Renesas RX Family

AWS Cloud Connectivity on CK-RX65N v2 with Wi-Fi DA16600 (CC-RX) – 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 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 OTA feature to update firmware via Cloud
- How to use Fleet Provisioning via Cloud

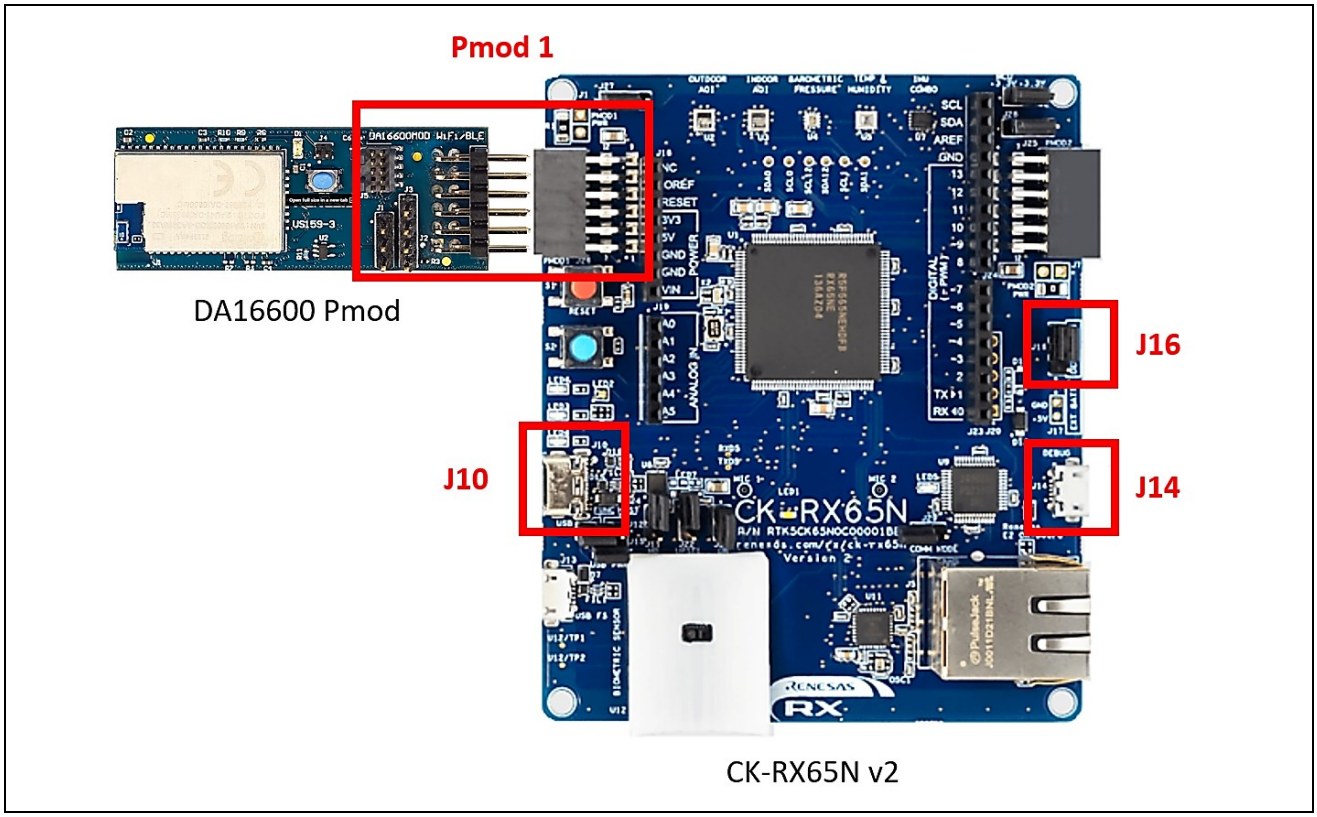

Figure 1. CK-RX65N v2 (with Wi-Fi DA16600 Pmod)

Contents

1. Terms

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 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 detail 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 detail at: US159-DA16600EVZ - Ultra-Low-Power Wi-Fi + Bluetooth® Low Energy Combo Pmod™ Board (Renesas Quick-Connect IoT) | Renesas
PC	Windows® 10 Google chrome	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	2024-01 or later
Compiler	CC-RX	V3.05
Communication Software	Tera term	Version 4.99
Emulator	E2 emulator Lite (on-board)	-
RTOS	AWS FreeRTOS	V202210.01
Python	-	V3.11.0 or later
Keygen tool	Win64 OpenSSL	V3.0.12
Flash programming tool	Renesas Flash Programmer	V3.12.00
Renesas Image Generator	Supplied with Firmware Update module Rev.2.01	V3.02

Note: For the GCC version software, refer to the Application Note “AWS Cloud Connectivity on CK-RX65N v2 with Wi-Fi DA16600 (GCC) – 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

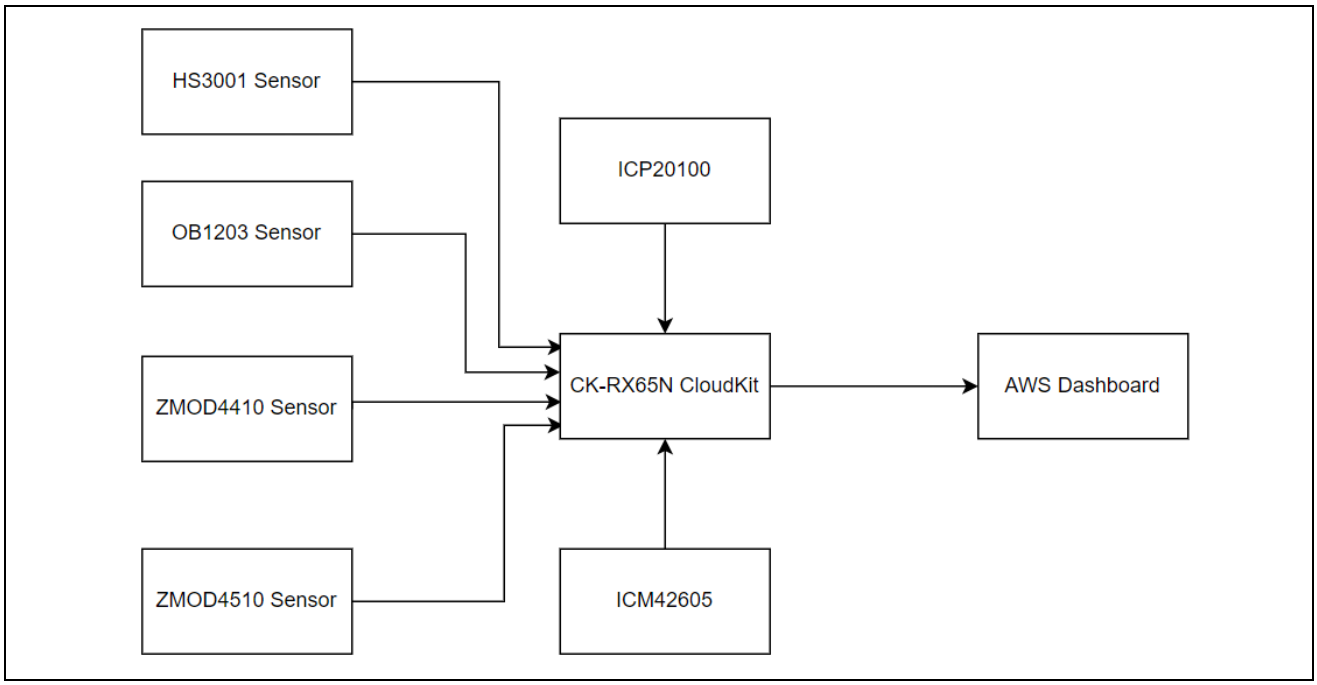

Figure 2. System Diagram

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.

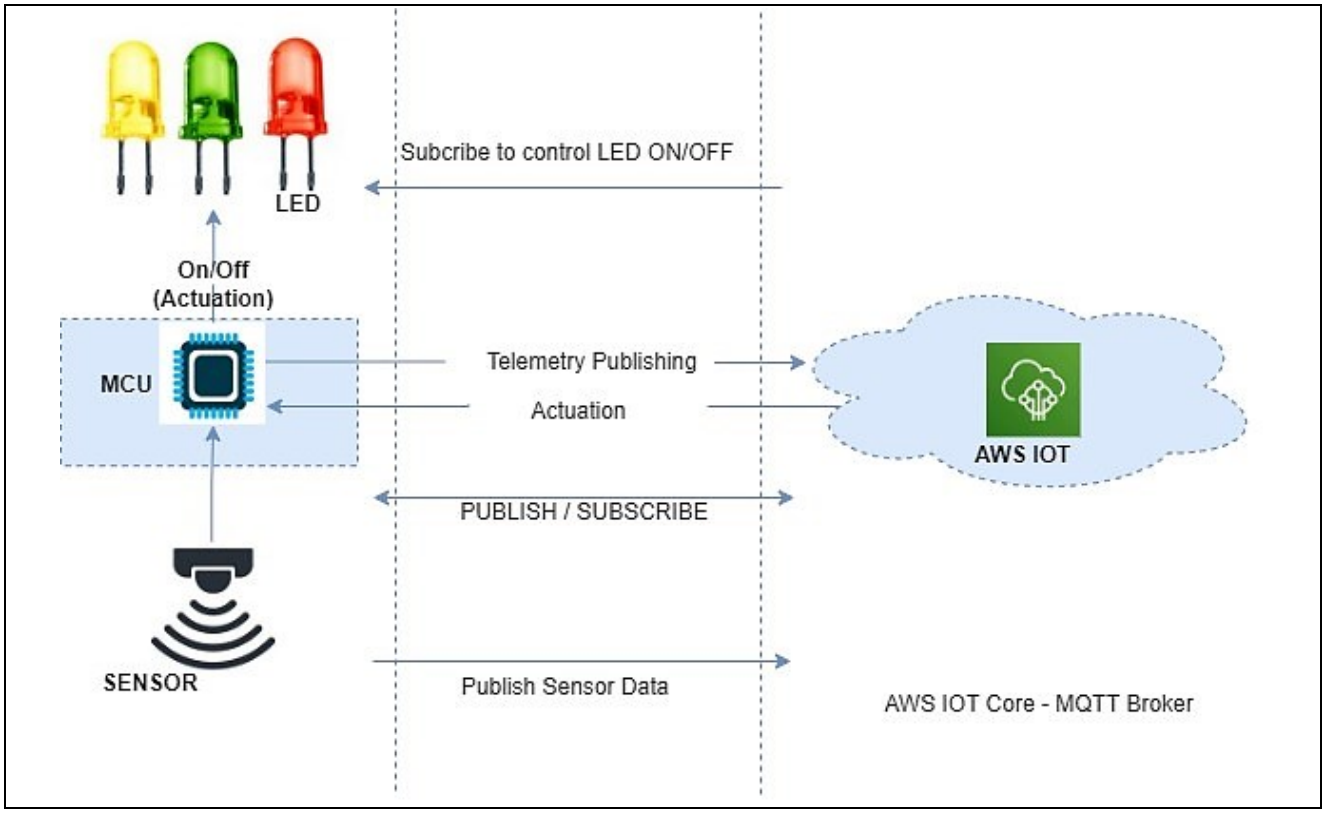

Figure 3. MQTT Publish/Subscribe to/from AWS IoT Core

Application also supports OTA over MQTT feature for updating new firmware (please refer to the section **6. OTA over MQTT),** and Fleet Provisioning (please refer to the section **Fleet Provisioning**) via AWS.

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 initialization code of the Wi-Fi connection, provisioning cloud credentials used in Cloud Connectivity, main function of 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.

No.	Filename	Purpose
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 console using UART
6.	src/application_code/CommandLine/console.h	Contains the function prototypes used to print data on 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 main menu on CLI
10.	src/application_code/CommandLine/menu_kis.h	Contains the function prototypes and macros used to get application's version, get UUID and help option for main menu on 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 application
15.	src/application_code/sensor_thread_entry.c	Contains the code for sensor thread (HS3001 + ICP20100 + ICM42605)
16.	src/application_code/ICM42605/ICM_42605.c	Contains the code for the 6-Axis MEMS Motion Tracking™ Sensor
17.	src/application_code/ICM42605/ICM_42605.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 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
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 the 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

No.	Filename	Purpose
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/softTimer.c	Contains the code for timer
44.	src/application_code/softTimer.h	Contains the common data structure's function prototypes for timer
45.	src/application_code/frtos_skeleton/ob1203_thread.c	Contains the ob1203 sensor thread (for oximeter sensor)
46.	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)
47.	src/application_code/frtos_skeleton/zmod_thread.c	Contains the ZMOD's thread (for Renesas ZMOD Internal Air Quality sensors)

No.	Filename	Purpose
48.	src/application_code/frtos_skeleton/task_function.h	Contains the common data structure's function prototypes for thread
49.	src/application_code/frtos_startup/freertos_object_init.c	Contains the source code for FreeRTOS thread
50.	src/application_code/frtos_startup/freertos_start.c	Contains FreeRTOS user-defined functions
51.	src/application_code/frtos_startup/freertos_start.h	FreeRTOS's user-defined functions header file
52.	src/application_code/frtos_config/*.h	Contains FreeRTOS configuration header file.
53.	src/application_code/sensorsData.h	Contains the common data structure's function prototypes for sensors
54.	Demos/SimplePubSub/simple_pub_sub_task.c	Contains code and functions used in MQTT interface for Cloud Connectivity.
55.	Demos/mqtt_agent/mqtt_agent_task.c	Contains the code for running the MQTT task
56.	Demos/OtaOverMqtt/OtaOverMqttDemoExample.c	Contains function for running OTA over MQTT
57.	Demos/Fleet_Provisioning_With_CSR_Demo	Contains function for running Fleet Provisioning
58.	Demos/cli/serial.c	Contains function for serial communication.
59.	Demos/cli/serial.h	Contains the common data structure's function prototypes for serial.c
60.	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.

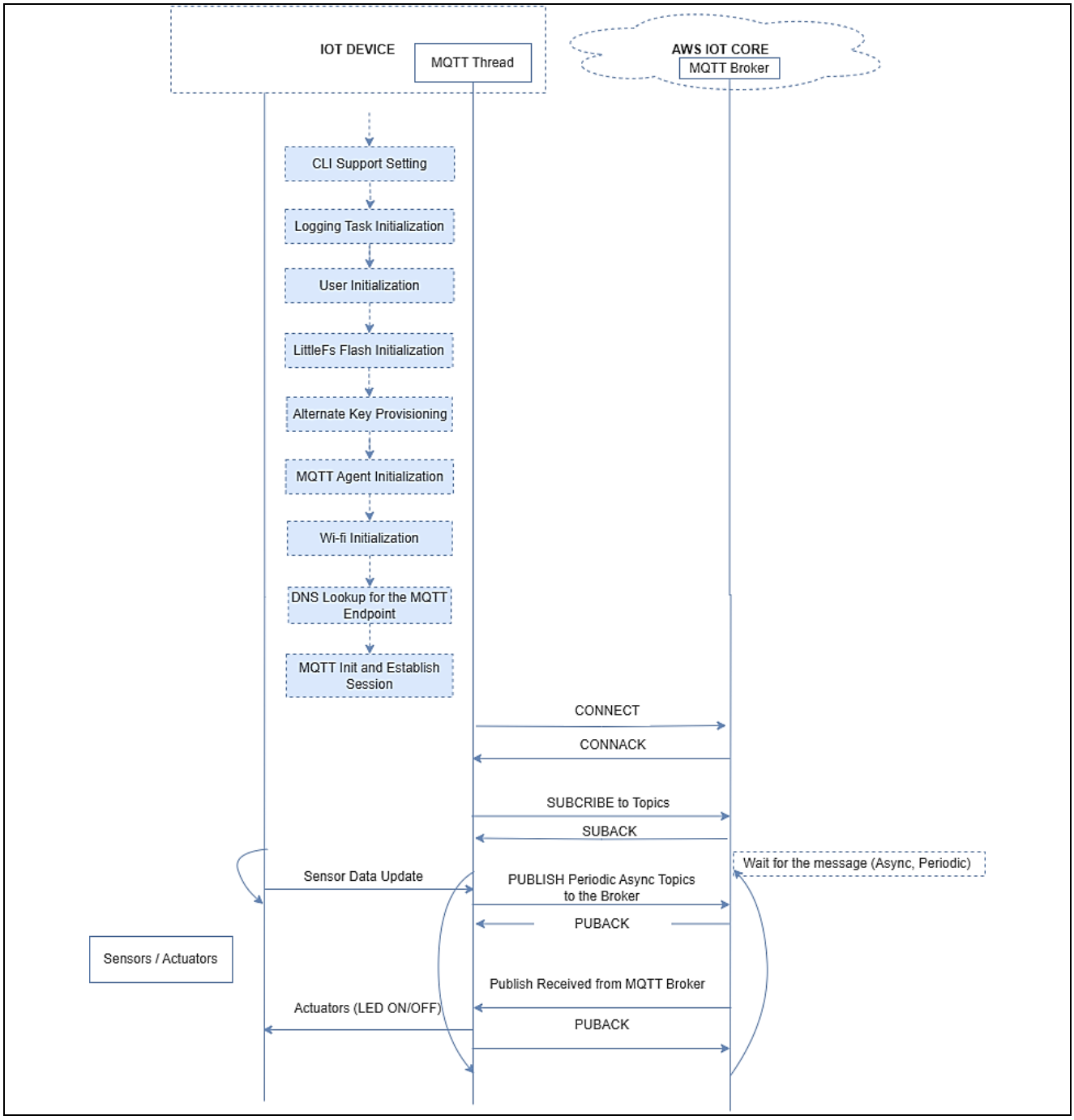

Figure 4. Application Example Implementation Details

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 establishes 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 SUBCRIBE 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 cloud with 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) in the device. When the network is down, the application will start re-connecting by resetting DA16600 Pmod and re-connecting to MQTT broker.

5. Connection to AWS

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

Renesas provides the 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 of way to connect AWS account.

- **Case 1:** For the users who want to use 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 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 account with application.

5.1 Hardware Setup and Import 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”**

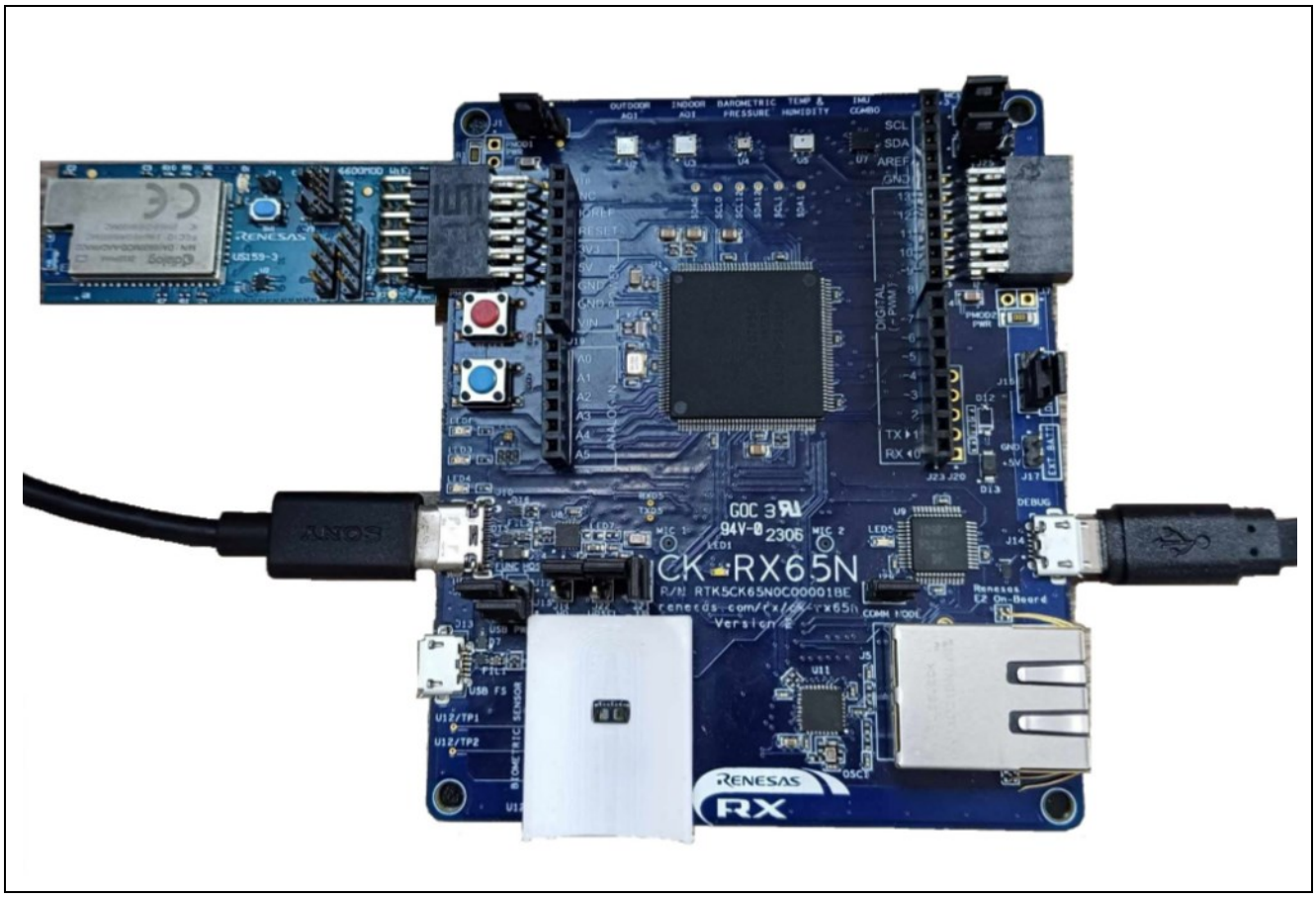

Figure 5. Connecting the USB and DA16600 Pmod

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.

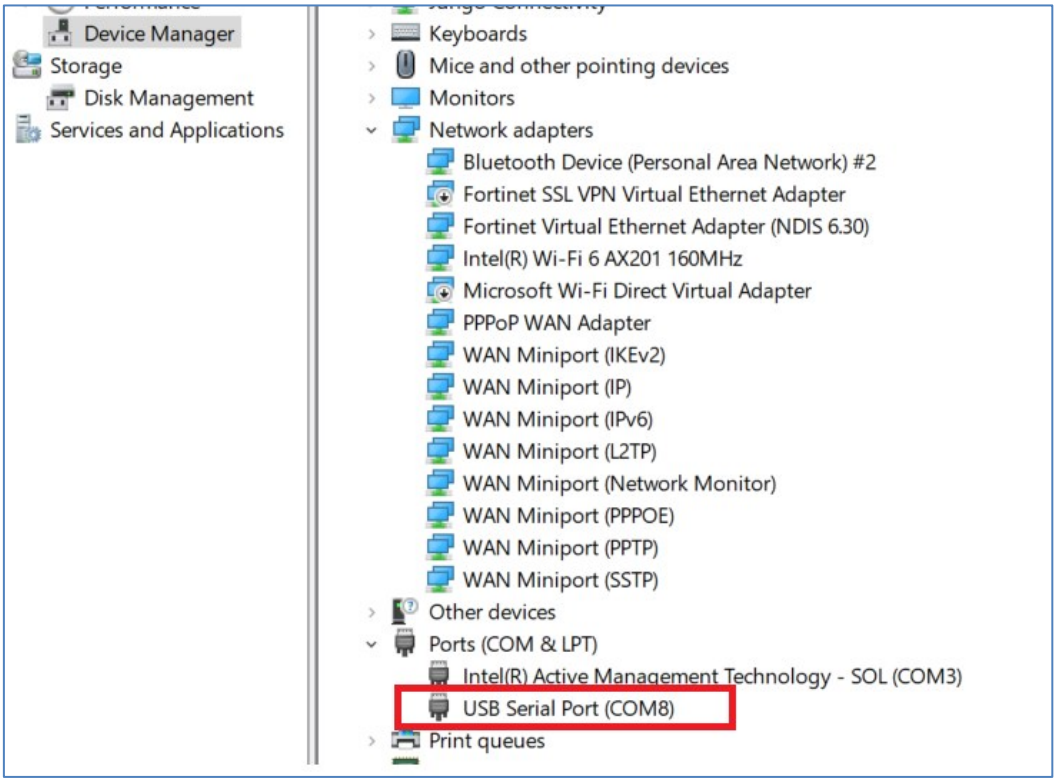

Figure 6. USB Serial Device in Windows Device Manager

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

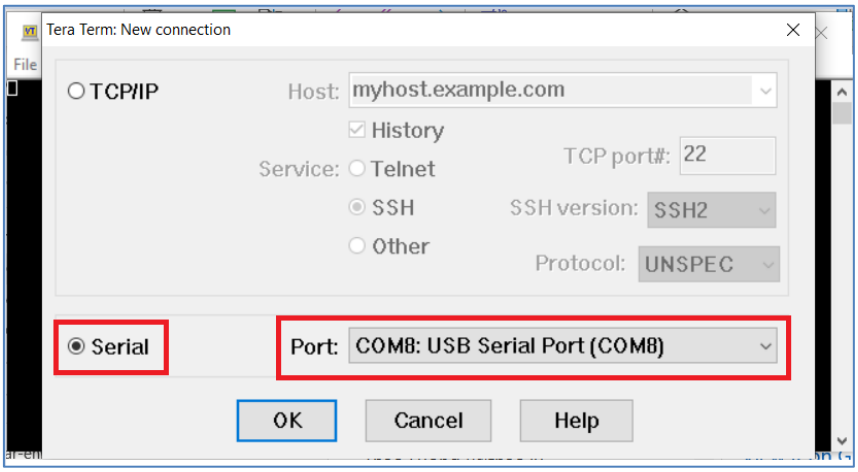

Figure 7. Selecting the Serial Port on Tera Term

2. Using the **Setup** menu, select **Setup** > **Terminal…** and select "**AUTO**" as Receive and, select "**CR**" as Transmit, as shown in Figure 8.

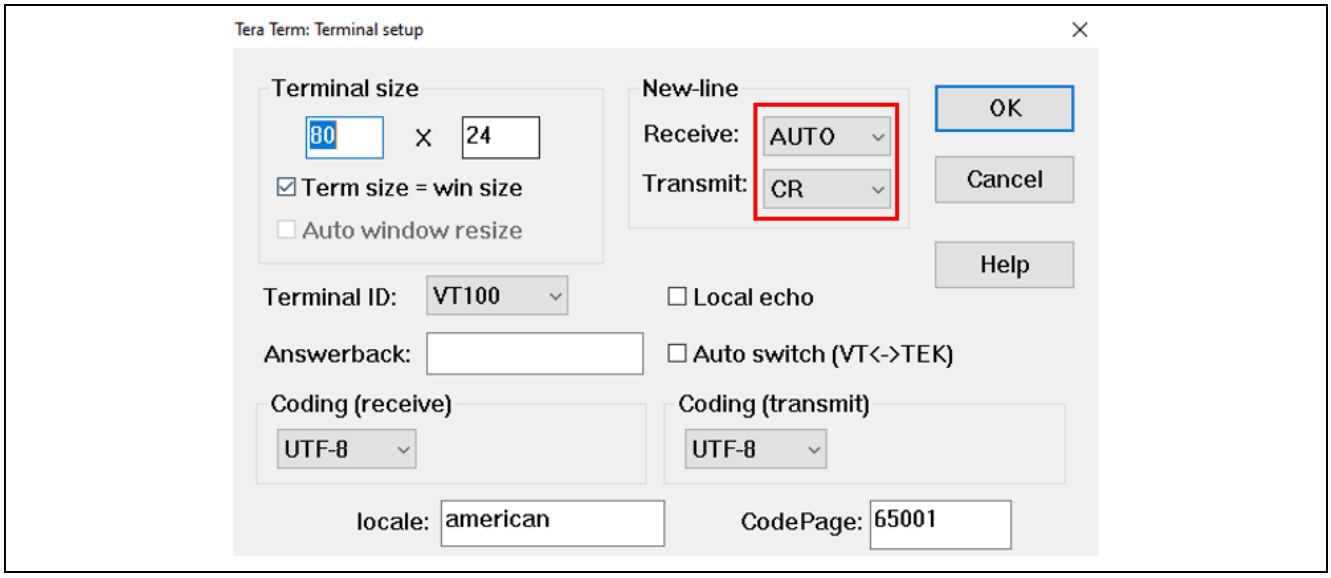

Figure 8. Select Receive: "Auto" and Transmit: "CR" on the Terminal Setting

3. Using the **Setup** menu pull-down, select **Serial port…** and ensure that the speed is set to 115200, as shown in Figure 9.

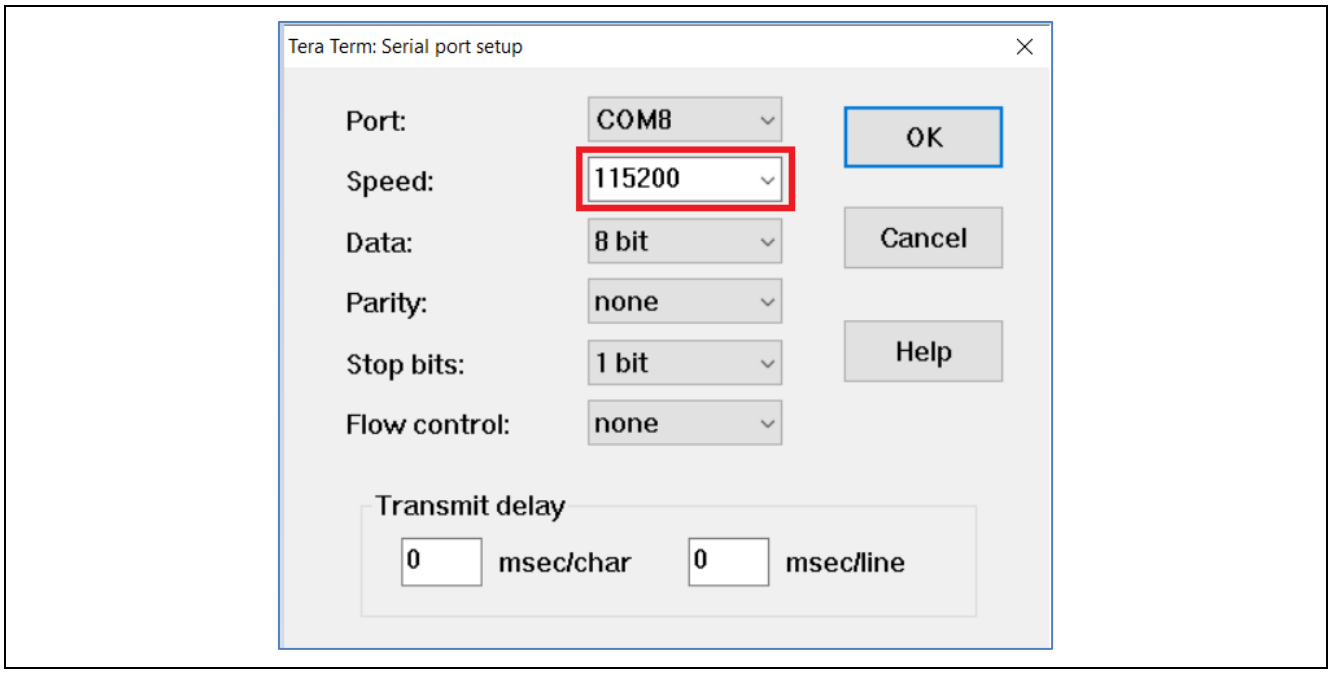

Figure 9. Select 115200 on the Speed Pulldown

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 copy them to the C drive.
   Please 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**.

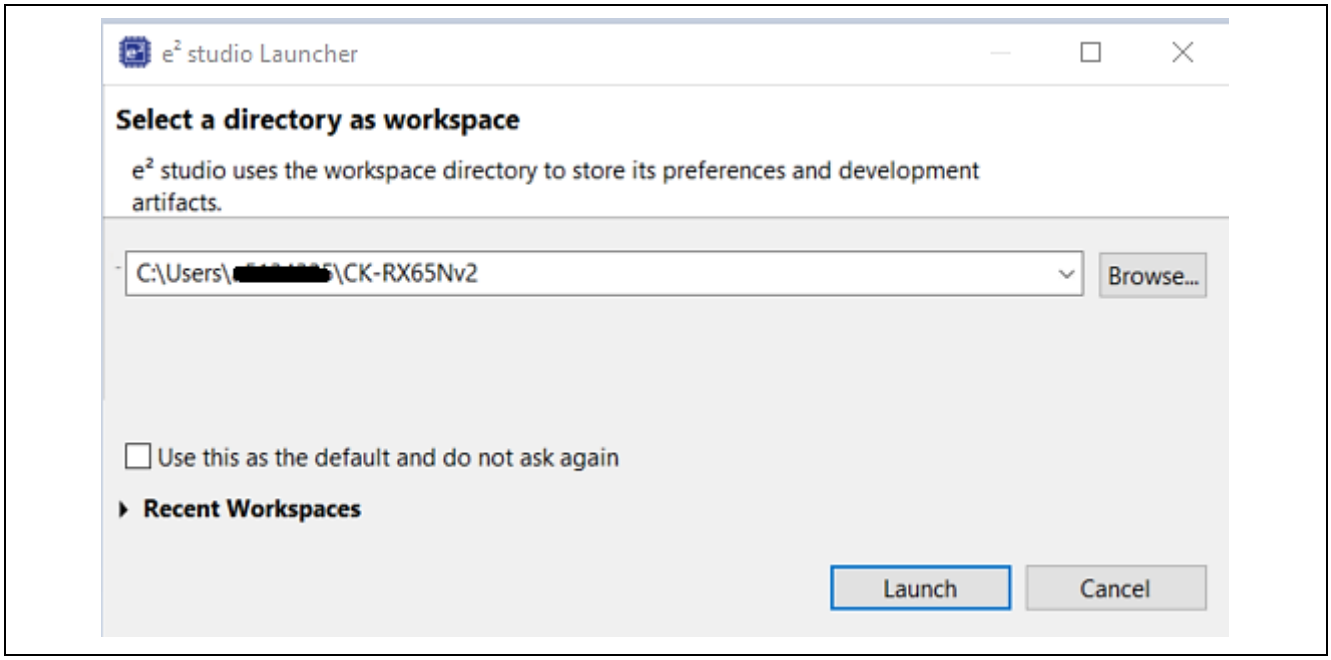

Figure 10. Launch e² studio

3. Select **File** > **Import…**

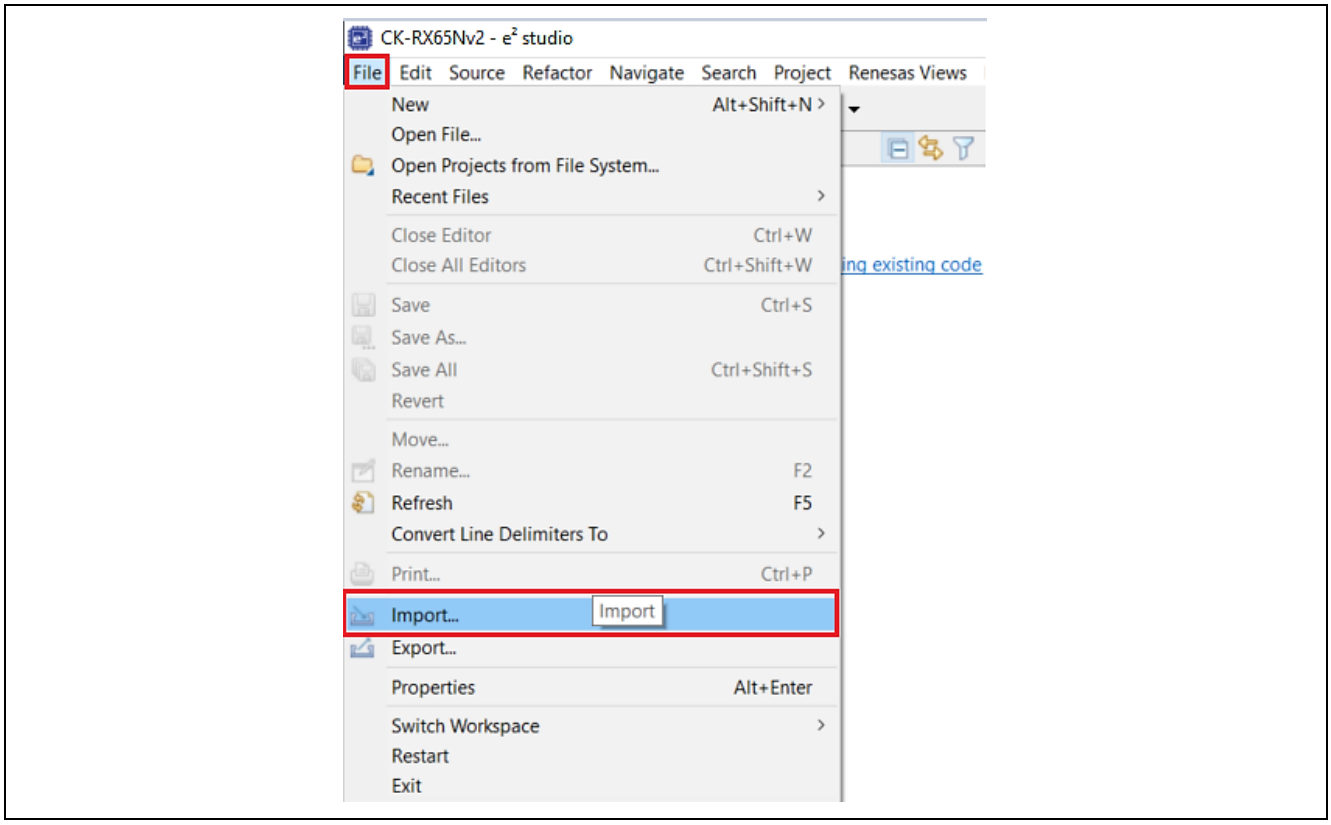

Figure 11. Select Import

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

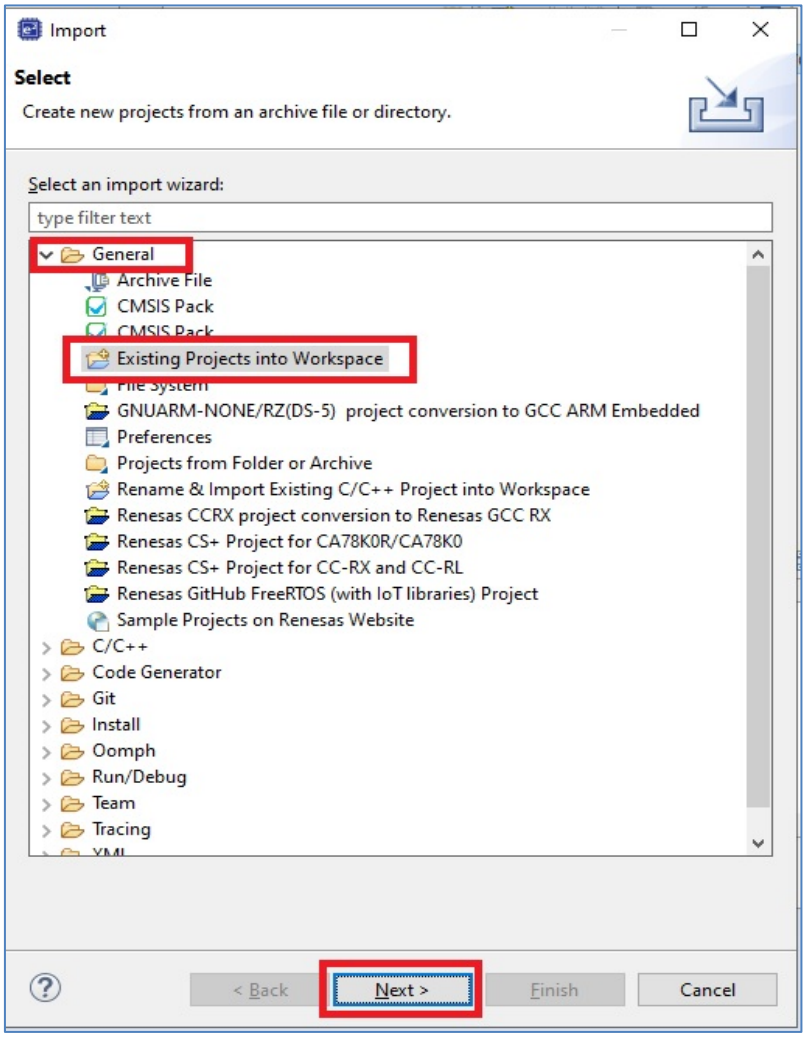

Figure 12. Select Existing Projects into Workspace

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

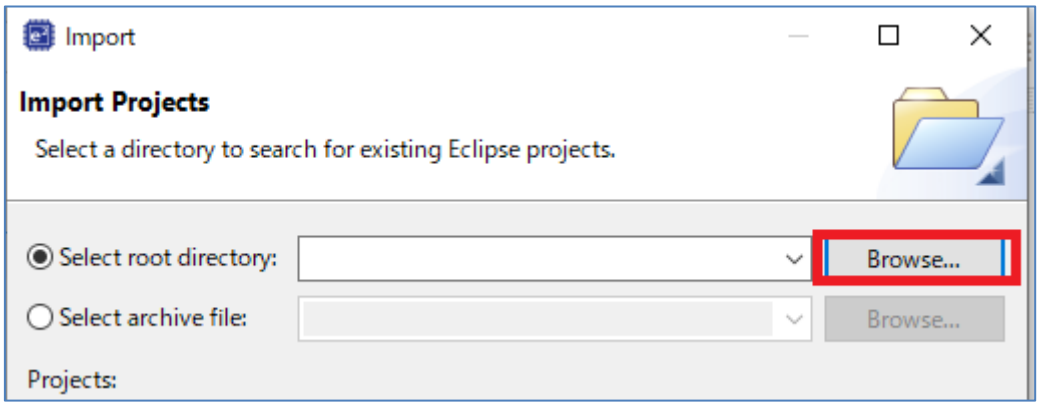

Figure 13. Find the Project

Please go to the **"[Project Root folder]\Projects\aws_da16600_ck_rx65n"** folder

Project Details

Project Name	Compiler	Connectivity
aws_da16600_ck_rx65n	CC-RX	Wi-Fi

Open the "**[Project Root folder]\Projects\aws_da16600_ck_rx65n\e2studio_ccrx**" folder.

