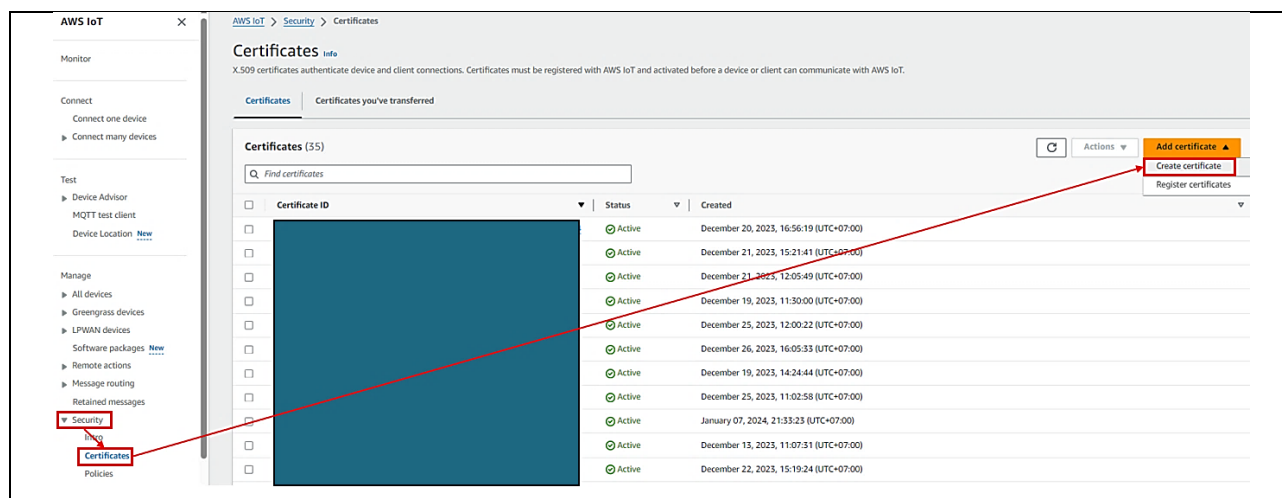


Figure 162. Create a Certificate

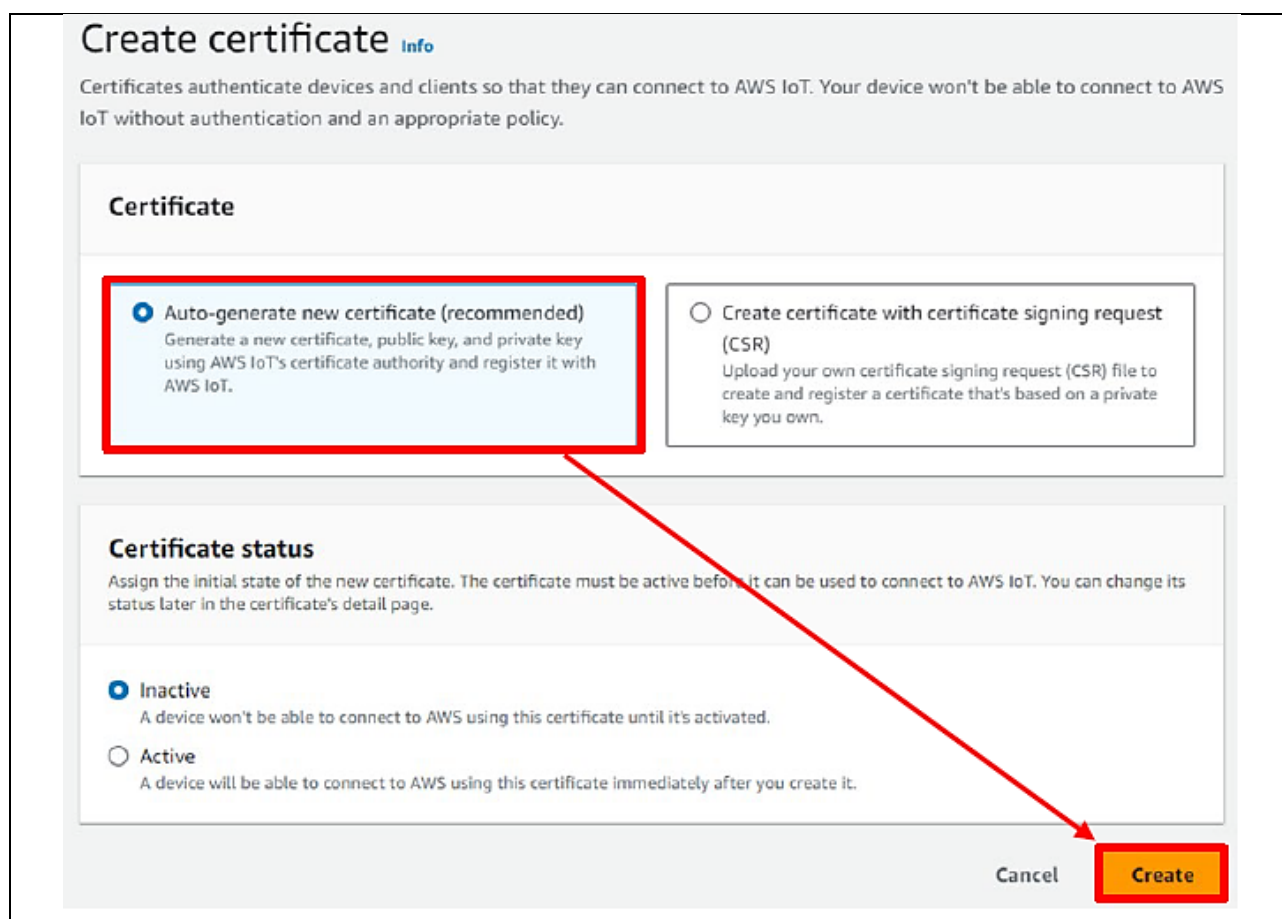


Figure 163. Creating a Certificate Automatically

Download the newly created certificate ① and key pair ②③, then click the **Continue** button.

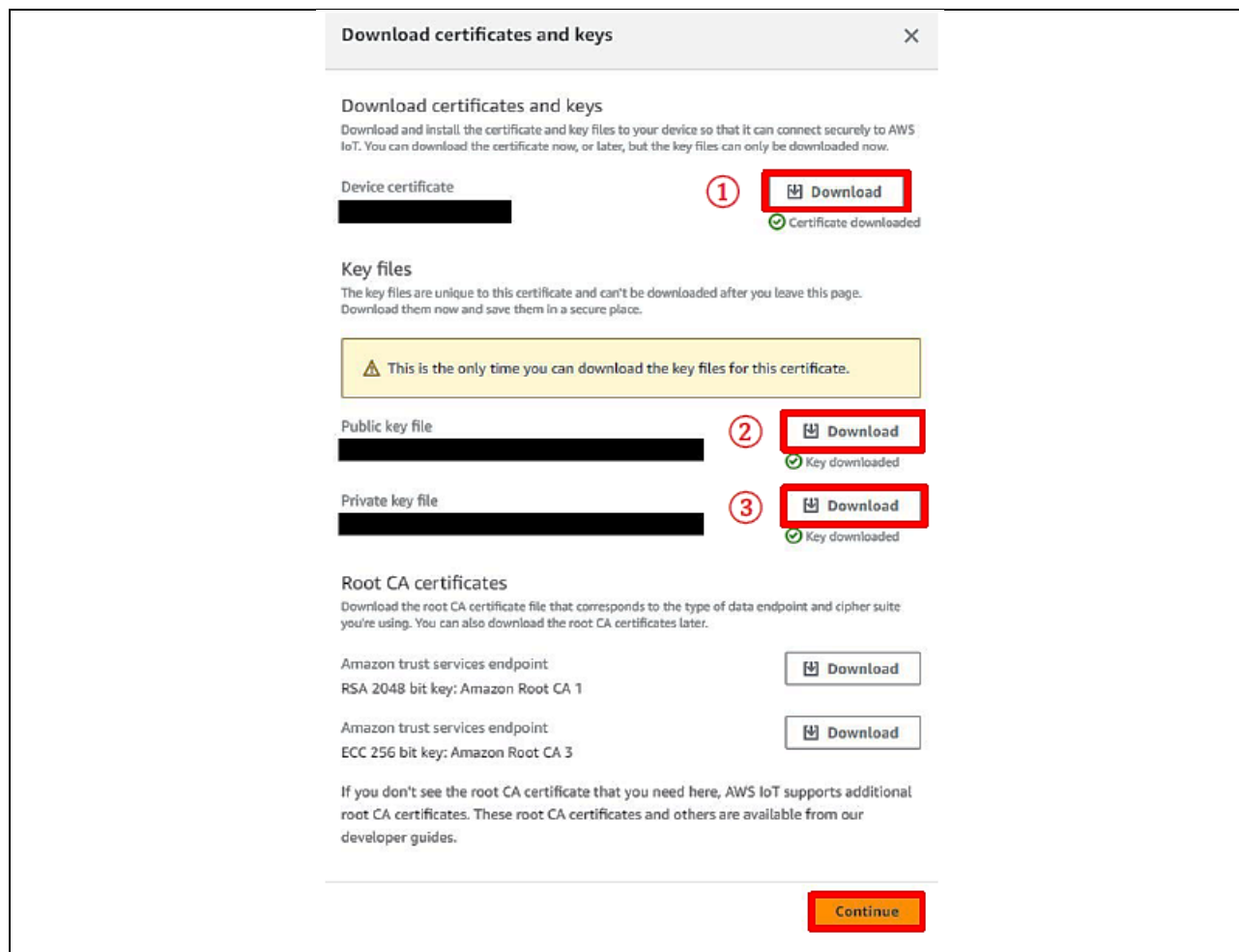


Figure 164. Downloading the Certificate and Key Pair

On the AWS console, select **Security** → **Certificates** and select the newly generated certificate ID.

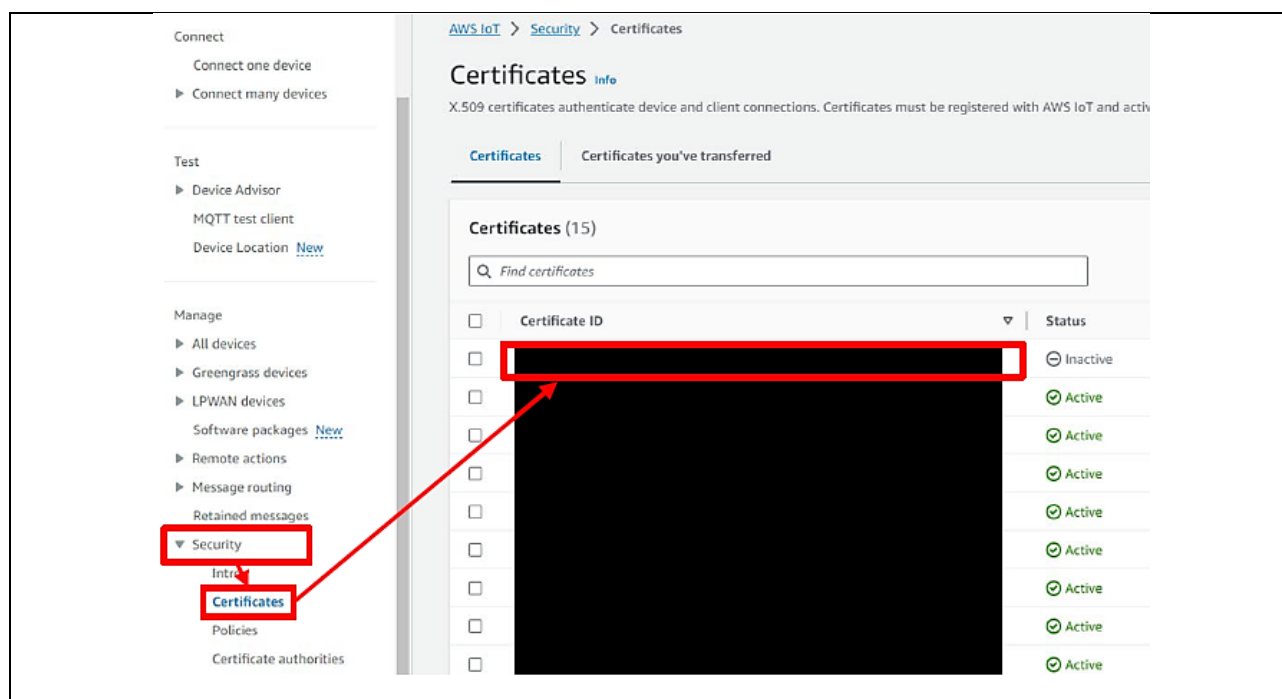


Figure 165. Certificate Settings

Click **Actions** → **Activate** to activate the certificate. Then, click the **Attach policies** button.

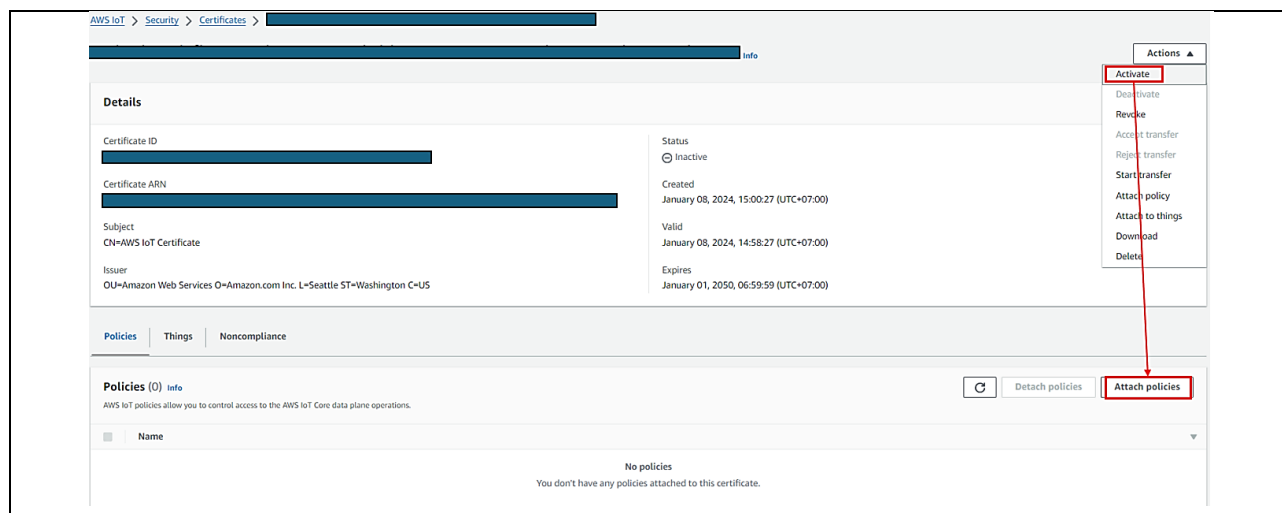


Figure 166. Certificate Settings: Attach Policies (1/2)

Clicking the **Attach policies** button opens the dialog box shown in Figure 167.