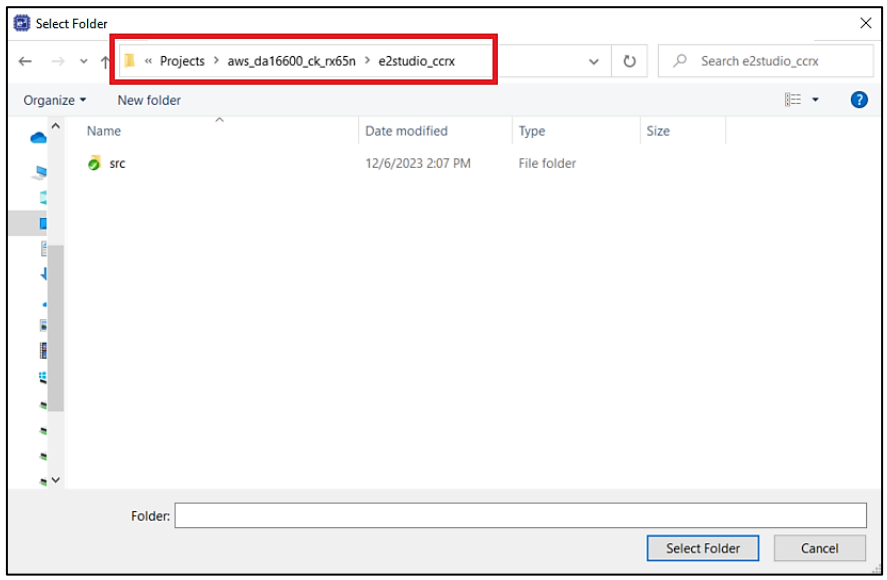

Figure 14. Select the Project Folder

Finally, click **Finish**.
Note: Make sure "Copy projects into workspace" is **unchecked**.

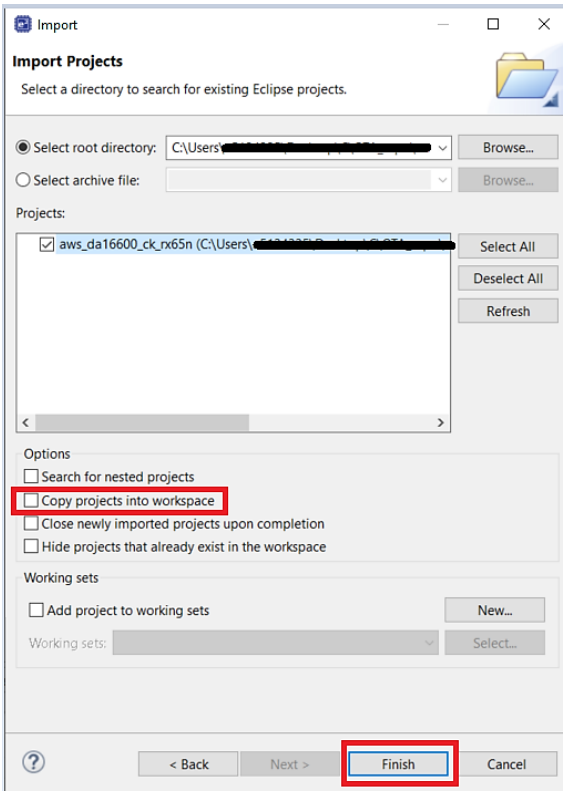

Figure 15. Import the Project

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**.
   Note: If the user's environment does not have the FIT component's version match with the application, please download it by choosing **aws_da16600_ck_rx65n.scfg** > **Components** > **downloading it**

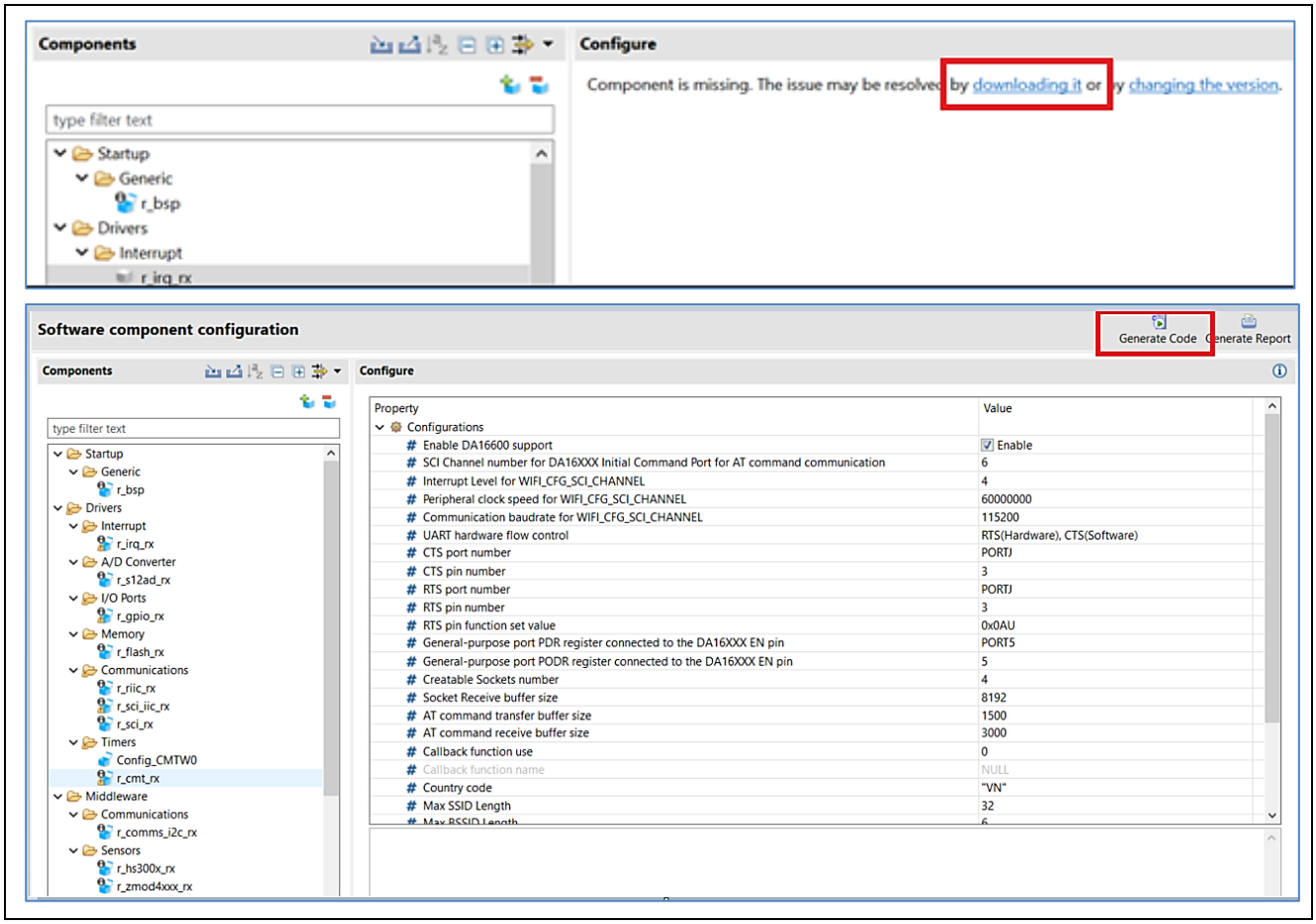

Figure 16. Generate Code

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**

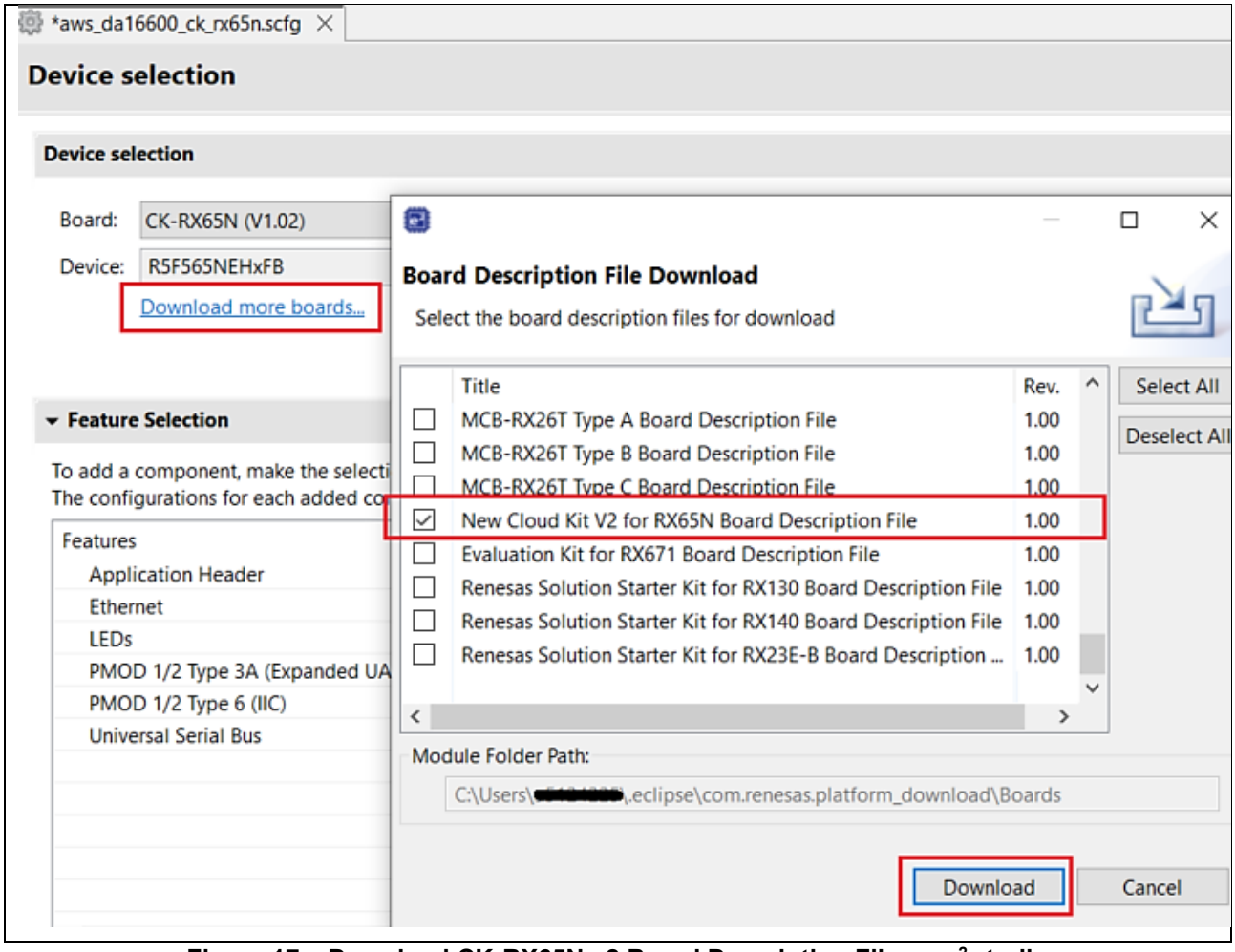

Figure 17. Download CK-RX65N v2 Board Description File on e² studio

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

Table 6. Components Configuration

No	Component	Configuration
1	**Startup→Generic→r_bsp**	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**
		User stdio charget function name: **my_sw_charget_function**
		Enable user stdio charput function: **Use BSP charput() function**
		User stdio charput function name: **my_sw_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**

No	Component	Configuration
		Interrupt priority for serial terminal: **Priority level 15 (highest)**
2	**Drivers→Interrupt→r_irq_rx**	Locking function for IRQ APIs: **Enable**
		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**	Resources→S12AD→S12AD1: ✓ →AN115 Pin: ✓ →AN117 Pin: ✓
4	**Drivers→I/O Ports→r_gpio_rx**	-
5	**Drivers→Memory→r_flash_rx**	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**	-
7	**Drivers→Communications→r_riic_rx**	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**	MCU supported channels for CH12: **Supported**
		Resource→SCI SCI12: ✓ →SSCL12 Pin: ✓ **Used** →SSDA12 Pin: ✓ **Used**
9	**Drivers→Communications→r_sci_rx**	Include software support for channel 5: **Include**
		Include software support for channel 6: **Include**

No	Component	Configuration
		ASYNC mode TX queue buffer size for channel 5: **80**
		ASYNC mode TX queue buffer size for channel 6: **2180**
		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	**Drivers→Timers→Config_CMTW0**	Count clock setting: **PCLK/8**
		Timer counter size: **32 bits**
		Counter clear: **CMWCNT is cleared by CMWCOR**
		Internal value: **1ms**
		Register value: **7499**
		Priority: **Level 15**
11	**Drivers→Timers→r_cmt_rx**	CMT interrupts priority level: **5**
12	**Middleware→ Communications→r_comms_i2c_rx**	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**

No	Component	Configuration
		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**
		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**
13	**Middleware→Sensors→r_hs300x_rx**	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**
14	**Middleware→Sensors→r_zmod4xxx_rx**	Number of ZMOD4xxx Sensors: **2**
		Operation mode of ZMOD4XXX Sensor0: **IAQ 1st Gen. (Continuous)**
		I2C Communication device No. for ZMOD4XXX sensor devices: **I2C Communication Device1**
		I2C callback function for ZMOD4XXX sensor device0: **zmod4xxx_user_i2c_callback0**
		IRQ callback function for ZMOD4XXX sensor device0: **zmod4xxx_user_irq_callback0**
		Enable IRQ from ZMOD4XXX sensor device 0: **Enabled**
		IRQ number for ZMOD4XXX sensor device0: **IRQ14**
		IRQ interrupt priority for ZMOD4XXX sensor device0: **10**
		Operation mode of ZMOD4XXX Sensor1: **OAQ 1st Gen.**
		I2C Communication device No. for ZMOD4XXX sensor devices: **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 interrupt priority for ZMOD4XXX sensor device0: **5**
15	**Middleware→Generic→r_byteq**	Memory allocation for queue control blocks: **Static memory allocation**
		Number of static queue control block: **32**
16	**Middleware→Generic→r_fwup**	Select the update mode: **Dual bank**

No	Component	Configuration
		Select the function mode: **user for User program**
		Main area start address: **0xFFF00000**
		Buffer area start address: **0xFFE00000**
		Install area size: **0xF0000**
17	**MIddleware→Generic→r_ob1203_rx**	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: o**b1203_comms_i2c_callback**
		Enable IRQ from OB1203 sensor device0: **Enabled**
		IRQ callback function for OB1203 sensor device0: **ob1203_irq_callback**
		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**
18	**Middleware→Generic→r_wifi_da16xxx**	Enable DA16600 support: **Enable**
		SCI Channel number for DA16XXX Initial Command Port 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**
		General-purpose port PDR register connected to the EN pin: **PORT 5**
		General-purpose port PODR register connected to the EN pin: **5**
		Creatable sockets number: **4**
		Socket Receive buffer size: **8192**
		Country code: **“VN”** **Note:** Depending on the user’s country

<table>
<tr><th>No</th><th>Component</th><th>Configuration</th></tr>
<tr><td></td><td></td><td>Timezone offet in hours (-12 ~ 12): 7
Note: Depending on the user’s time zone</td></tr>
<tr><td></td><td></td><td>Use FreeRTOS logging functionality: Disable</td></tr>
<tr><td rowspan="16">19</td><td rowspan="16">RTOS→RTOS Kernel→FreeRTOS_Kernel</td><td>RTOS scheduler: Preemptive</td></tr>
<tr><td>Maximum number of priorities to the application task: 7</td></tr>
<tr><td>The frequency of the RTOS tick interrupt: (TickType_t) 1000</td></tr>
<tr><td>The size of the stack used by the idle task: 768</td></tr>
<tr><td>The configTOTAL_HEAP_SIZE_N: 256</td></tr>
<tr><td>The maximum permissible length of name: 12</td></tr>
<tr><td>Idle should yield: ✓ Used</td></tr>
<tr><td>Mutex functionality: ✓ Used</td></tr>
<tr><td>Counting semaphore functionality: ✓ Enable</td></tr>
<tr><td>Software timer functionality: ✓ Enable</td></tr>
<tr><td>Priority of the software timer task: 6</td></tr>
<tr><td>The length of the software timer command queue: 5</td></tr>
<tr><td>Kernel interrupt priority: 1</td></tr>
<tr><td>Maximum syscall interrupt priority: 4</td></tr>
<tr><td>Tick vector: _CMT0_CMI0</td></tr>
<tr><td>Record stack high address: ✓ Enable</td></tr>
<tr><td>Support dynamic allocation: ✓ Enable</td></tr>
<tr><td>Support static allocation: ✓ Enable</td></tr>
<tr><td>20</td><td>RTOS→RTOS Object→FreeRTOS_Object→</td><td>
<table>
<tr><th>Initialize</th><th>Task Code</th><th>Priority</th><th>Stack Size</th></tr>
<tr><td>kernel start</td><td>ob1203_thread</td><td>3</td><td>1024</td></tr>
<tr><td>kernel start</td><td>zmod_thread</td><td>2</td><td>1024</td></tr>
<tr><td>kernel start</td><td>sensor_thread</td><td>2</td><td>2048</td></tr>
</table>
</td></tr>
</table>

7. Data Publishing Interval Settings (Optional)
   Data publish interval can be set by the user. Default publishes interval time 2 sec.

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

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

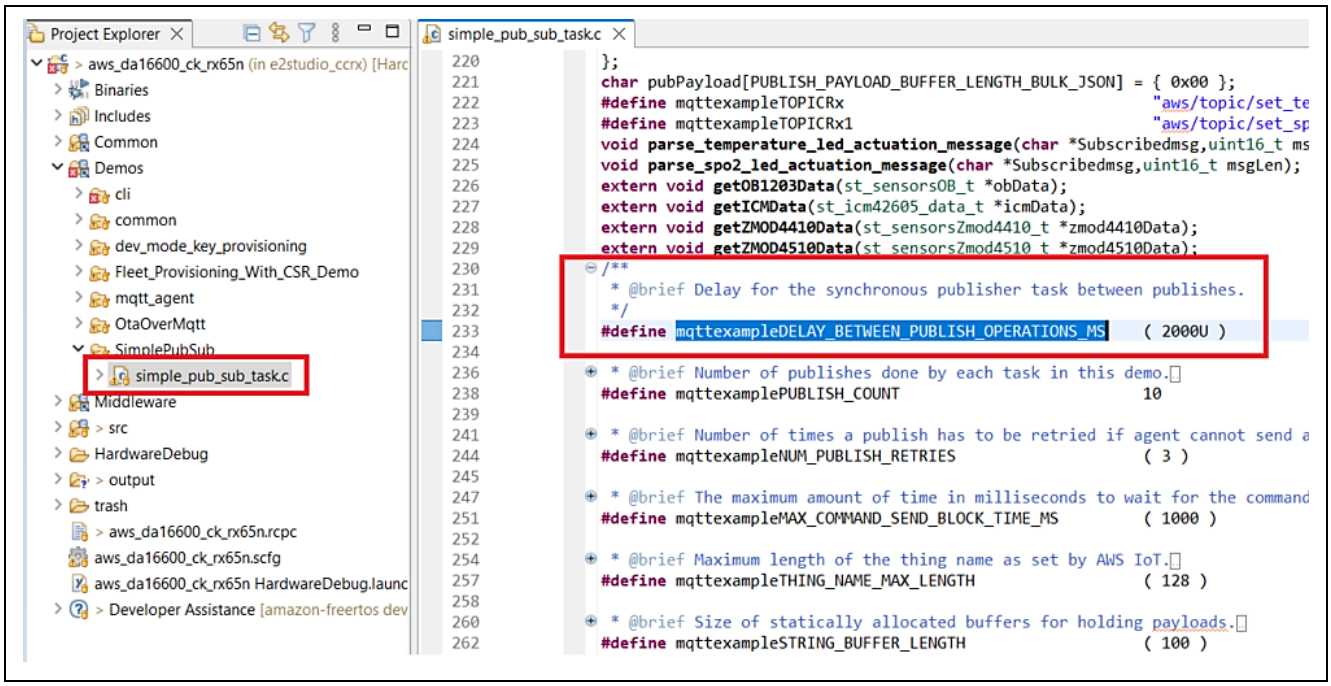

Figure 18. Data Publishing Interval Settings (Optional)

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

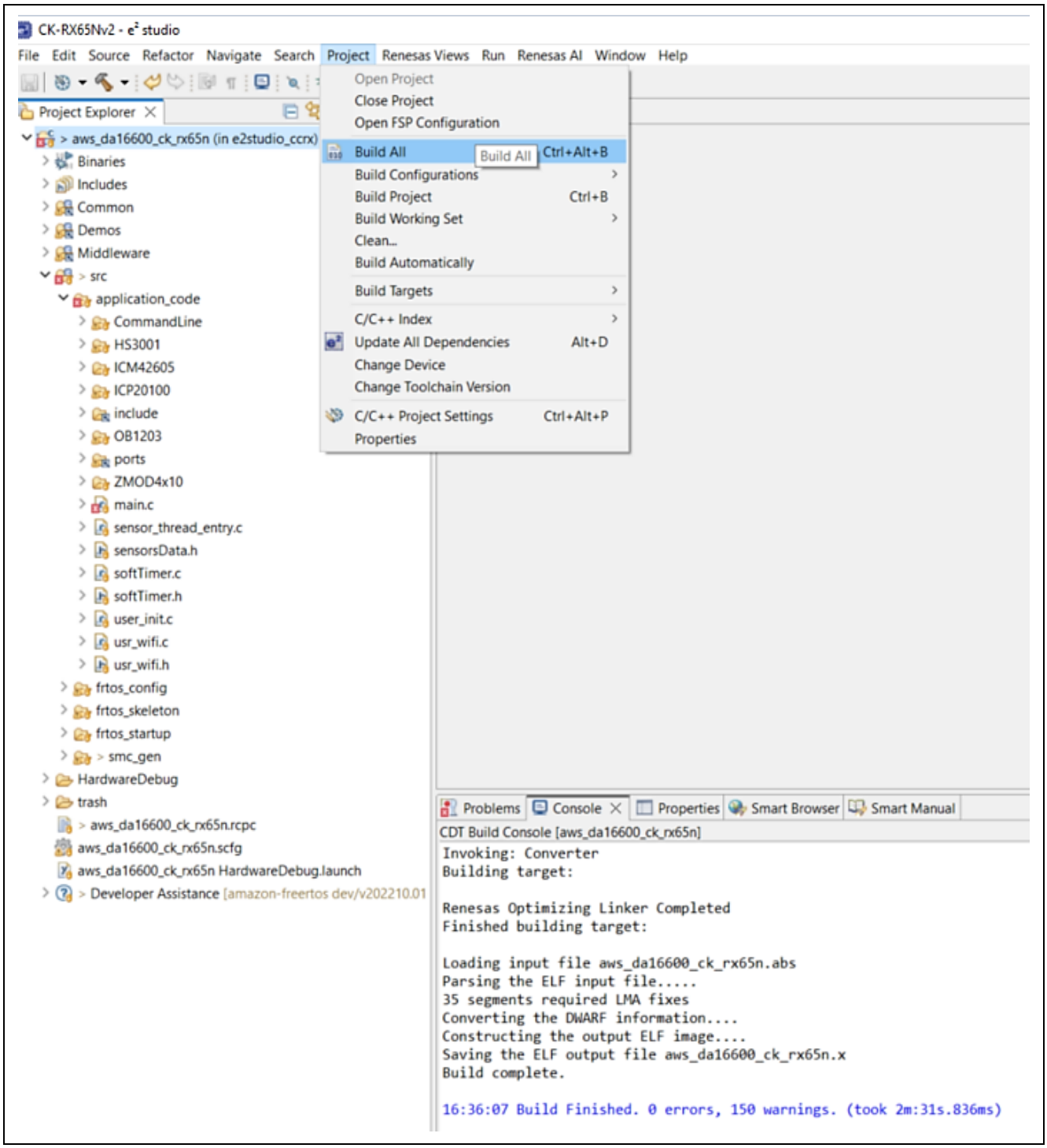

Figure 19. Build the Project

9. Debug configuration and load image to board:
   Open Debug configuration: Right click on project **> Debug As > Debug Configurations**…

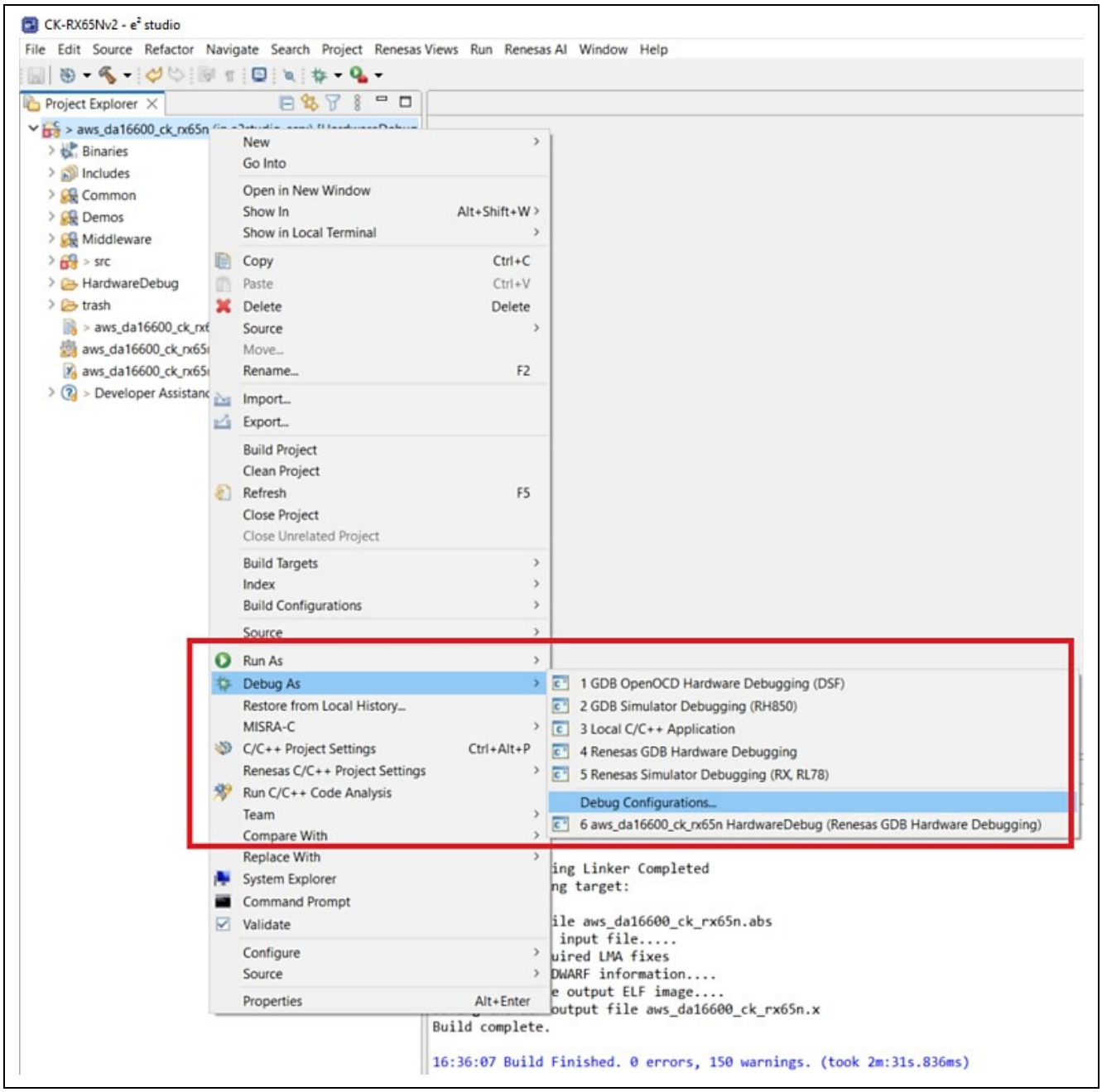

Figure 20. Configuration Debugger 1/2

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

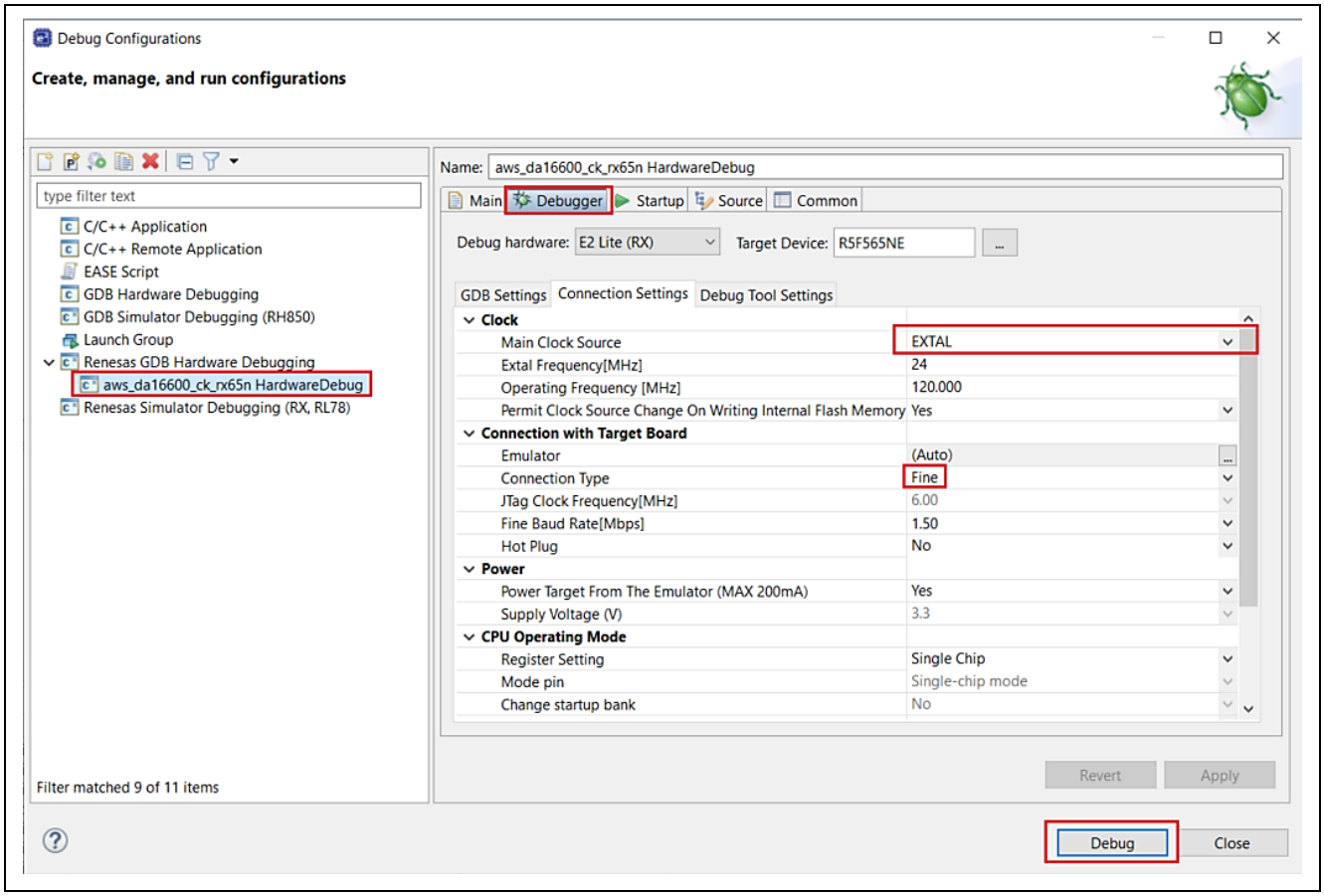

Figure 21. Configuration Debugger 2/2

5.1.4 Running the Application Project

To run the Application project, use the following instructions.

In Serial port console of the PC already setup in section 5.1.2, Tera Term, the console will display as below.

Figure 22. Start the Application

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

Note: After 10s, the application will run automatically with option **"6. Run Sensor App with MQTT"** in application's menu. User can modify this time's value by changing the value of **WAIT_USER_TIME** macro in file: **Projects\aws_da16600_ck_rx65n\e2studio_ccrx\src\application_code\CommandLine\menu_main.h** before building the project:

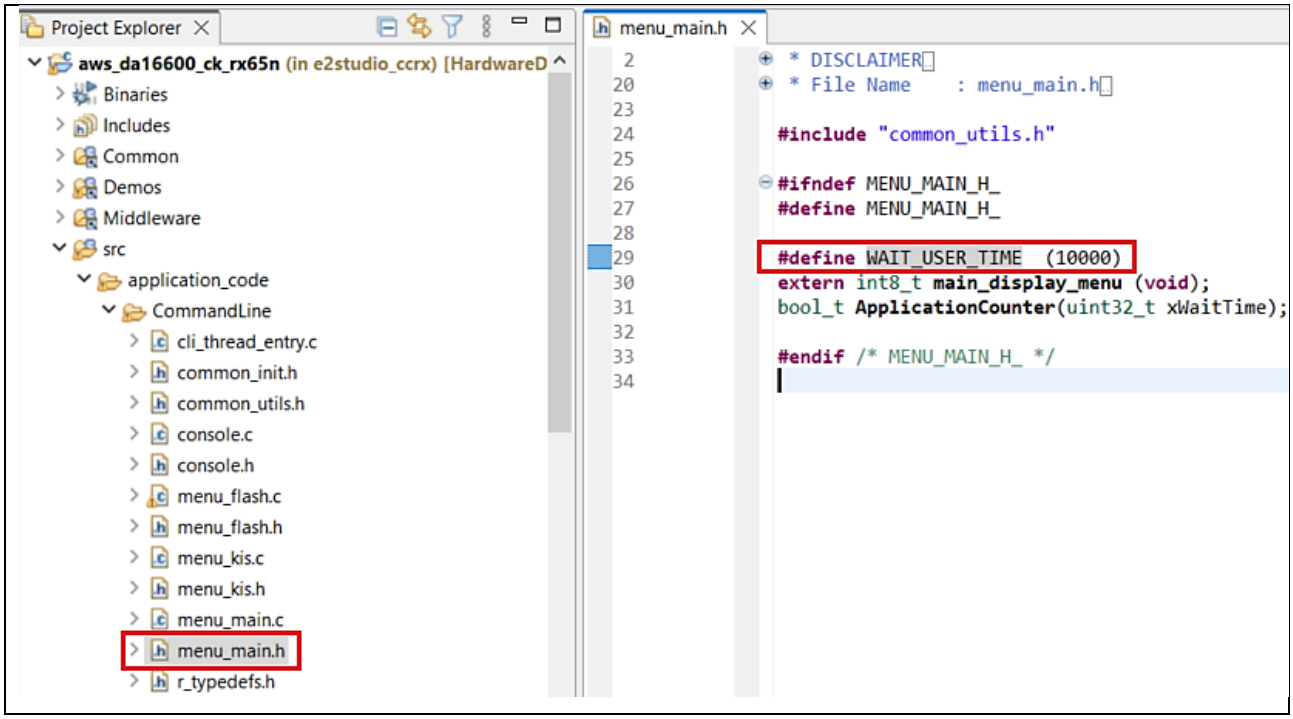

Figure 23. Modify macro to Wait for User Input!

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

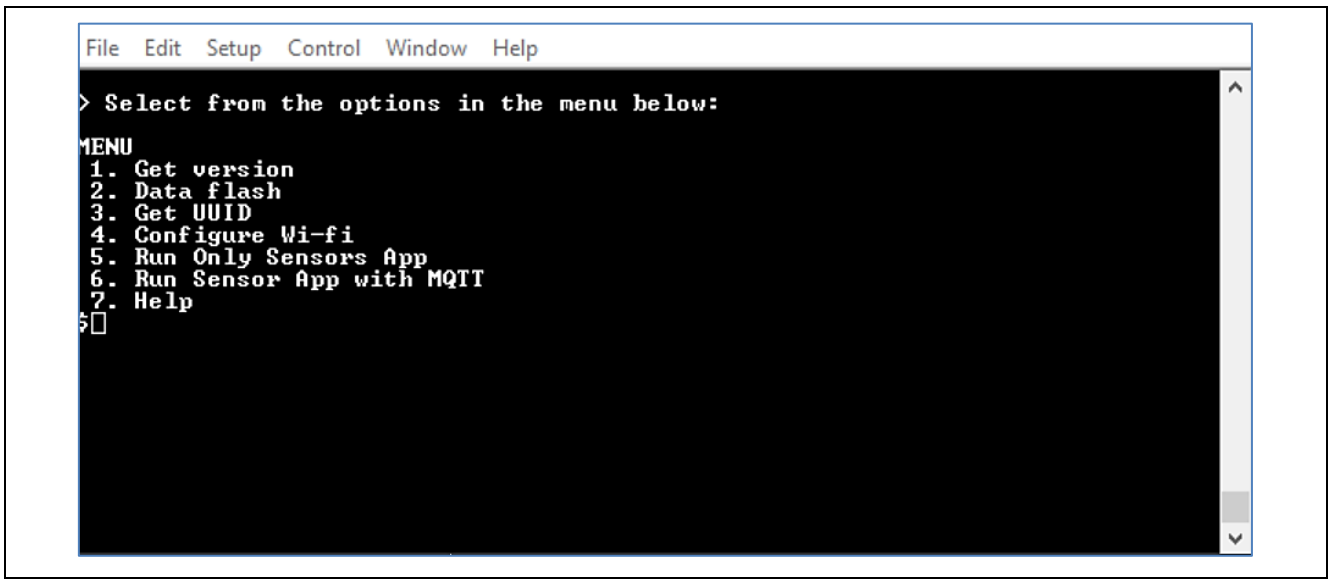

Figure 24. Main Menu

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.

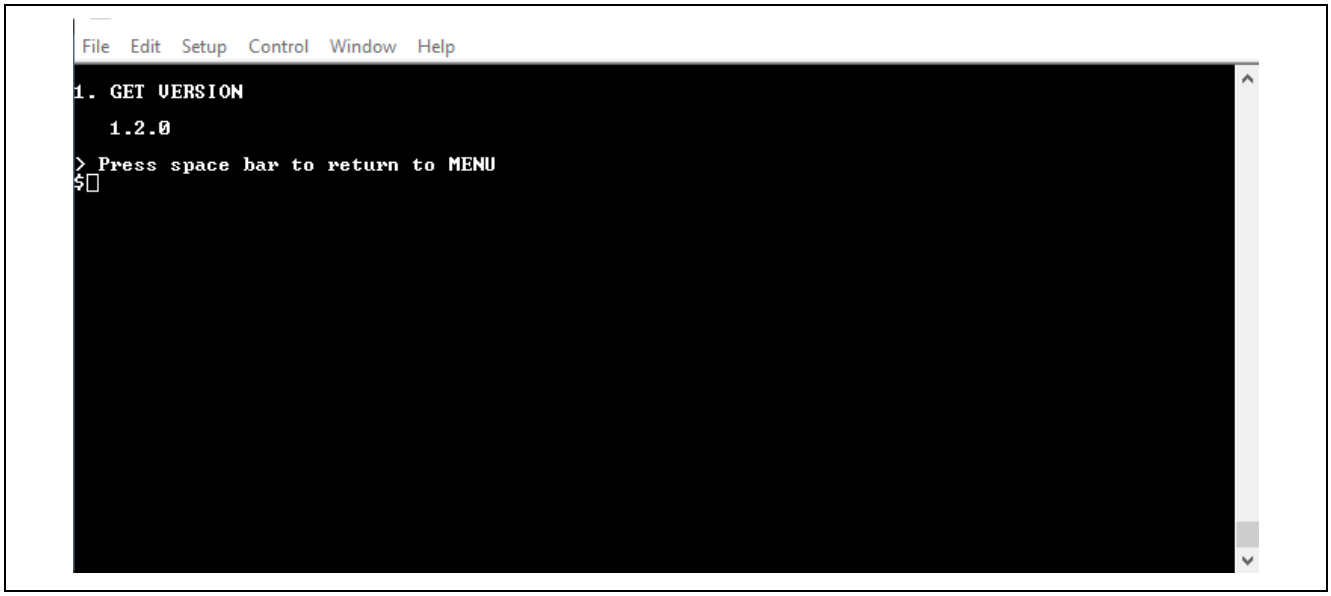

Figure 25. Get Version Information

The next section provides an introduction to the setup for the AWS account. Please refer to section **5.2 For Users Using the Provided Dashboard and AWS Account of Kit**, refer to section **5.4 For User who use their own AWS account**.

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

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

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.

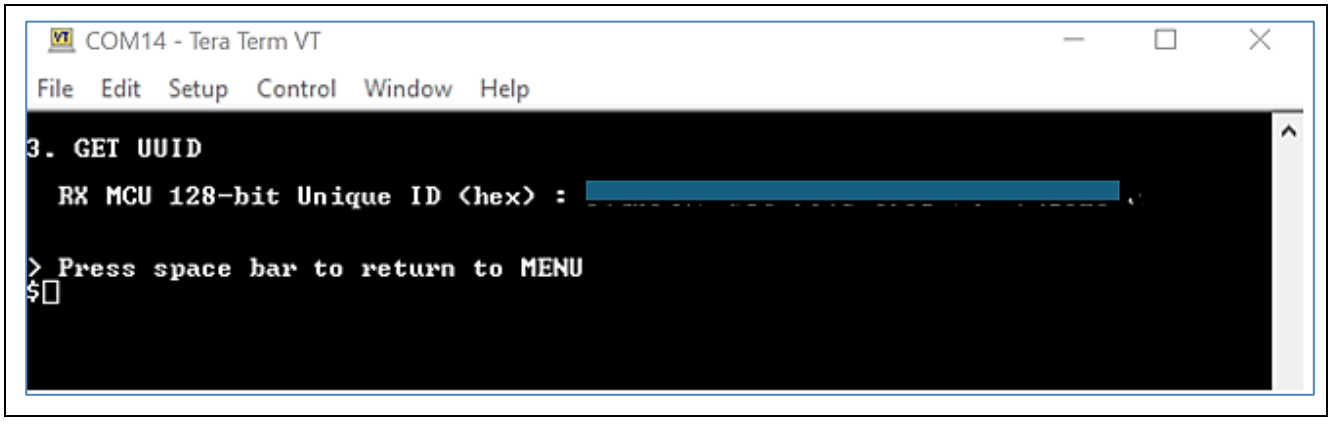

Figure 26. Getting Board UUID Information

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 Instance.

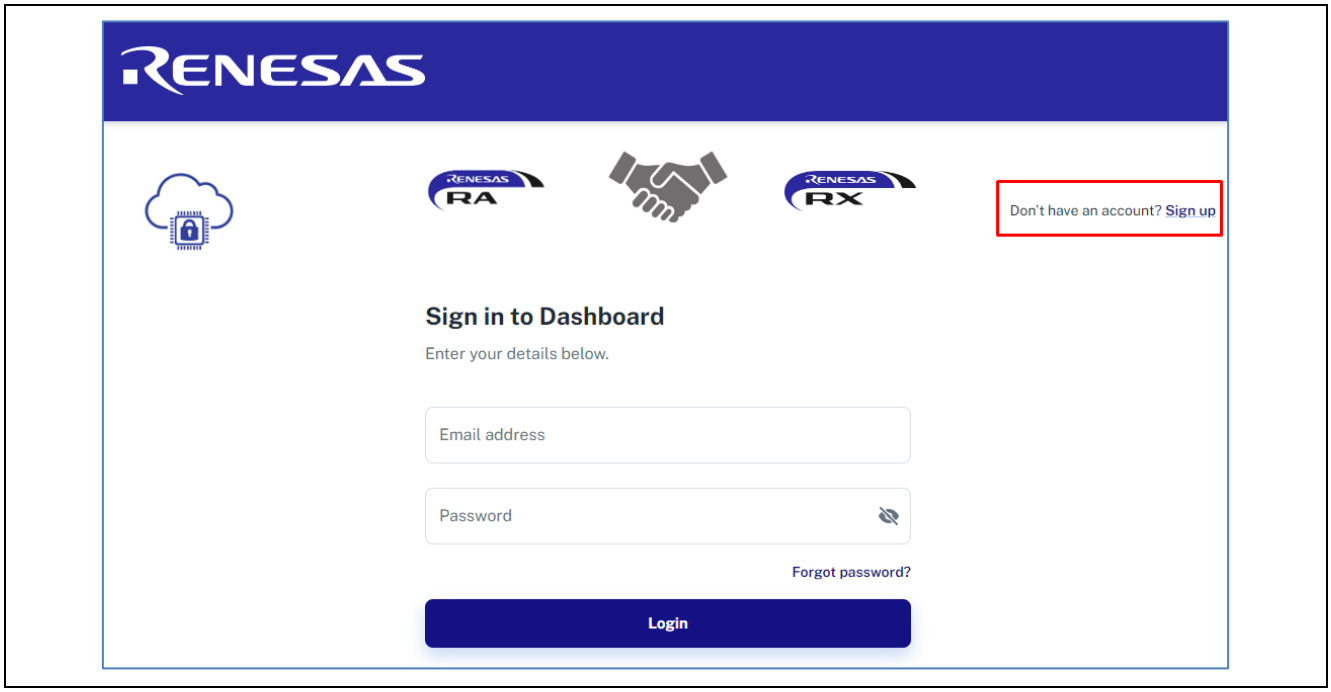

Figure 27. Get the Account 10 USD of Trial of AWS (1/2)

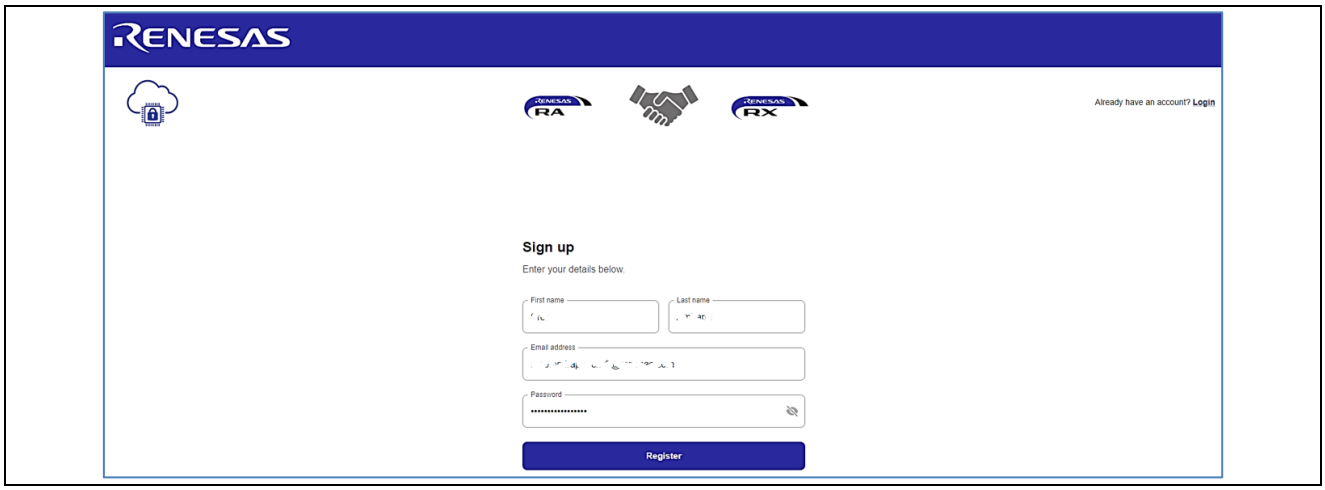

Figure 28. Get the Account 10 USD of Trial of AWS (2/2)

2. Wait for AWS verification email (~10 min). Then put on the email and UUID to register the kit in below window. 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.

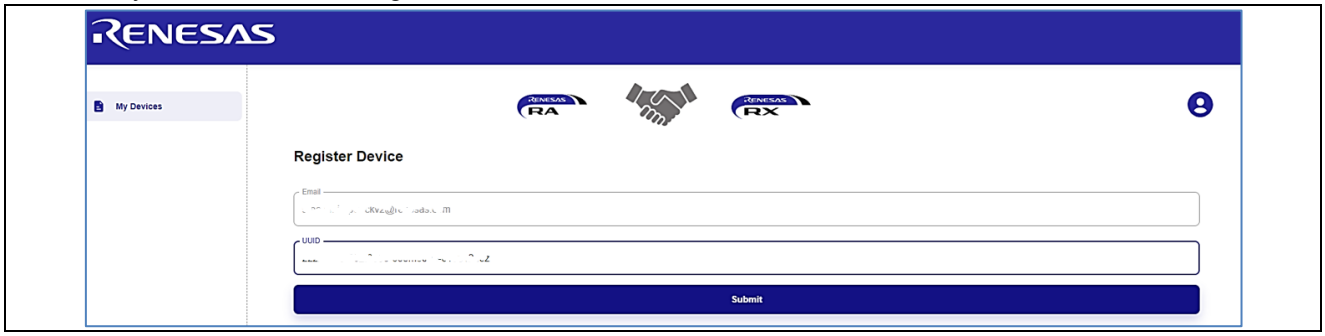

Figure 29. Register Device

3. **Verify the AWS account** in your email that you registered.
4. 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.

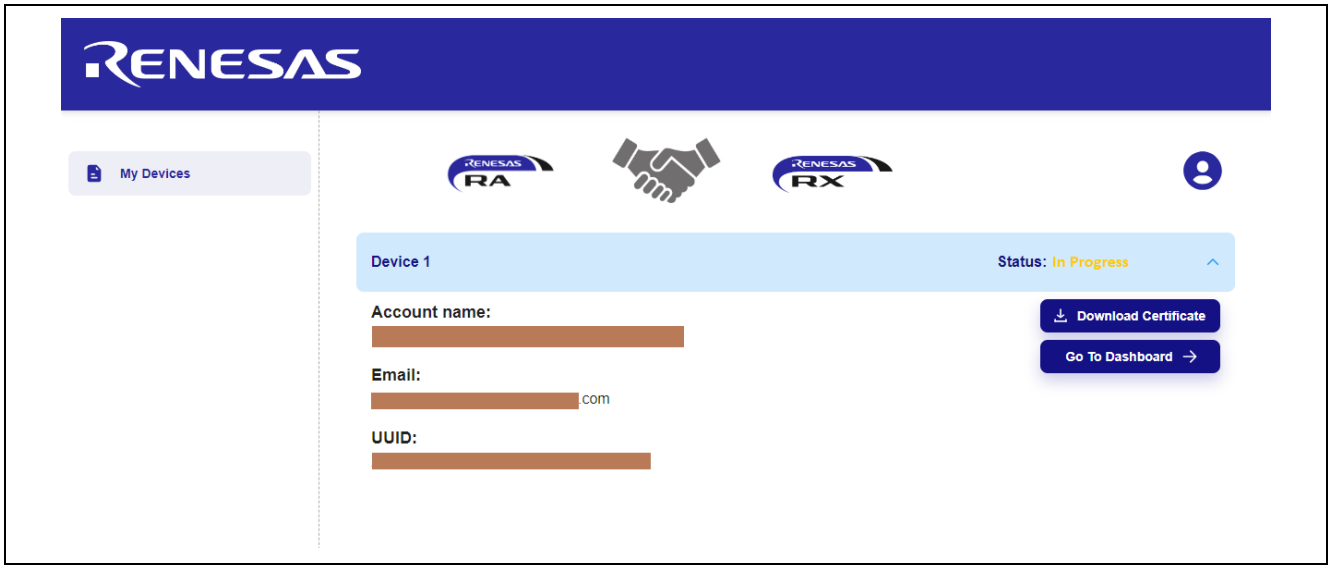

Figure 30. Dashboard Build In Progress

5. Once the account status shows up as online in the registration page, click on device to see device UUID.

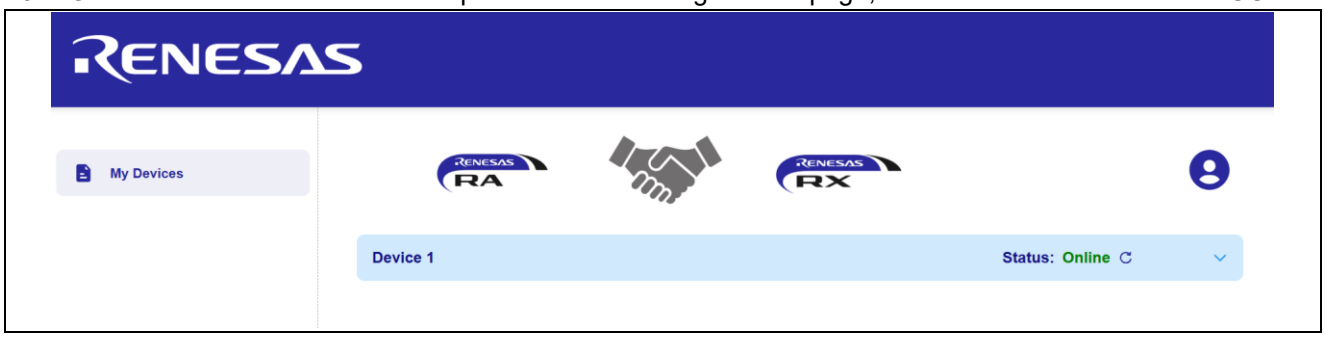

Figure 31. Active Device

6. After finishing the progress, you can get the file of certification to connect dashboard from **"Download Certificate"** button. It is used for installation on the application demo of kits that you got previous step.

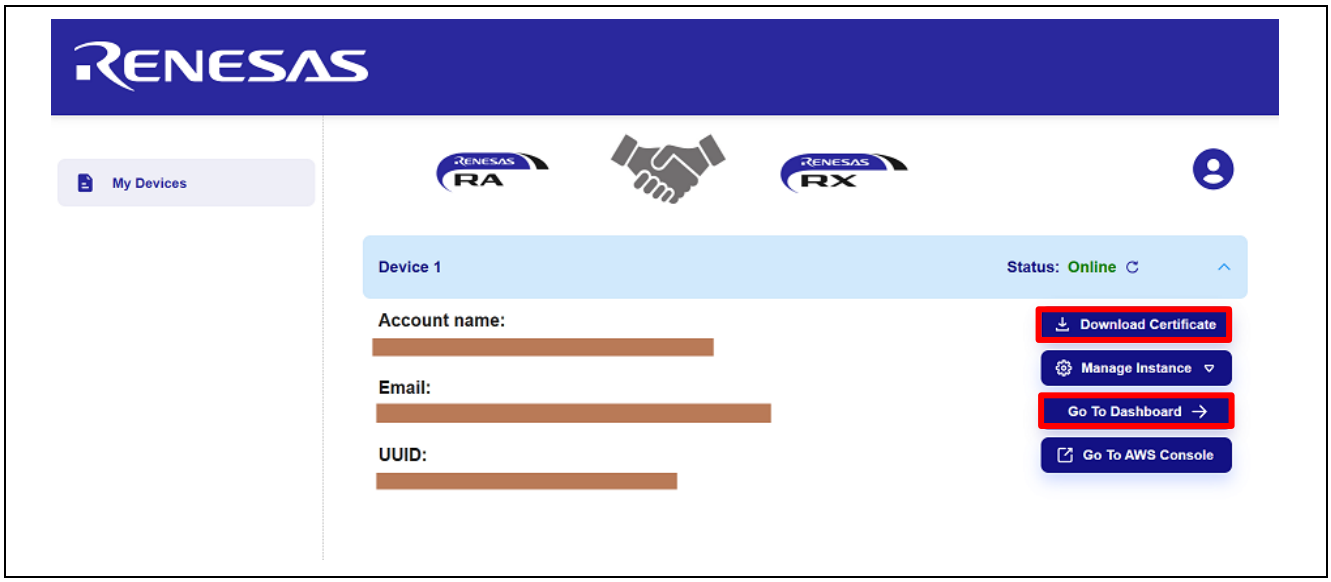

Figure 32. Dashboard Build Complete

7. 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.

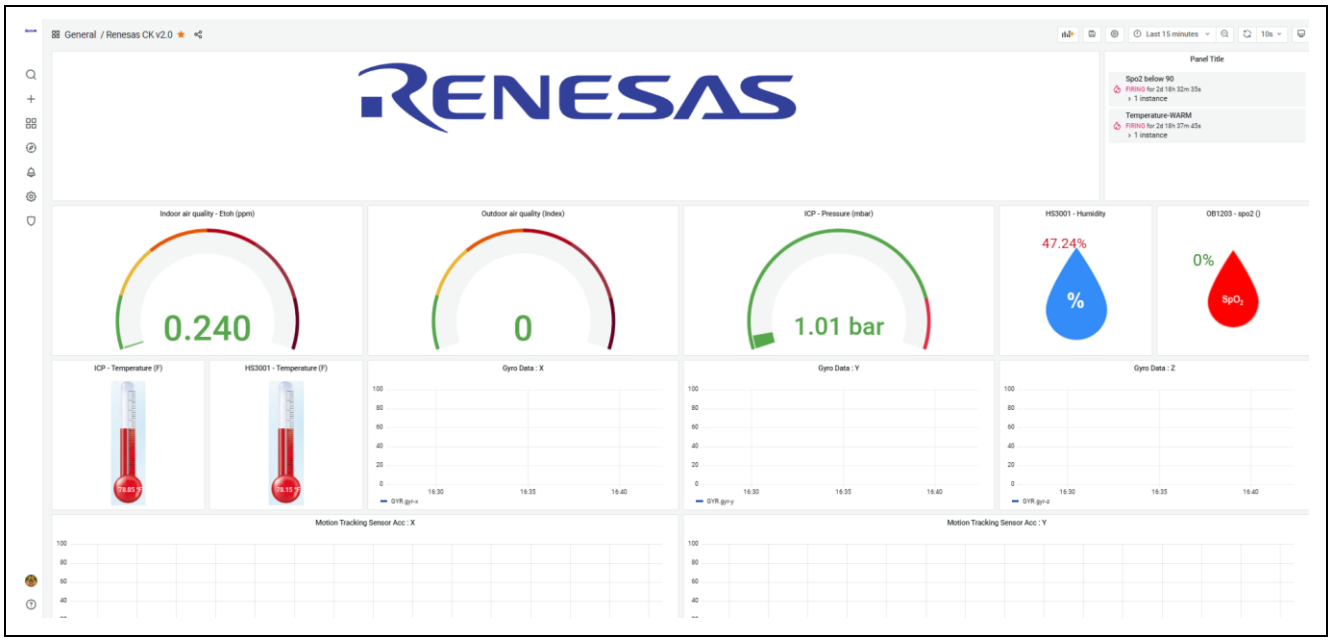

Figure 33. Dashboard for this Application

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.

1. 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.

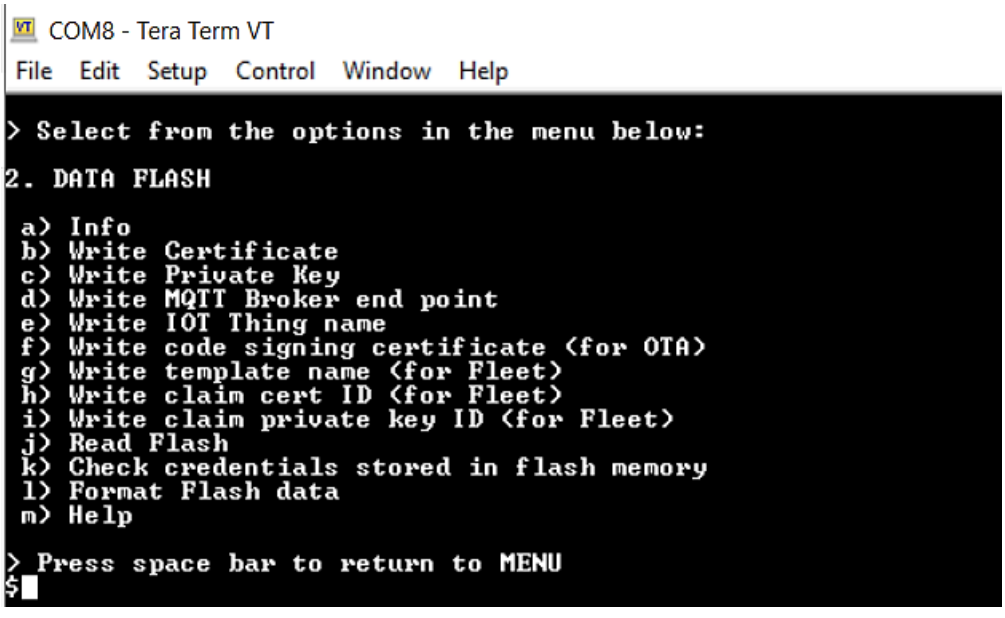

Figure 34. Data Flash Related Menu and Commands

2. Please unzip the `cert.zip` from dashboard.

3. To store the **Device Certificate**, press the option '**b**' and 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.

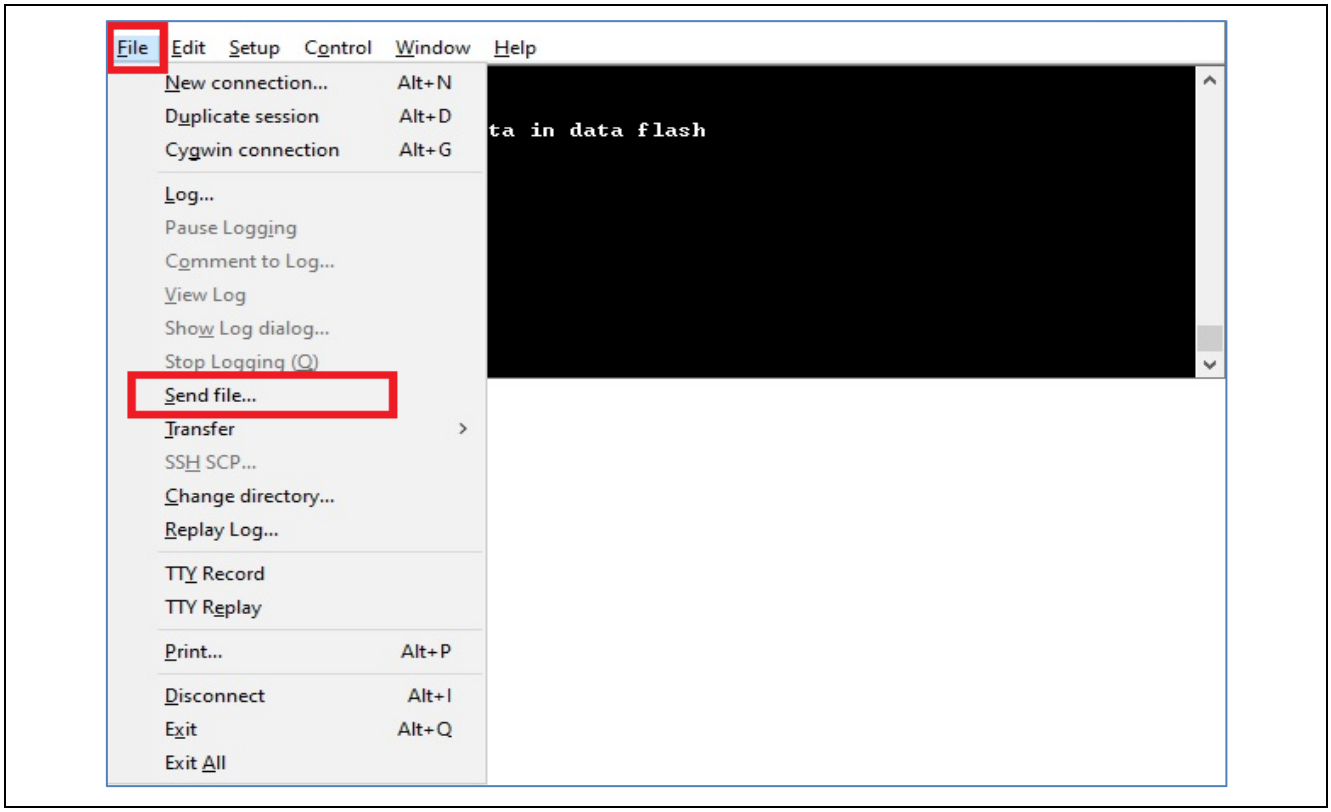

Figure 35. Accessing the Device Certificate

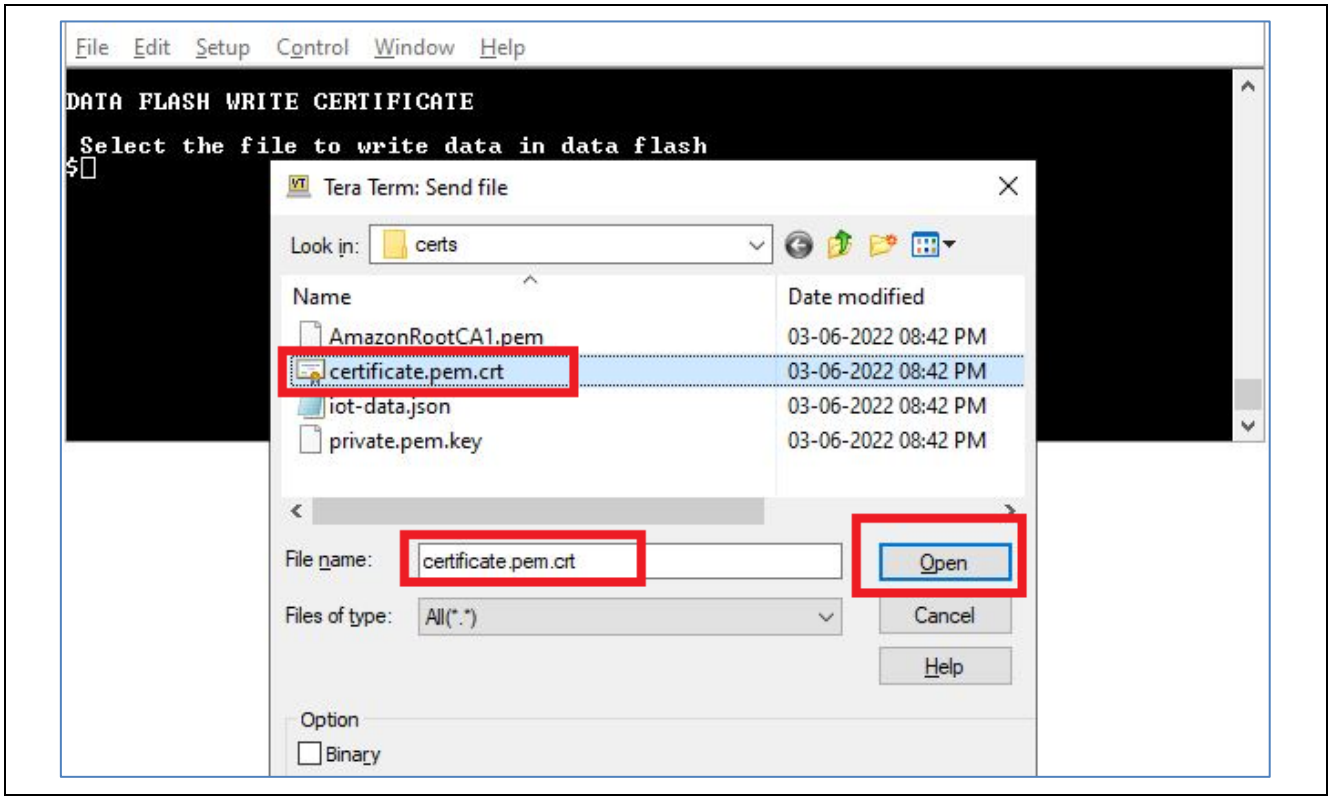

Figure 36. Downloading the Device Certificate into the Data Flash

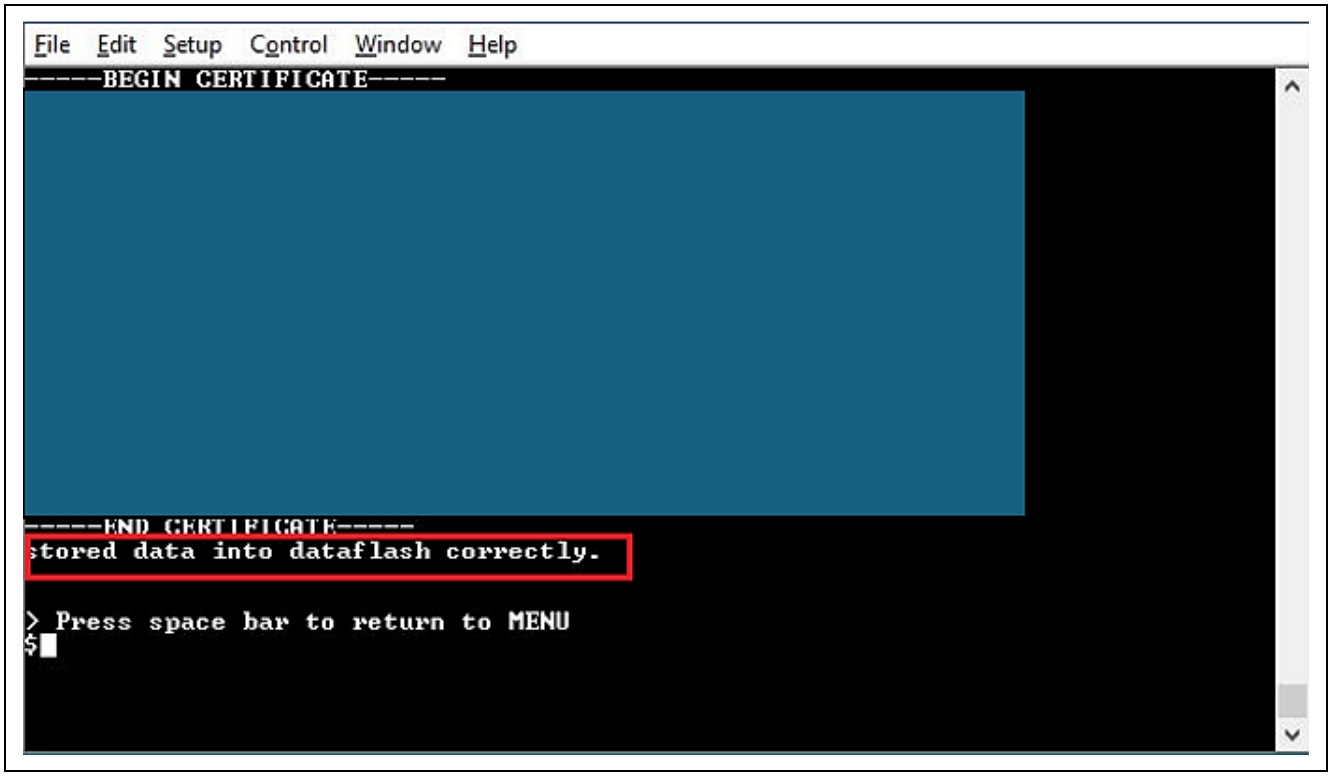

Figure 37. Status of the Downloaded Device Certificate into the Data Flash

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.

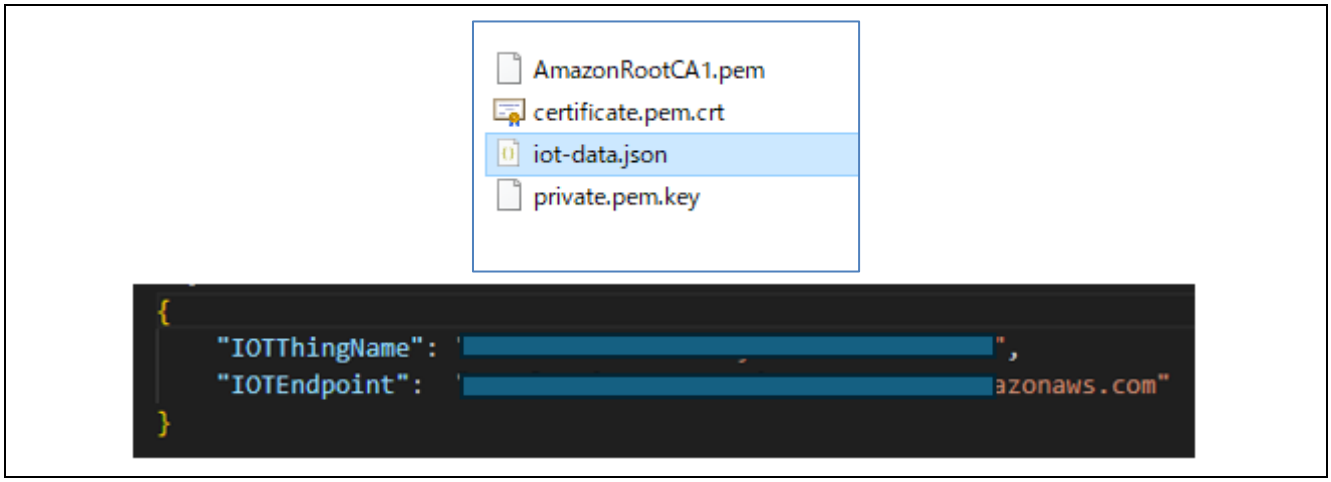

Figure 38. Getting the IoT Things Name and IoT Endpoint Information

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 "".

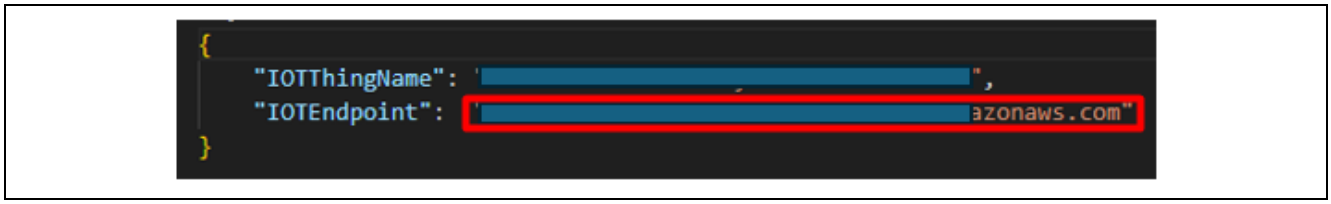

Figure 39. Copy IoT Endpoint Information

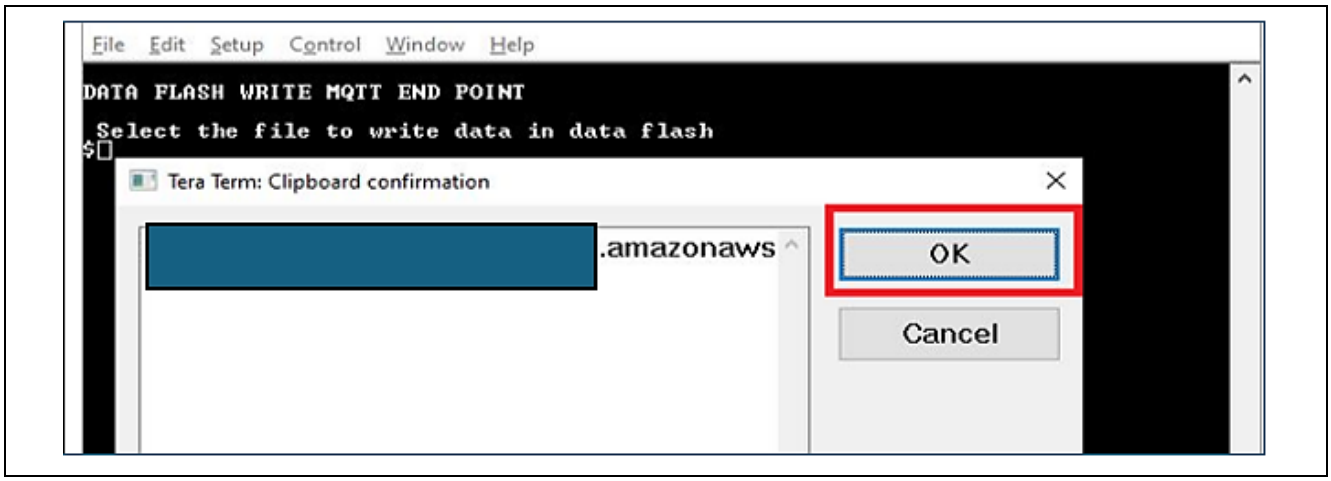

Figure 40. Storing the MQTT IoT Endpoint into the Data Flash

7. To store the IOT Thing Name, copy the Thing Name string between the quotes xxxxxxxx-xxxx-xxxxxxxx-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 41. Copy the IOTThingName

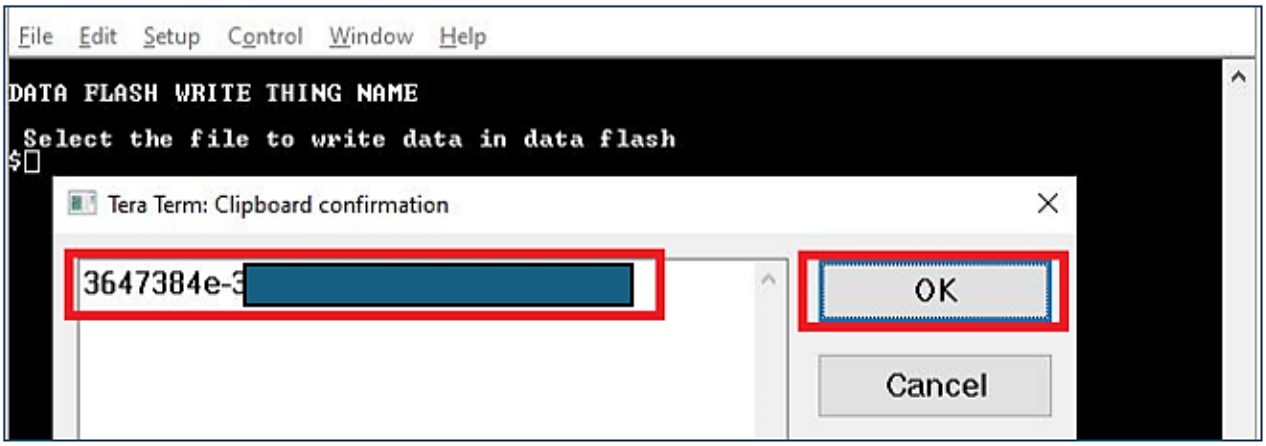

Figure 42. Storing the Thing Name into the Data Flash

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 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 value in data flash. Please be careful when using this option in 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 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.

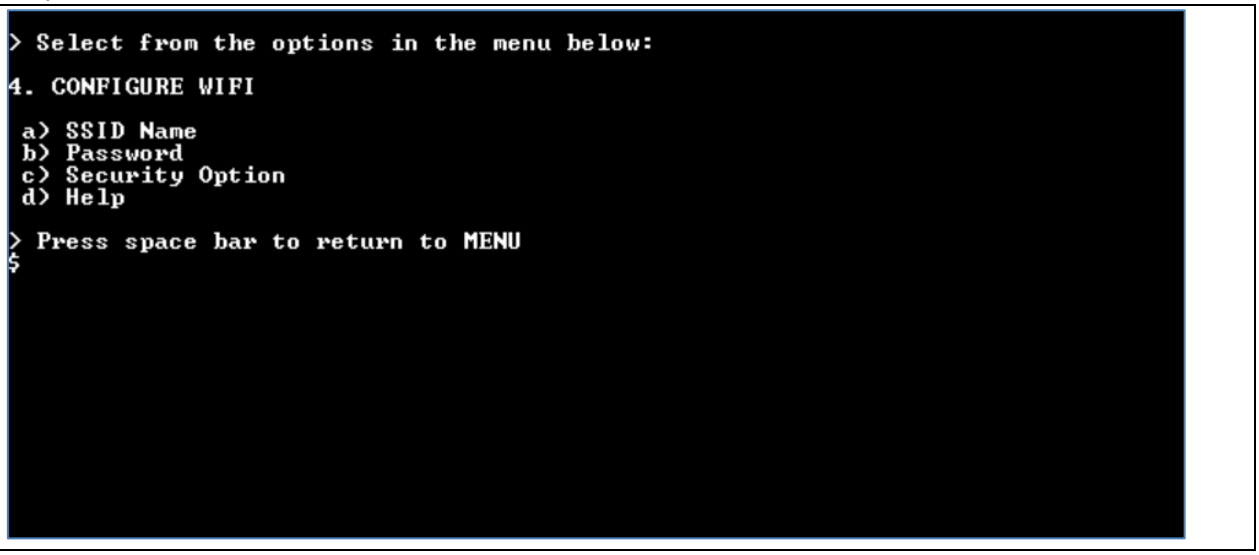

Figure 43. Configure Wi-Fi related Menu and Commands

2. To store the SSID Name, in the Wi-Fi menu, press the option **'a'.**
   Note: Max length of SSID is 32 characters and the min 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.