Select the policy to be used when fleet provisioning is run, created in 7.2.1, Policy Settings, and then click the **Attach policies** button to attach it to the certificate. For example, the name of policy, which is created for Fleet demo, is **"Fleet_policy"**.

This completes the settings related to generation of the claim certificate and claim key pair.

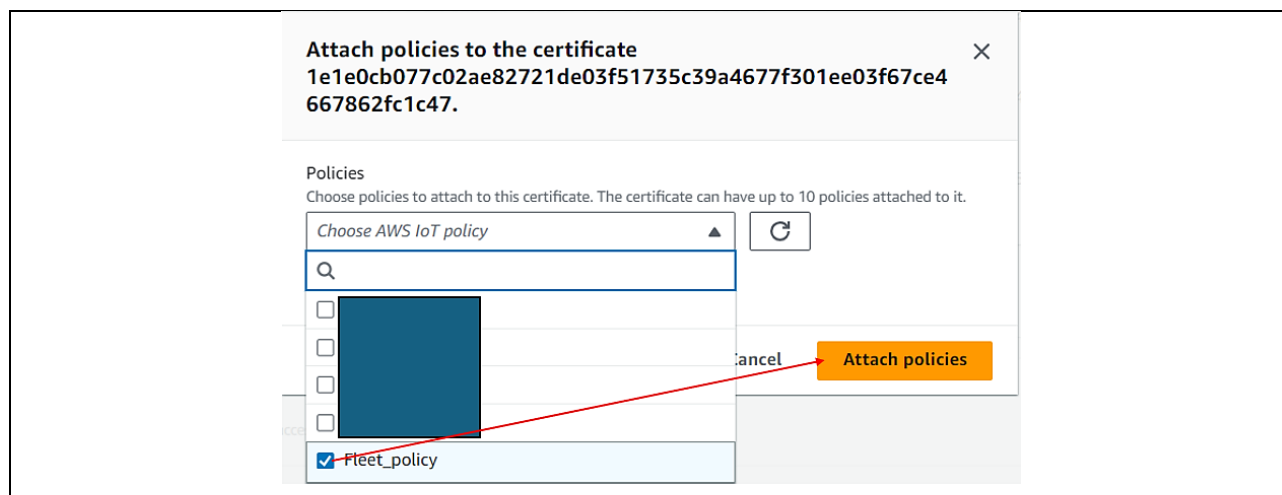


Figure 167. Certificate Settings: Attach Policies (2/2)

7.2.3 Creating a Fleet Provisioning Template

Select **Connect many devices** → **Connect many devices**, then click the **Create provisioning template** button.

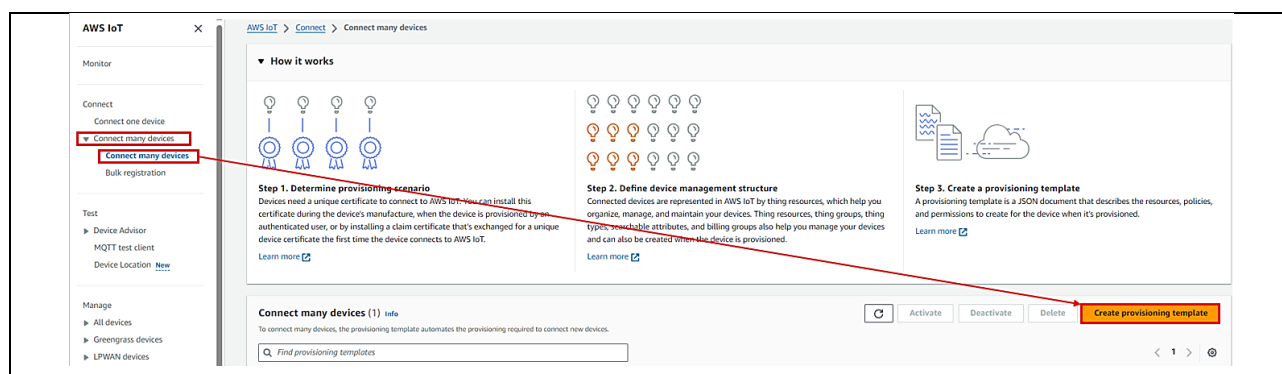


Figure 168. Creating a Provisioning Template (1/7)

Select **Provisioning devices with claim certificates**, then click the **Next** button.

Create provisioning template

AWS IoT supports three device-provisioning scenarios to accommodate different device manufacturing and installation processes. If you're not sure which scenario to choose, refer to the [developer guide](#) for information.

Provisioning scenario

Choose the provisioning scenario that fits your device manufacturing and installation processes the best. [Learn more](#)

☐ **Provisioning devices with unique certificates (JITP) - recommended**
 Your IoT devices will be installed with unique device certificates already on the device. This scenario is also known as just-in-time provisioning (JITP).

☐ **Provisioning devices by authorized users**
 Your IoT devices don't have unique certificates when they are installed. Authorized installers or end users use an app to provision the devices before they are connected to AWS IoT. In this scenario, you provide the installation app to configure the device during installation and the device's firmware must support this provisioning process. This is also known as fleet provisioning with user.

☒ **Provisioning devices with claim certificates**
 Choose this option if your IoT devices are delivered with claim certificates that are shared with other devices. The devices use their claim certificates to connect to AWS IoT for the first time. The claim certificate is replaced with a unique device certificate after provisioning. This option is also known as fleet provisioning with certificate.

To provision devices with claim certificates [Learn more](#)

1. Configure claim certificates
Configure the claim certificates that you'll install on your IoT devices and configure the policies attached to those certificates to allow provisioning.

2. Set provisioning actions
Configure how AWS IoT should provision your IoT device when it uses the claim certificate to connect to AWS IoT. You describe the AWS IoT resources and permissions that AWS will create for your device in a provisioning template that provisions your device when it uses the claim certificate to connect.

3. Connect devices
When your IoT devices use the claim certificate to connect to AWS IoT, the device is provisioned according to the provisioning template. During the provisioning process, a unique device certificate is created and installed on your IoT device for all subsequent connections to AWS IoT.

Cancel **Next**

Figure 169. Creating a Provisioning Template (2/7)

On the template creation screen, specify the provisioning template status, template name, and provisioning role. For **Provisioning template status** select **Active** and enter the name of the provisioning template. Then click the **Create new role** button and enter the role name.

Describe provisioning template [Info](#)

The details on this page describe the general aspects of the provisioning template that you're creating.

Provisioning template properties [Info](#)

Provisioning template status

The provisioning template status determines whether the template can be used to provision a new device. Only active templates can provision devices.

☐ **Inactive**
 Inactive templates can't provision any devices that are configured to use it. You can create an inactive template to prevent devices from being provisioned until you're ready.

☒ **Active**
 An active template can provision the devices that are configured to use it.

Provisioning template name

Enter_template_name

The name can have up to 36 characters and must not contain spaces. Valid characters: A-Z, a-z, 0-9, and _ (underscore) and - (hyphen).

Description - optional

A description of the provisioning template you're creating.

500 character remaining

Provisioning role

The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalf.

Choose an IAM role ▼ ↺ View ↗ **Create new role**

☒ Attach managed policy to IAM role

► **Tags - optional**

Figure 170. Creating a Provisioning Template (3/7)

For the **Claim certificate policy**, select the policy to be used when fleet provisioning is run, created in 7.2.1. For the **Claim certificate**, select the certificate created in 7.2.2, and click the **Next** button.

Claim certificate policy Info

The claim certificate requires a policy that authorizes it to connect to AWS IoT and perform the actions that provision the device. This policy doesn't apply to the device certificate that will be provisioned. You'll configure the policies for the provisioned device certificate later.

Claim certificate provisioning policy

Choose the AWS IoT policy that authorizes the claim certificate to connect and provision the IoT device. This policy is attached to the claim certificates you choose in the next section.

Fleet_policy

Q Fleet_policy ✓

We recommend that you use multiple claim certificates to provision your device fleet. Using each claim certificate in a limited number of IoT devices limits your exposure in case a claim certificate is compromised.

Claim certificates - optional (1/15) Info

Choose the claim certificates to attach the policy to, or attach the policy later by editing the provisioning template's provisioning initiator. Claim certificates must be active and have the claim certificate provisioning policy attached.

Activate Deactivate Delete Upload

Q 1 match < 1 >

☒	Certificate ID	Status
☒	[Redacted]	Active

Cancel Next

Figure 171. Creating a Provisioning Template (4/7)

For **Pre-provisioning actions**, select **Don't use a pre-provisioning action**. Also, under **Automatic thing creation**, turn on **Automatically create a thing resource when provisioning a device**, and if necessary, enter a character string of your choice as the thing name prefix. The thing name registered with AWS will be generated from this character string and the serial number set by the program. After entering the prefix, click the **Next** button.

Note: The demo does not use pre-provisioning actions. Refer to the page linked to below for information on using pre-provisioning actions: [Using pre-provisioning hooks with the AWS CLI](#)

Set provisioning actions [Info](#)

The provisioning actions describe the actions that take place during the provisioning process.

Pre-provisioning actions (recommended) [Info](#)

Before a new device is provisioned, you can run a Lambda function to verify the device should be provisioned. We recommend use this function to control access to your AWS account.

Pre-provisioning action

☐ Use a pre-provisioning action (recommended)
Perform actions prior to provisioning the device. For example, to check the device against a known device database to prevent unauthorized devices from connecting to your account.

☒ **Don't use a pre-provisioning action**
No actions will be performed prior to provisioning the device and the device is given access to your AWS account.

We recommend that you use a pre-provisioning action
We recommend that you use a pre-provisioning action when using a claim certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. [Learn more](#)

Automatic thing creation - optional

Create a thing resource to represent the device in AWS IoT. Your devices will need thing resources to use AWS IoT device management features such as thing groups, billing groups, and Device Shadows.

☒ **Automatically create a thing resource when provisioning a device**

Thing name prefix
The thing name prefix forms the beginning of each thing resource created by this provisioning template.

The name can't contain spaces. Valid characters: A-Z, a-z, 0-9, and _ (hyphen)

Additional configurations
You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ Thing type - optional
- ▶ Searchable thing attributes - optional
- ▶ Thing groups - optional
- ▶ Billing group - optional

Device configuration data - optional
You can define device configuration data that consists of key/value pairs that are sent to the device during provisioning.

You don't have any device configuration attributes for this template.

[Add configuration attribute](#)

Cancel Previous **Next**

Figure 172. Creating a Provisioning Template (5/7)

For **Set device permissions**, check the box next to the policy attached to newly created things, which were created in 5.4.4, then click the **Next** button.

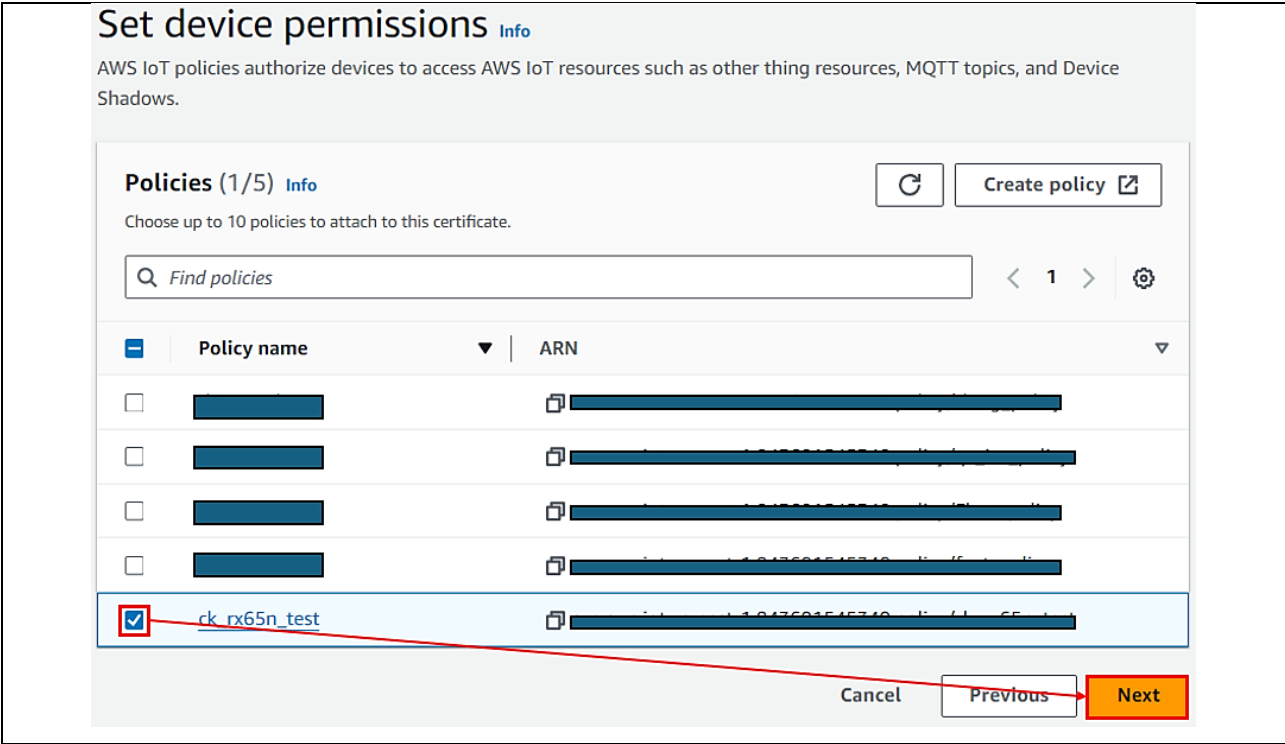


Figure 173. Creating a Provisioning Template (6/7)

Click the **Create template** button to complete the process of creating a fleet provisioning template.