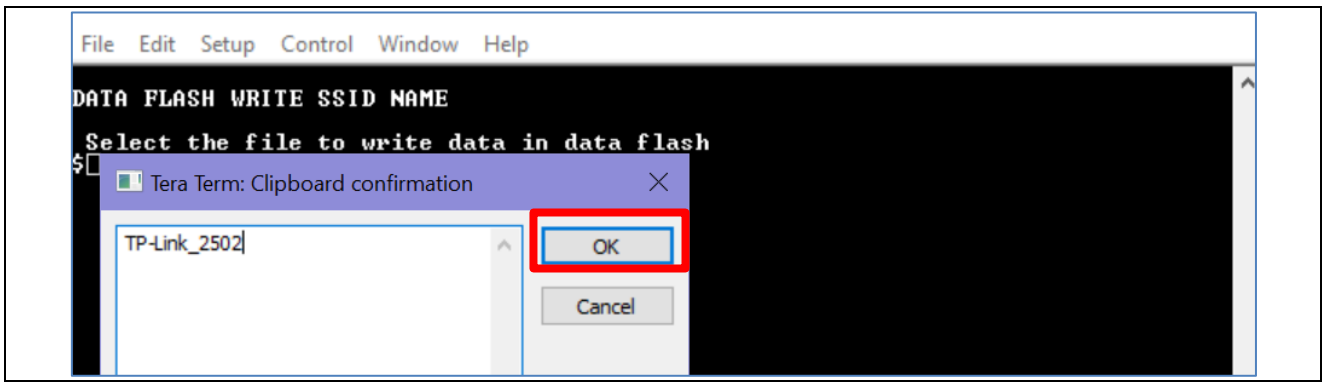

Figure 44. Storing the SSID Name into the Data Flash 1/3

— If the user did not copy the SSID before this step, they can input directly to the Tera Term. And for checking easily the SSID, click the **Setup > Terminal > Local echo** of the Tera Term as below.

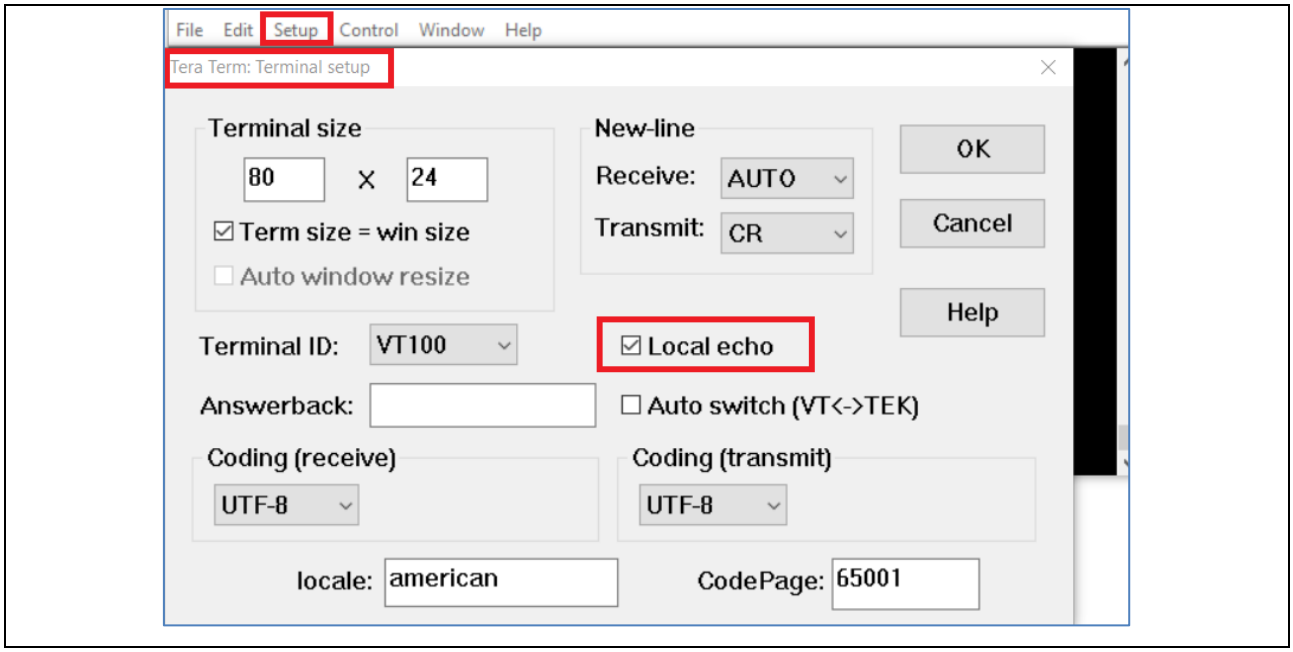

Figure 45. Storing the SSID Name into the Data Flash 2/3

— Input SSID and press **Enter**.

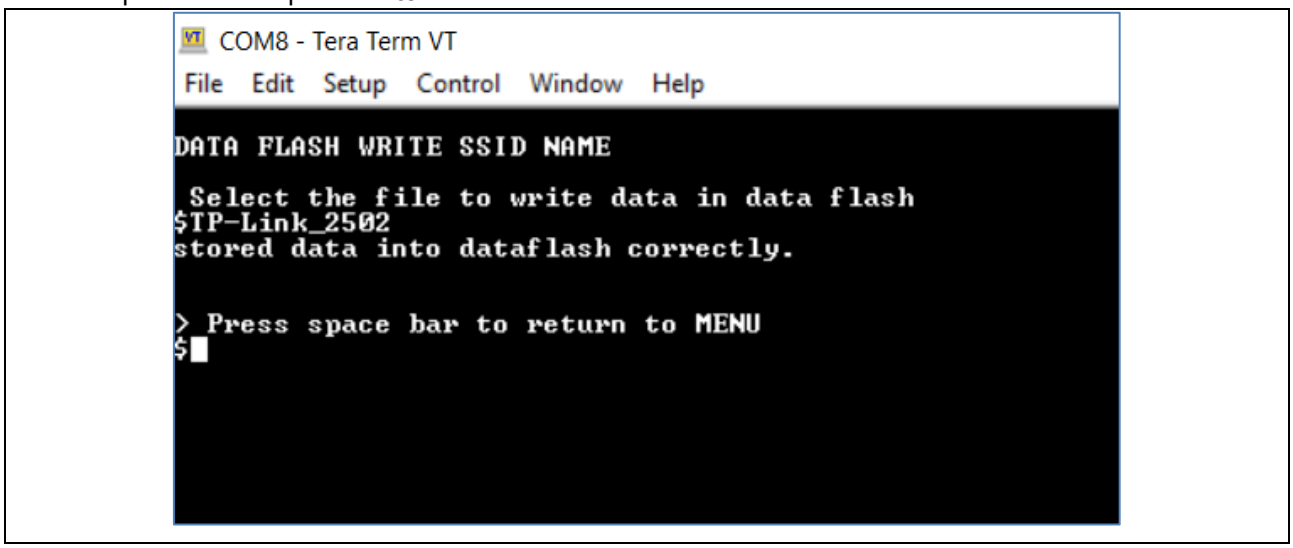

Figure 46. Storing the SSID Name into the Data Flash 3/3

3. To store the password, in the Wi-Fi menu, press the option **'b'**, input the password, and confirm the valid string then press Enter. Max length of password is 32 characters, and the min length is 8 characters.

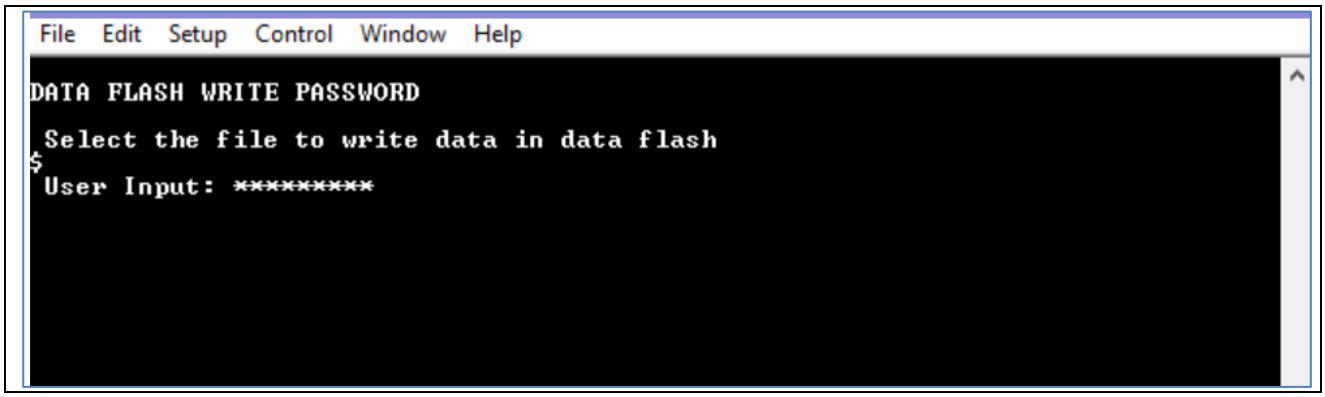

Figure 47. Storing the Wi-Fi Password into the Data Flash

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**, **'c'** for **WPA2 security** to choose the correct security for Wi-Fi configuration.

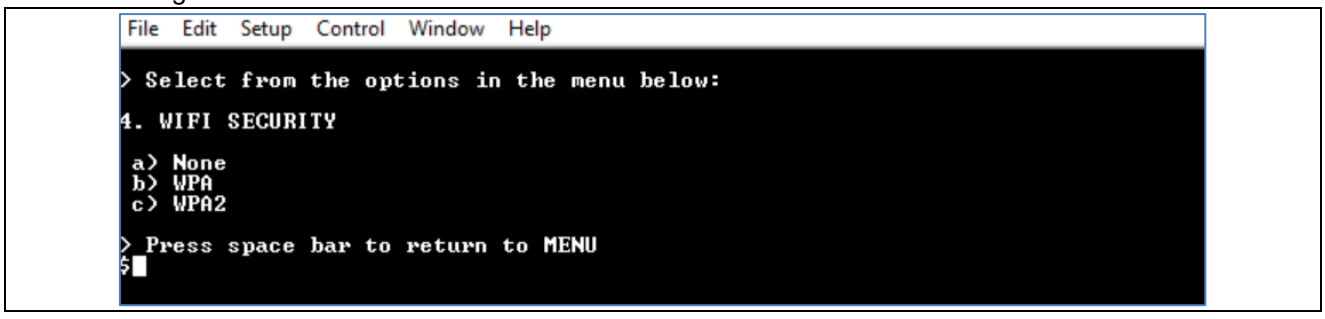

Figure 48. Storing the Security into the Data Flash

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.

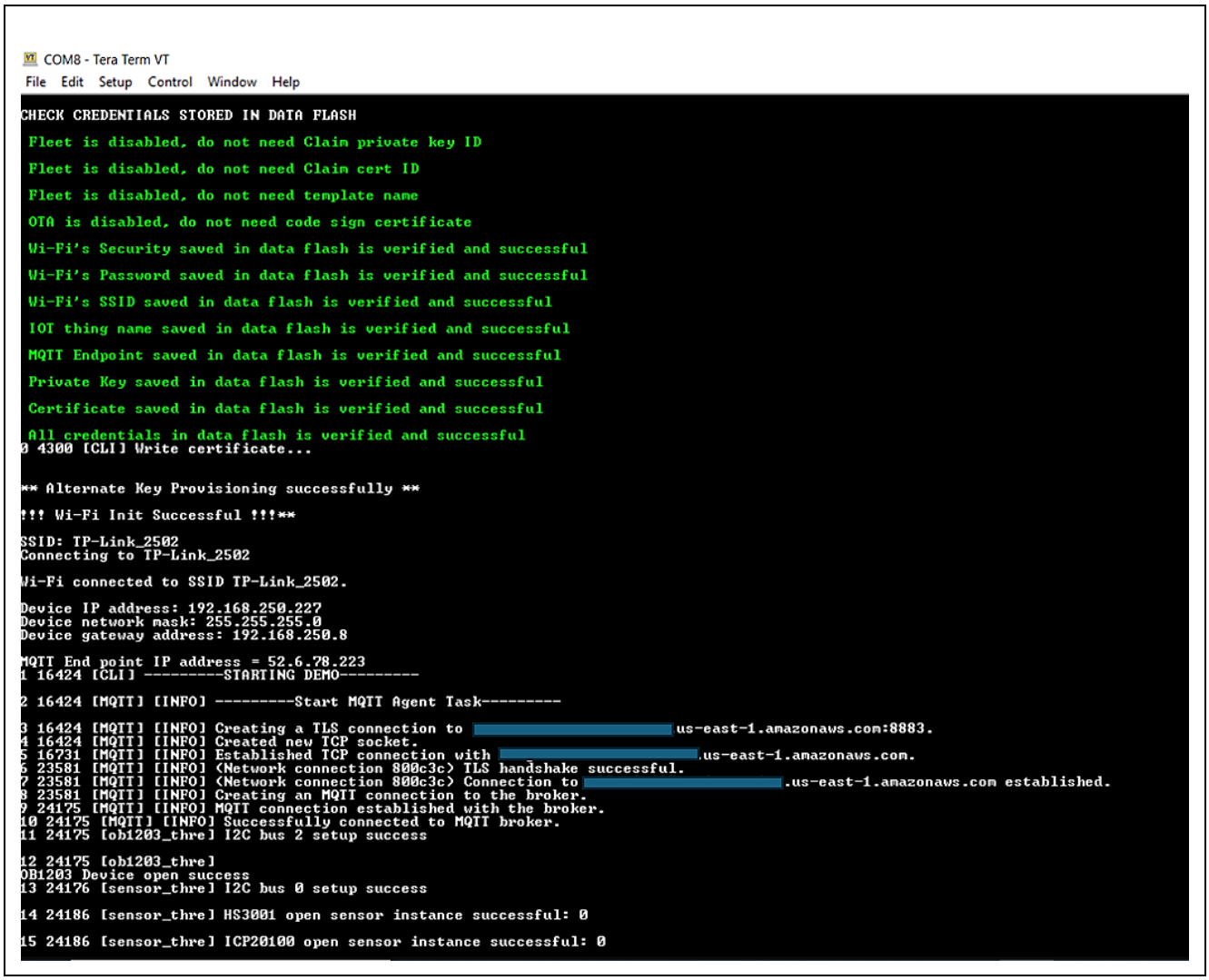

Figure 49. Welcome Screen on the Console

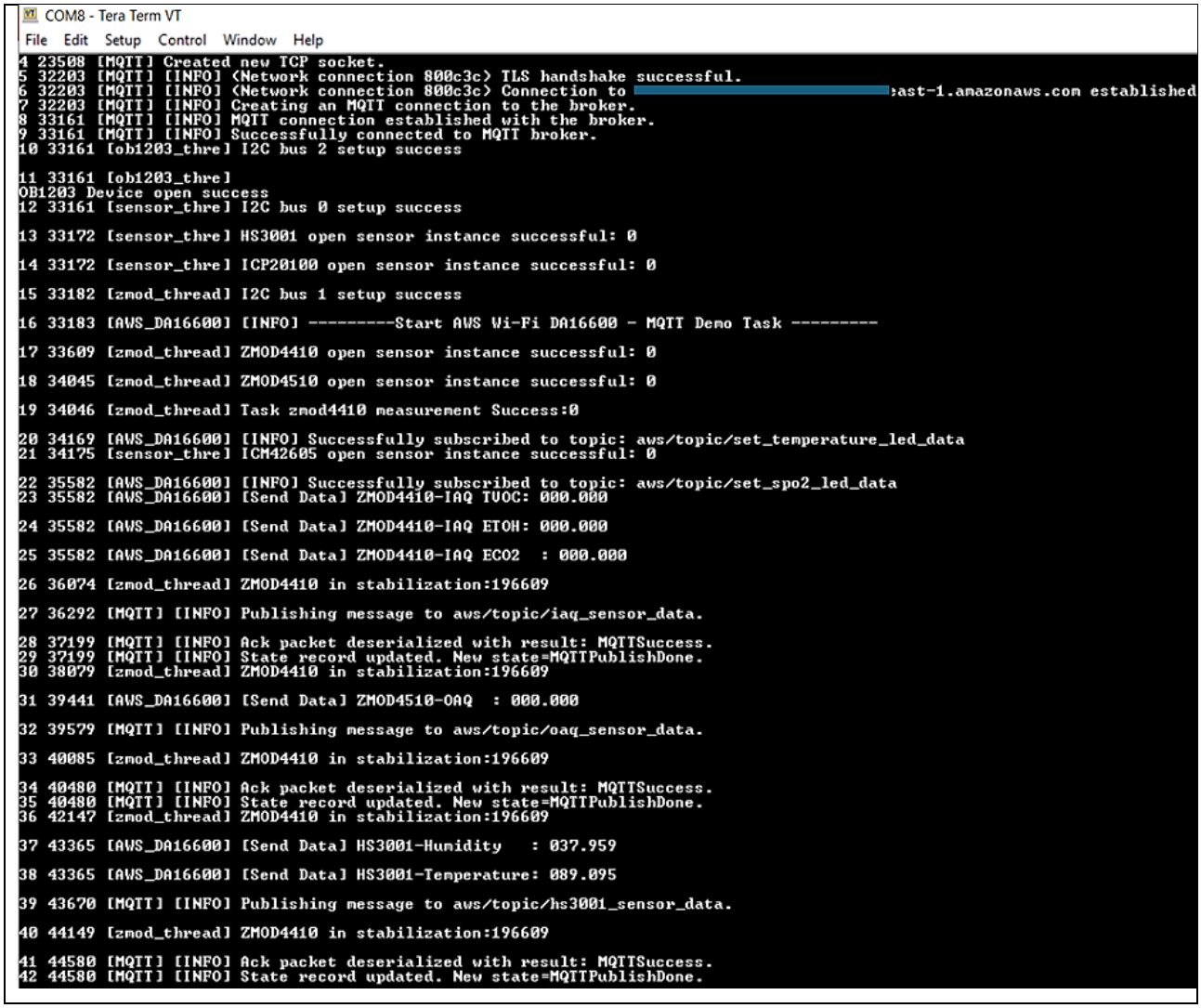

Figure 50. Application with MQTT

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 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.

About the detail 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.

When considering using AWS, you can use the AWS free tier.

AWS Free Tier

5.4.2 Log in to the AWS Management Console

Amazon Web Services > My Account > AWS Management Console

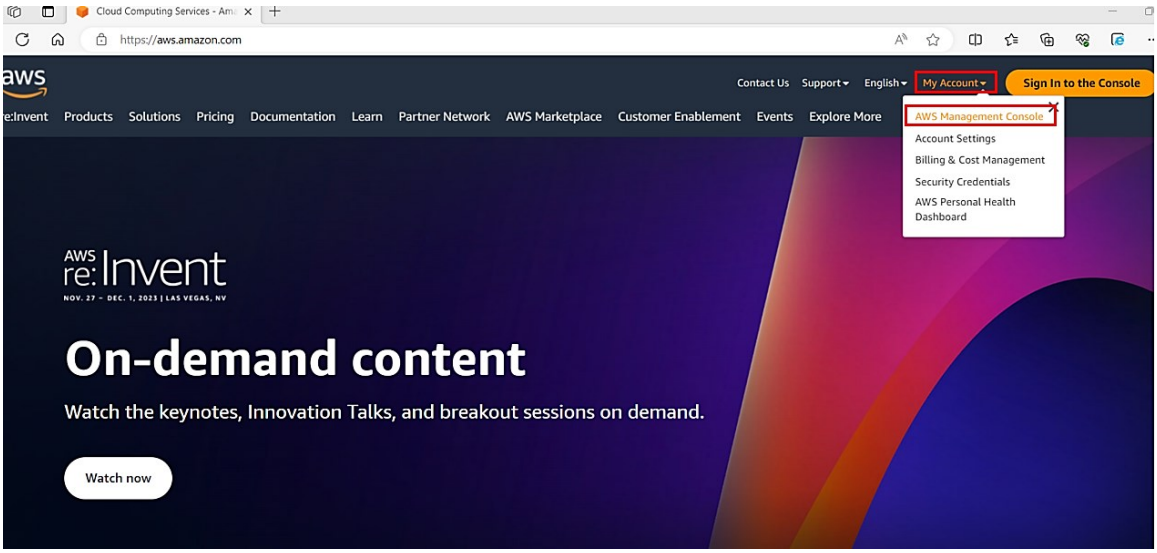

Figure 51. Login the AWS

5.4.3 Move to IoT Core Control Panel

AWS services > All services > IoT Core

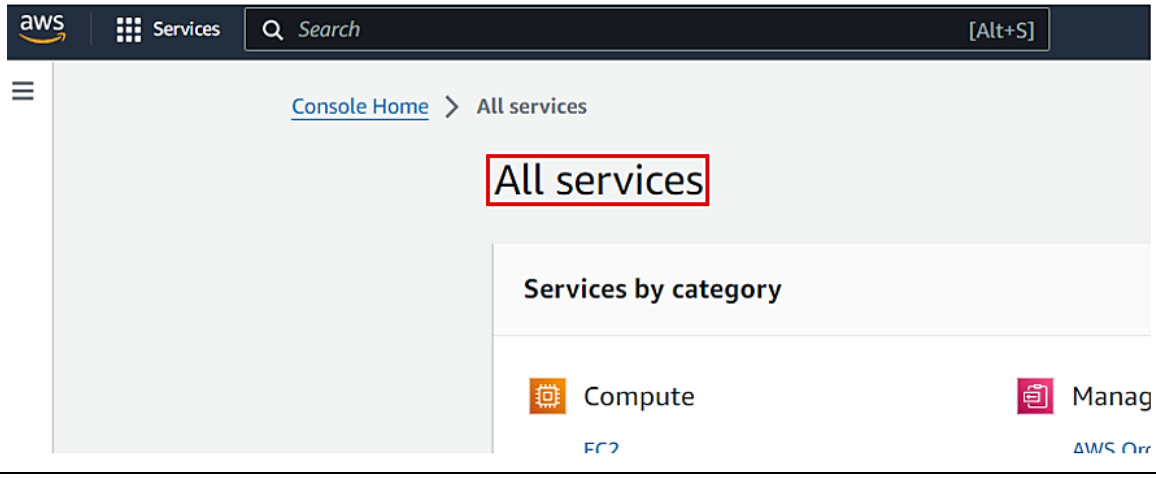

Figure 52. Search the IoT Core (1/2)

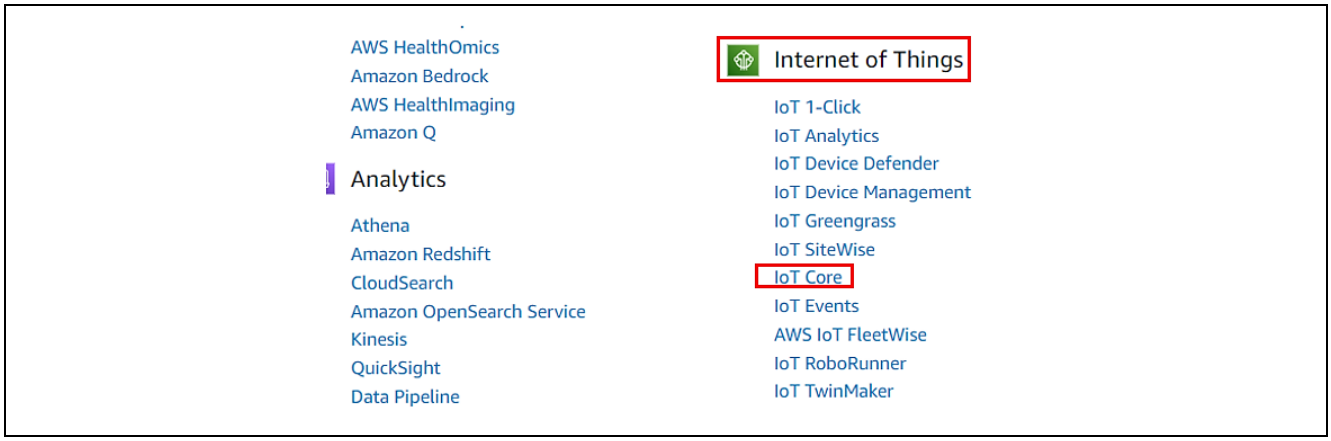

Figure 53. Search the IoT Core (2/2)

5.4.4 Create a Security Policy

Secure > Policies > Create a policy.

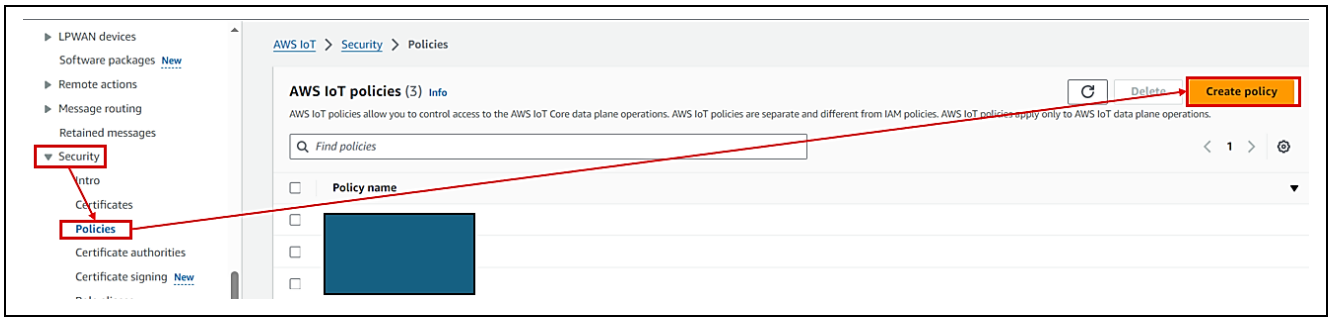

Figure 54. Create the Policy (1/3)

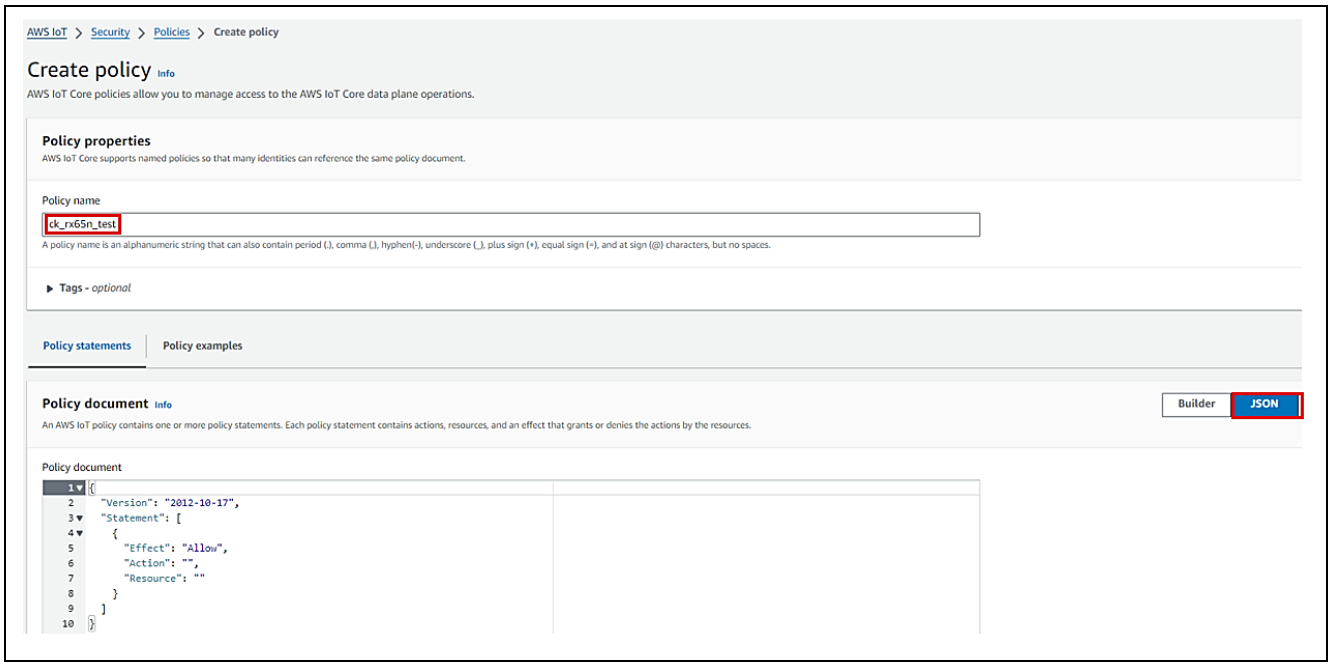

Figure 55. Create the Policy (2/3)

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

Paste the copied code into the policy syntax > Create.

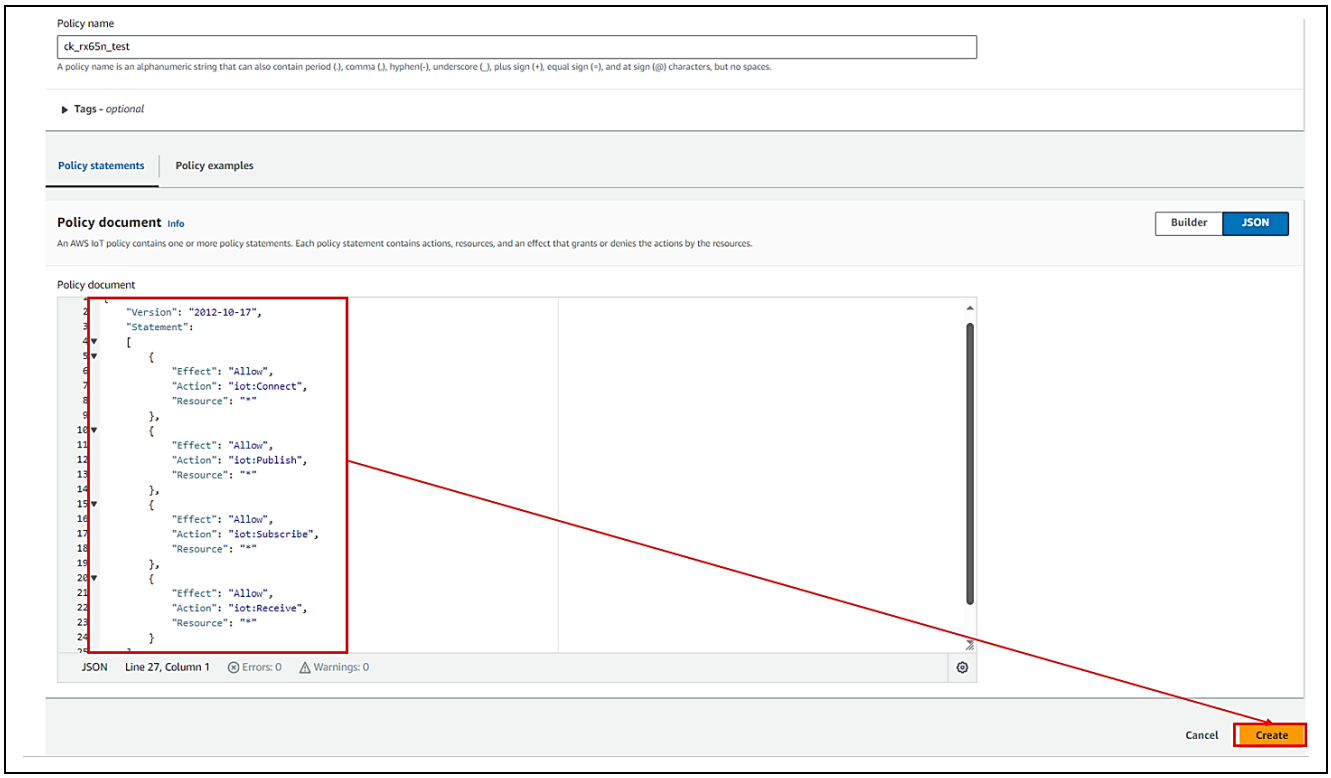

Figure 56. Create the Policy (3/3)

5.4.5 Register your device (thing) with AWS IoT

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

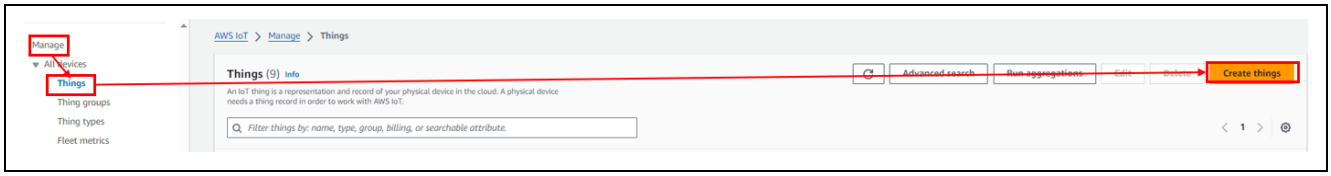

Figure 57. Creating the Things (1/5)

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

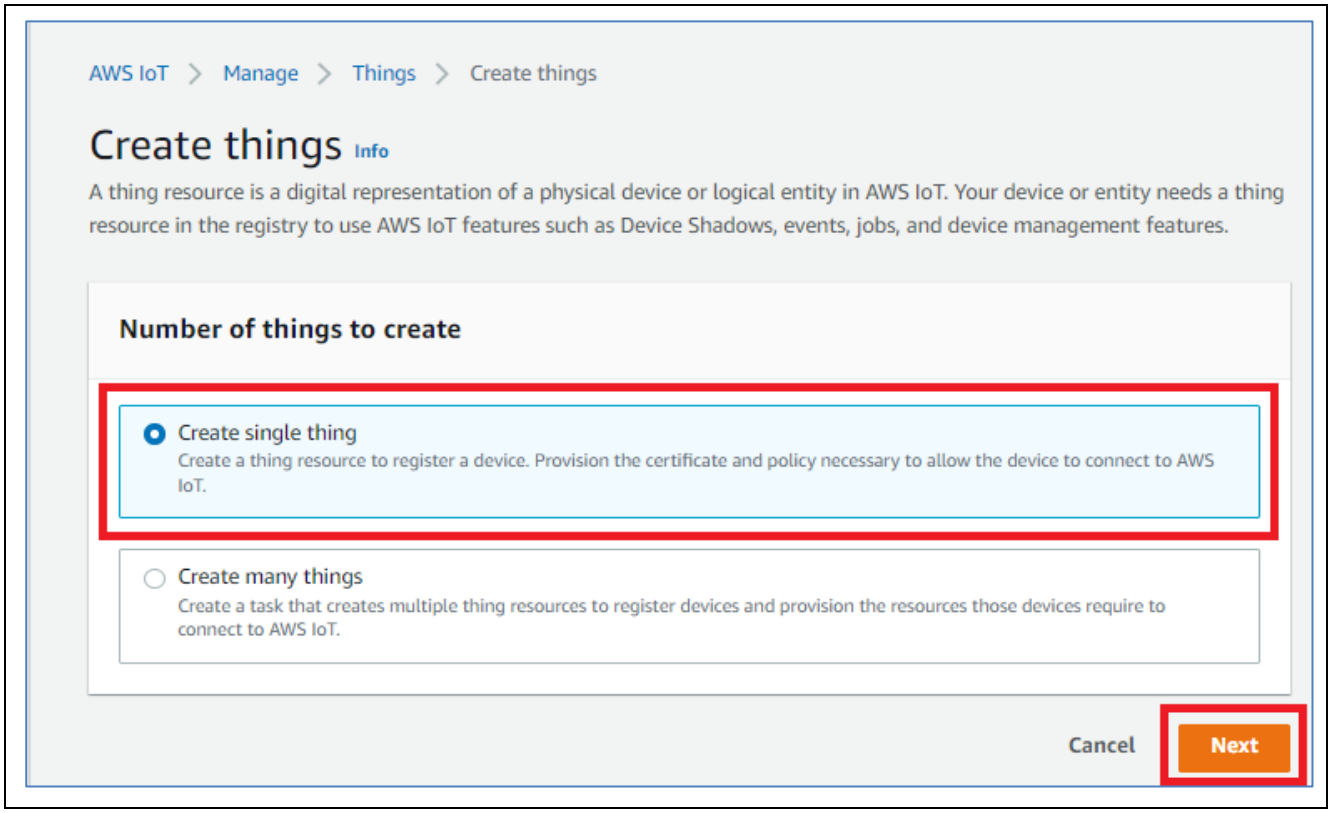

Figure 58. Creating the Things (2/5)

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

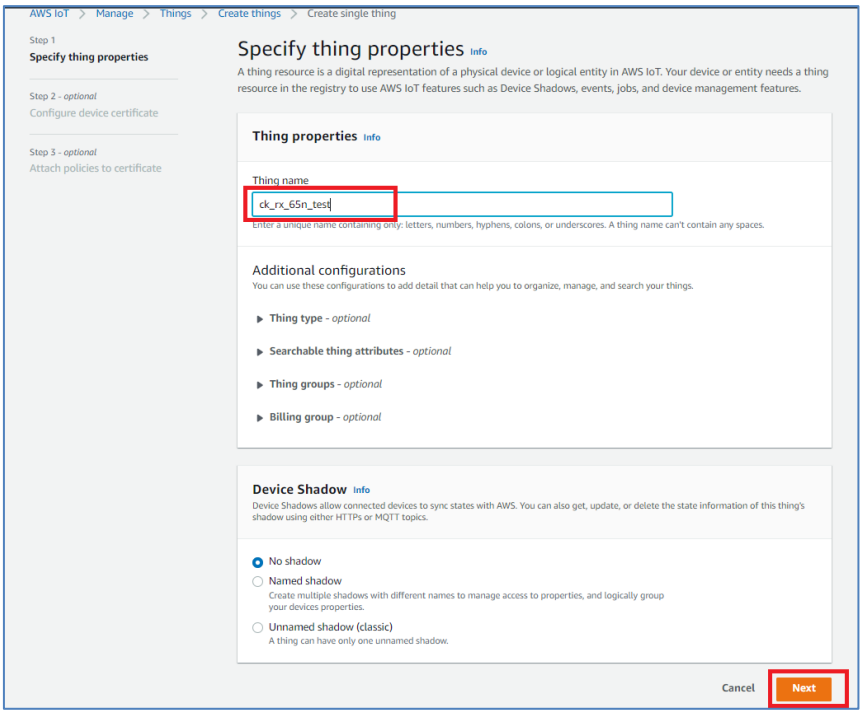

Figure 59. Creating the Things (3/5)

Auto-generate a new certificate.

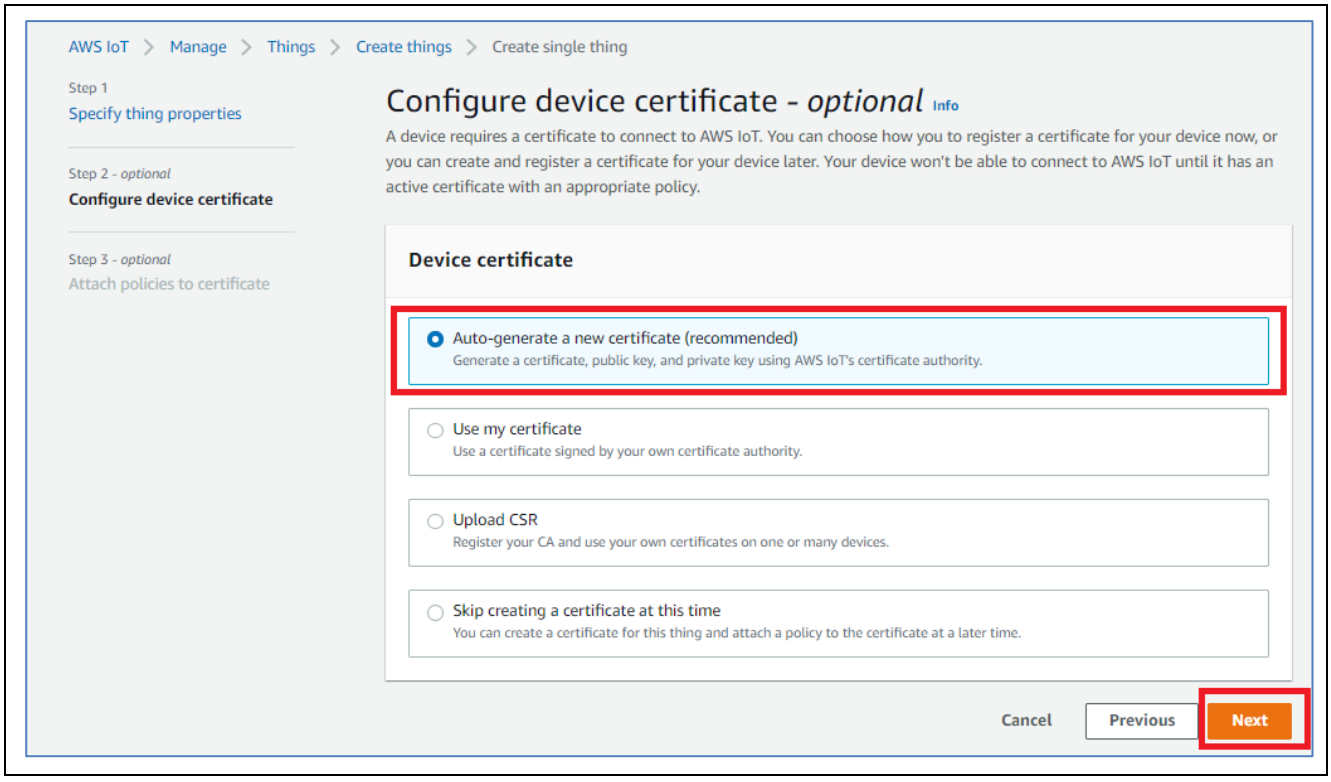

Figure 60. Creating the Things (4/5)

- Add a policy for your thing

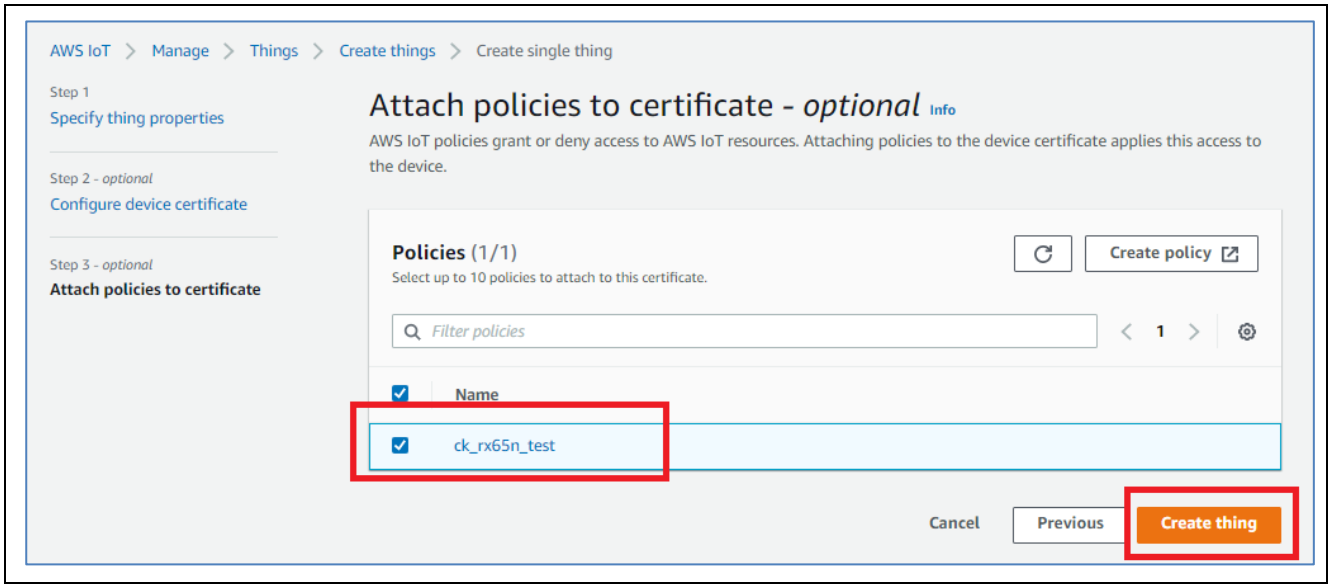

Figure 61. Creating the Things (5/5)

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

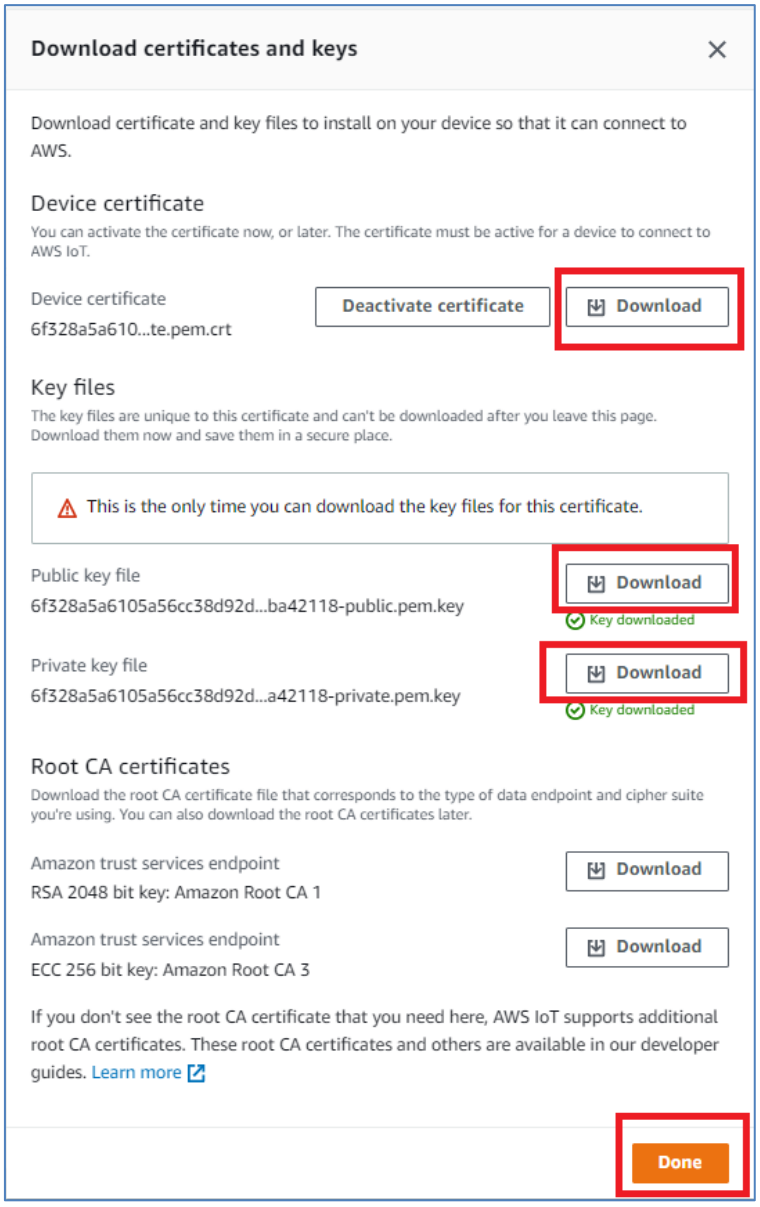

Figure 62. Download A Certificate for this Thing/A Public Key/A Private Key

5.4.6 Check AWS IoT Endpoints

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

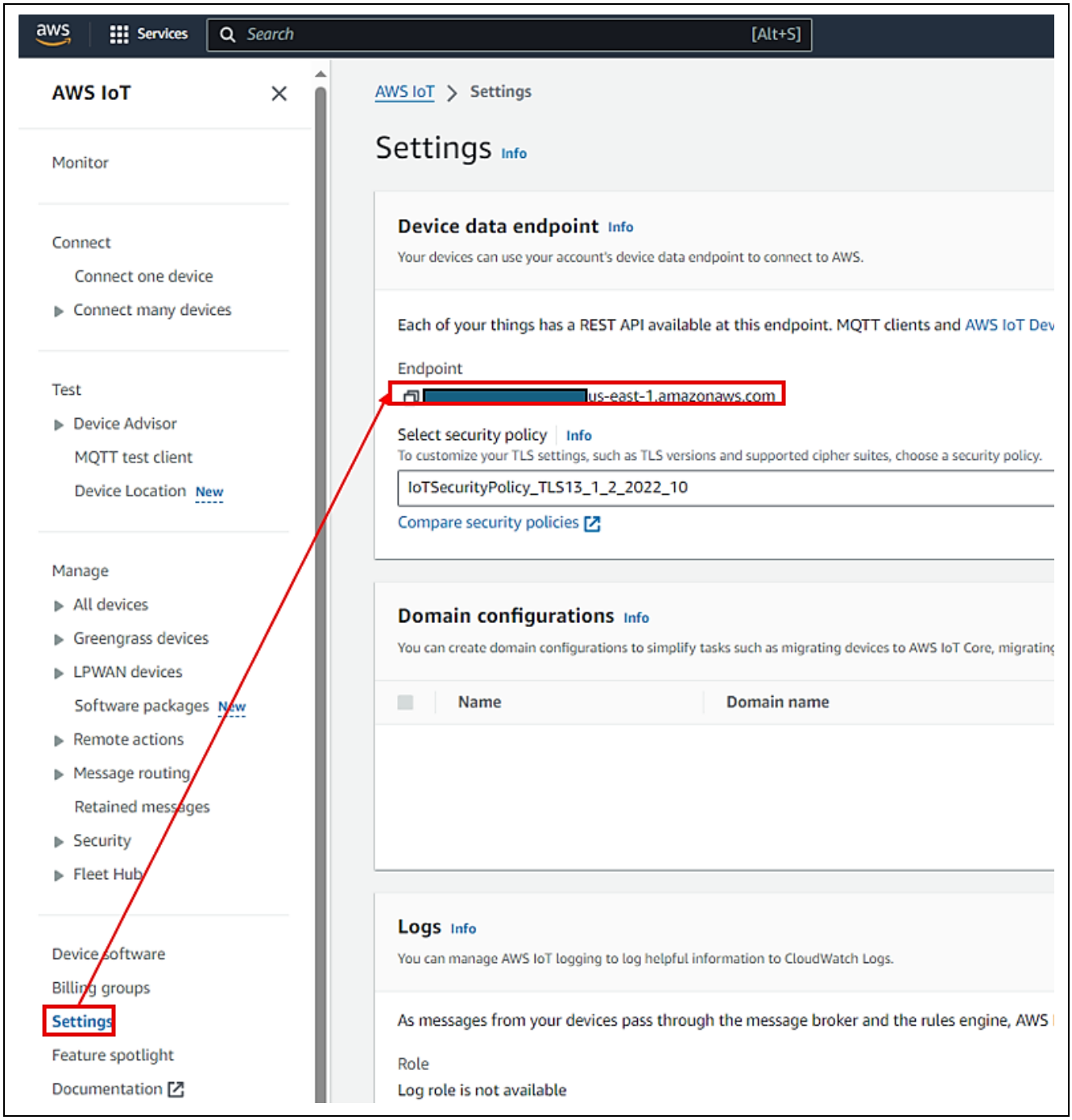

Figure 63. Check AWS IoT Endpoints

Reference

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

Install for the credential in application as instruction 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 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 application.

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

5.5 Verifying the Application Project using AWS Dashboard and Renesas Dashboard

5.5.1 Subscribe to a Topic Messages on the AWS IoT

This section describes the steps on 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 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 customized view of all the Sensor Data.

For the 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 are shown below.

Note: The messages shown below are **case sensitive**; users need to take care of this 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.

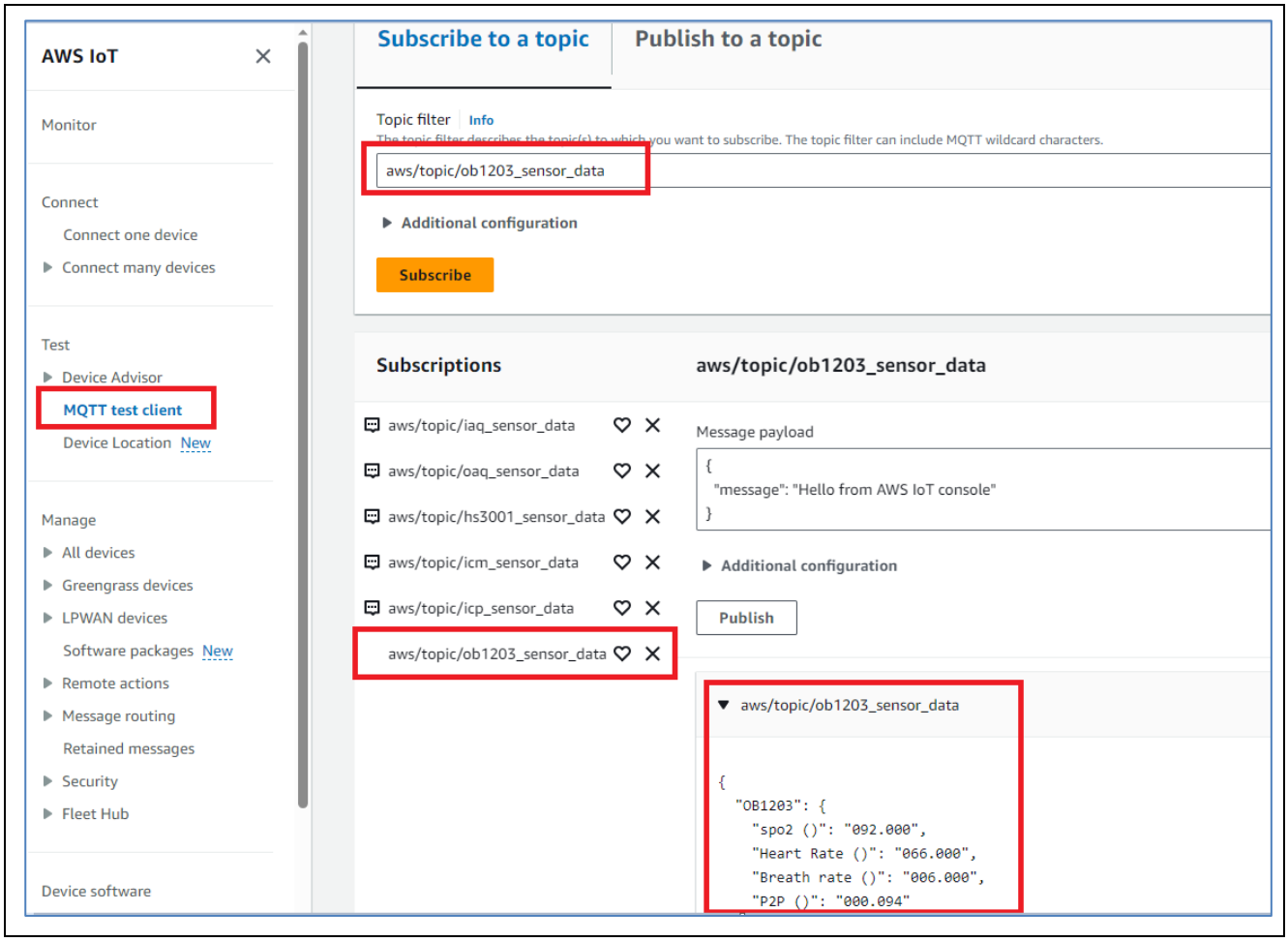

Figure 64. Subscribe to a Topic Messages on the AWS IoT Screen

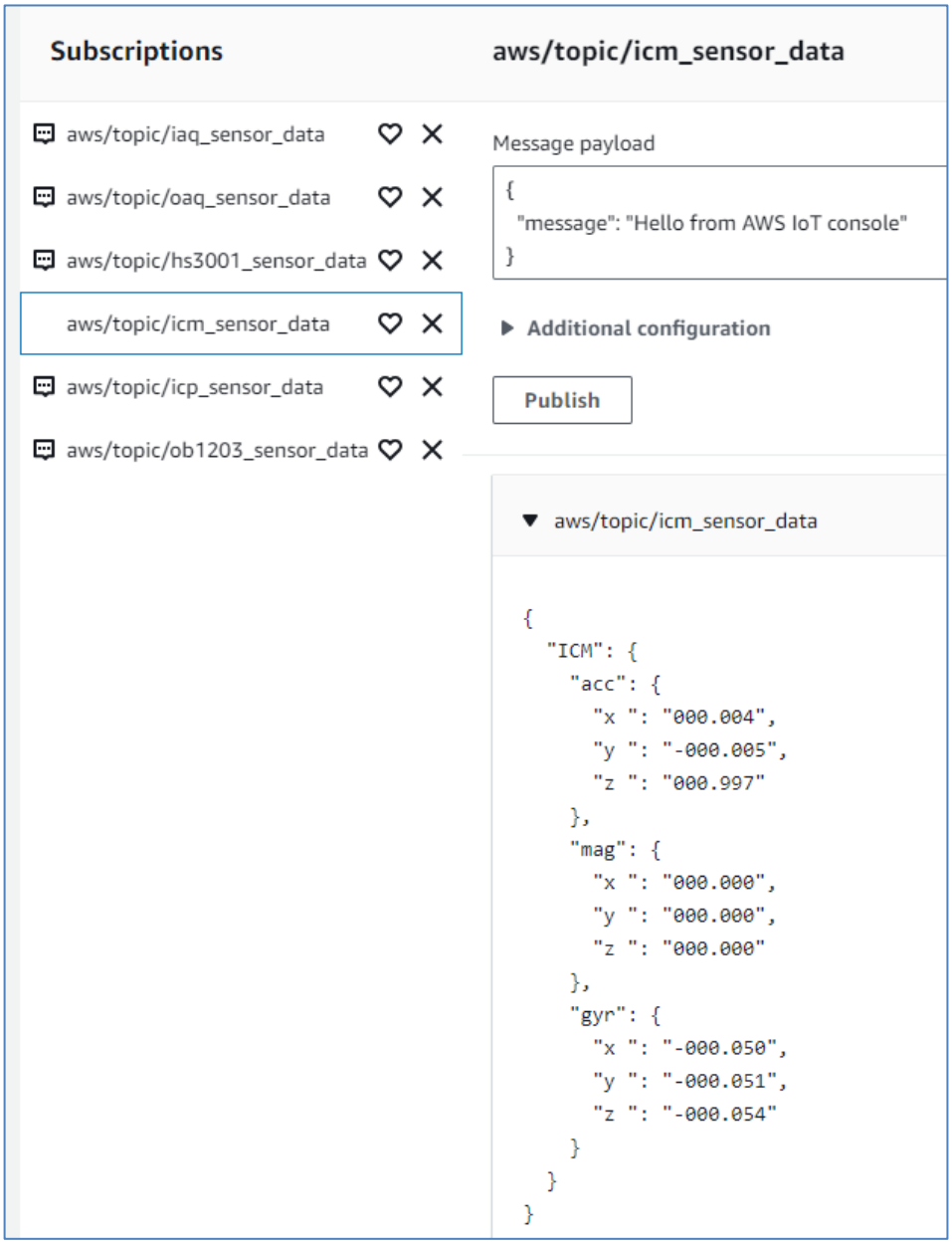

Figure 65. Subscribed Messages on the AWS IoT Screen

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**

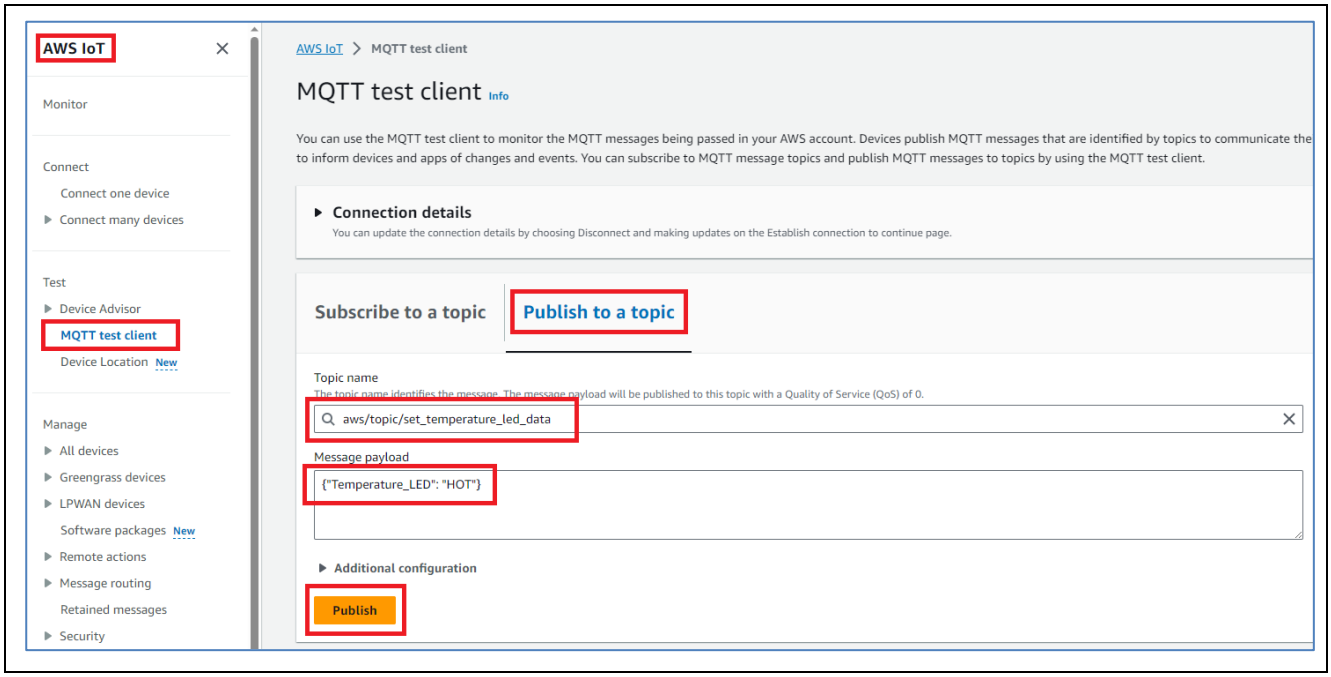

Figure 66. Publish the MQTT Message (1/2)