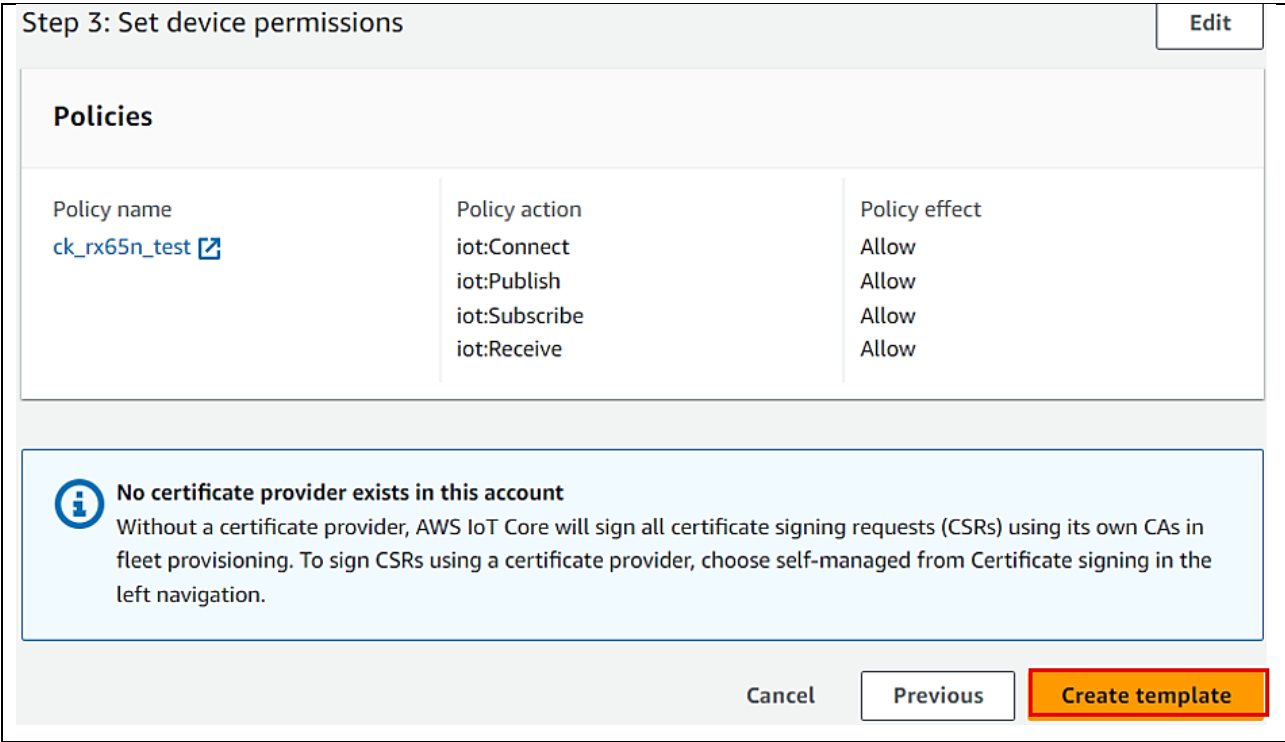


Figure 174. Creating a Provisioning Template (7/7)

7.3 Setting up the Project

Change the value of **ENABLE_FLEET_PROVISIONING_DEMO** to 1 in **aws_da16600_ck_rx65n/src/frtos_config/demo_config.h**:

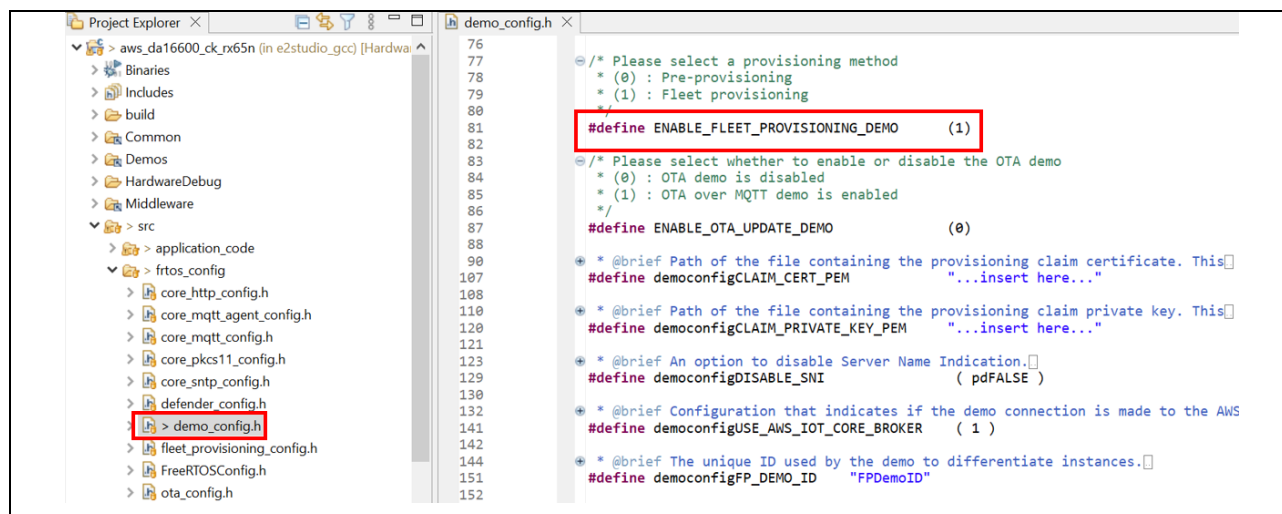


Figure 175. Enable Fleet Provisioning macro

Build and Debug the project as instructed in Topic: **5.1.3 Import the Project (number 8, 9)**

7.4 Running Fleet Provisioning

Note: User can run both Fleet Provisioning and OTA (Section 6) in this application. This section only describes the Fleet Provisioning feature.

1. After loading debugging to board, press any key to configure the application. (in case users have not config Cloud's credential for Fleet Provisioning before, or users want to save another value for Cloud's credential)

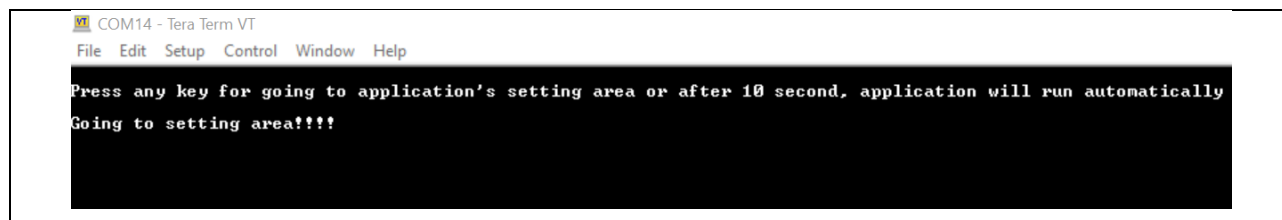


Figure 176. Save credentials for Fleet Provisioning (1/8)

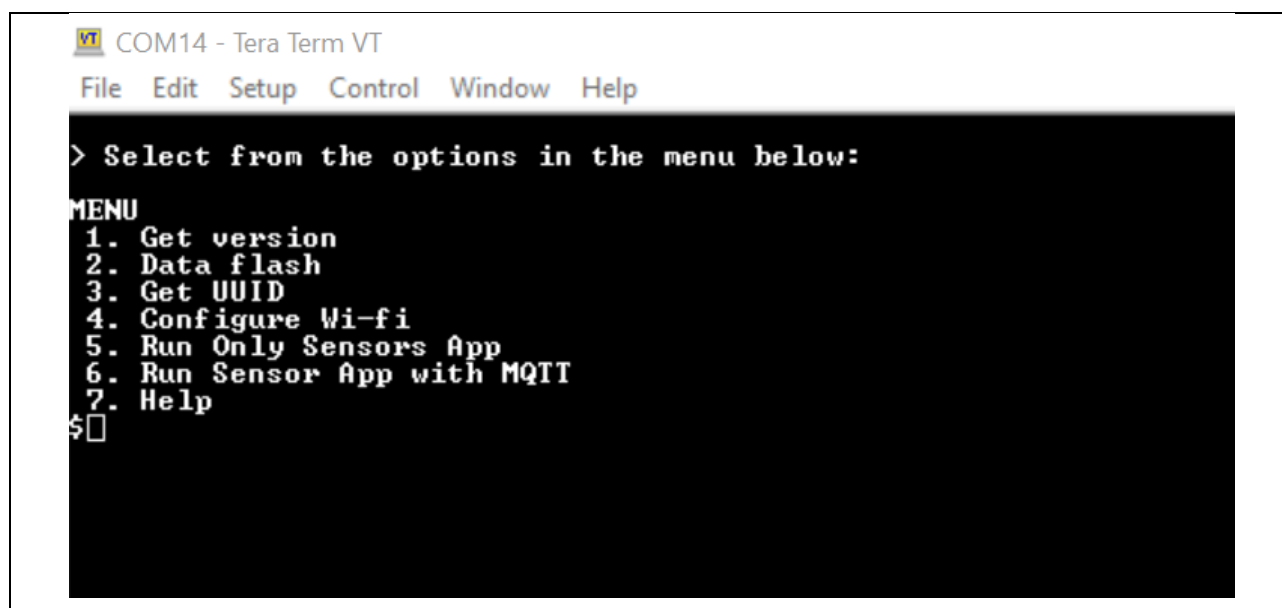


Figure 177. Save credentials for Fleet Provisioning (2/8)

- Press '2' to choose "**2. Data flash**" → press 'l' to choose "**l) Format Flash data**" to erase all stored information in flash (if application was run previously) (**optional**)

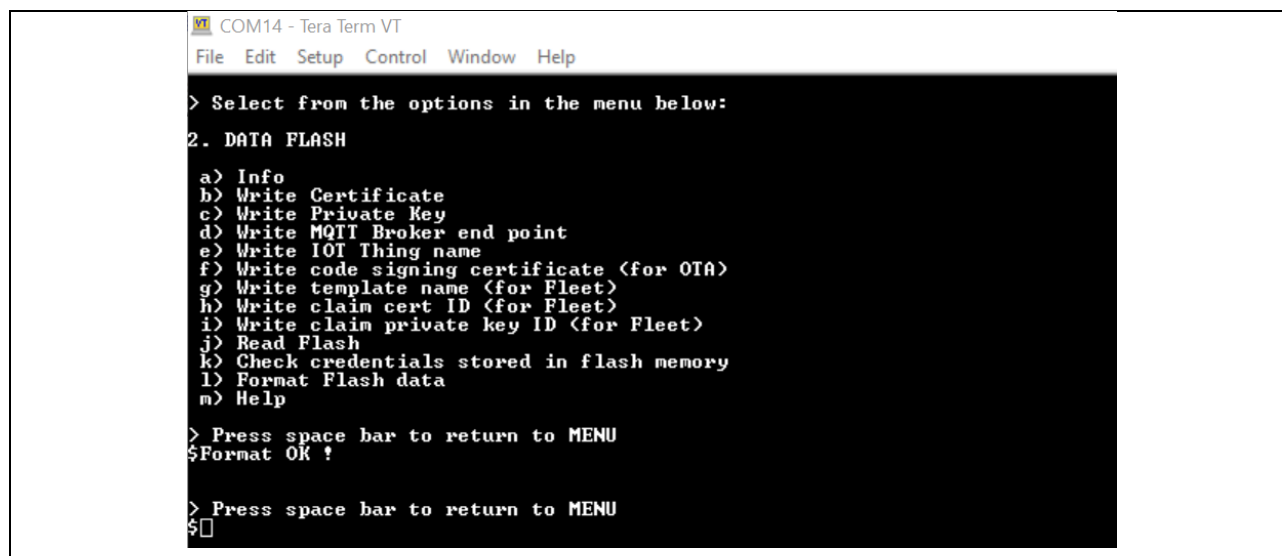


Figure 178. Save credentials for Fleet Provisioning (3/8)

Different from normal provisioning, Fleet Provisioning requires credentials: MQTT broker endpoint, fleet template name, claim cert ID, and claim private key ID.

- Get the MQTT endpoint and save it to flash like the topic 5.4.6 and 5.3.1:

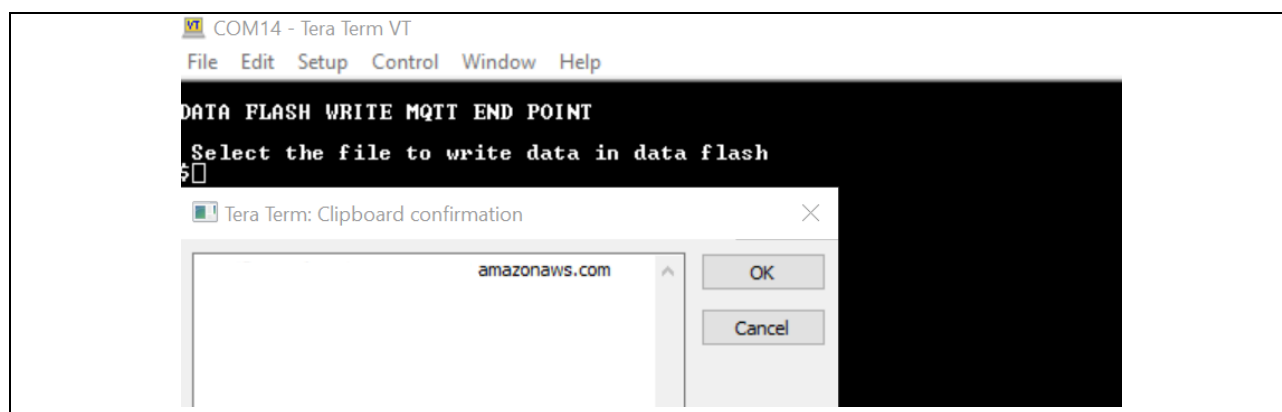


Figure 179. Save credentials for Fleet Provisioning (4/8)

- Next, storing fleet template name that user created at 7.2.3, copy this value, press the option 'g' and click the **Edit** tab of the Tera Term and "**Paste<CR>**" and verify and confirm the valid string and press OK.

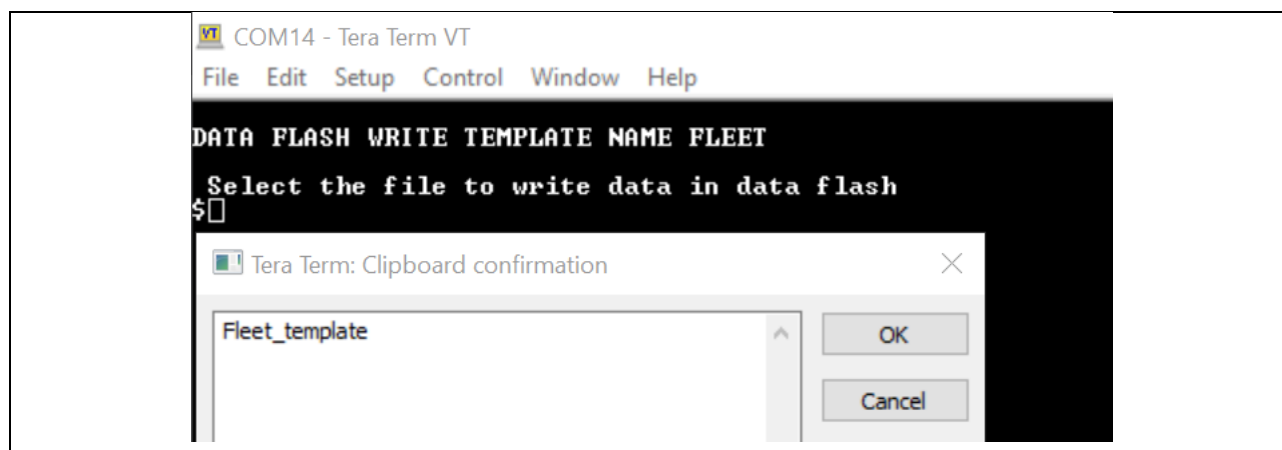


Figure 180. Save credentials for Fleet Provisioning (5/8)

5. Next, storing fleet claim cert ID, claim private key ID that users created at **7.2.2**, users downloaded (**Figure 164. Downloading the Certificate and Key Pair**):

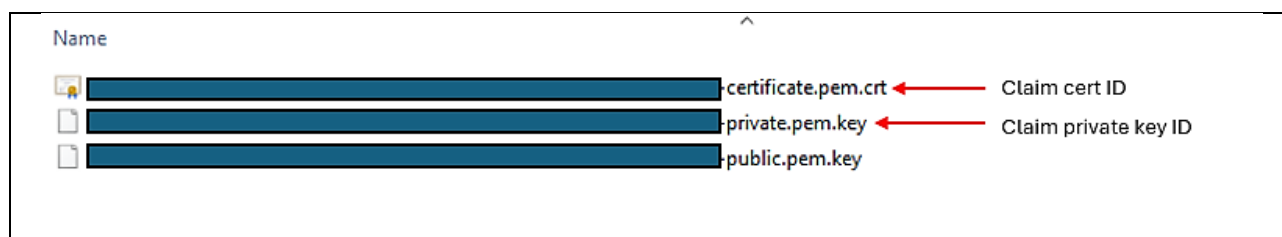


Figure 181. Save credentials for Fleet Provisioning (6/8)

From **"2. DATA FLASH"** menu, press **'h'** to save **Claim cert ID**. Then click the **File** tab of the Tera Term and **Send File** option and choose the downloaded Device certificate file **"xxxxxcertificate.pem.crt"**.