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
```

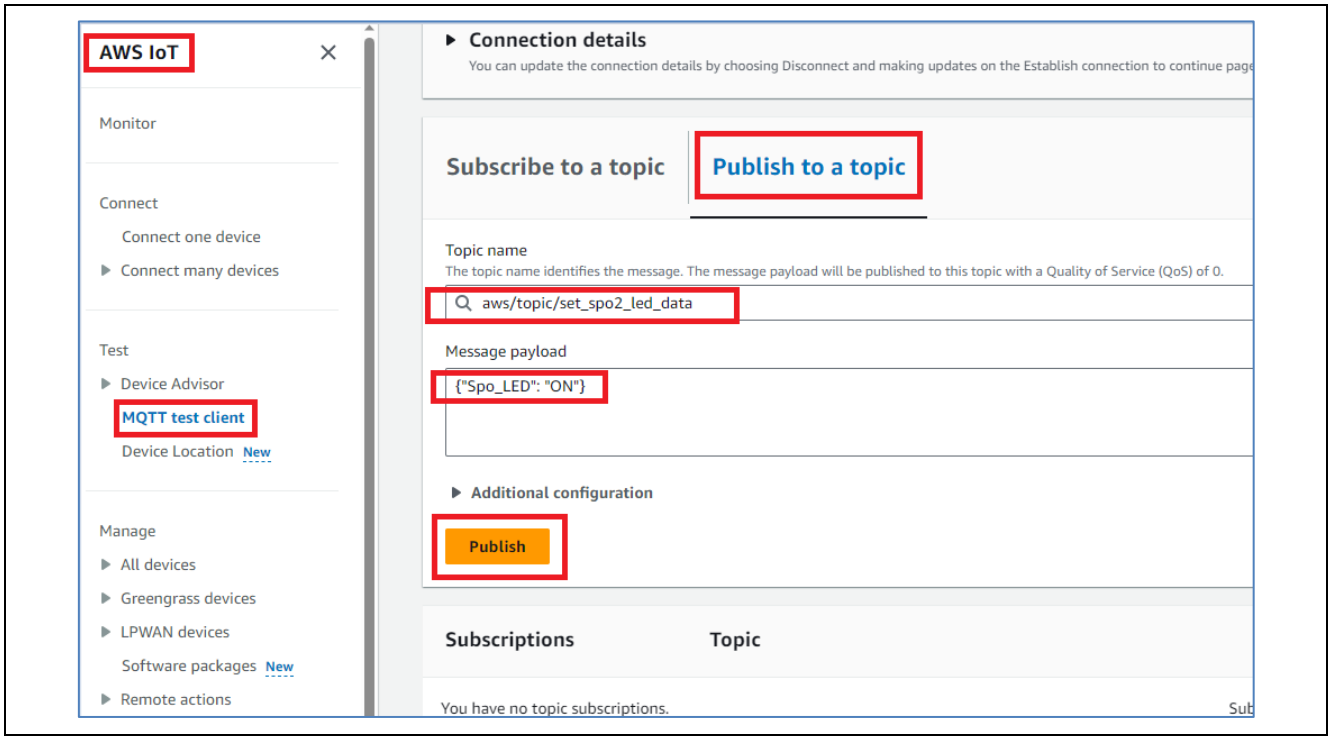

Figure 67. Publish the MQTT Message (2/2)

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 range from 65 to 85
  - Temperature – Hot: above 85

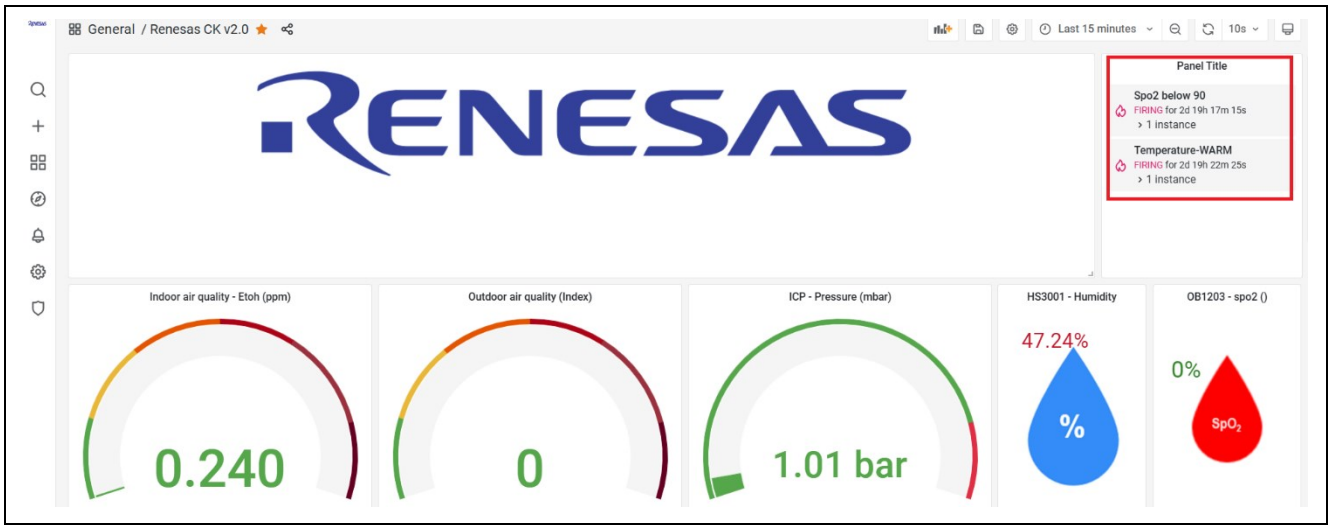

Figure 68. Sensor Status Feedback

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 on 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 account with free 10$ credit and dashboard that are provided by Renesas, because of some limitation with this account's permission.

6.2 Prerequisites

6.2.1 Installing Python

1. Access the Python download web site: [Python downloaded website] and Download the Python 3.11.0 installer: Click the **Download** link for Python 3.11.0

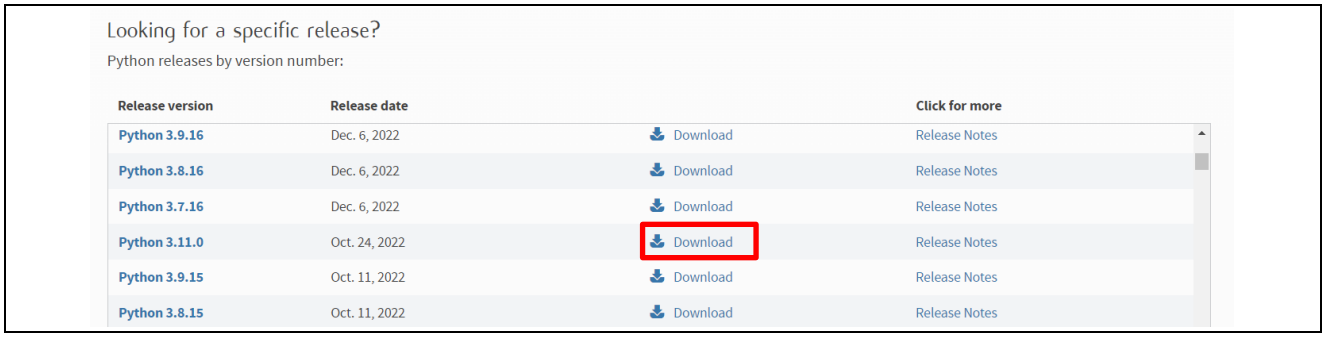

Figure 69. Python Download Website

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

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

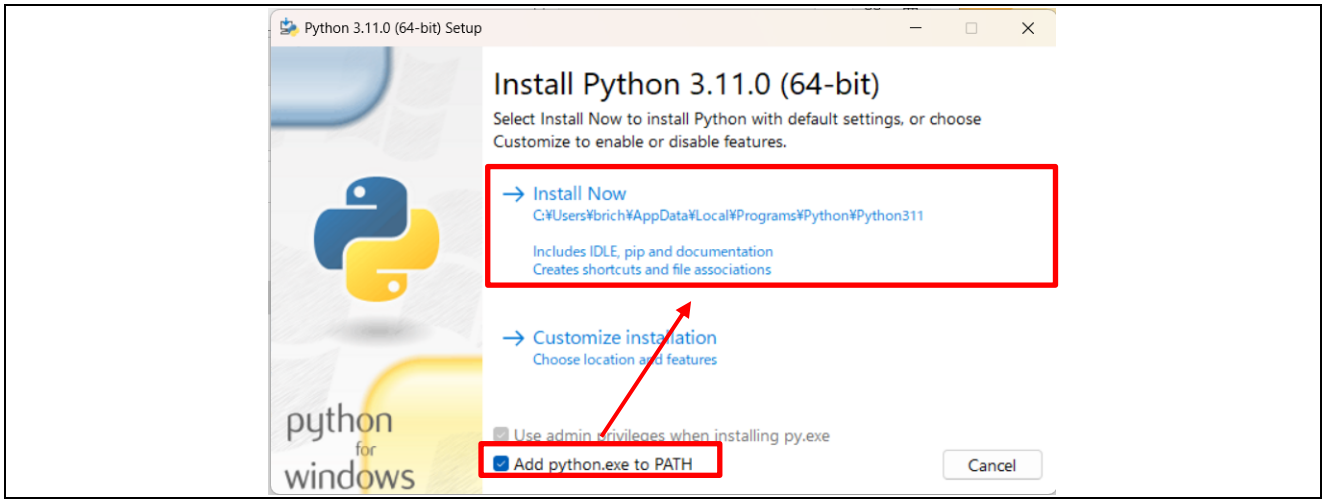

Figure 70. Python 3.11.0 installer

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 71. Checking version of Python

4. Install the Python encryption library (pycryptodome)

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

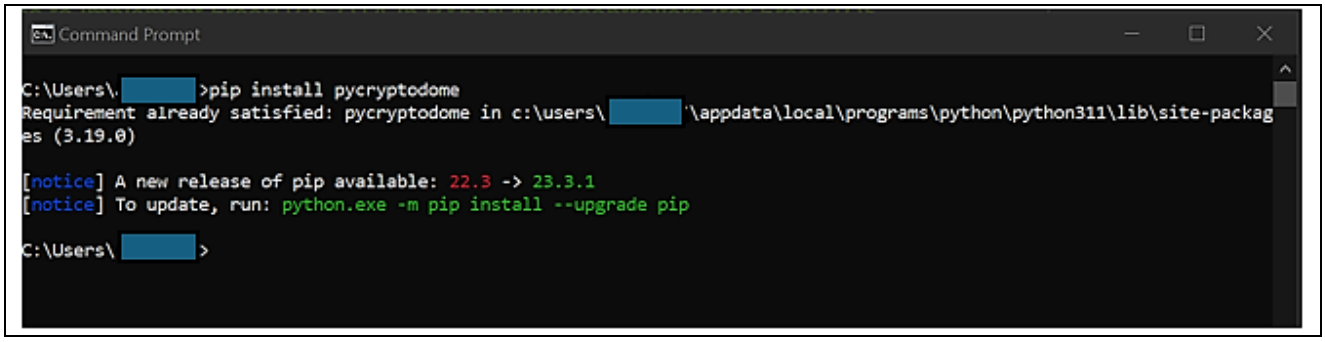

Figure 72. Installing Python encryption library

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.

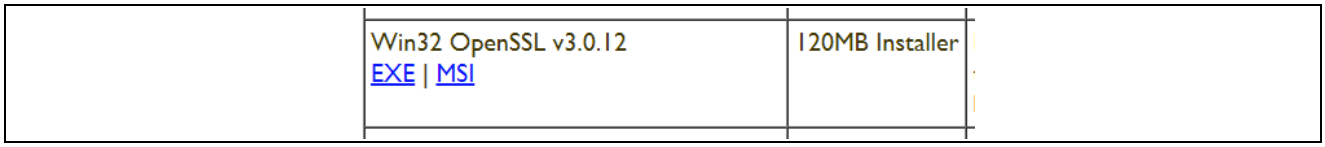

Figure 73. OpenSSL Download Website

2. Run the installer and follow the prompts to install OpenSSL.
3. Open the Win64 OpenSSL Command Prompt and confirm that OpenSSL is installed.

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

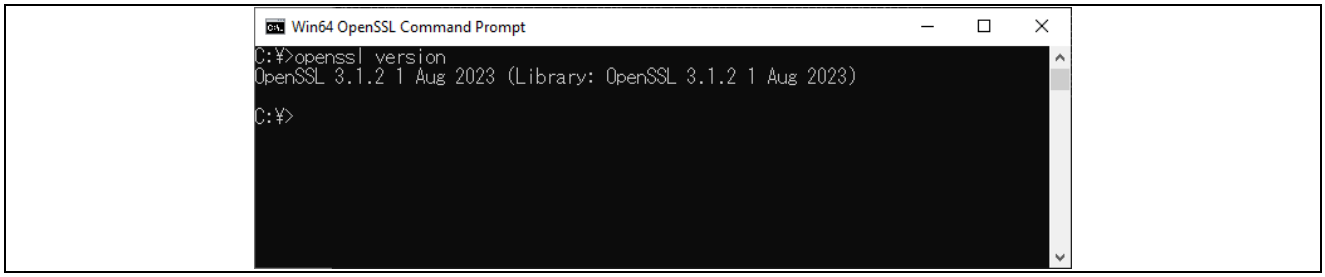

Figure 74. Checking version of OpenSSL

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.2.01 - Sample Code | Renesas] and download the firmware update module.

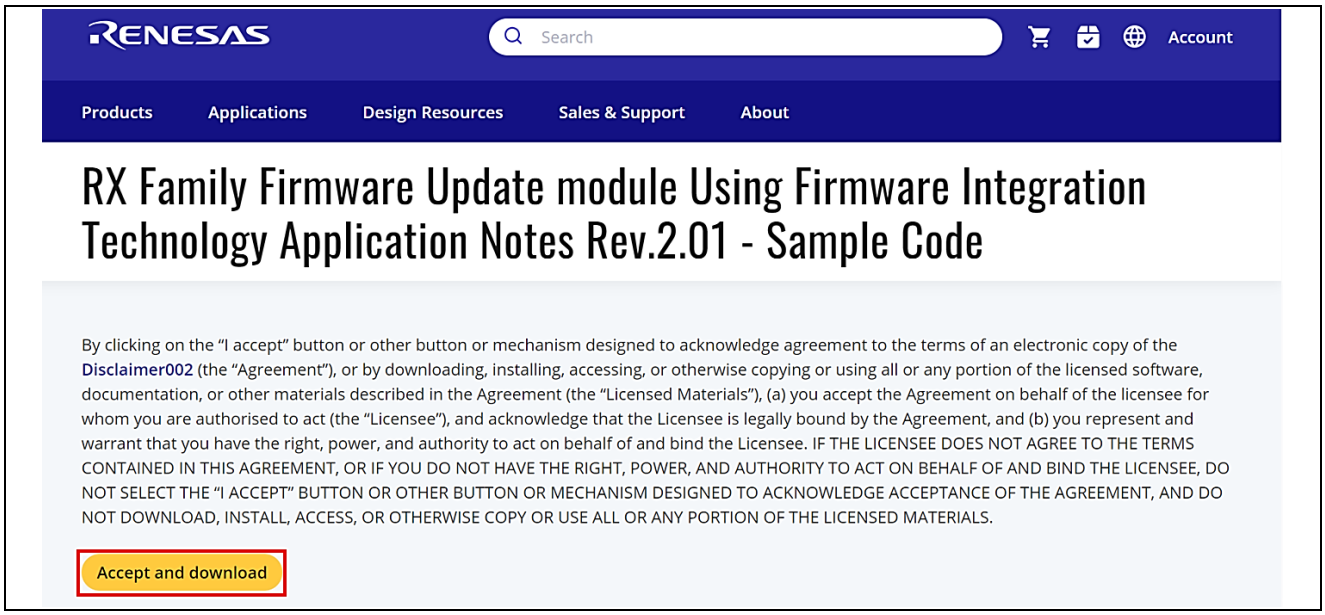

Figure 75. Renesas Image Generator Downloading (1/2)

2. Extract the downloaded firmware update module.

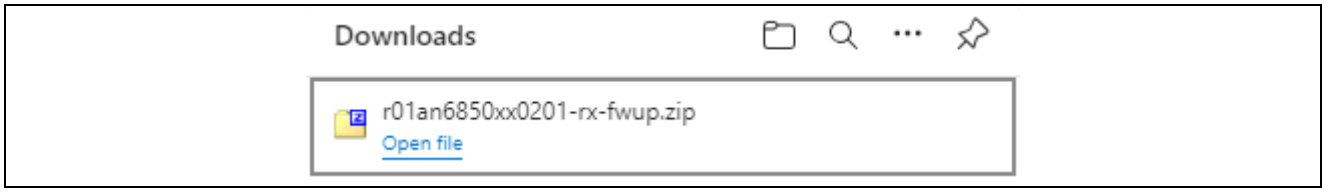

Figure 76. Renesas Image Generator Downloading (2/2)

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).

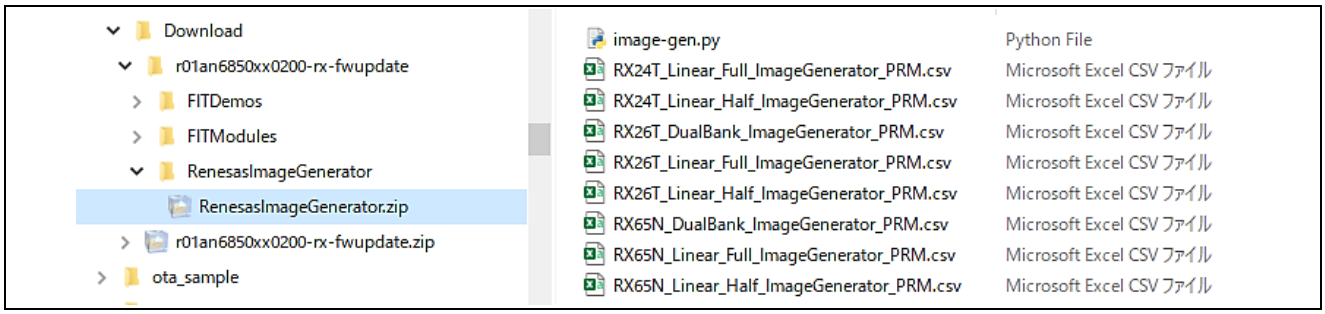

Figure 77. Renesas Image Generator package

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.**

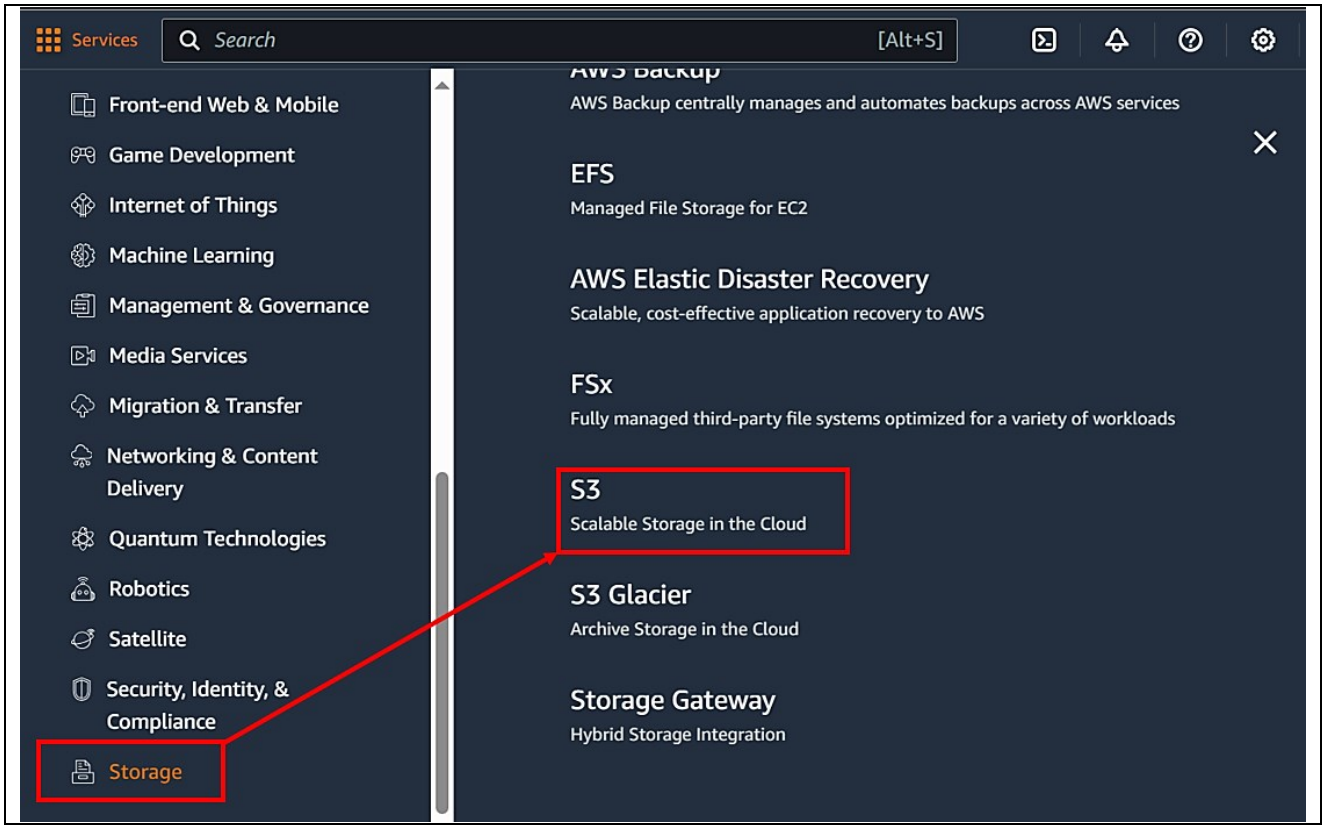

Figure 78. Create AWS S3 bucket (1/2)

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

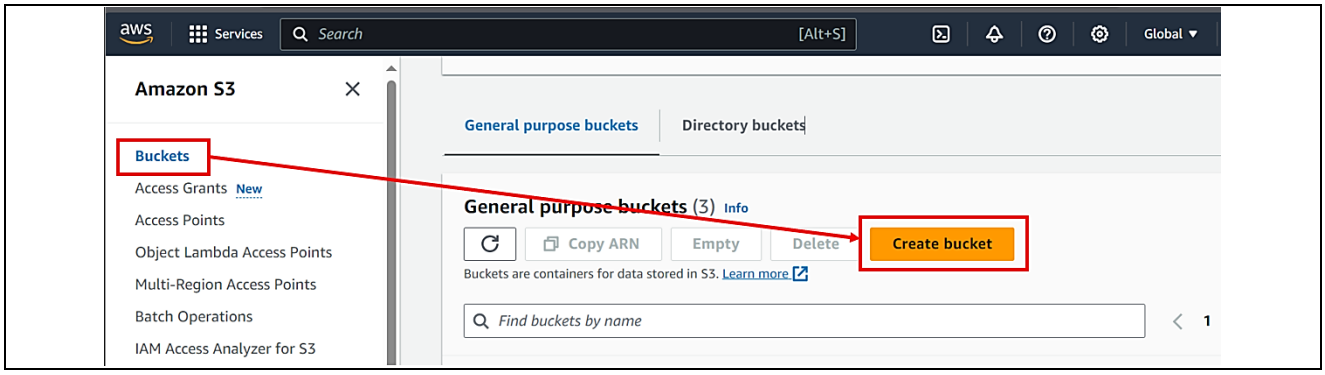

Figure 79. Create AWS S3 bucket (2/2)

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

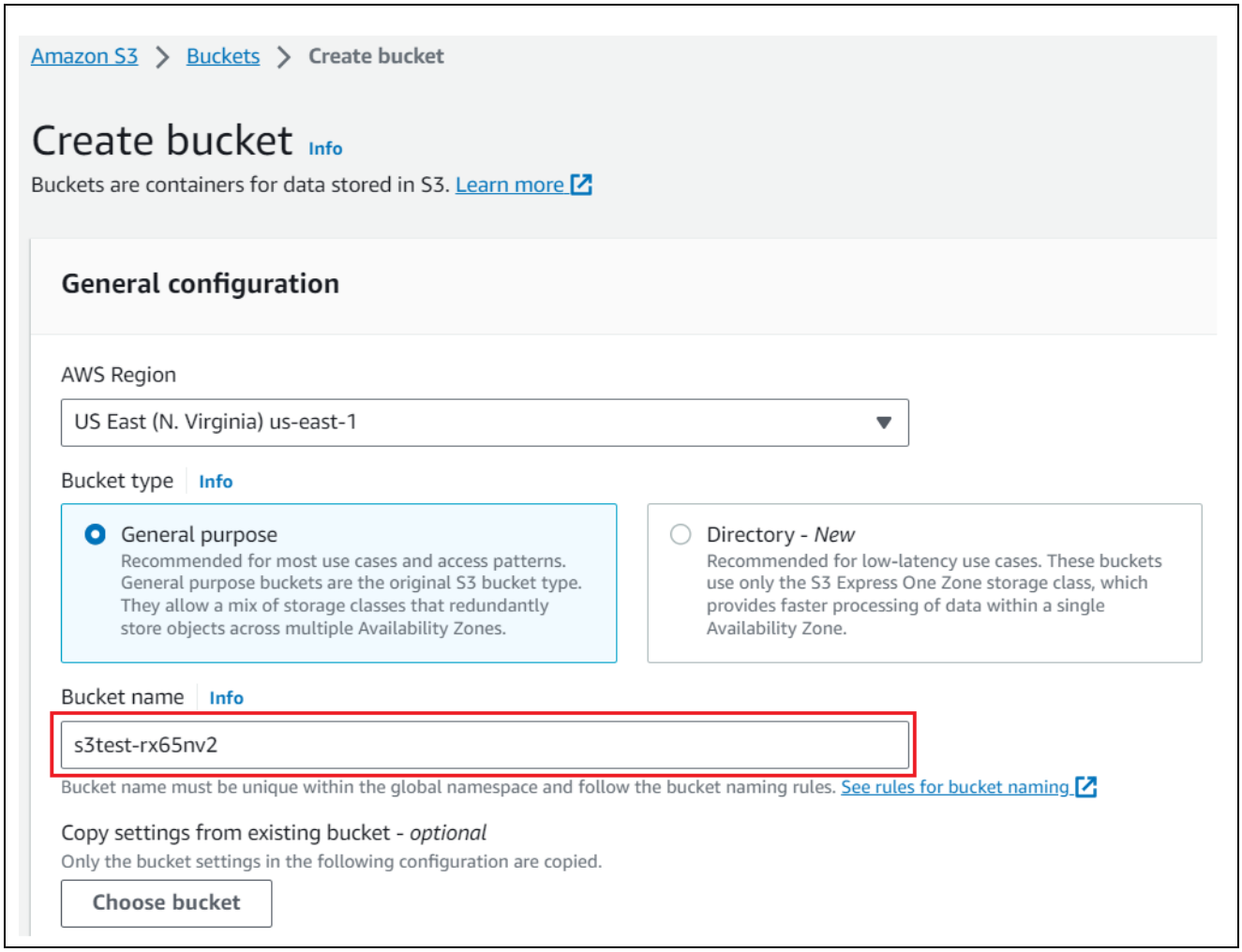

Figure 80. Naming for bucket

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

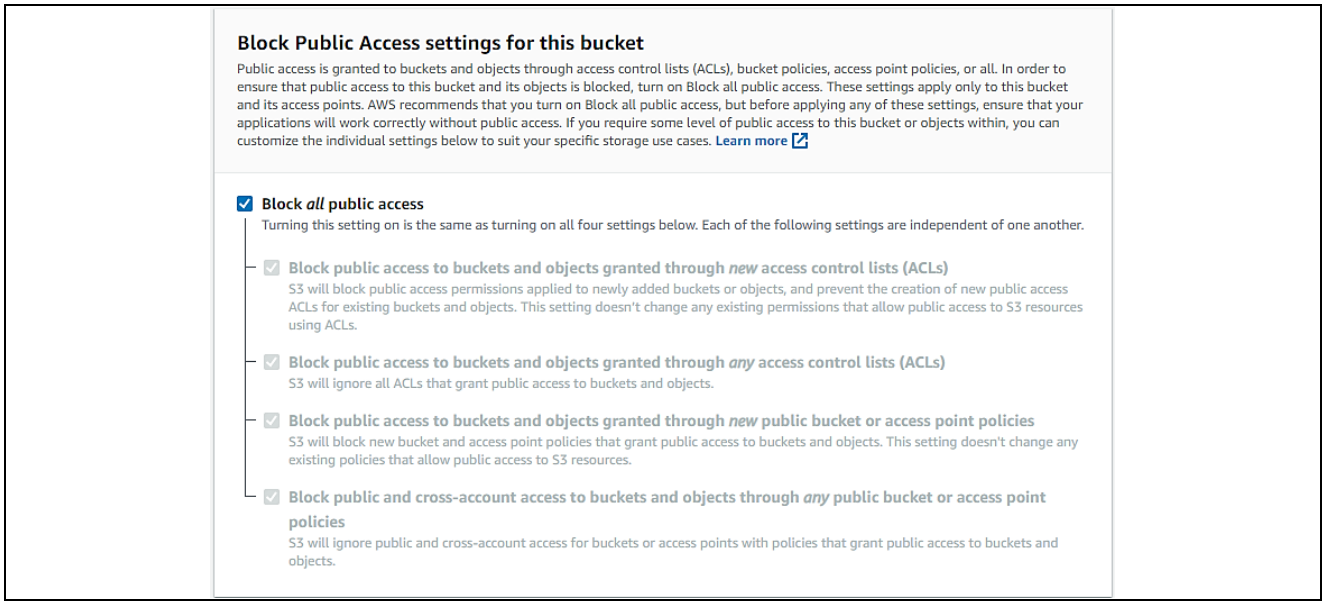

Figure 81. Bucket setting (1/2)

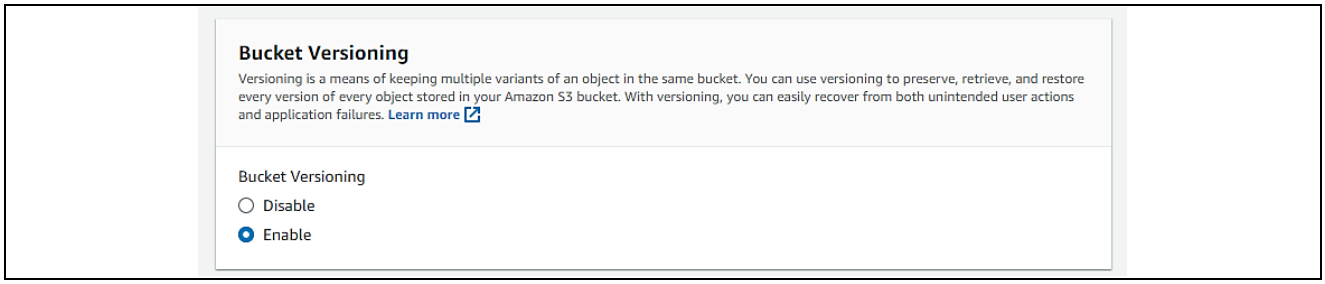

Figure 82. Bucket setting (2/2)

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.

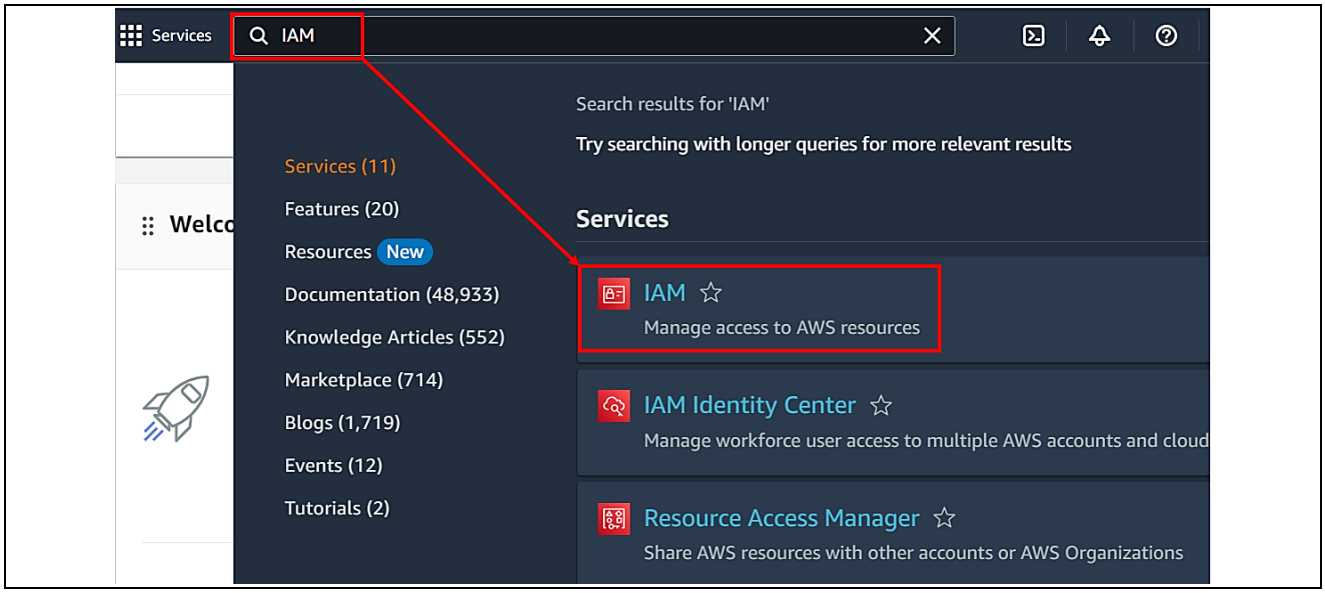

Figure 83. IAM search box

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

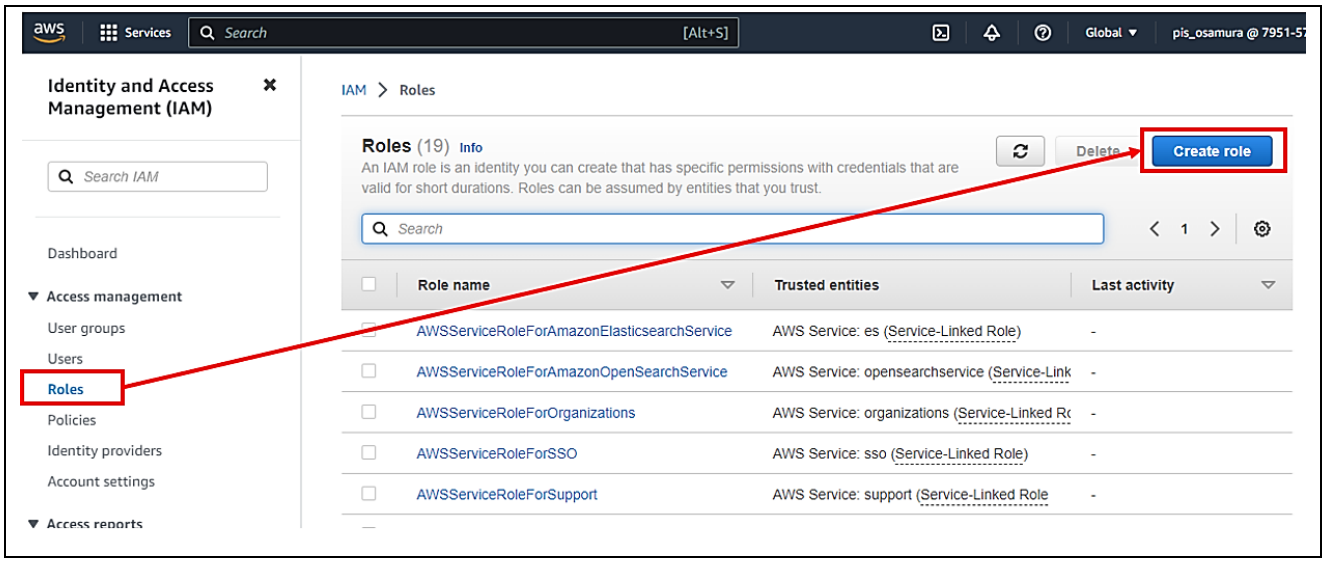

Figure 84. Creating a role

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

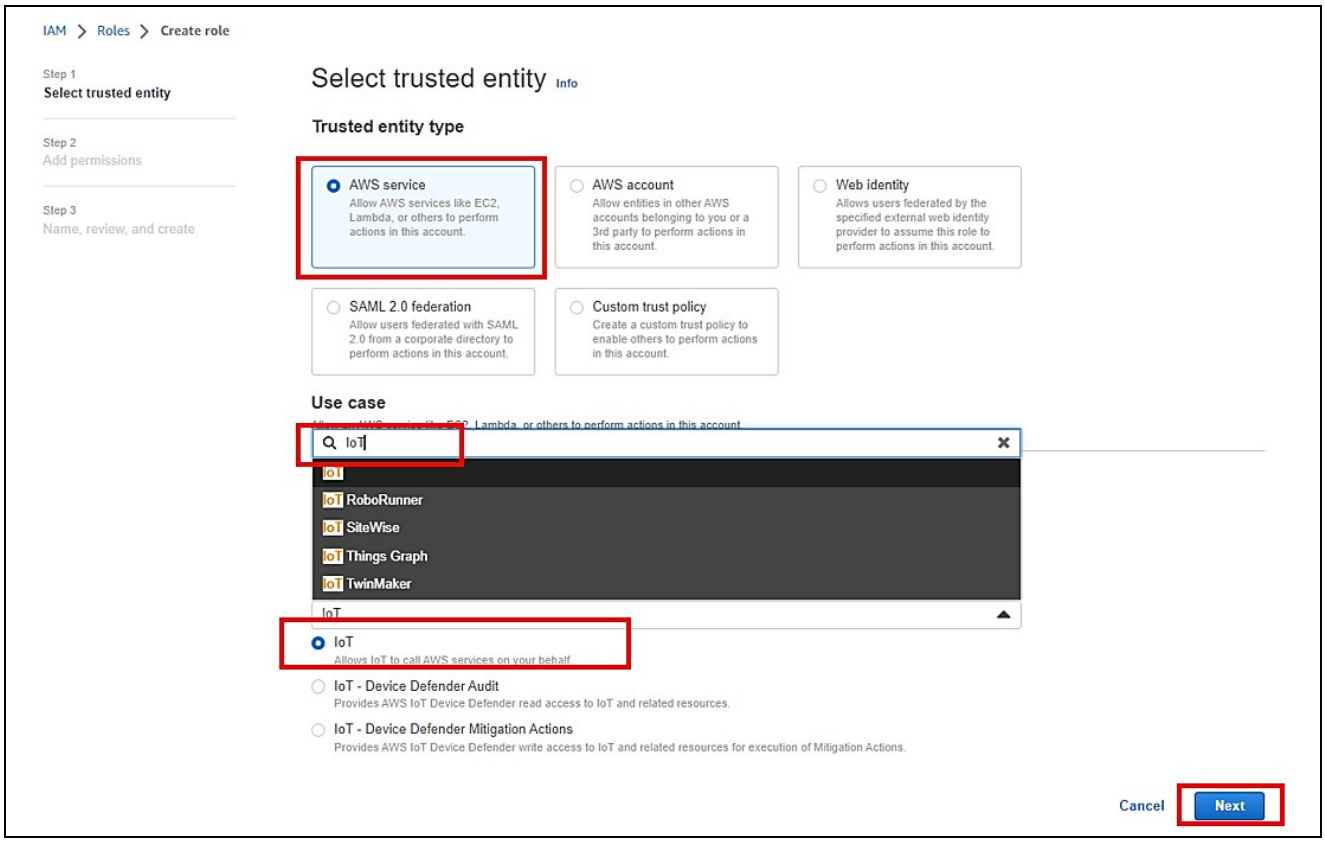

Figure 85. Selecting trusted entity

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

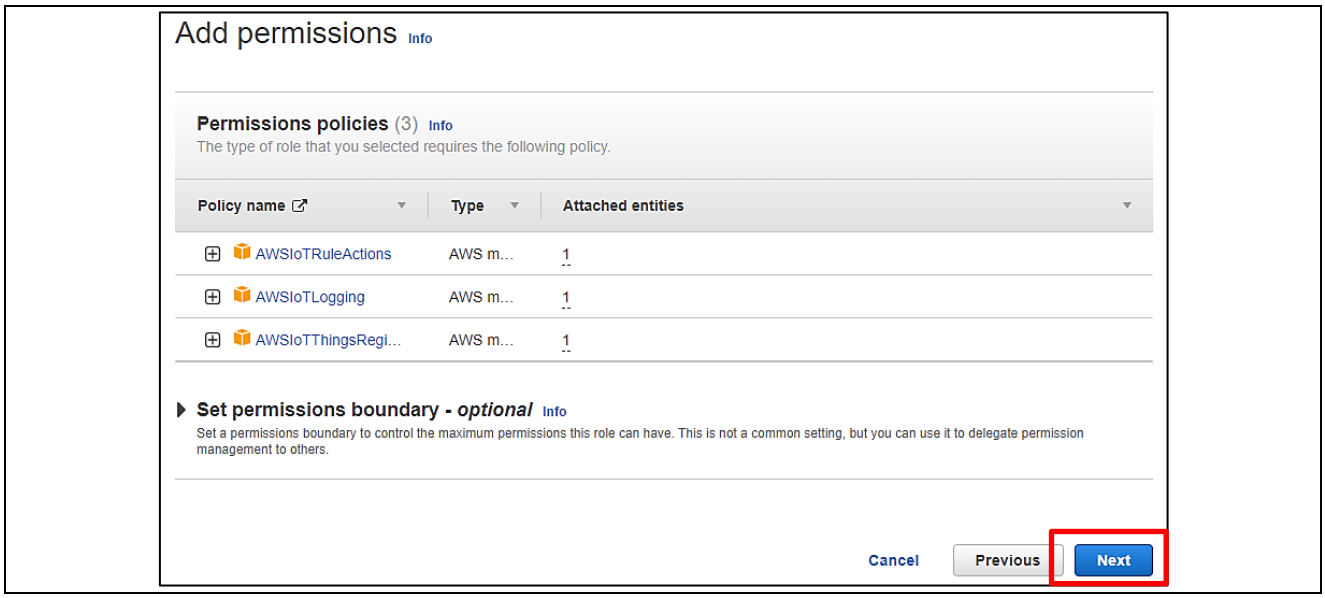

Figure 86. Add Permissions for Role

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

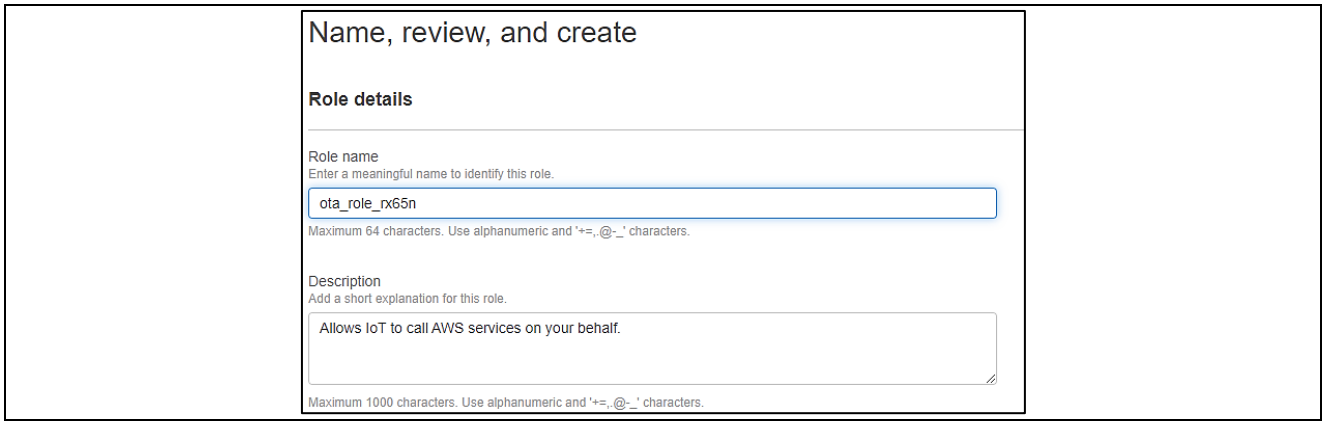

Figure 87. Naming for Created Role

6. Click on the role you created.

Figure 88. Created Role

7. Select Attach policies.

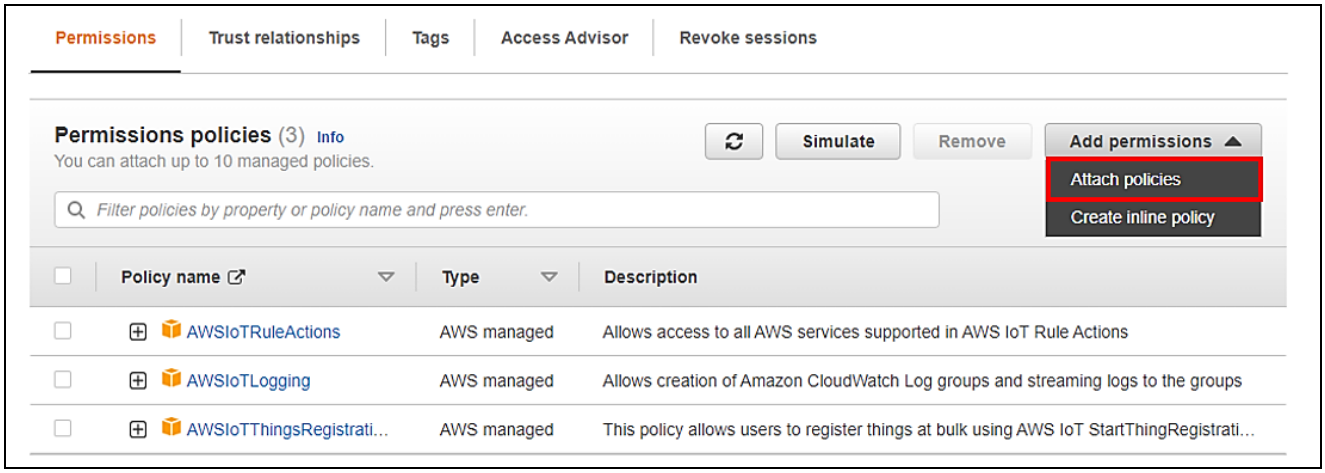

Figure 89. Add permission (1/3)

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

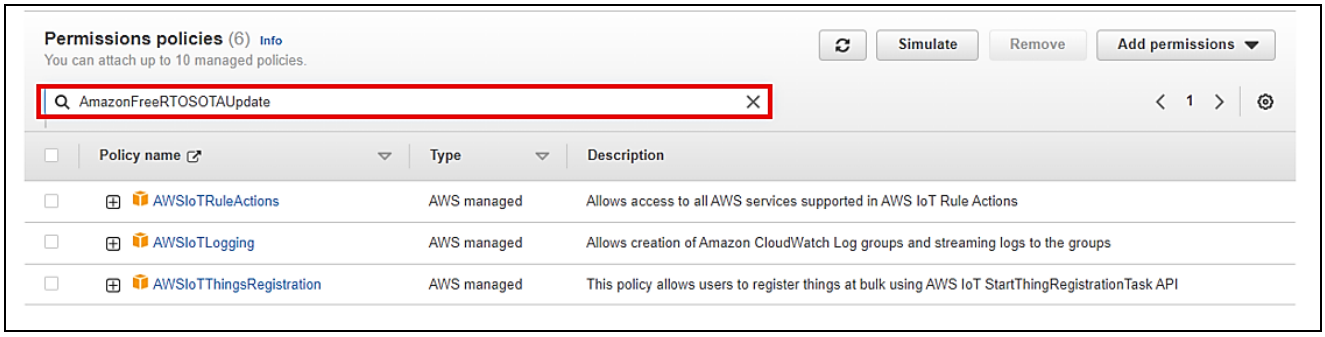

Figure 90. Add permission (2/3)

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

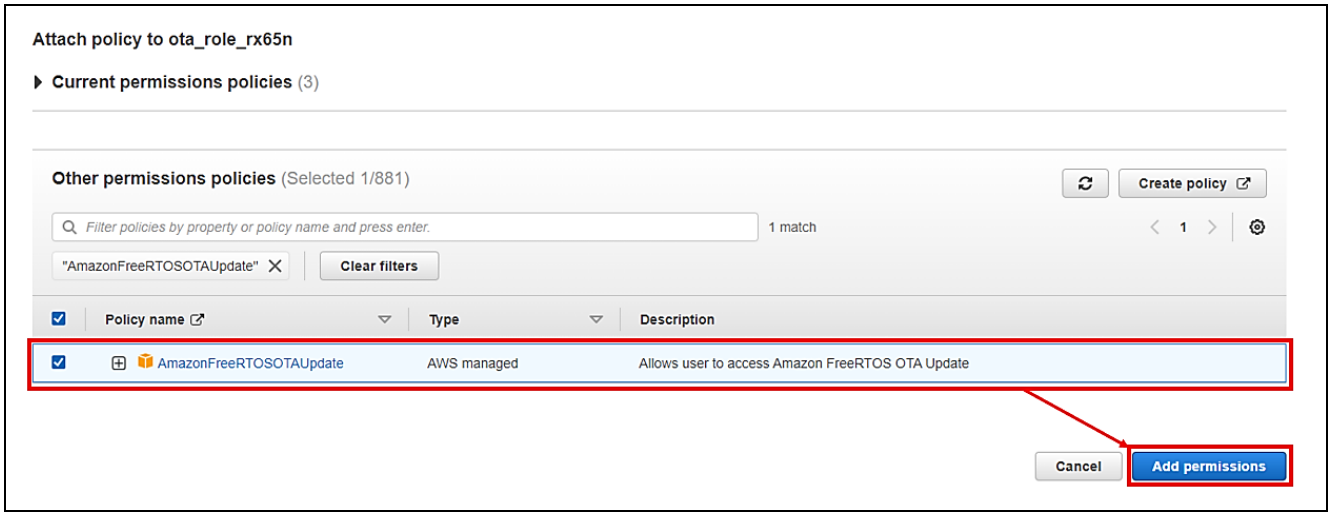

Figure 91. Add permission (3/3)

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

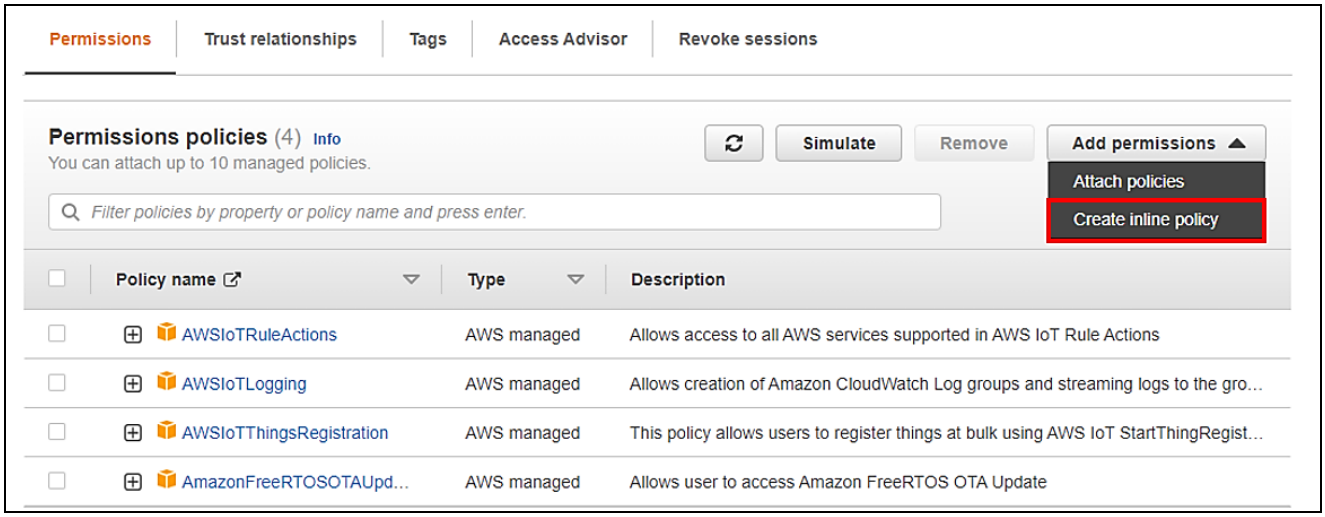

Figure 92. Create inline policy (1/3)

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": "*"
    }
  ]
}
```

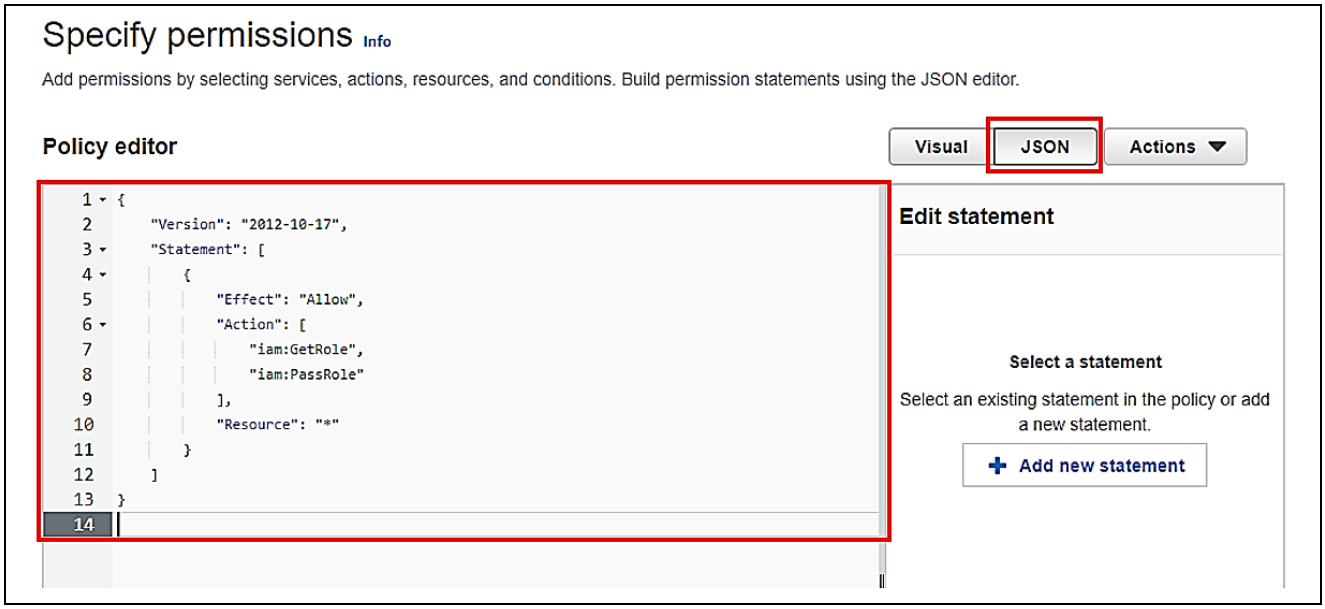

Figure 93. Create inline policy (2/3)

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

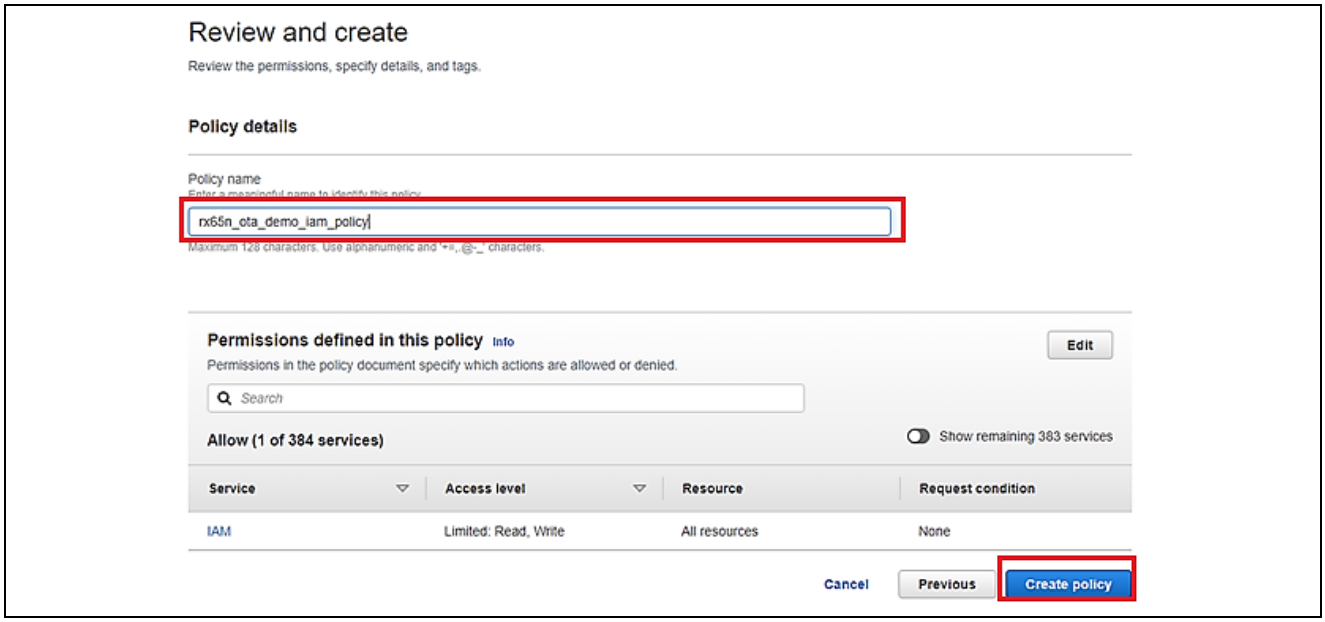

Figure 94. Create inline policy (3/3)

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

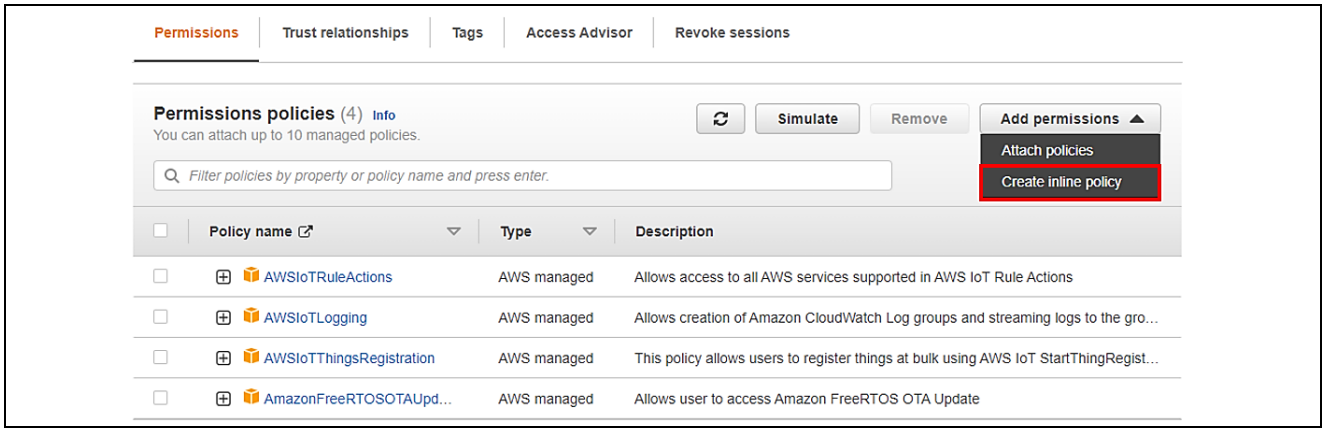

Figure 95. Create inline policy (1/3)

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": [
            "*"
        ]
    }
  ]
}
```

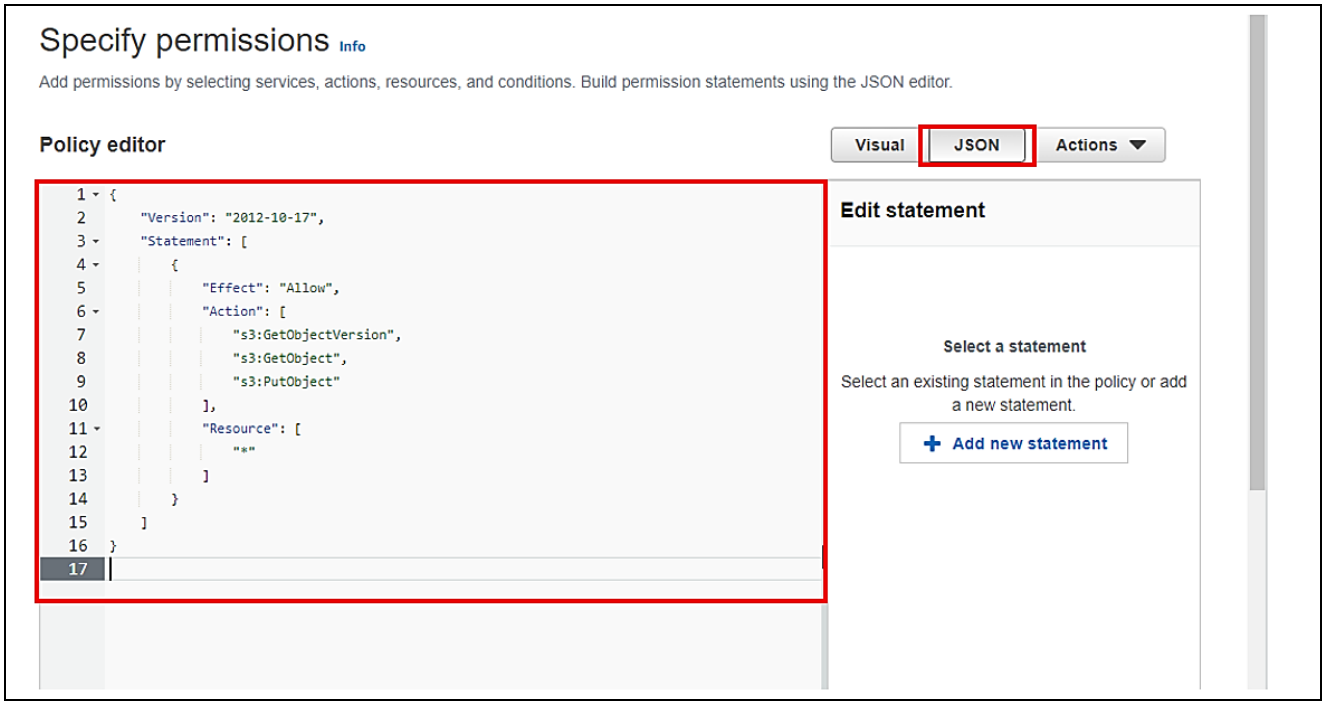

Figure 96. Create inline policy (2/3)

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

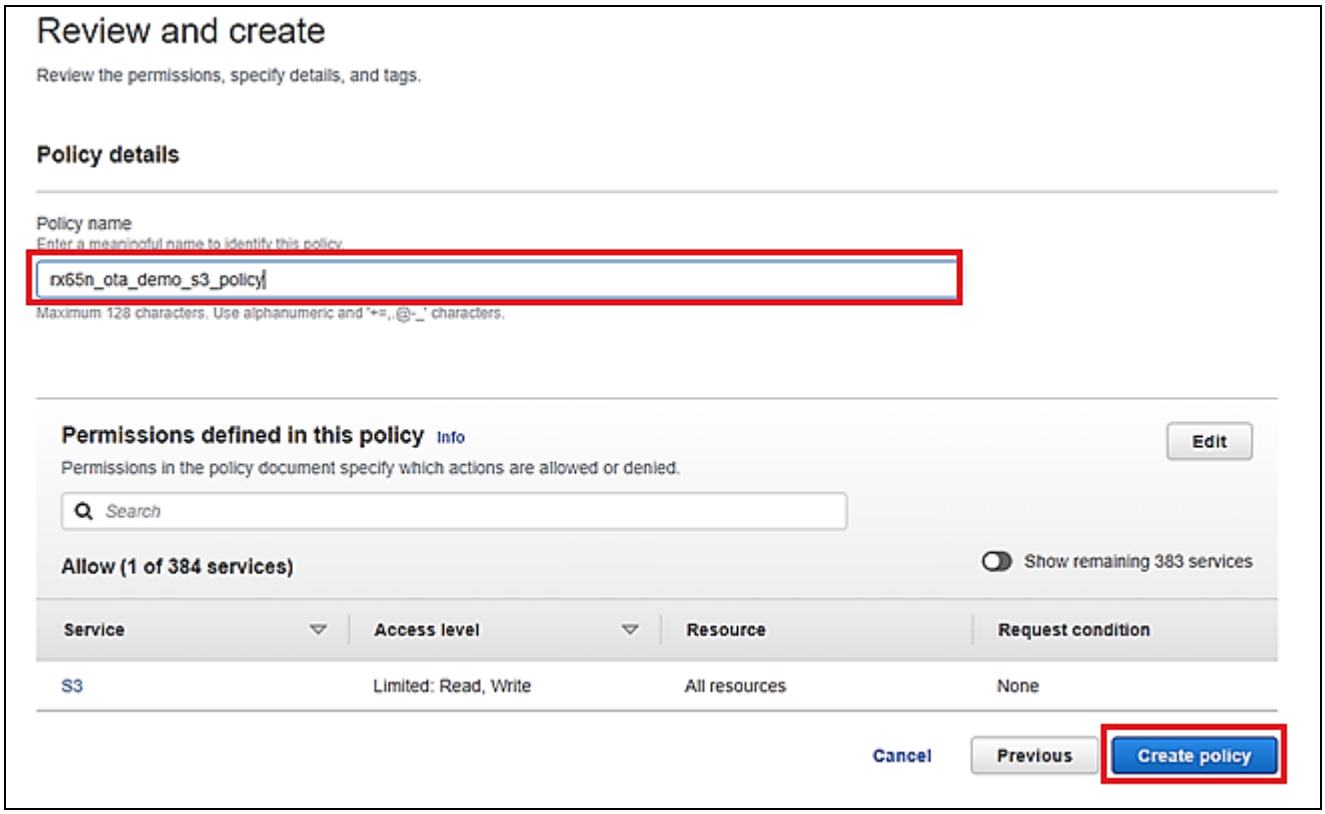

Figure 97. Create inline policy (3/3)

6.4 Setting up the Device

6.4.1 Generating Key Pairs and Certificates

1. Open the **Win64 OpenSSL Command Prompt** and create a CA private key using ECDSA.

Execute the following command: *$openssl ecparam -genkey -name secp256r1 -out ca.key*

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

Figure 98. Creating CA private key

2. Create a CA certificate from the created CA private key.

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