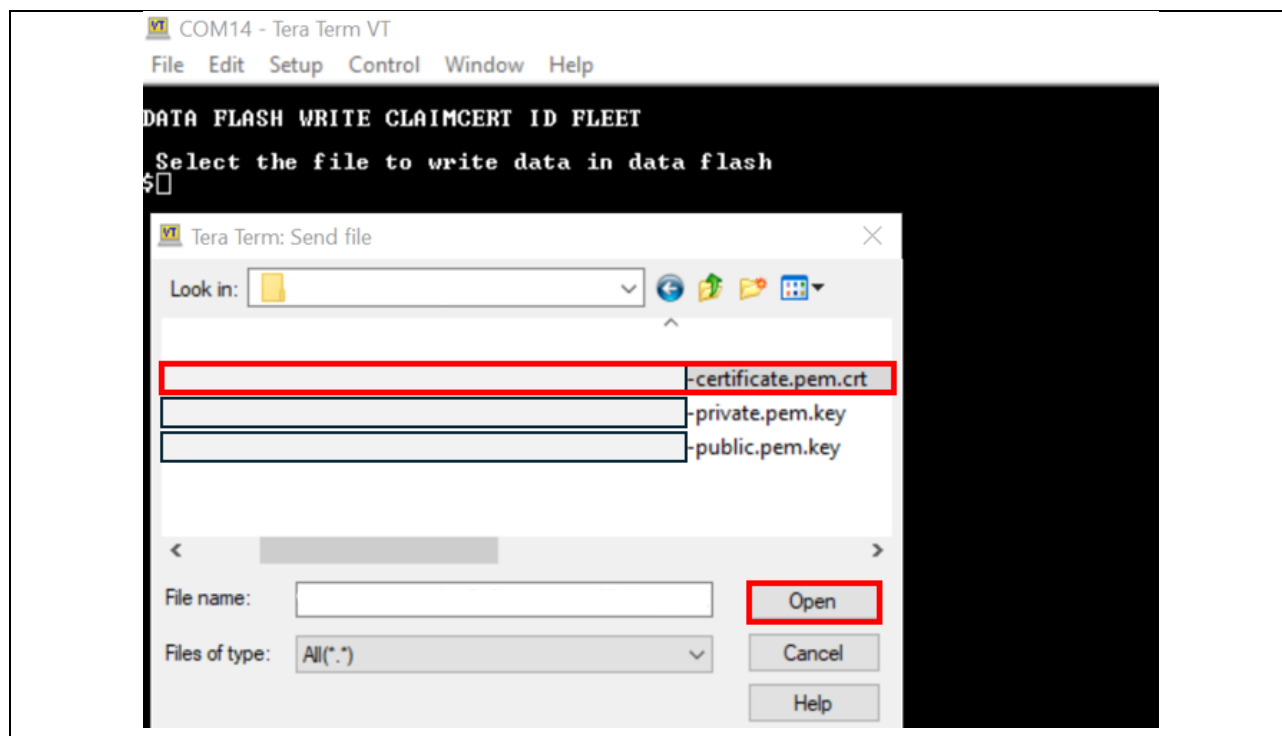


Figure 182. Save Credentials for Fleet Provisioning (7/8)

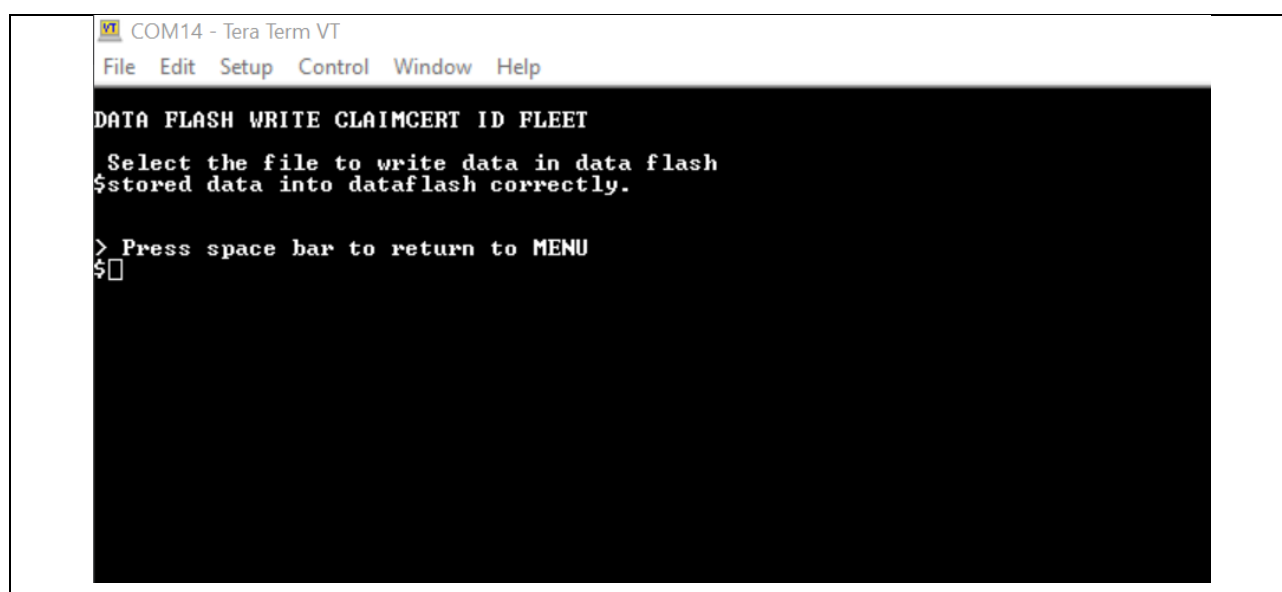


Figure 183. Save Credentials for Fleet Provisioning (8/8)

To store the **Claim private key ID**, press the option 'i' and click the **File** tab of the Tera Term and **Send File** option, choose "**xxxxxprivate.pem.key**".

6. Store Wi-Fi's credentials to Flash (refer to topic 5.3.2)

7. Start application with Fleet Provisioning:

Back to the Application Menu and press '6' to run application:

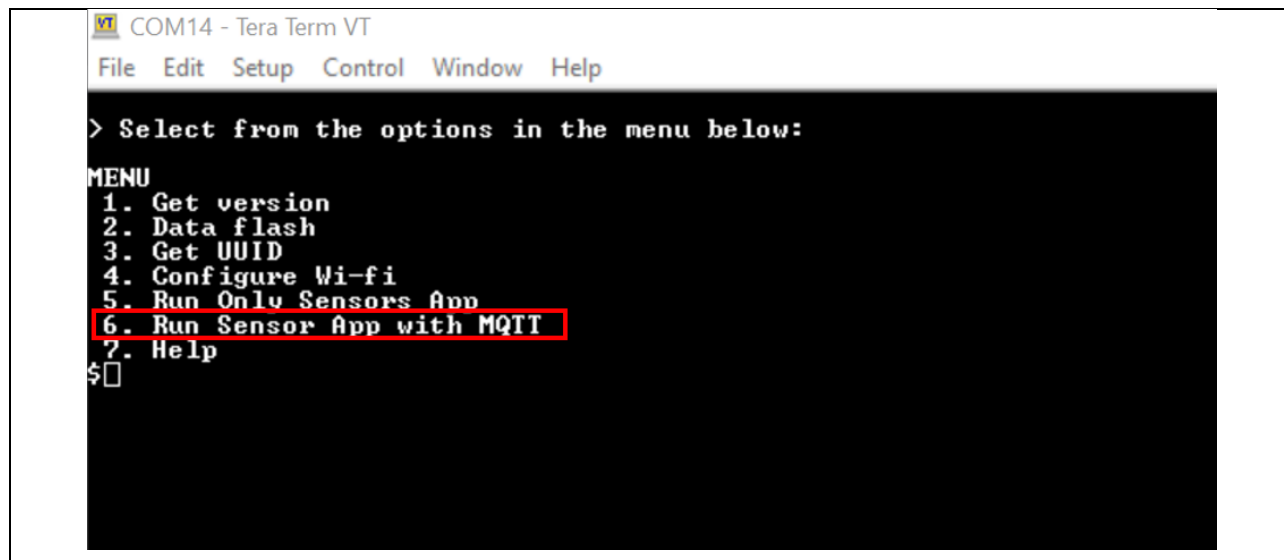


Figure 184. Running Application with Fleet (1/8)

Application will check stored data flash. When it has necessary credentials enough, application will start. Otherwise, the application will send a notice to the user and the user has to save the lacking credentials for running application.

```

CHECK CREDENTIALS STORED IN DATA FLASH
Claim private key ID saved in data flash is verified and successful
Claim cert ID saved in data flash is verified and successful
template name saved in data flash is verified and successful
OTA is disabled, do not need code sign certificate
Wi-Fi's Security saved in data flash is verified and successful
Wi-Fi's Password saved in data flash is verified and successful
Wi-Fi's SSID saved in data flash is verified and successful
Fleet is enabled, IOT thing name is not available, it will be updated by Fleet
MQTT Endpoint saved in data flash is verified and successful
Fleet is enable, do not need Private Key <2b, 2c section>
Fleet is enable, do not need Certificate <2b, 2c section>
All credentials in data flash is verified and successful

*** Wi-Fi Init Successful ***
SSID: TP-Link_2502
Connecting to TP-Link_2502
Wi-Fi connected to SSID TP-Link_2502.
Device IP address: 192.168.250.227
Device network mask: 255.255.255.0
Device gateway address: 192.168.250.8
MQTT End point IP address = 54.174.15.244
0 99937 [CLI] -----STARTING DEMO-----

1 99938 [DemoTask] [INFO] -----Start Fleet Provisioning Task-----
2 100336 [DemoTask] [INFO] Establishing MQTT session with claim certificate...
3 100337 [DemoTask] [INFO] Using default rootCA cert.
4 100337 [DemoTask] [INFO] Create a TCP connection to .us-east-1.amazonaws.com:8883.
5 100353 [DemoTask] [INFO] Created new TCP socket.
6 100806 [DemoTask] [INFO] Established TCP connection with .us-east-1.amazonaws.com.
7 108311 [DemoTask] [INFO] <Network connection 800144> TLS handshake successful.
8 108311 [DemoTask] [INFO] <Network connection 800144> Connection to .us-east-1.amazonaws.com established.
9 109156 [DemoTask] [INFO] MQTT connection established with the broker.
10 109156 [DemoTask] [INFO] MQTT connection successfully established with broker.

11 109156 [DemoTask] [INFO] Established connection with claim credentials.
12 109425 [DemoTask] [INFO] SUBSCRIBE topic $aws/certificates/create-from-csr/chor/accepted to broker.

13 109943 [DemoTask] [INFO] MQTT_PACKET_TYPE_SUBACK.

14 114909 [DemoTask] [INFO] SUBSCRIBE topic $aws/certificates/create-from-csr/chor/rejected to broker.

```

Figure 185. Running Application with Fleet (2/8)

If the text string “**Demo completed successfully.**” Appears at the end of the log, the fleet provisioning demo completed successfully. Successful completion of the demo means that a new thing has been registered on AWS IoT Core and an individual device certificate assigned to it.

```

59 7796782 [DemoTask] [INFO] Successfully established connection with provisioned credentials.
60 7796866 [DemoTask] [INFO] <Network connection 800144> TLS close-notify sent.
61 7796969 [DemoTask] Closed Socket: Socket Number = 0.
62 7796970 [DemoTask] [INFO] Demo iteration 1 is successful.
63 7796970 [DemoTask] [INFO] Demo completed successfully.
64 7796970 [DemoTask] [INFO] -----Fleet Provisioning Task Finished-----

```

Figure 186. Running Application with Fleet (3/8)

After running the fleet provisioning demo, users can use the individual device certificate and private key obtained from AWS to run the MQTT with sensors demo:

```

14 20257 [DemoTask] [INFO] Demo iteration 1 is successful.
15 20257 [DemoTask] [INFO] Demo completed successfully.
16 20257 [DemoTask] [INFO] -----Fleet Provisioning Task Finished-----
17 20257 [MQTT] [INFO] -----Start MQTT Agent Task-----
18 20257 [MQTT] [INFO] Creating a TLS connection to .iot.us-east-1.amazonaws.com:8883.
19 20258 [MQTT] [INFO] Created new TCP socket.
20 20259 [DemoTask] [INFO] Deleting Fleet Provisioning Demo task.
21 24856 [MQTT] [INFO] <Network connection 800c3c> TLS handshake successful.
22 24856 [MQTT] [INFO] <Network connection 800c3c> Connection to .iot.us-east-1.amazonaws.com established.
23 24856 [MQTT] [INFO] Creating an MQTT connection to the broker.
24 25894 [MQTT] [INFO] MQTT connection established with the broker.
25 25894 [MQTT] [INFO] Successfully connected to MQTT broker.
26 25894 [oh1203_thre] I2C bus 2 setup success
27 25894 [oh1203_thre]
28 25894 [sensor_thre] I2C bus 0 setup success
29 25905 [sensor_thre] HS3001 open sensor instance successful: 0
30 25905 [sensor_thre] ICP20100 open sensor instance successful: 0
31 25915 [zmod_thread] I2C bus 1 setup success
32 25916 [AWS_DA16600] [INFO] -----Start AWS Wi-Fi DA16600 - MQTT Demo Task -----
33 26348 [zmod_thread] ZMOD4410 open sensor instance successful: 0
34 26783 [zmod_thread] ZMOD4510 open sensor instance successful: 0
35 26784 [zmod_thread] Task zmod4410 measurement Success:0
36 26900 [sensor_thre] ICH42605 open sensor instance successful: 0
37 26962 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
38 28775 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
39 28775 [AWS_DA16600] [Send Data] ZMOD4410-IAQ TUC: 000.000

```

Figure 187. Running Application with Fleet (4/8)

Users can check on the thing registered by the fleet provisioning demo from AWS IoT console. When running fleet provisioning successfully, please check the log in Tera Term with line: “Received AWS IoT Thing name: test_fleetFPDemoID_xxxxxxxxxx”

```

40 7774240 [DemoTask] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
41 7774240 [DemoTask] [INFO] State record updated. New state=MQTTPubAckSend.
42 7774240 [DemoTask] [INFO] Received accepted response from Fleet Provisioning RegisterThing API
43 7778845 [DemoTask] [INFO] Received AWS IoT Thing name: test_fleetFPDemoID_xxxxxxxxxx
44 7778869 [DemoTask] [INFO] AWS IoT Thing name is saved to Data Flash
45 7779171 [DemoTask] [INFO] UNSUBSCRIBE sent topic $aws/provisioning-templates/Fleet_template/provision/chor/accepted to broker.

```

Figure 188. Running Application with Fleet (5/8)

Note: Prefix **test_fleet** is the name which is set by the user when creating the Fleet template.

Under **All devices**, select **Things**. The registered IoT thing will appear under the header **Name** as shown in the image below:

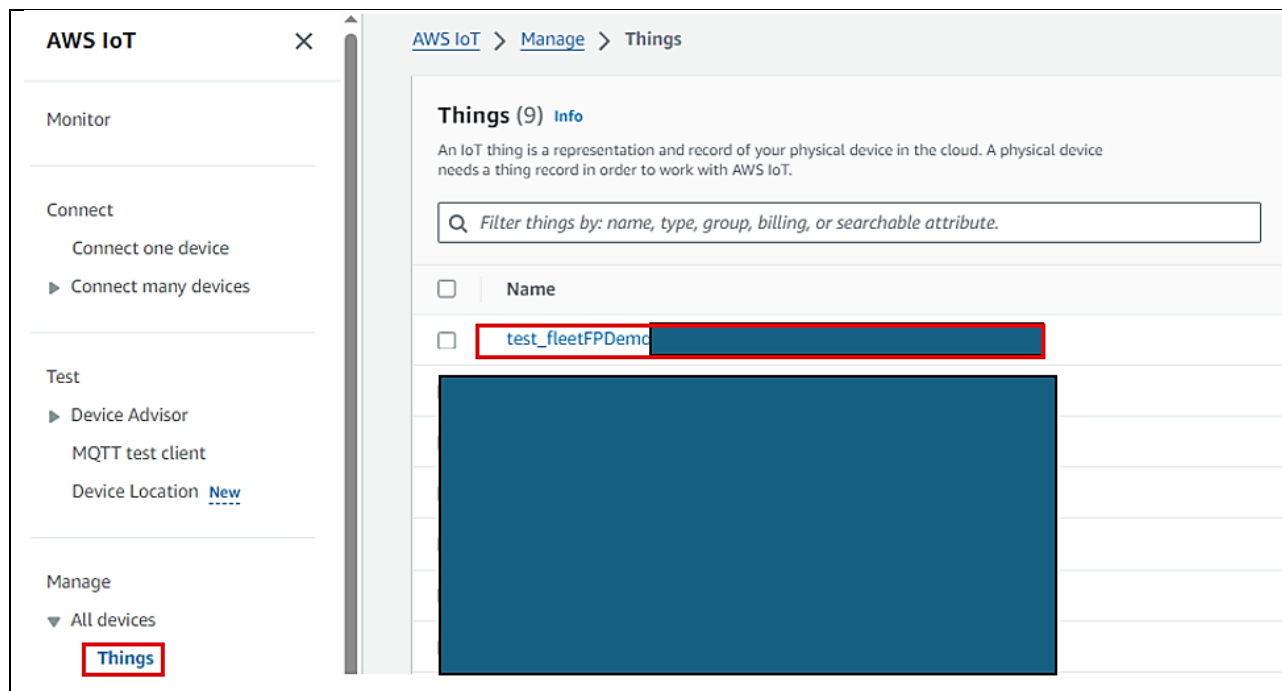


Figure 189. Running Application with Fleet (6/8)

By checking the registered things, user can confirm that the individual device certificate generated and assigned by fleet provisioning has been attached and activated:

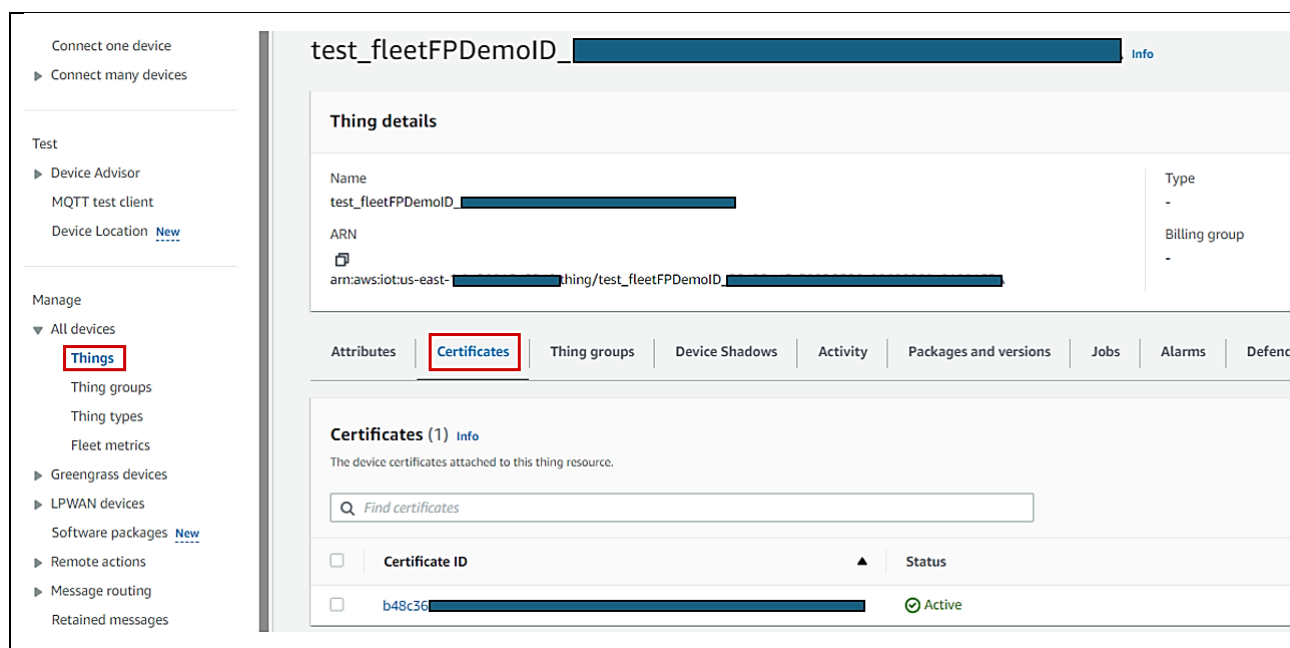


Figure 190. Running Application with Fleet (7/8)

This Certificate ID will be shown in Tera Term with line “Received certificate with Id: xxxxxxxxxxxxxxxxxxxx”.



Figure 191. Running Application with Fleet (8/8)

8. Note and Trouble Shooting

8.1 About Stabilization Time for Sensor

There is a stabilization time for each sensor. It cannot read correct values during the time.

The following table gives details of the stabilization time for each sensor.

Table 9. Sensor Stabilization Time

Sensor Name	When Powered up the First Time	After Soft or Hard Reset
ZMOD4410 IAQ	Up to 1 minute	Up to 1 minute
ZMOD4510 OAQ	Up to 1.5 hours	Up to 1 hour
OB1203	Up to 20 minutes (After putting finger on sensor, it may take up to 60 seconds to sense data)	Up to 20 seconds (After putting finger on sensor, it may take up to 60 seconds to sense data)
HS3001	Up to 30 seconds	Up to 10 seconds
ICP	Up to 30 seconds	Up to 10 seconds
ICM	Up to 30 seconds	Up to 10 seconds

8.2 When Build Errors Occur

If a ‘No such file or directory’ error occurs, the project path may be too long. When the path is longer than 256 characters, e² studio outputs errors at build time.

When this error occurs, move the project to a shorter path location (for example, under C:\)

8.3 When Run Errors Occur

In case of an error, relate to `wifi_init()`: "[ERR] In Function: wifi_init(), ** Wi-Fi Init Failure **", SDK version in DA16600 Pmod is old. It is required to use DA16600 SDK v3.2.7.1 or later for the application to work correctly. To confirm DA16600 SDK version and update for it if required, see the guideline in [DA16200/DA16600 SDK Update Guide](#).

If SDK version is suitable, user need to press button **S1** to reset the board and run the application again.

8.4 Wi-Fi Access Point

Depending on the configuration of the Wi-Fi router/modem, the application may not work correctly. Please retry with another access point (for example, by using the hotspot on a mobile phone).

8.5 Renesas AWS Dashboard account credits and quarantine

Every kit registered to the dashboard will include a \$10 AWS credit. When the credit is exhausted the account is quarantined. At this time, the user cannot download certificate and go to Dashboard.

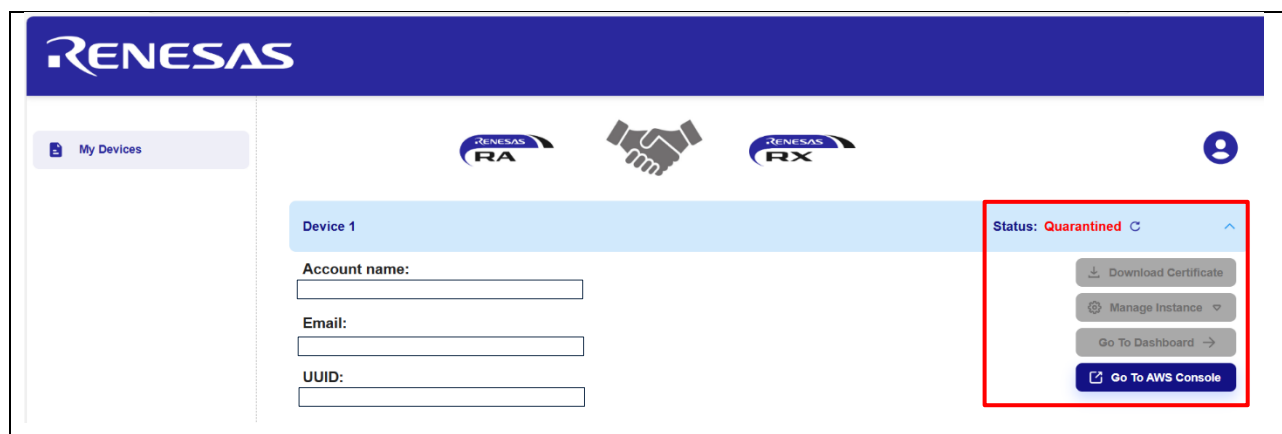


Figure 192. Account Used Up Trial 10 USD

To avoid the account from being quarantined, set up the following:

1. Users can access their AWS account from the dashboard main page by clicking on Go to AWS Console.

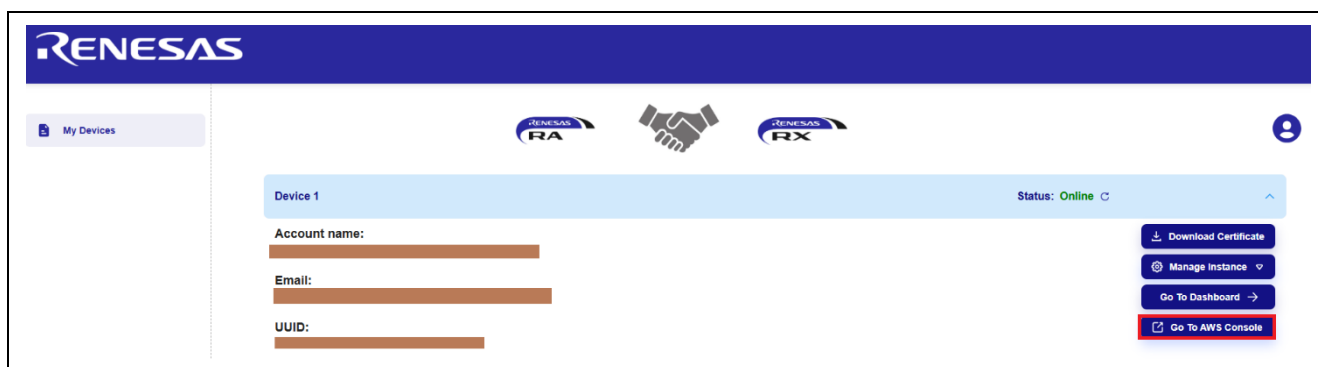


Figure 193. Accessing AWS Account from the Dashboard

2. Make sure to unblock the pop-ups on the browser to allow the AWS console to open.

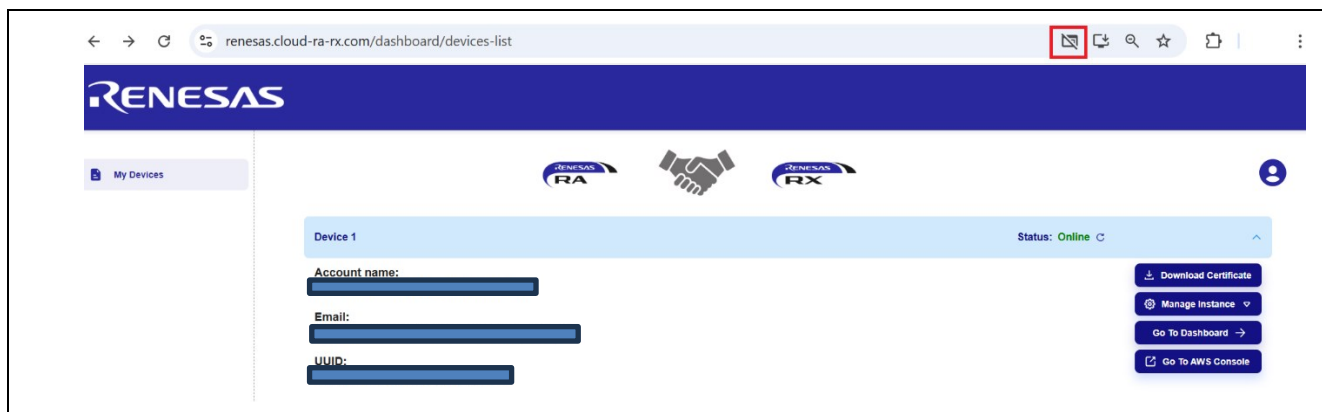


Figure 194. Pop Ups on the browser

- After login you are redirected to AWS access portal page. The AWS sub account can be accessed from the AWSAdministratorAccess link.

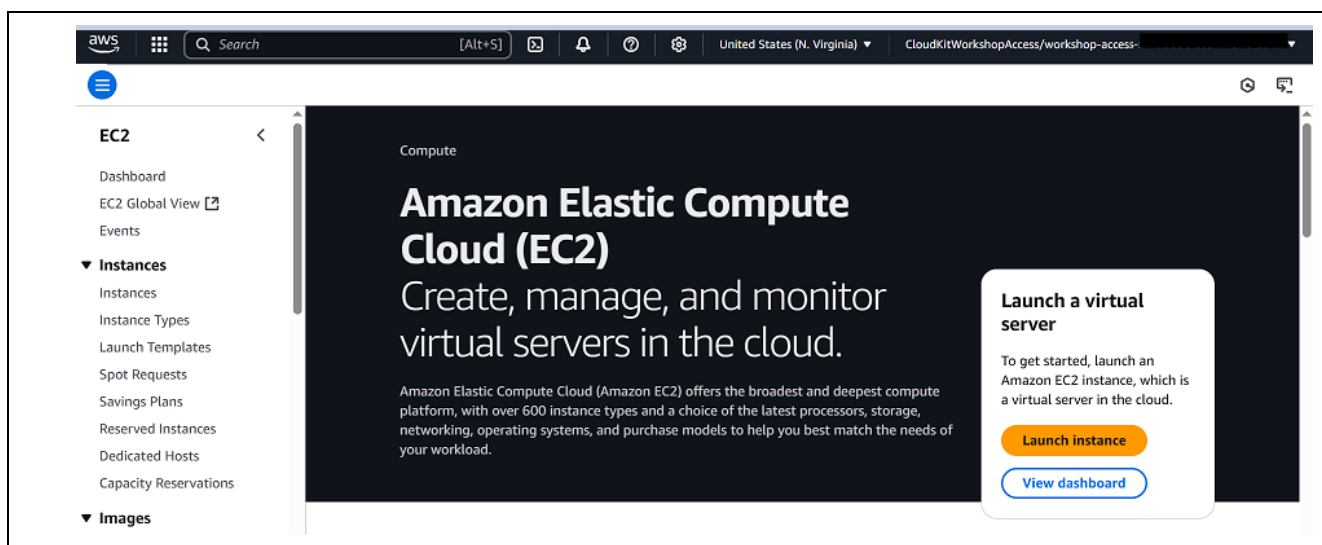


Figure 195. AWS account

- Payment options can be accessed from the 'Billing and Cost Management' section of the console.

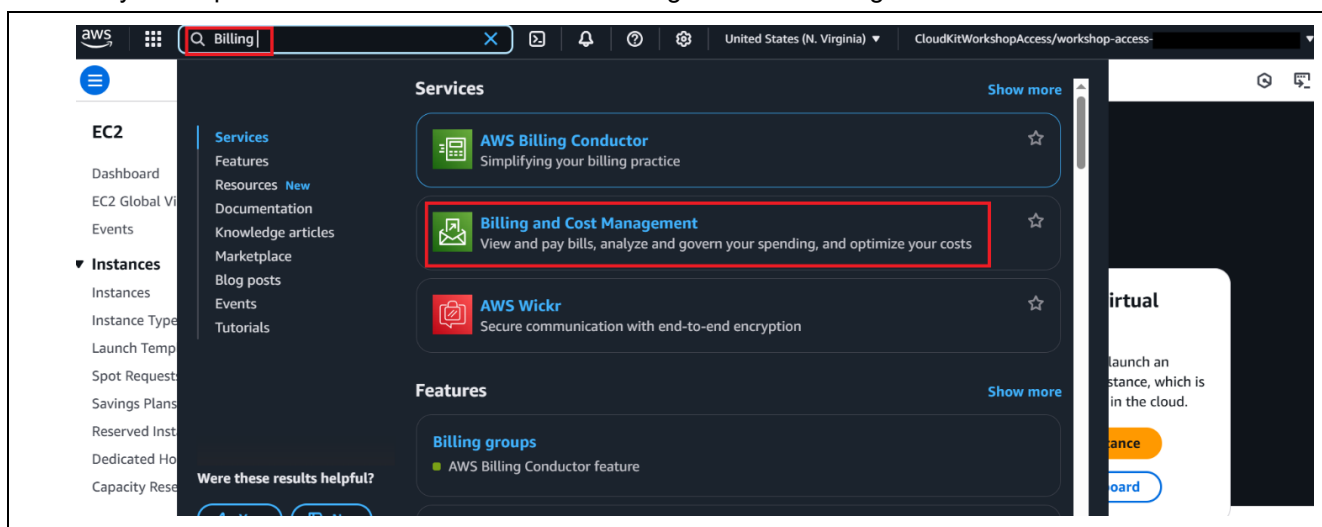


Figure 196. Billing and Cost Management

- Scroll to the 'Preferences and Settings' section and click on 'Payment Preferences'.

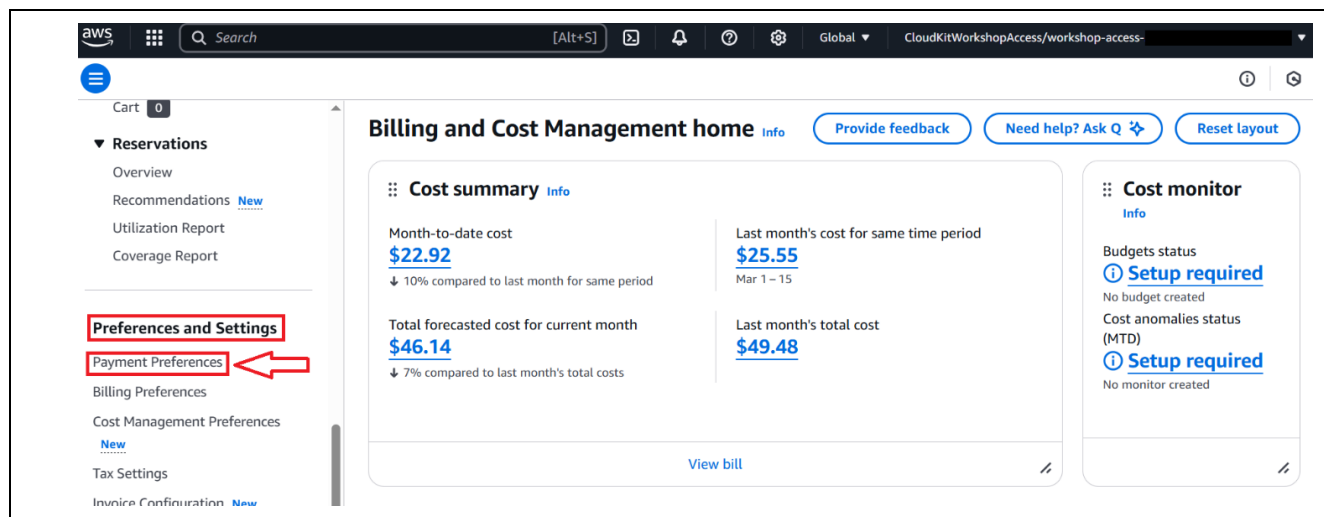


Figure 197. Payment Preferences

6. Add payment method.

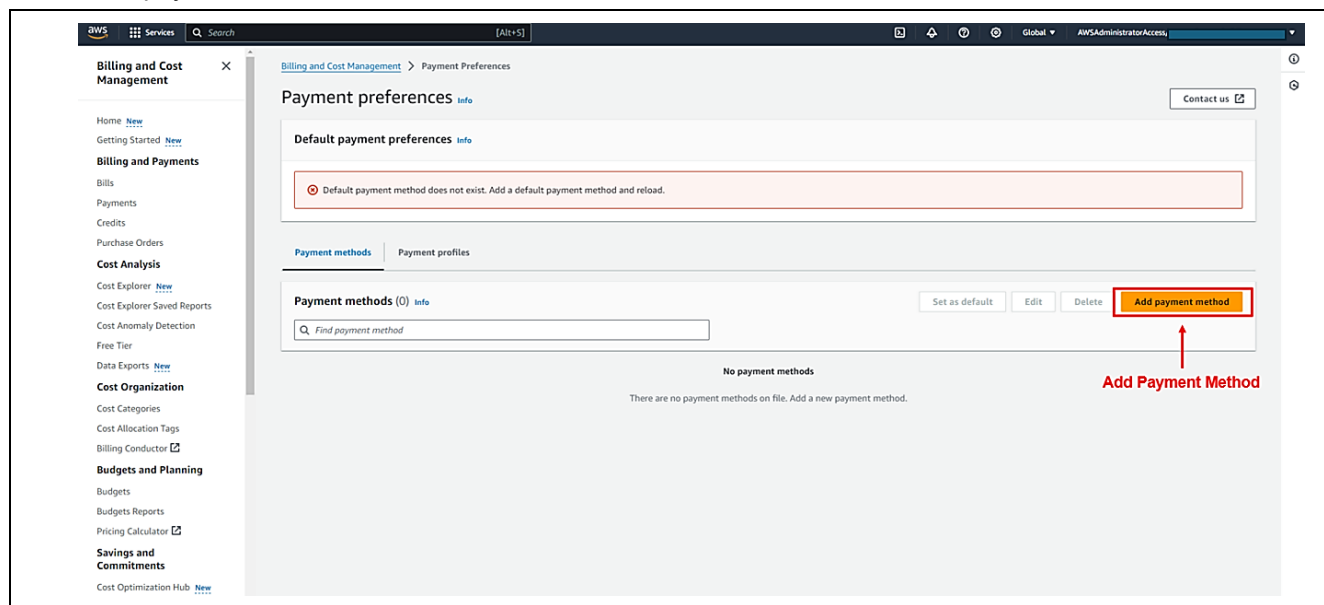
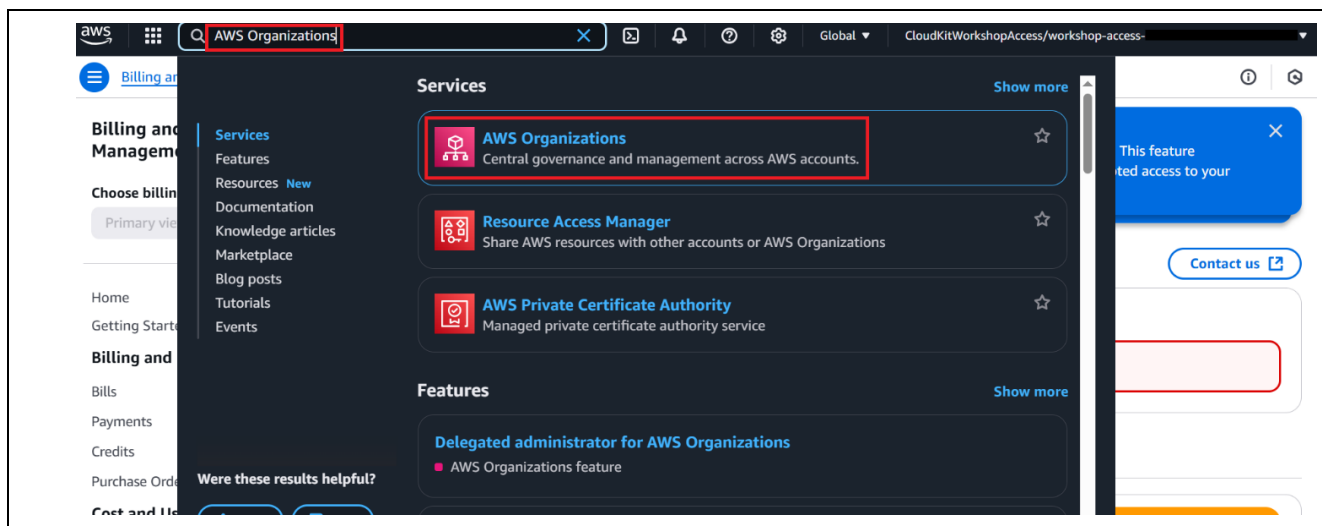
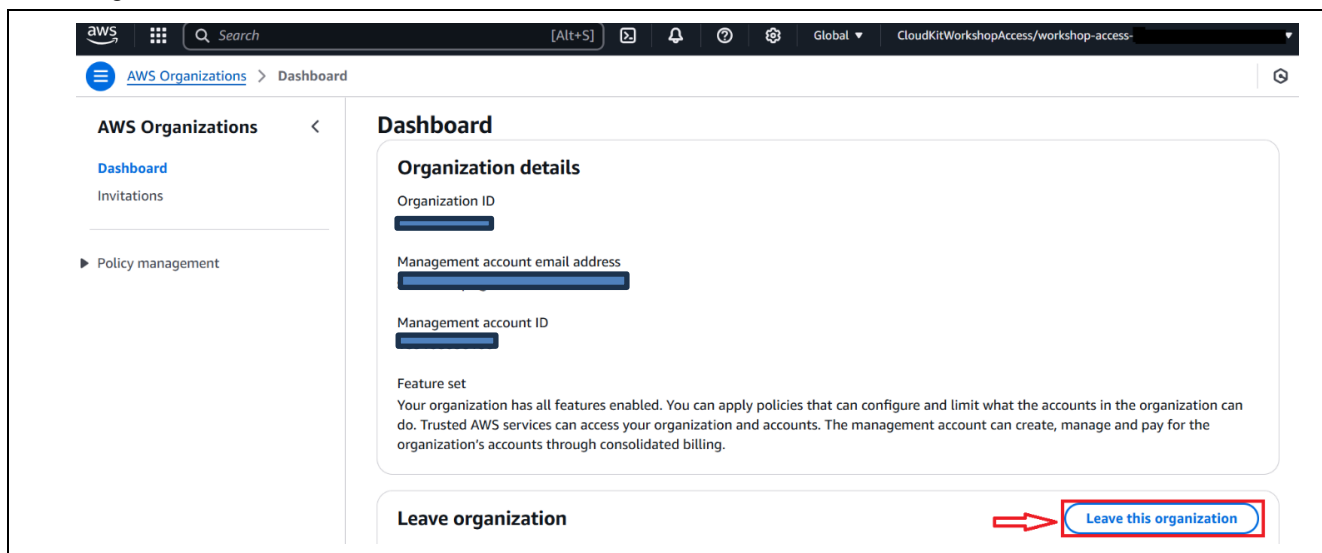


Figure 198. Add Payment Method

7. Go to 'AWS Organizations'.

**Figure 199. AWS Organizations**

8. Leave the organization. This allows user accounts to not get affected by the \$10 credit limit set by the organization.

**Figure 200. Leaving the organization**

9. If the account is missing the prerequisites for leaving the organization, a dialog box appears prompting for the next steps. Click on 'sign-in to the account'.

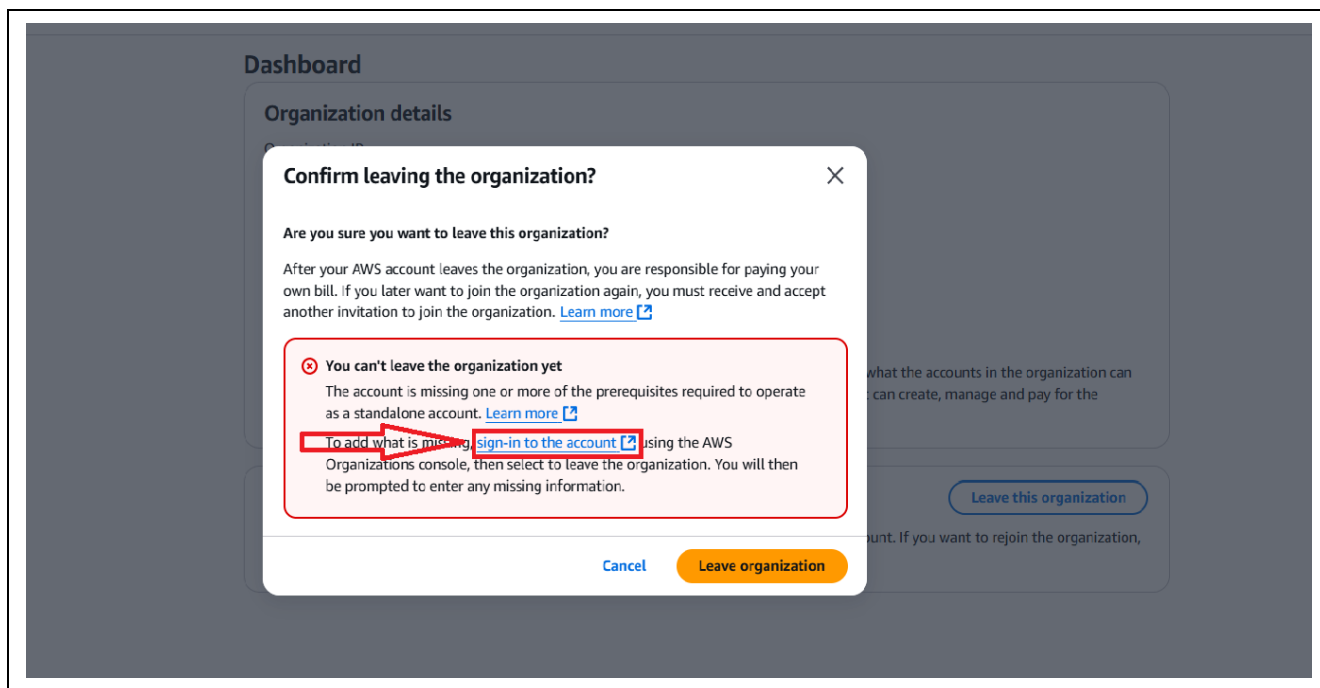


Figure 201. AWS Sign up process

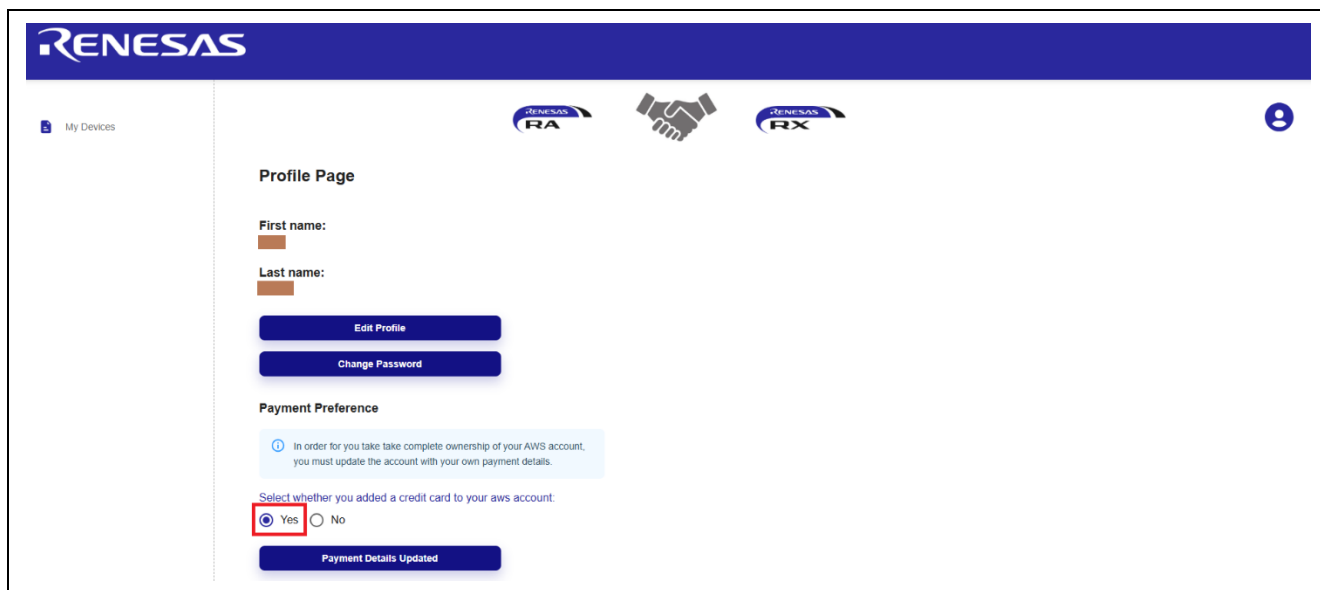
10. Follow the steps to complete the requirements for sign-up.

Figure 202. Prerequisites for leaving the organization

11. When the dialog box stating the sign up process is complete appears, choose '**Leave organization**'.
12. A confirmation dialog box appears. Confirm your choice to remove the account. You are redirected to the **Getting Started** page of the AWS Organizations console, where you can view any pending invitations for your account to join other organizations.
13. Remove the IAM roles that grant access to your account from the organization.

In addition, the dashboard must be updated with the payment preference from the user profile on the dashboard page.

1. Go to the user profile as shown in Figure 201.
2. Update the payment preference.



The screenshot shows the Renesas dashboard interface. At the top is the Renesas logo. Below it, there are navigation links for 'My Devices', 'RA', a handshake icon, and 'RX'. The main content area is titled 'Profile Page'. It contains fields for 'First name' and 'Last name', both with redacted text. Below these are buttons for 'Edit Profile' and 'Change Password'. The 'Payment Preference' section includes a blue information box stating: 'In order for you take complete ownership of your AWS account, you must update the account with your own payment details.' Below this is a prompt: 'Select whether you added a credit card to your aws account.' There are two radio buttons: 'Yes' (selected) and 'No'. At the bottom of this section is a 'Payment Details Updated' button.

Figure 203. Payment Preference Update

Note: Failure to complete either of these steps will result in the account being quarantined. Quarantined accounts must leave the organization and remove the IAM roles as mentioned in this section to regain access to the accounts.

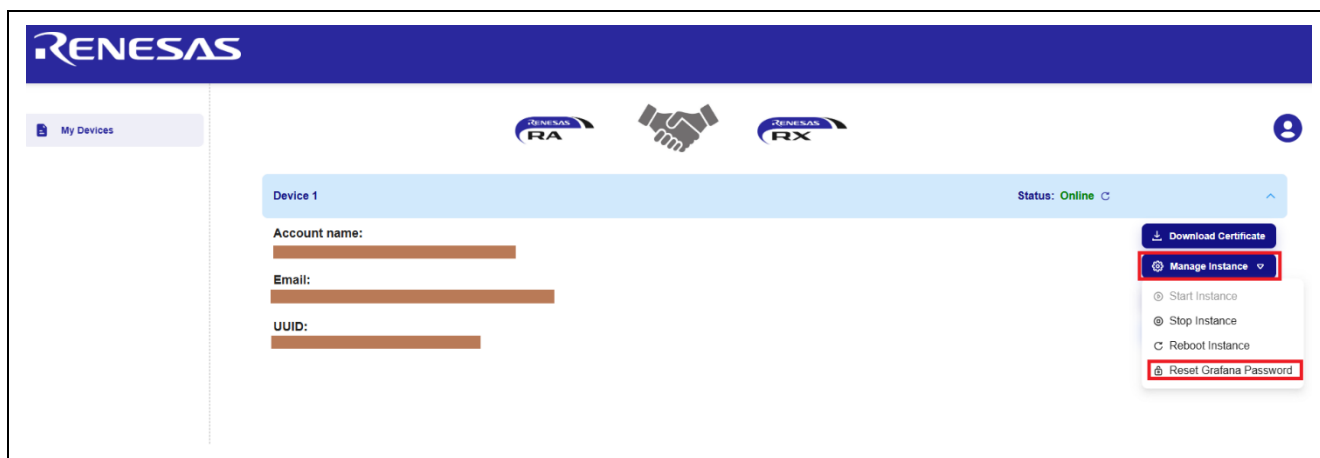
8.6 When Unable to Log in to the Dashboard (Grafana account)

If you cannot log in to the Dashboard with the password you changed in step 7 of the section 5.2.2, To Get the Account 10 USD of Trial of AWS, try the following.

- Set “admin” in the Email or username field and set the changed password in the password field.

When changing the password for the initial session, the username is not changed from admin. Therefore, “admin” must be entered in the username field. To enable users to log in with their own username and email address, please change the user information in the Server Admin menu after logging in.

To reset the forgotten password, choose ‘Reset Grafana Password’ from the ‘Manage instance’ dropdown menu.



The screenshot shows the Renesas dashboard interface. At the top is the Renesas logo. Below it, there are navigation links for 'My Devices', 'RA', a handshake icon, and 'RX'. The main content area shows details for 'Device 1'. It includes fields for 'Account name', 'Email', and 'UUID', all with redacted text. To the right of these fields is a 'Status: Online' indicator. A 'Download Certificate' button is visible. Below it is a 'Manage Instance' dropdown menu, which is open, showing options: 'Start Instance', 'Stop Instance', 'Reboot Instance', and 'Reset Grafana Password'. The 'Reset Grafana Password' option is highlighted with a red box.

Figure 204. Grafana Password Reset

8.7 How to Enable/Disable EC2 Instance

AWS trial accounts start billing immediately after device registration. To avoid excessive billing or AWS credit usage when the device is not in use, manage the EC2 instance from the dashboard main page.

1. When the device is not in use stop the instance from the 'Manage Instance' dropdown menu.

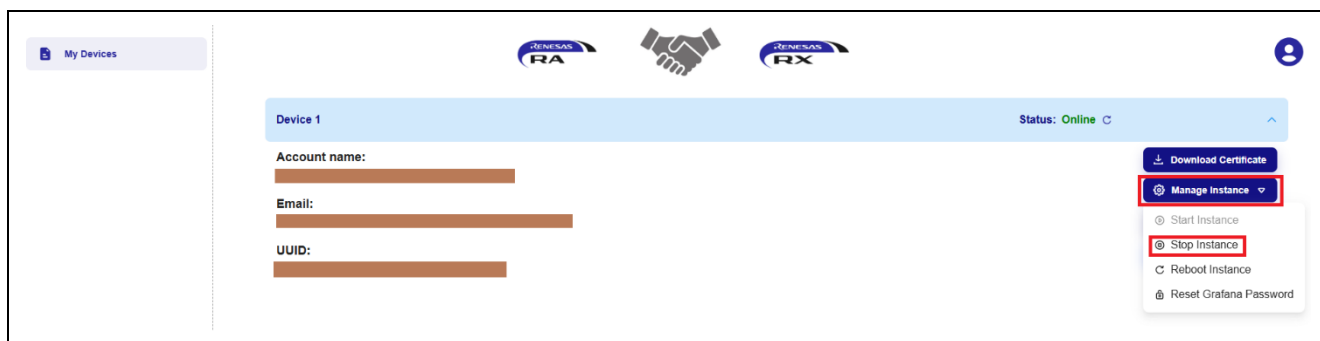


Figure 205. Stop EC2 instance

2. The screen prompt indicates the instance is stopping and the dashboard cannot be accessed.

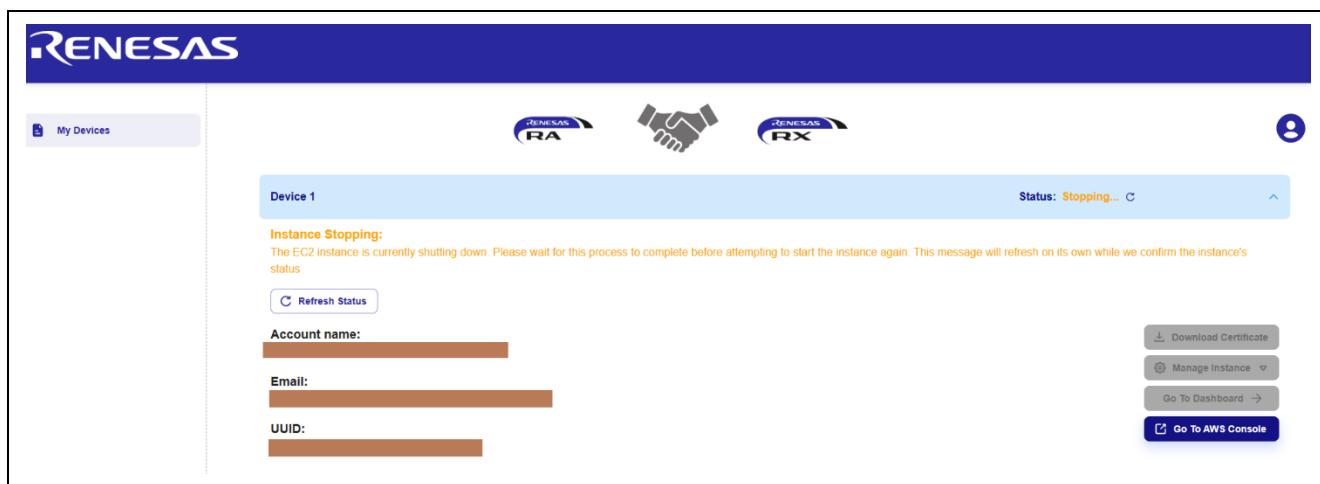


Figure 206. EC2 instance status

3. When the device is back in use, restart the EC2 instance.

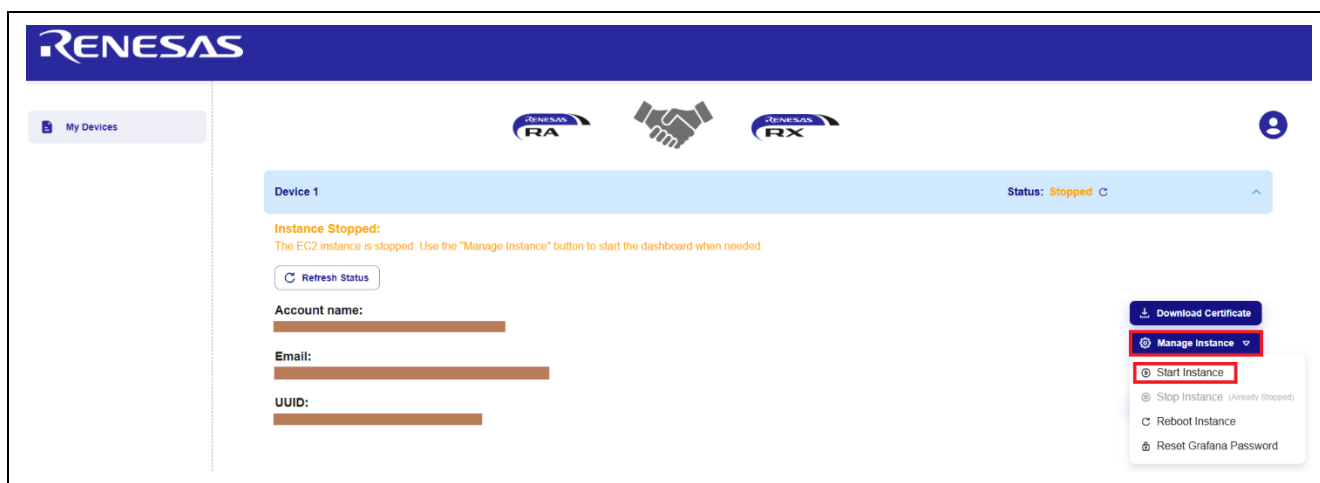


Figure 207. Restarting the EC2 Instance State

User can disable the EC2 instance directly on AWS account by following steps:

1. Access AWS account (Refer to Figure 193. Accessing AWS Account from the Dashboard)
2. From the **Services** menu, select **Compute** and then choose **EC2**.

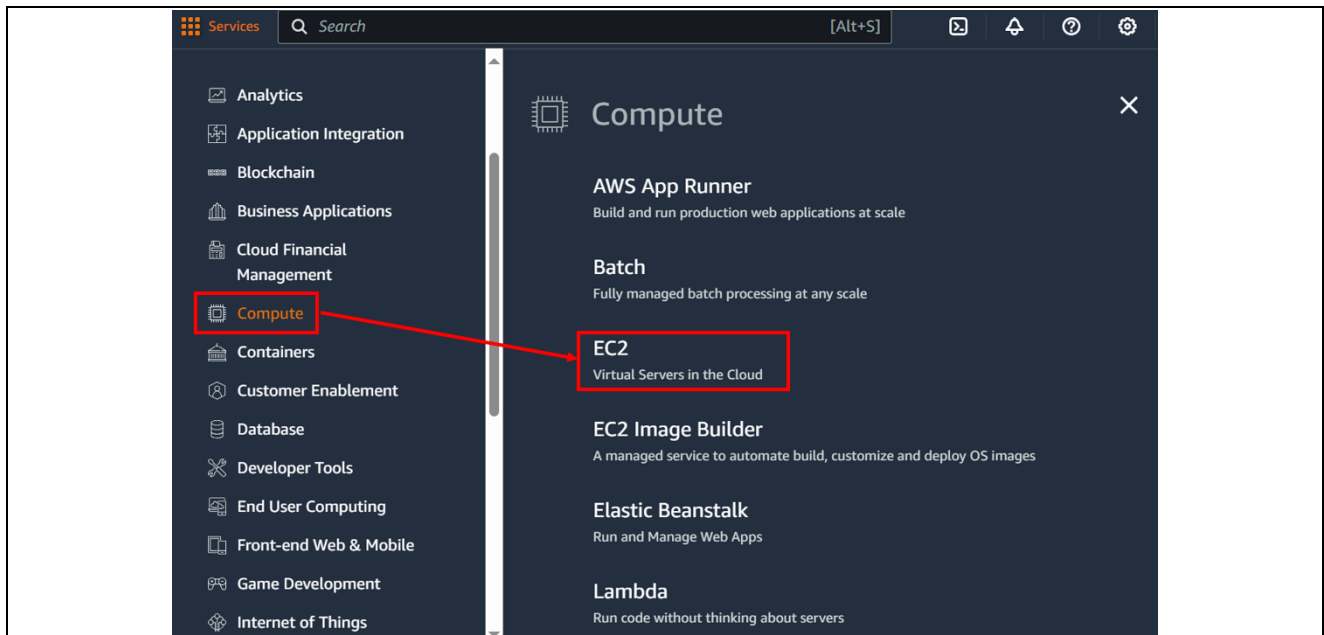


Figure 208. EC2 AWS Service

3. Choose the instance then change the **Instance State** to **Stop state**.

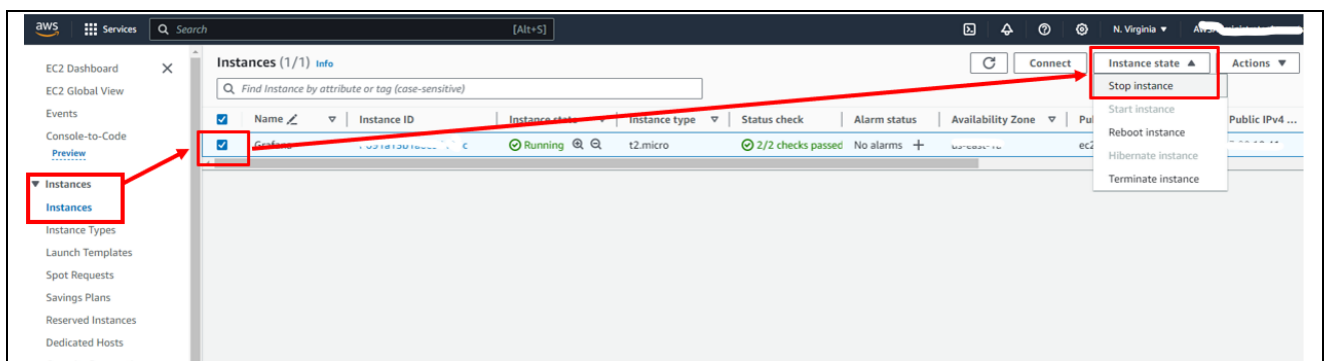


Figure 209. Disable Instance

8.8 Grafana dashboard display is different from the one in application note

Depending on the version of the cloud kit being used, you can choose one of the dashboard types: Renesas CK v1.0 or Renesas CK v2.0. **Renesas CK v2.0** is the default, if not choose Renesas CK v2.0.

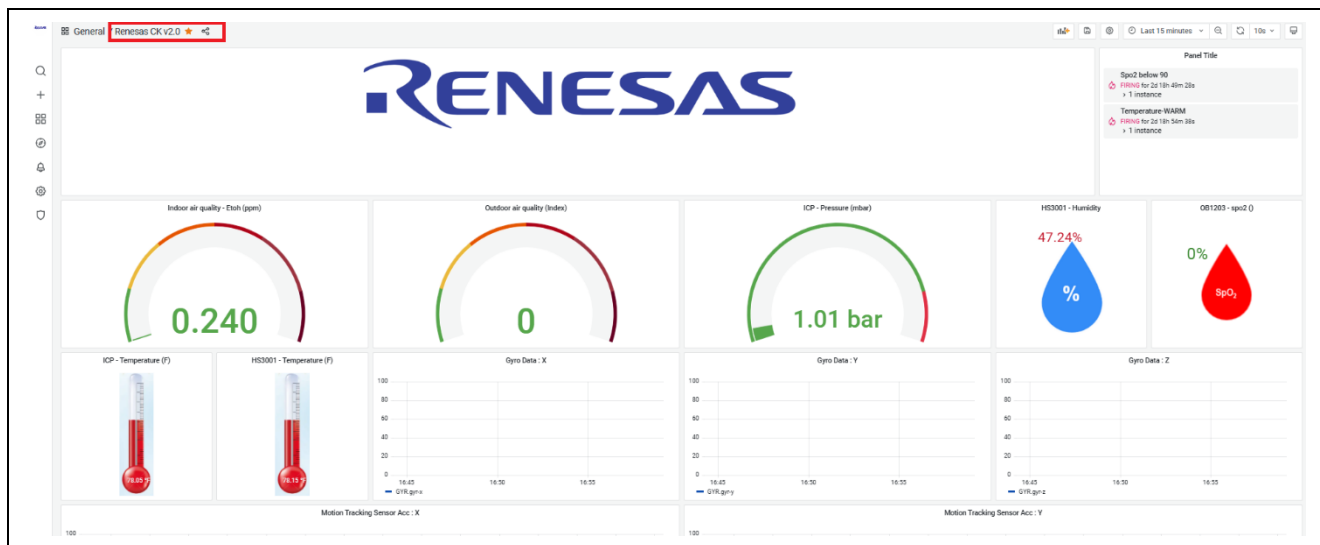


Figure 210. Renesas AWS Cloud Dashboard Types

Choose Renesas CK v2.0.

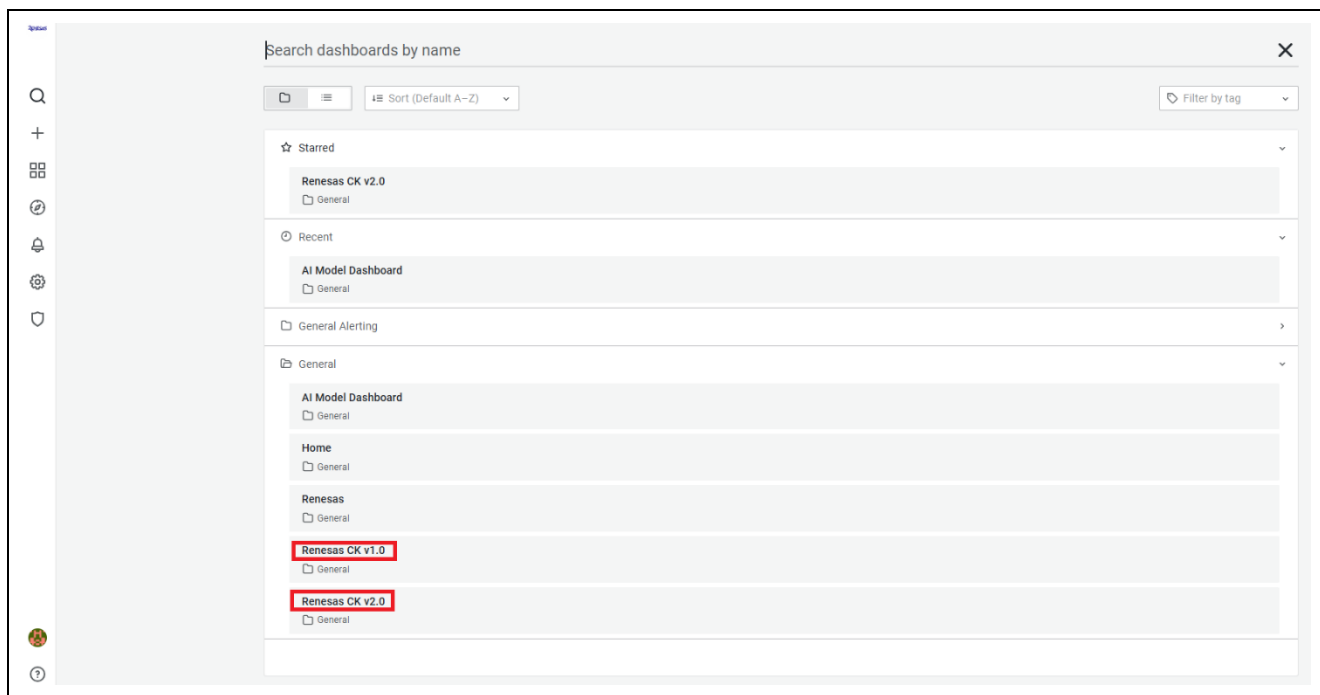


Figure 211. Choosing Renesas dashboards based on the cloud kit version

8.9 How to check the total amount spent in the AWS account

Access the AWS account from <https://cloud-ra-rx.awsapps.com/start/#/> using the dashboard credentials.

Go to **Account > Bills**.

User can choose the “**Billing period**” to see the amount spent during that period.

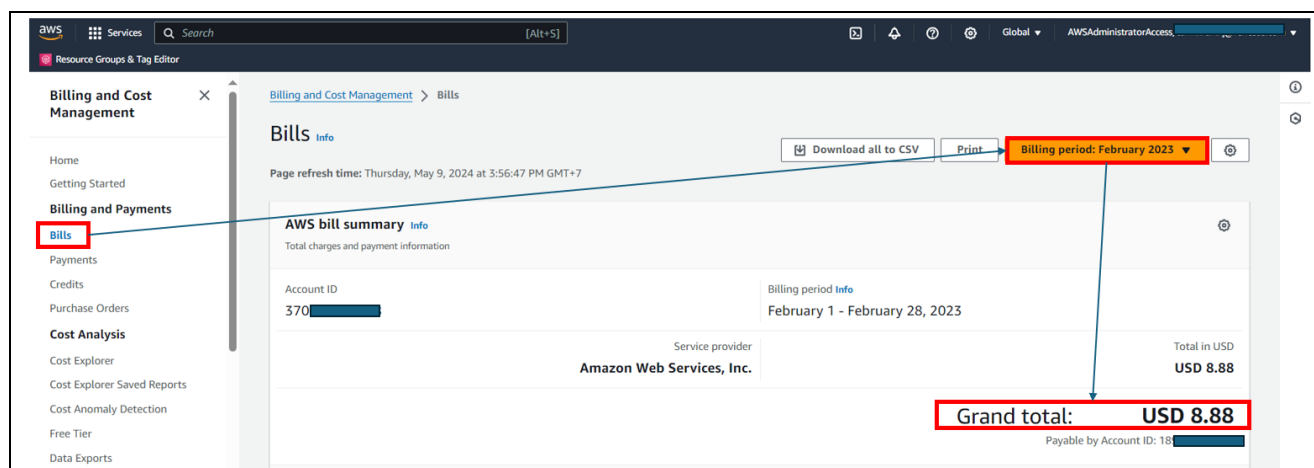


Figure 212. Check the amount spent in AWS Account

8.10 An error occurs when connecting to AWS

The AWS IoT information is not set yet or is set incorrectly. Please check and set AWS IoT information again. (5.3.1)

8.11 Command to create to create the initial firmware fails (OTA)

The cause of this issue is the Python installation folder is not set correctly in the Path variable or the encryption library is not installed.

Users have to re-install Python. Also, make sure that the Add python.exe to PATH check box is selected when you perform the step in 6.2.1 and install the encryption library.

8.12 Initial firmware cannot be written/ does not start. (OTA)

Make sure that the jumper on J16 of CK-RX65N v2 board is on pins 1-2 (debug mode) when writing initial firmware and on pin 2-3 (run mode) when starting the initial firmware.

8.13 Firmware does not start after starting the bootloader (OTA)

Please review the public key setting in the bootloader because it is not correctly set in the bootloader.

8.14 Firmware does not start after an OTA update (OTA)

Users can review the public key setting in the firmware because the public key is not set correctly in the firmware. If not, please review the device settings in the firmware and the boot loader.

Website and Support

Visit the following vanity URLs to learn about key elements of the RX family, download components and related documentation, and get support.

CK-RX65N v2 Kit Information	renesas.com/rx/ck-rx65n
RX&RA Cloud Solutions	renesas.com/cloudsolutions
RX Cloud solution web	renesas.com/rx-cloud
RX Product Information	renesas.com/rx
RX Product Support Forum	renesas.com/rx/forum
RX Driver Package	renesas.com/RDP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.21.25	—	Initial release

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.