```
C:¥openssl>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 []
```

Figure 99. Creating CA certificate

3. Create an ECDSA key pair.

Execute the following command: *$openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair*

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

Figure 100. Creating ECDSA key pairs

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

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

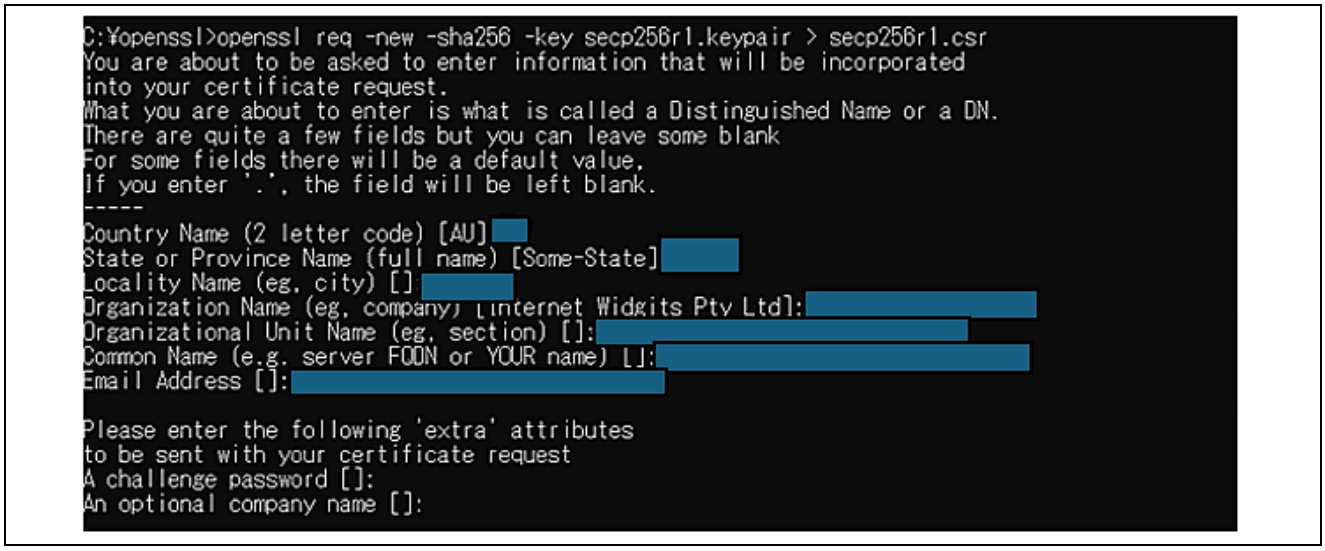

Figure 101. Creating certificate signing request

5. 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*

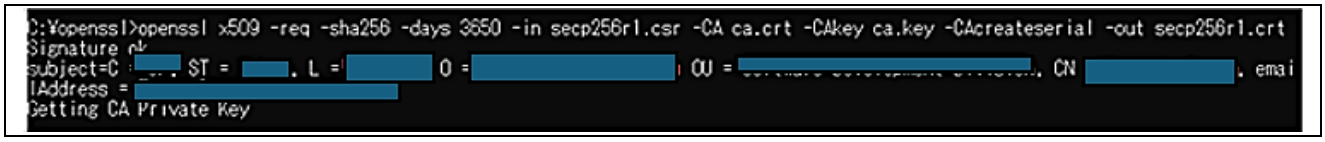

Figure 102. Creating code signing certificate

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

Execute the following command:

$openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey

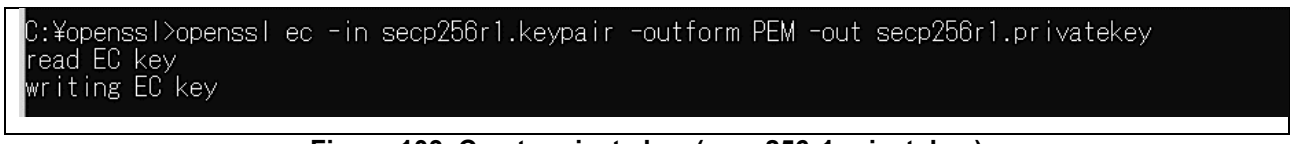

Figure 103. Create private key (secp256r1.privatekey)

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

Execute the following command:

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

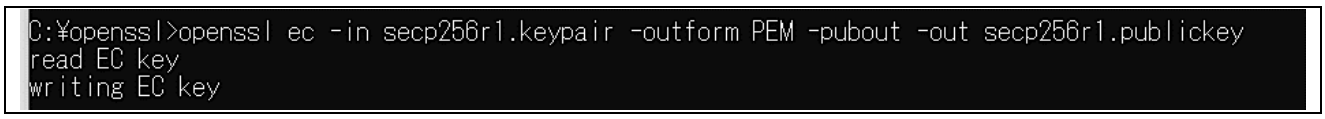

Figure 104. Create public key (secp256r1.publickey)

8. After creating, please check the result:

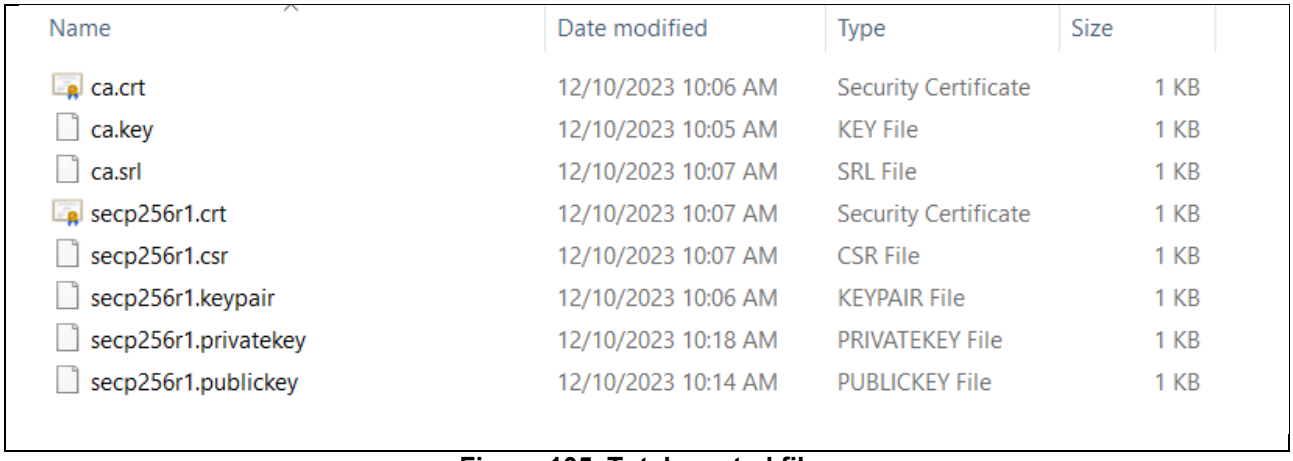

Figure 105. Total created files

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 **boot_loader_ck_rx65n_v2** project:

Make sure that configuration in **boot_loader_ck_rx65n.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**	Enable user stdio charput function: **Use user charput() function**
2	**Drivers→Memory→r_flash_rx**	Enable code flash programming: **Includes code to program ROM area**
		Enable code flash self-programming: **Programming code flash while executing from another segment in ROM**
3	**Drivers→Communications→r_sci_rx**	Include software support for channel 5: **Include**
4	**Middleware→Generic→r_fwup**	Select the function mode: **user for Boot Loader**
		Main area start address: **0xFFF00000**
		Buffer area start address: **0xFFE00000**
		Install area size: **0xF0000**

Check the boot loader device.

And also in the **boot_loader_ck_rx65n.scfg** file, click the **Board** tab. Confirm that R5F565NEHxFB_DUAL appears in the **Board** field.

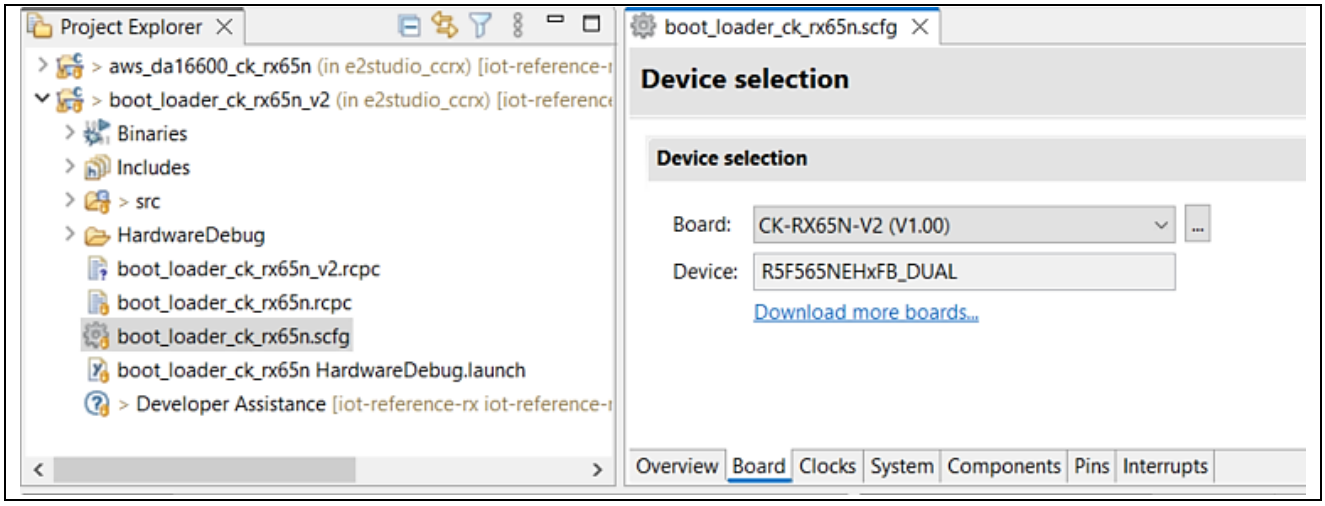

Figure 106. bootloader's device selection

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

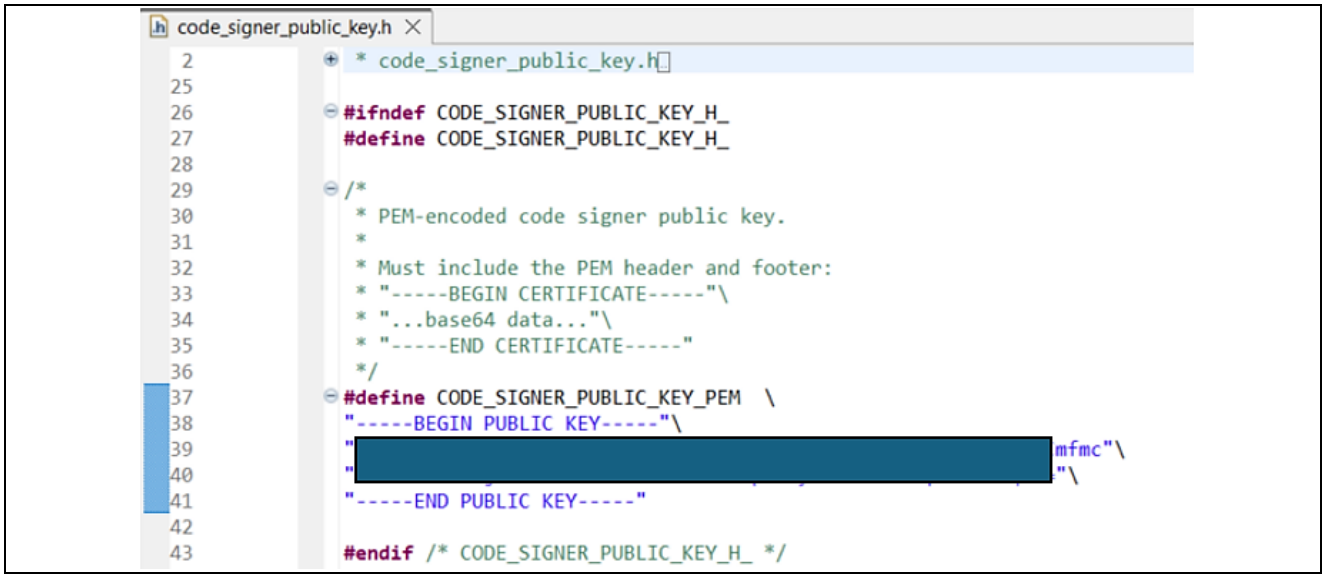

Figure 107. Add Code Signing Public Key to Boot Loader Project

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

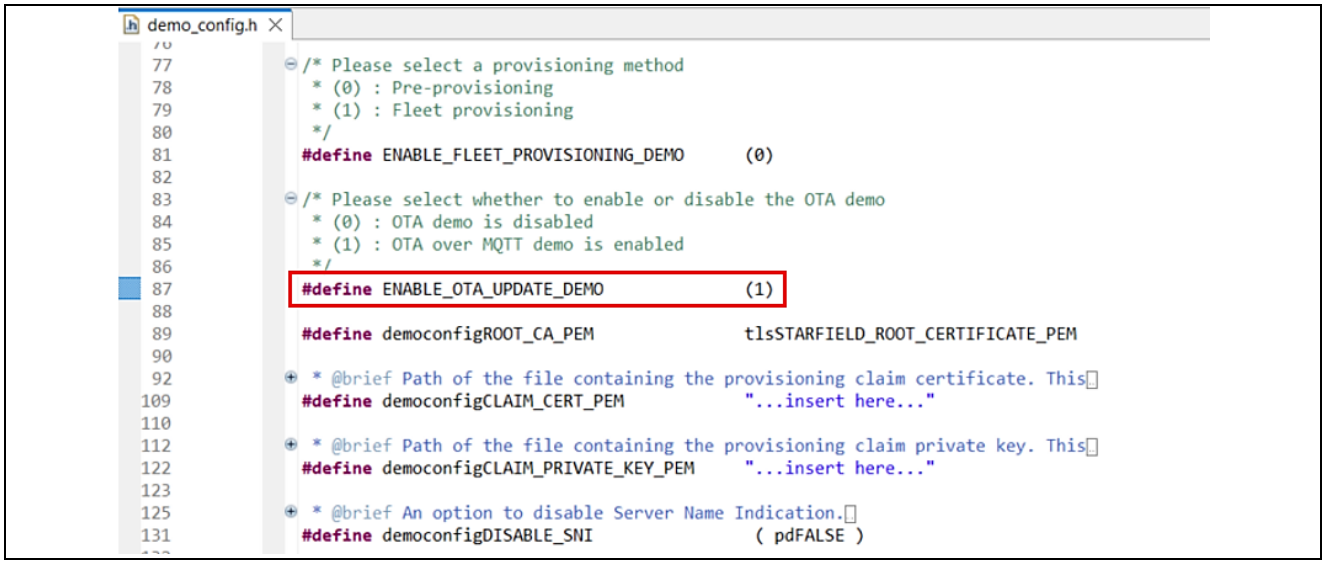

Figure 108. Enable OTA Demo

3. 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 **Toolchain** tab, confirm that the toolchain is **Renesas CC-RX**

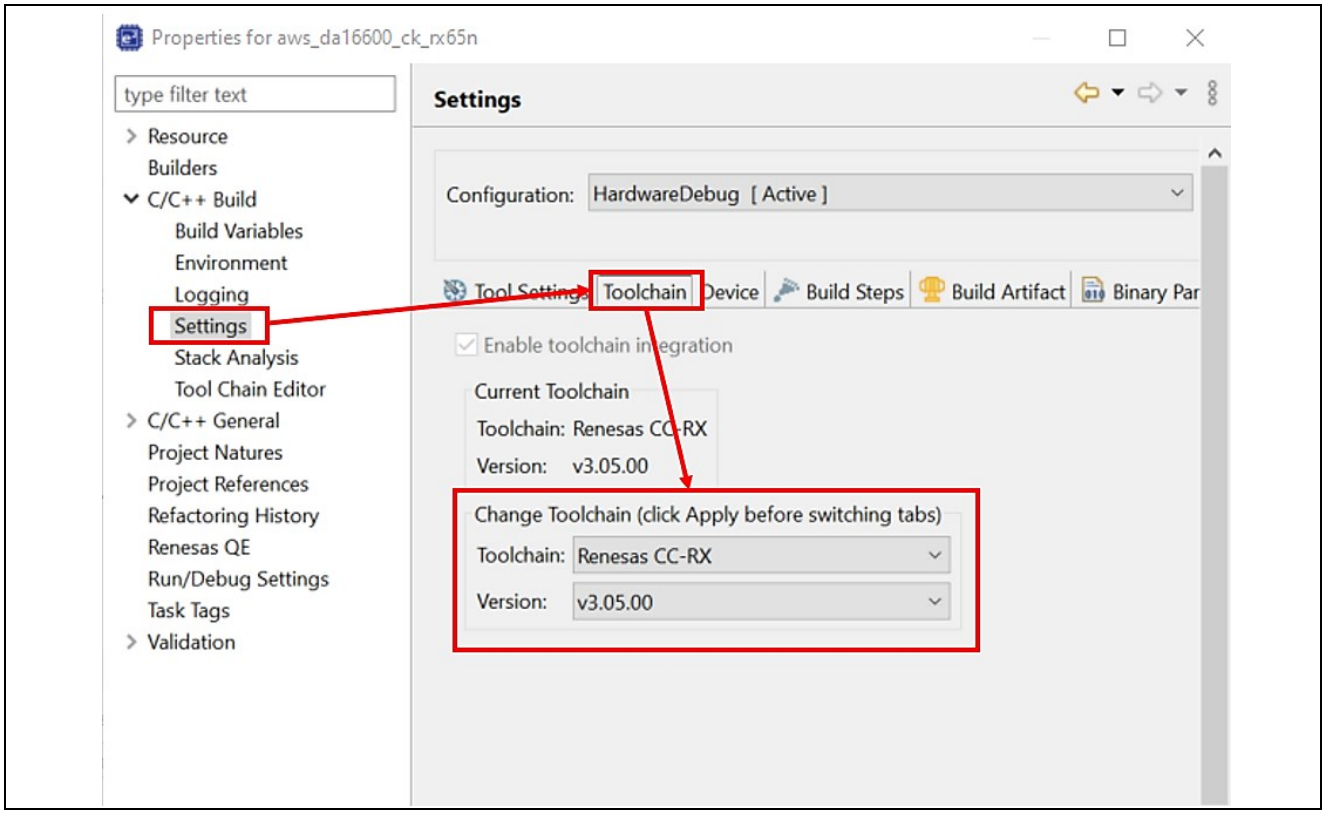

Figure 109. Project toolchain

4. On the **Tool Settings** tab, expand the **Converter** menu and select **Output**. Confirm that the **Motorola S format file** check box is selected.

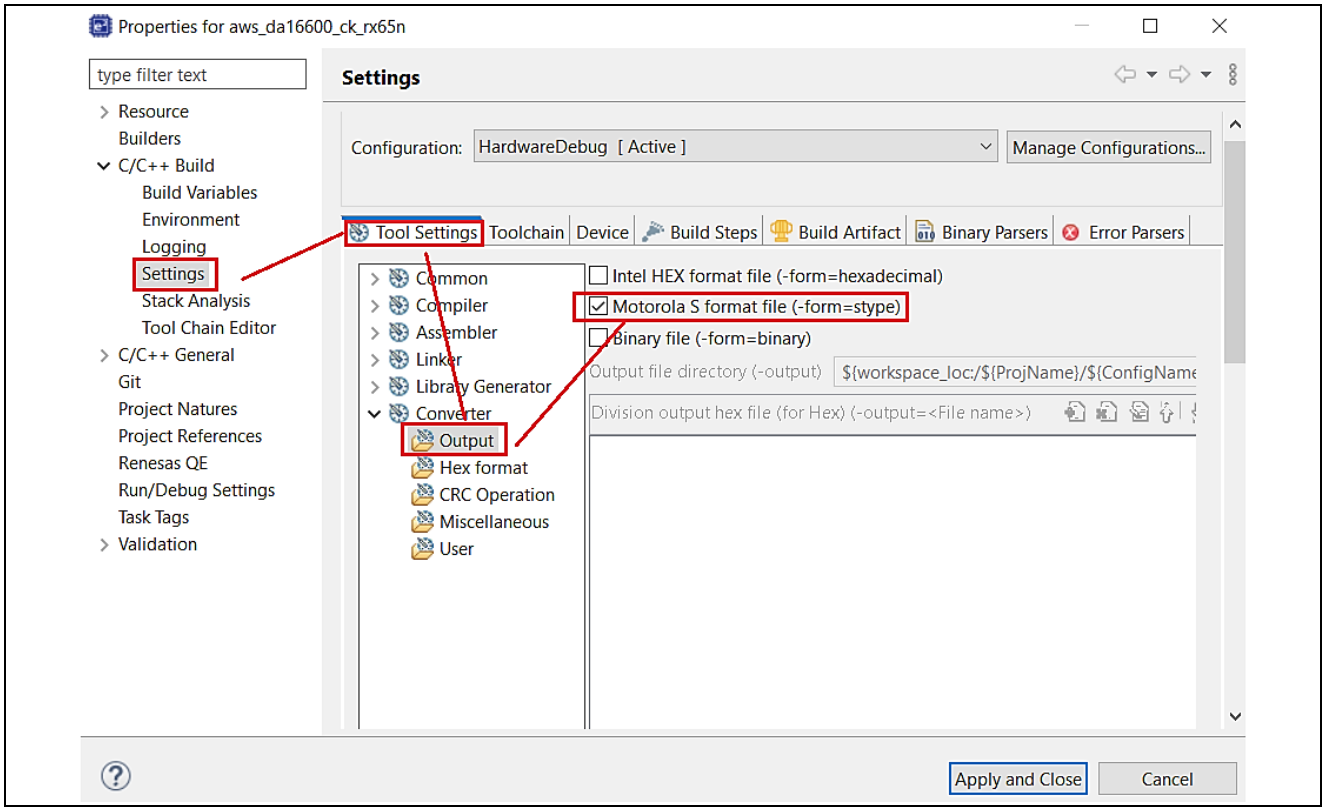

Figure 110. Project setting

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.

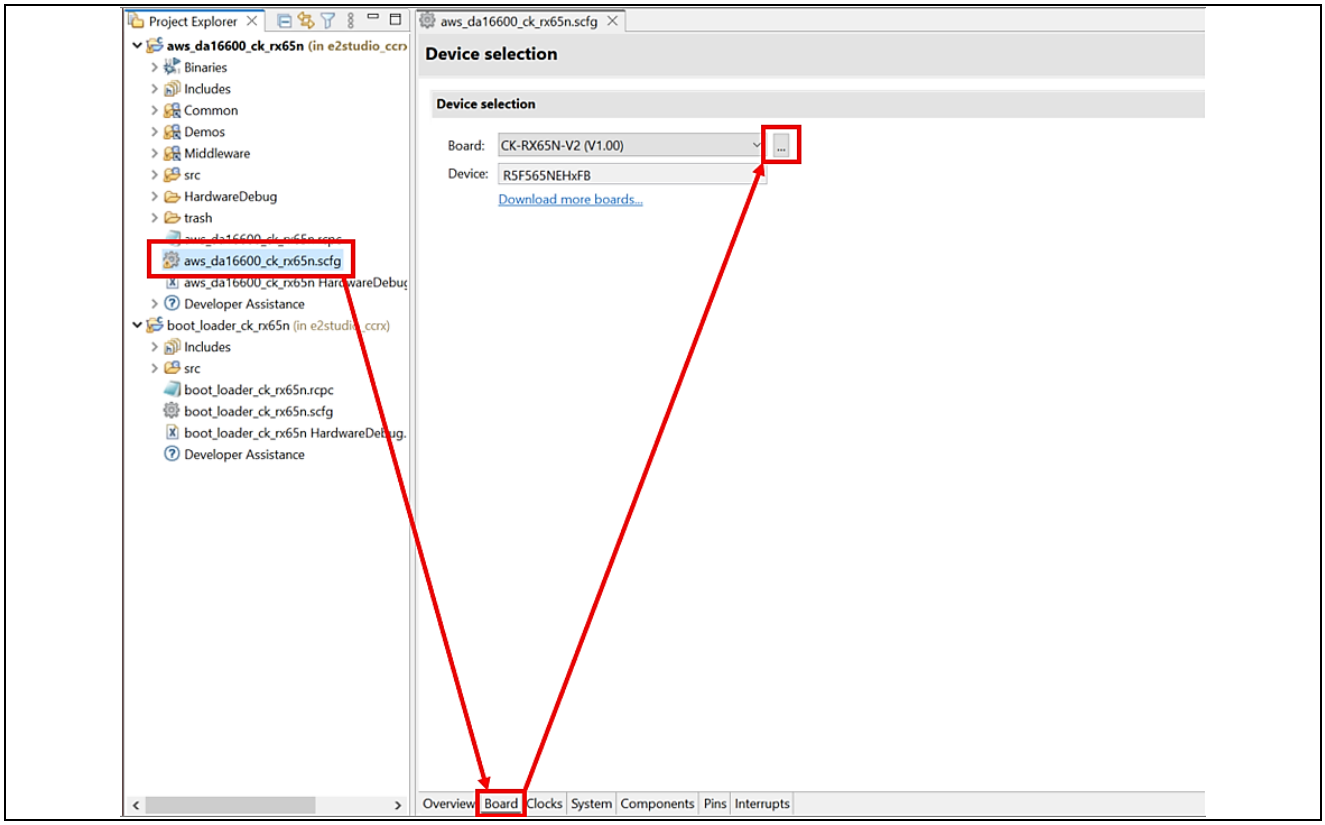

Figure 111. Device selection (1/3)

Click the ellipsis (**...**) beside the **Target Device** field and select R5F565NEHxFB_DUAL. The value in the **Target Board** drop-down list changes to **Custom**.

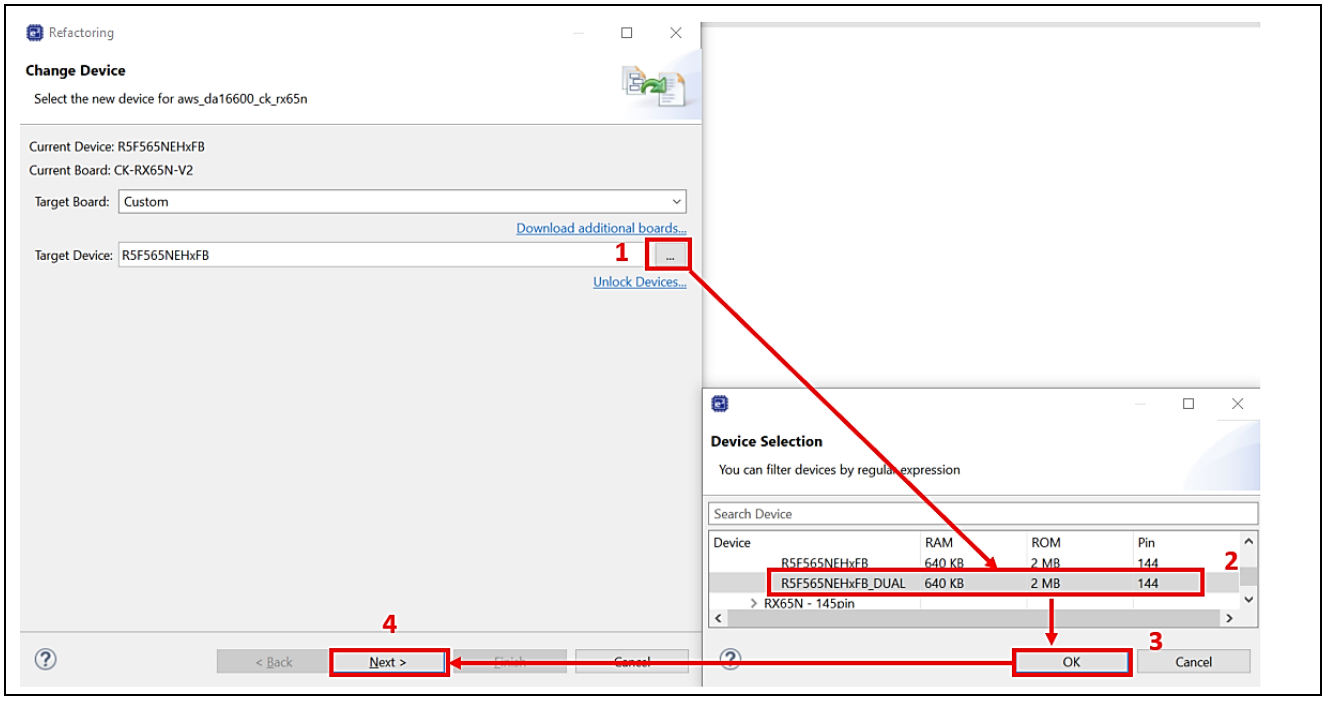

Figure 112. Device selection (2/3)

Under **Build Settings** > **HardwareDebug** > **Toolchain Settings**, clear the **ROM to RAM mapped section (-rom)** and **Sections (-start)** check boxes and then click **Finish**.

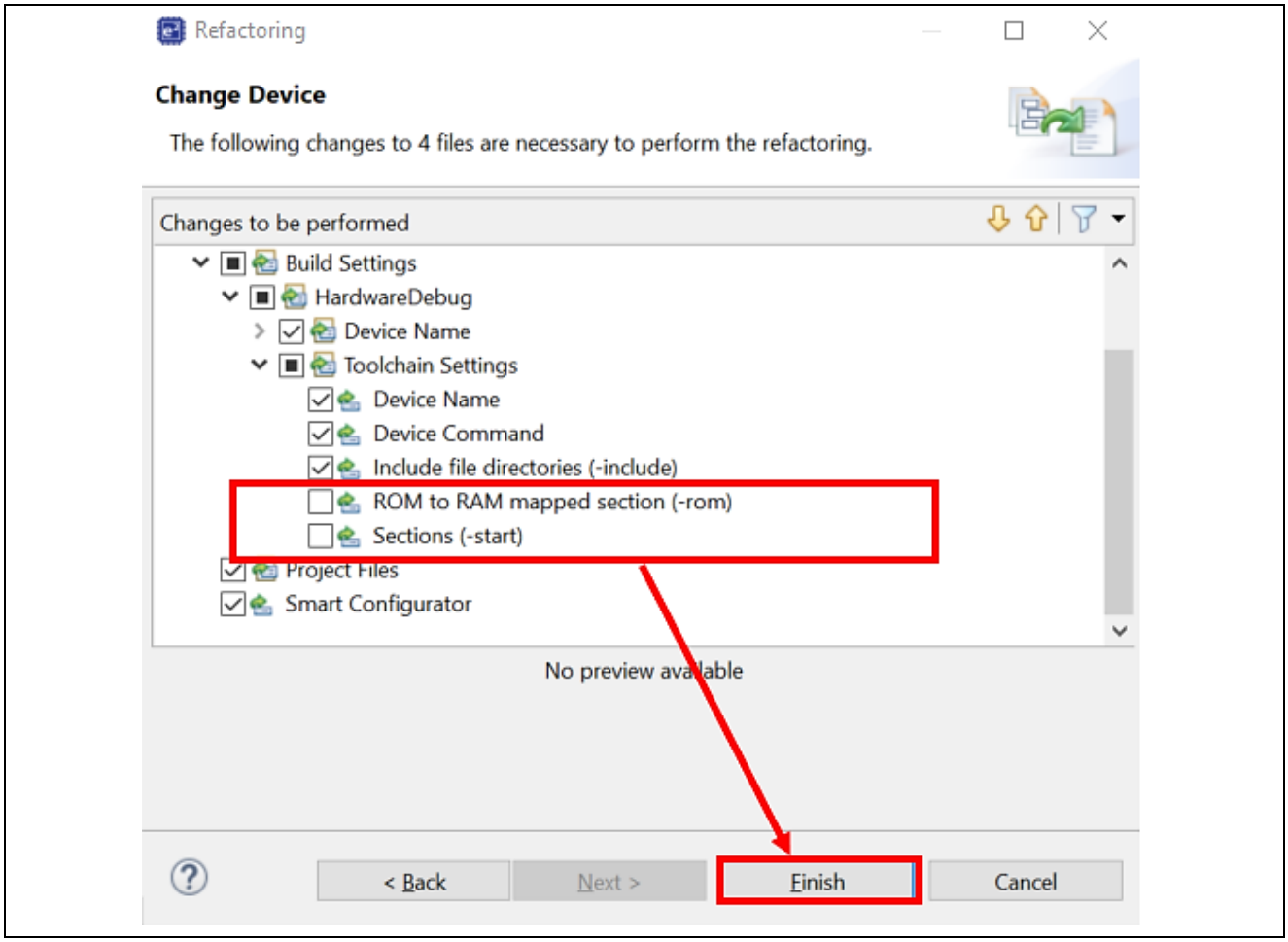

Figure 113. Device selection (3/3)

6. Change the firmware (aws_da16600_ck_rx65n) vector.

   Open the aws_da16600_ck_rx65n project and select **Project** and then **Properties**.

Expand the **C/C++ Build** menu and click **Settings**. In the menu tree on the **Tool Settings** tab, expand the **Linker** menu and click **Section**, and open the Section Viewer. **Allocate EXCEPTVECT to 0xFFFEFF80 and RESETVECT to 0xFFFEFFFC.**

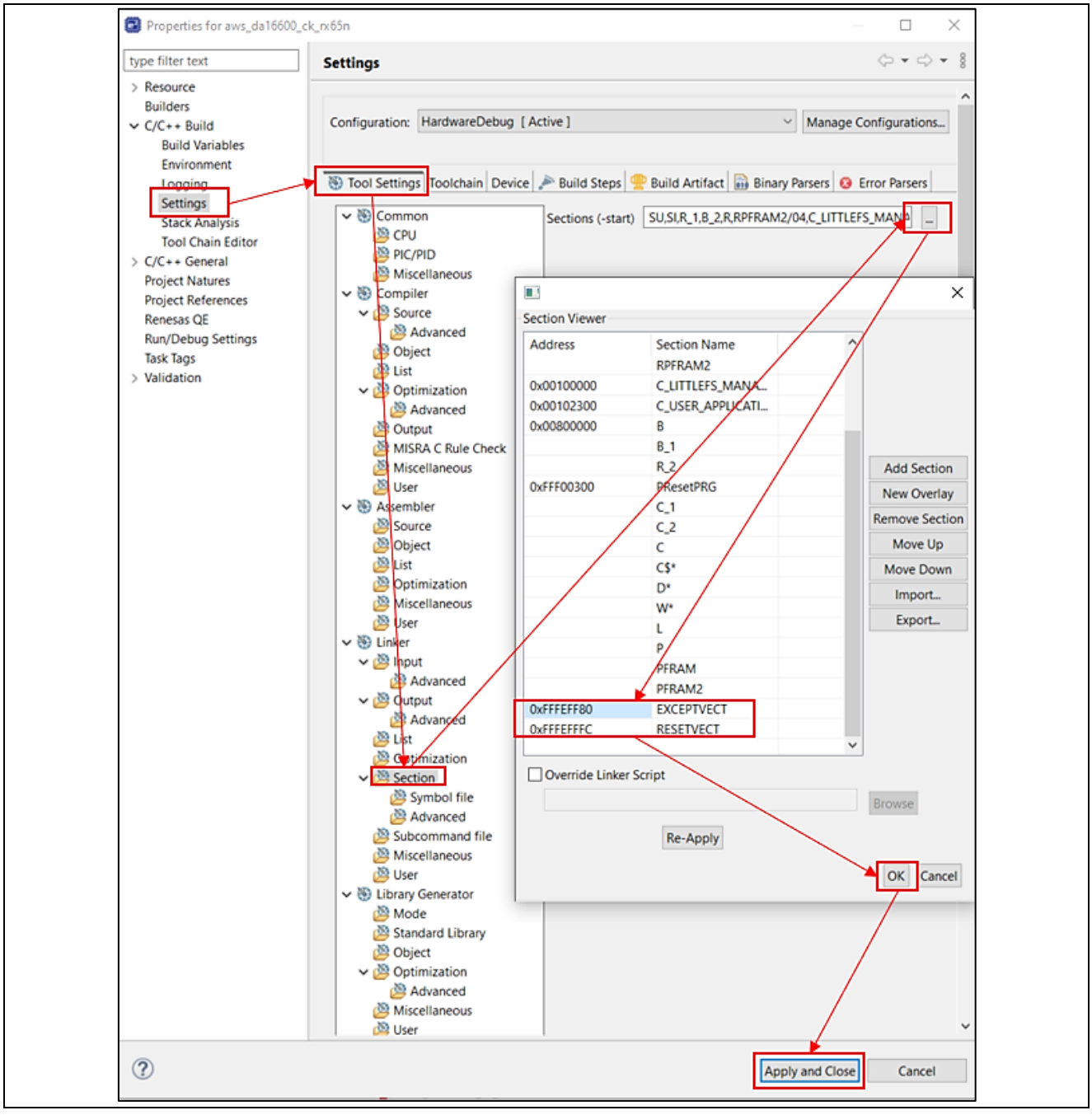

Figure 114. Vector setting

Build the project as instruction in Topic: **5.1.3.Import the Project**

7. Generate the initial firmware.

   Place the following files in the Renesas Image Generator folder:

- The results of the 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:

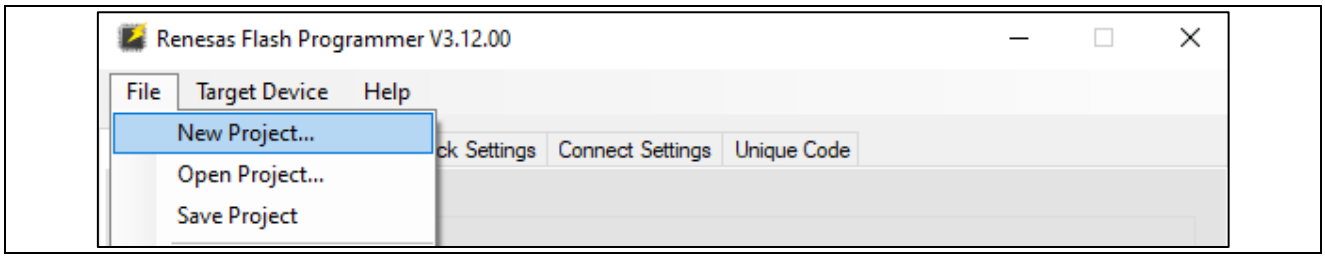

Figure 115. Create New Flash Project (1/4)

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 “Debug” mode.

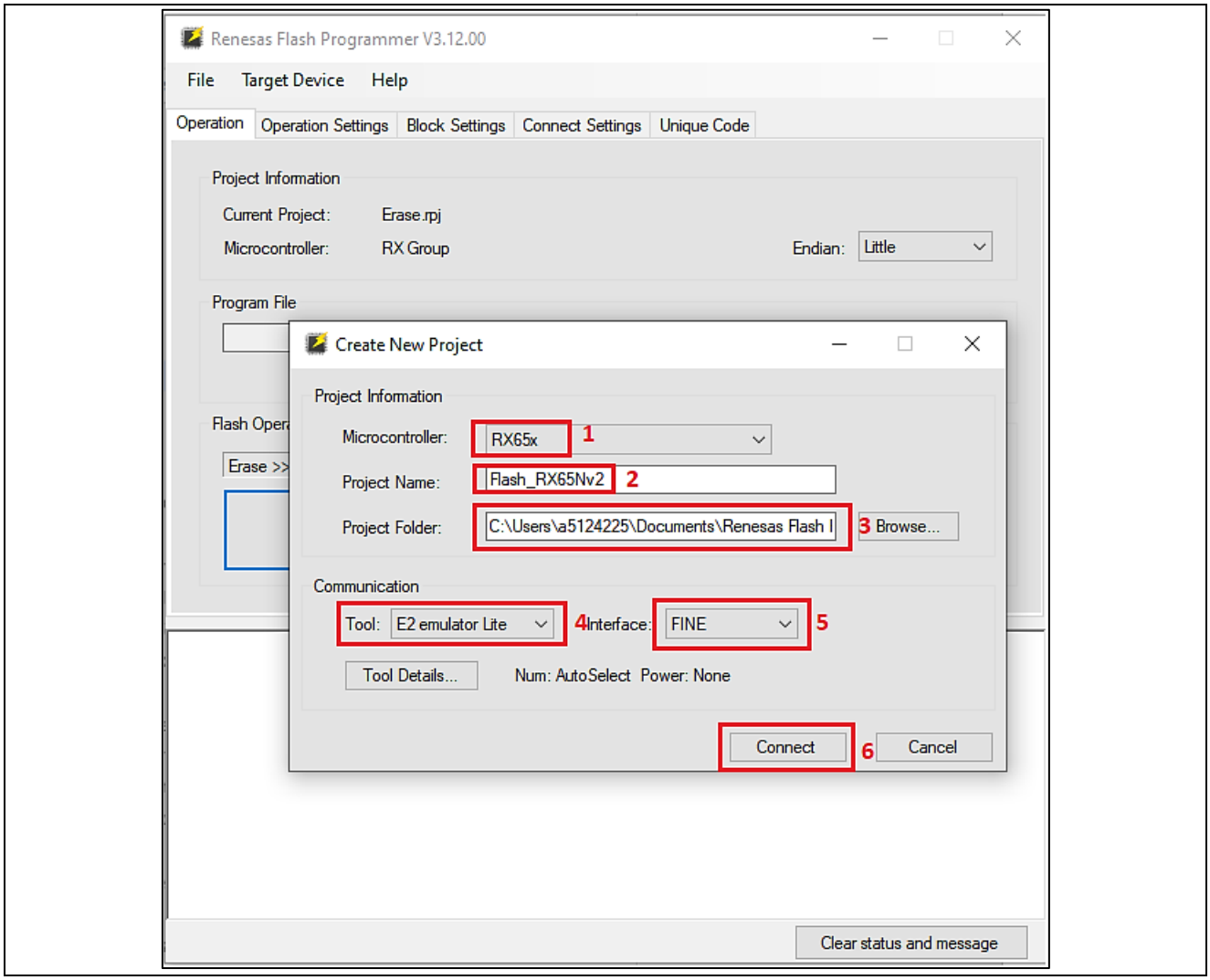

Figure 116. Create New Flash Project (2/4)

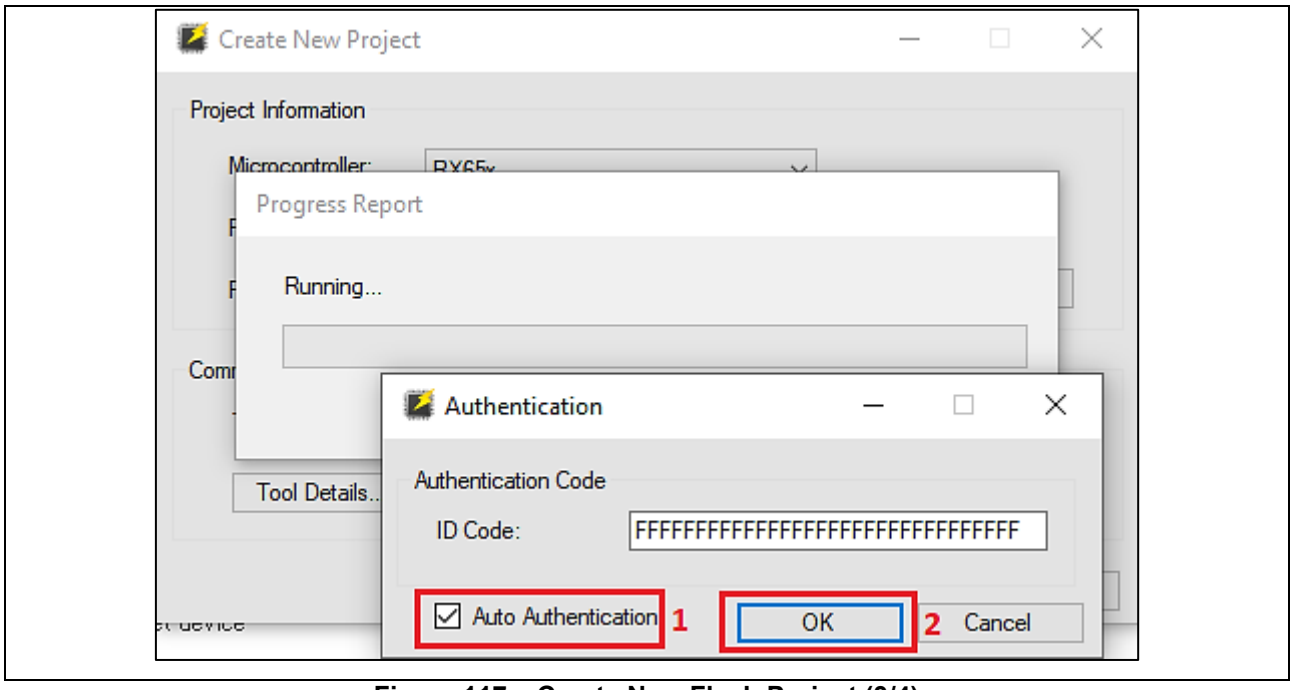

Figure 117. Create New Flash Project (3/4)

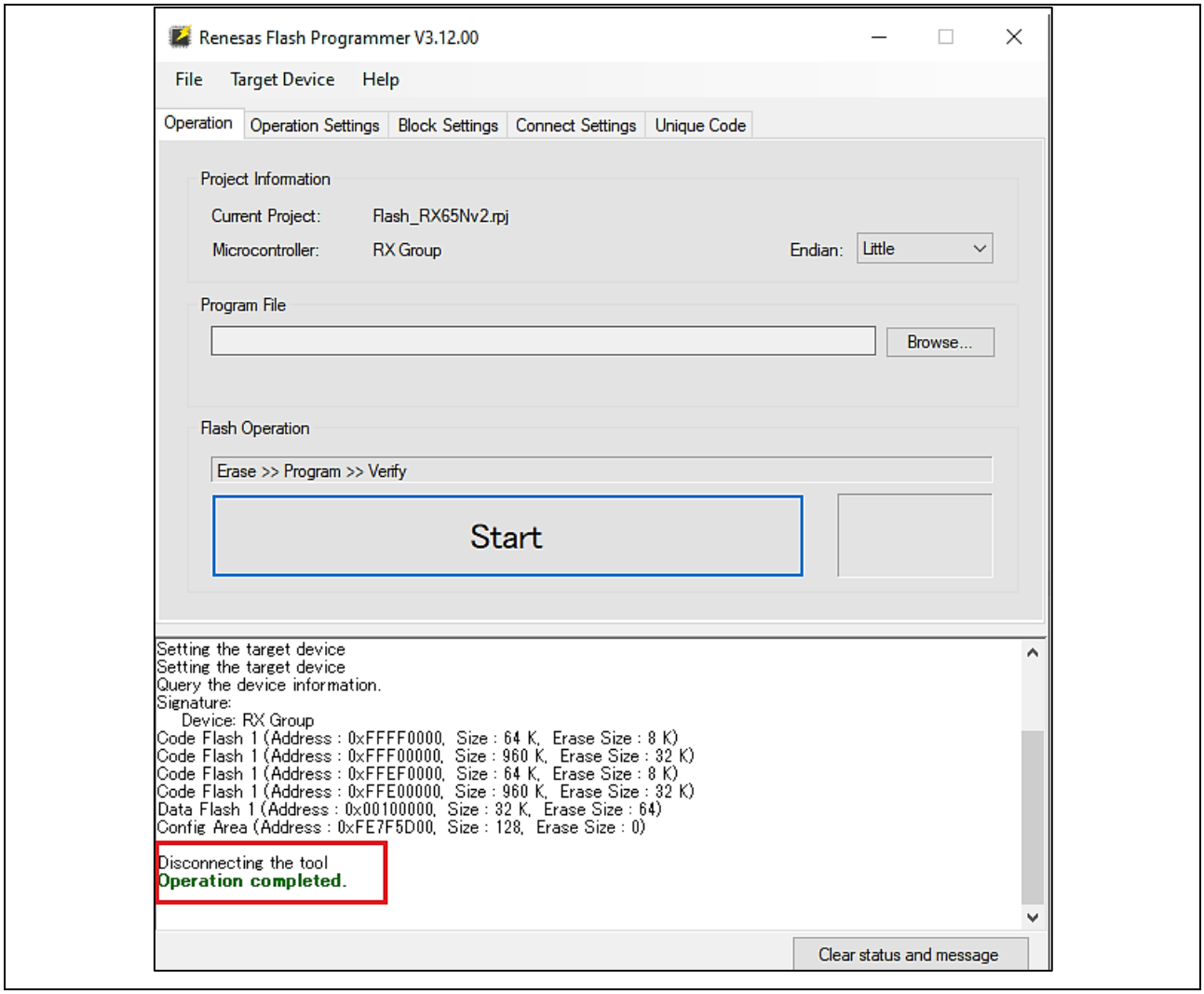

Figure 118. Create New Flash Project (4/4)

2. Choose code flash area in configuration:

   Only tick option **"Select"** for **"Code Flash 1".**

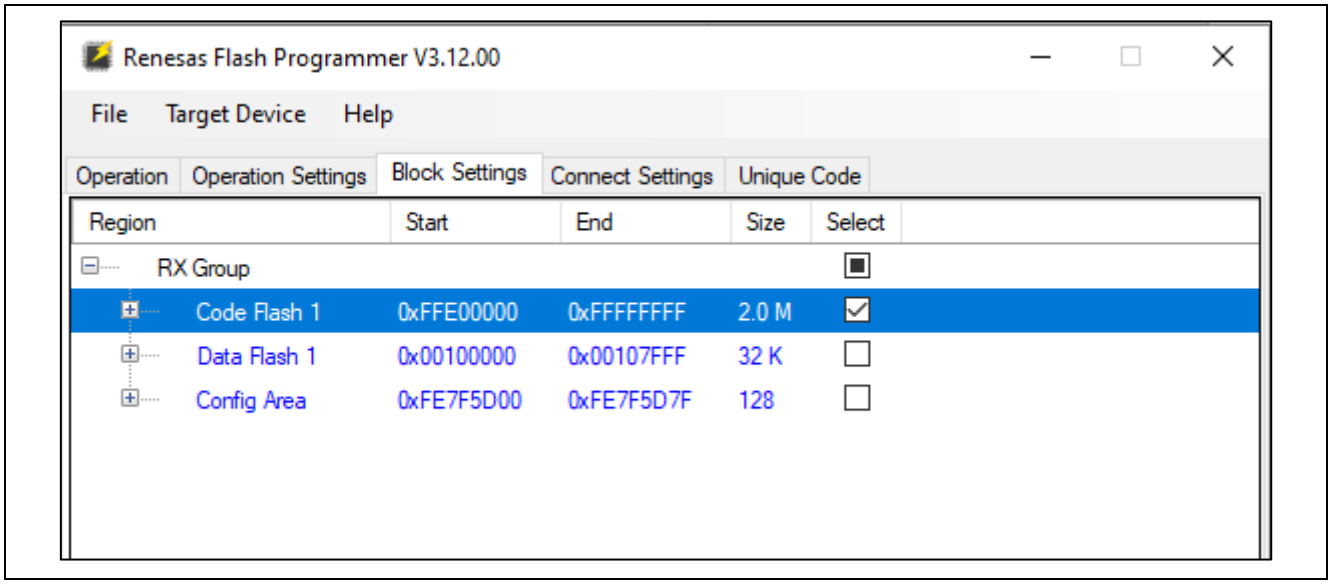

Figure 119. Choose Code Flash Area

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

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

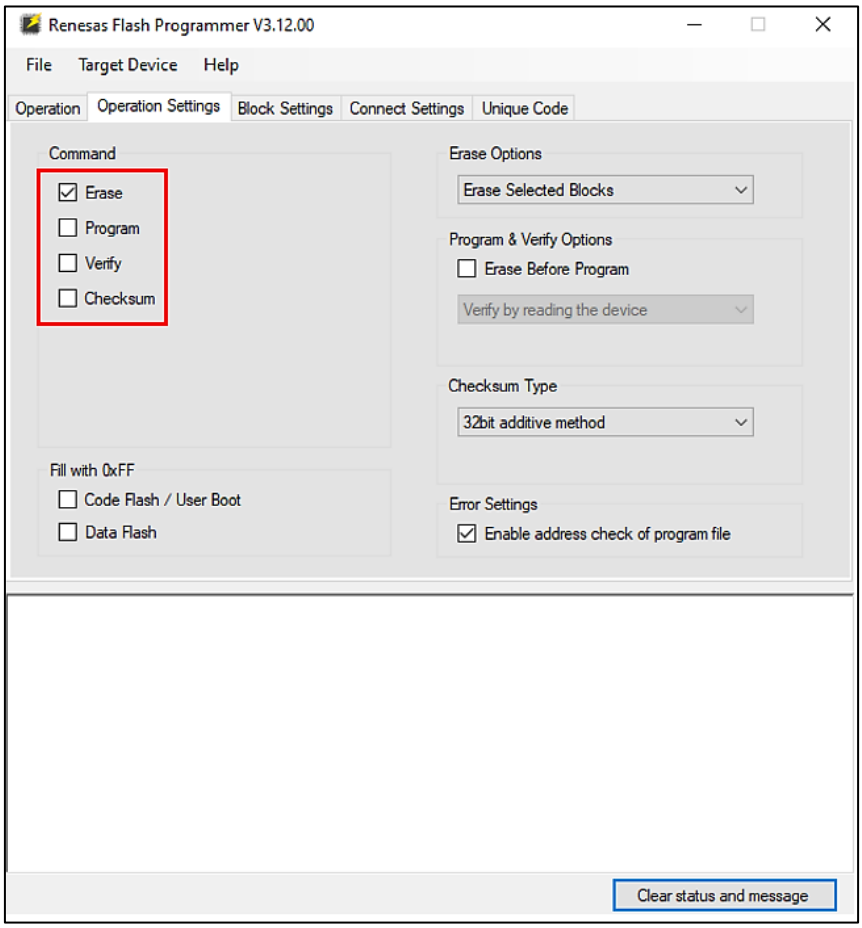

Figure 120. Erase Code Flash (1/2)

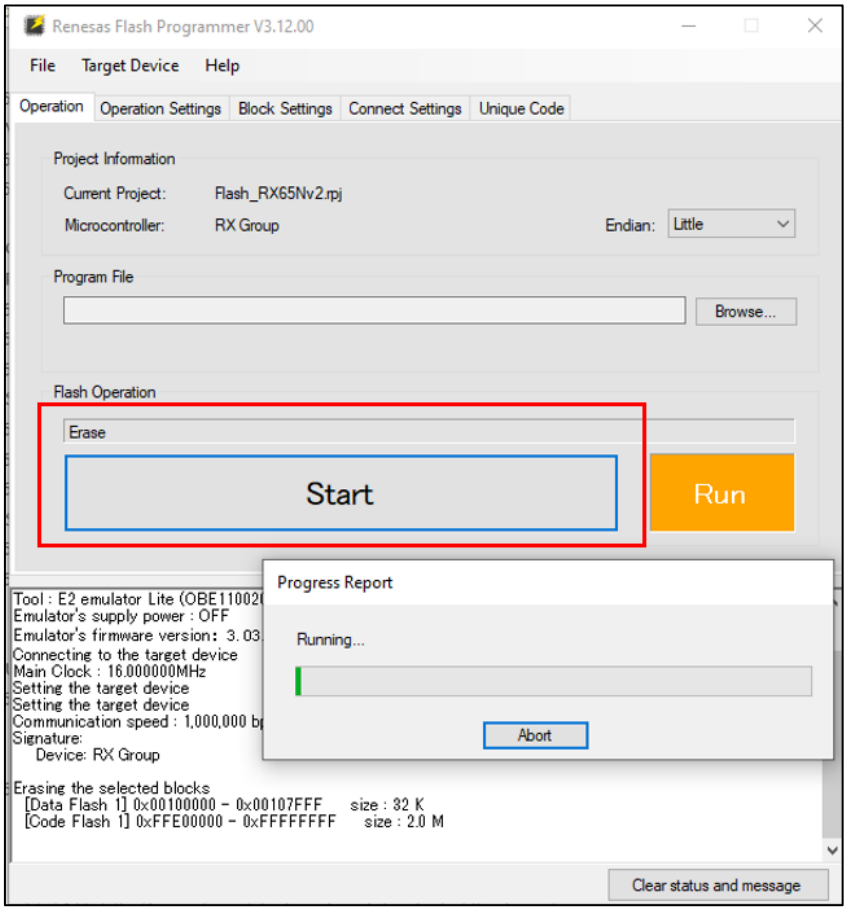

Figure 121. Erase Code Flash (2/2)

4. Write the initial firmware (userprog.mot)

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

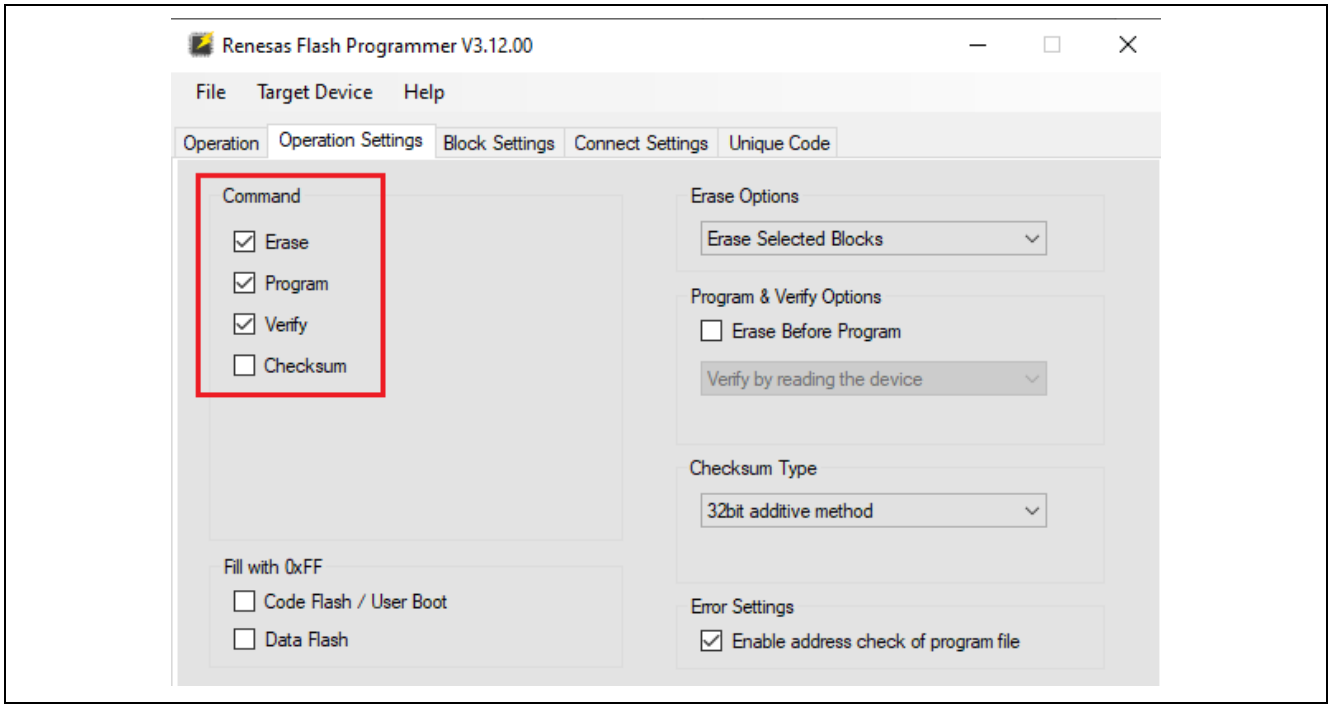

Figure 122. Flash the Firmware (1/3)

Add firmware's path (users have created in 6.4.2.1) to "Program File" and click "Start":

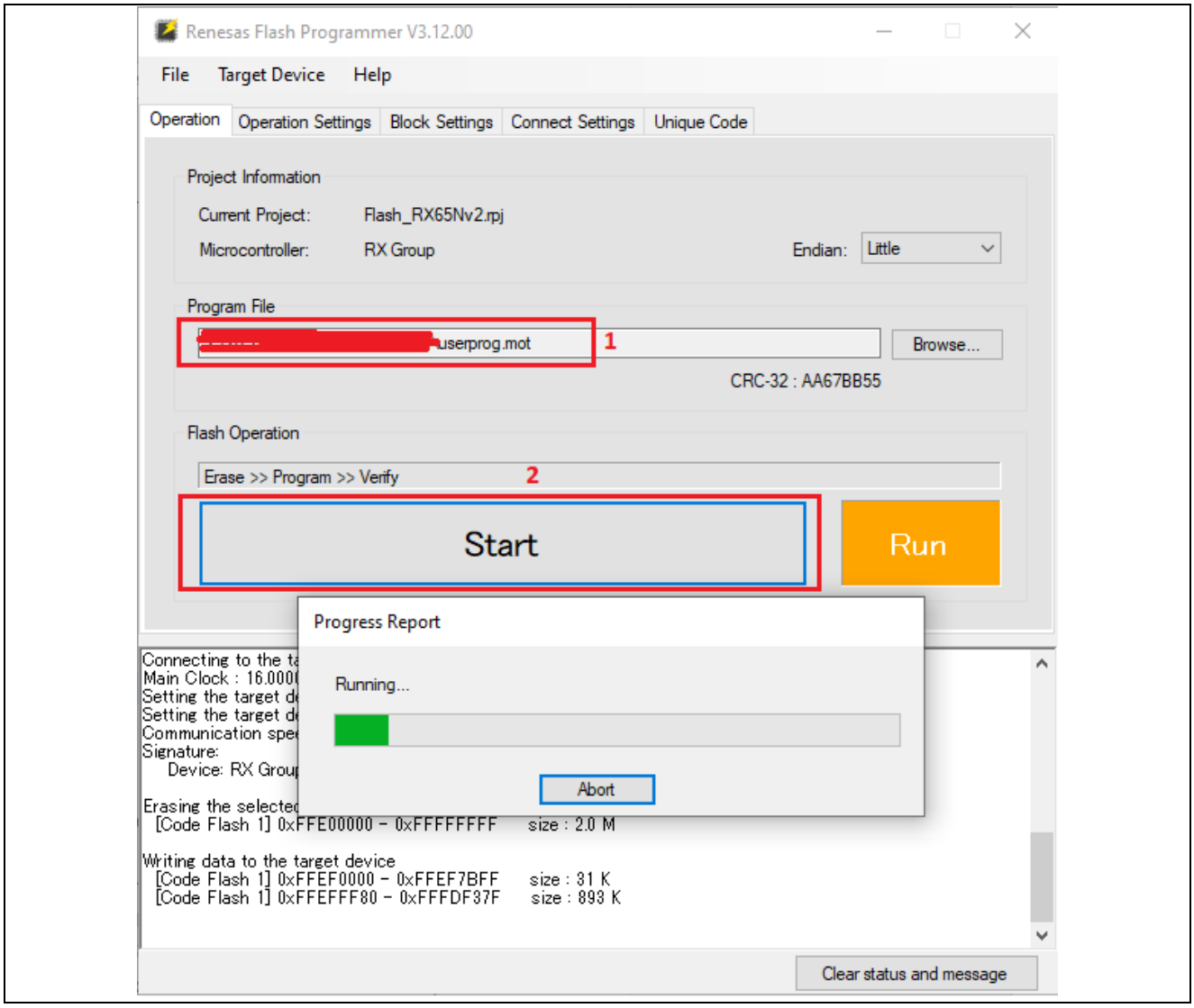

Figure 123. Flash the Firmware (2/3)

If it is successful, it will display “Operation completed”:

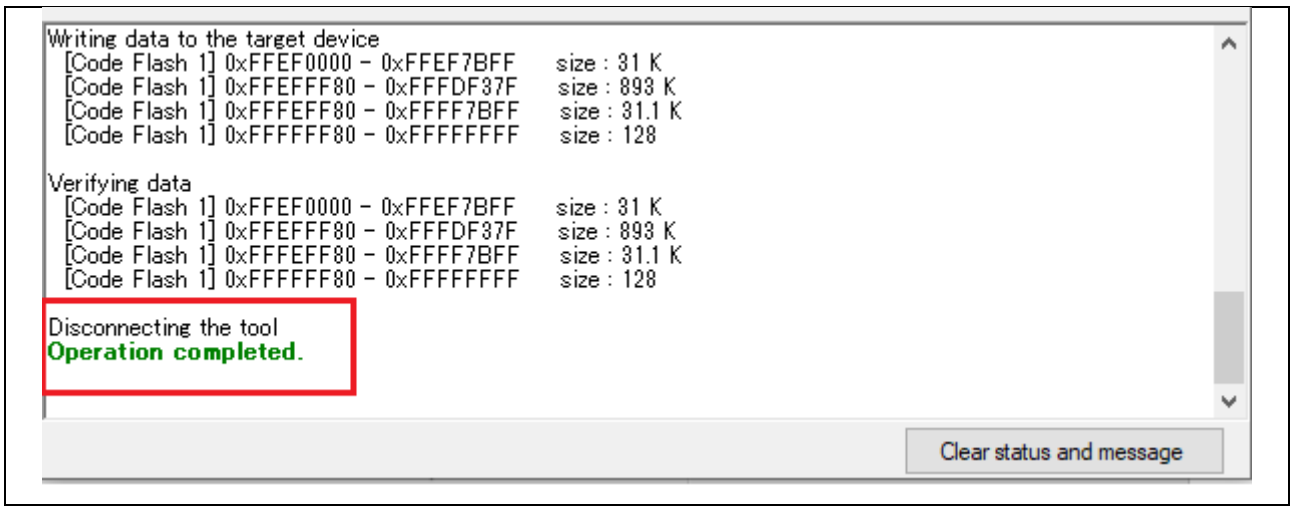

Figure 124. Flash the Firmware (3/3)

5. Running the application:

   Set jumper of J16 is “Run” mode. The application will run as below (like application in section **5.3.3 Starting the Application,** but have activity’s log of boot loader):

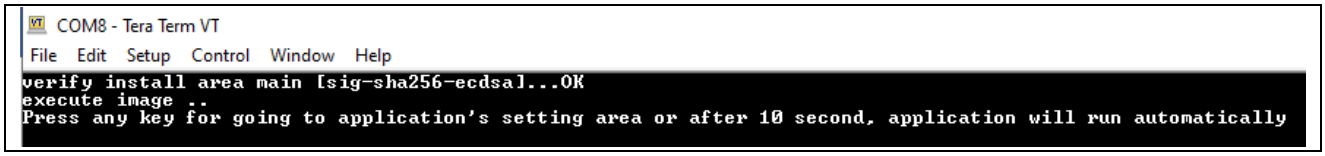

Figure 125. Start Application with OTA

Press any key to configure the application. For setting Cloud’s credentials for IoT Device, please refer to the **5.3.3 Starting the Application.**

Besides that, for OTA, please store the code signing certificate (user created **“secp256r1.crt”** file at step: ***Create a certificate from the certificate signing request, CA certificate, and CA private key***).

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

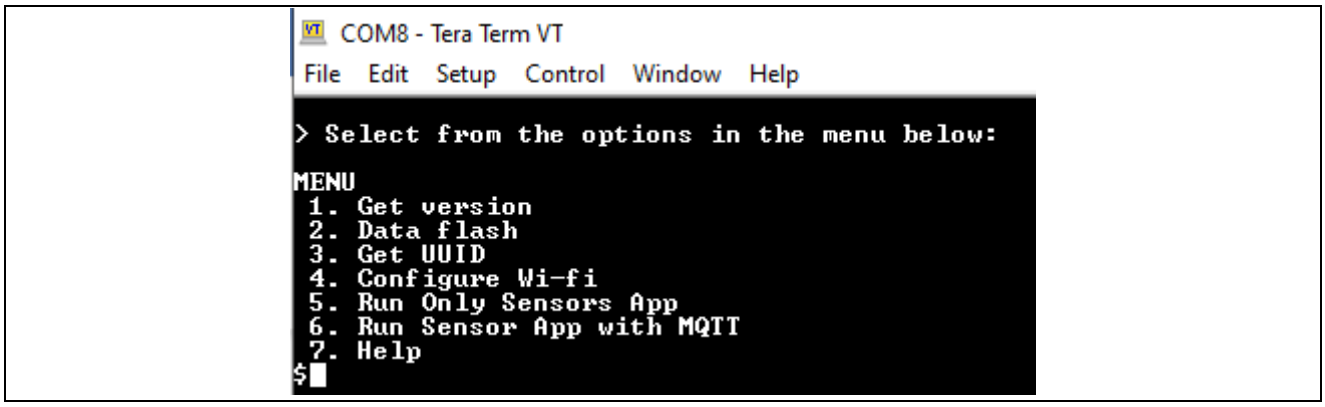

Figure 126. Main Menu

Press ‘f’ for storing code signing certificate:

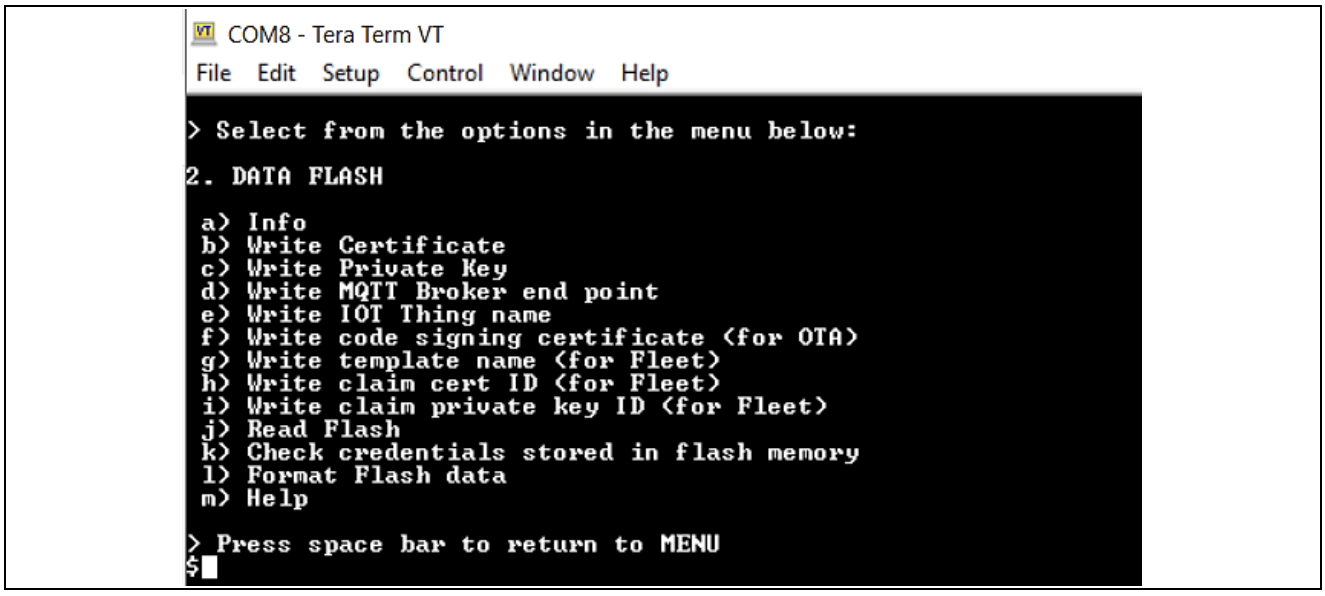

Figure 127. Data Flash Menu

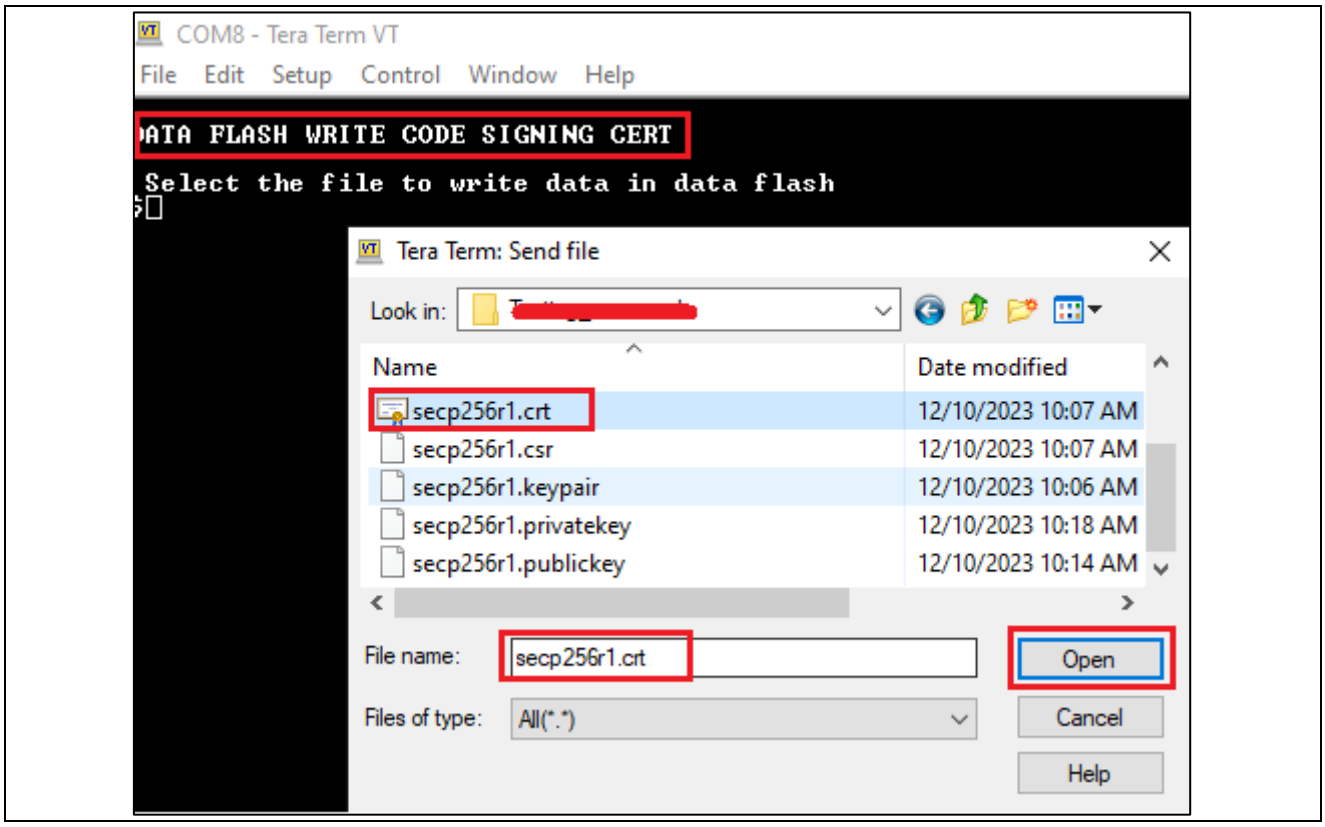

Figure 128. Store code signing certificate into flash (1/2)

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

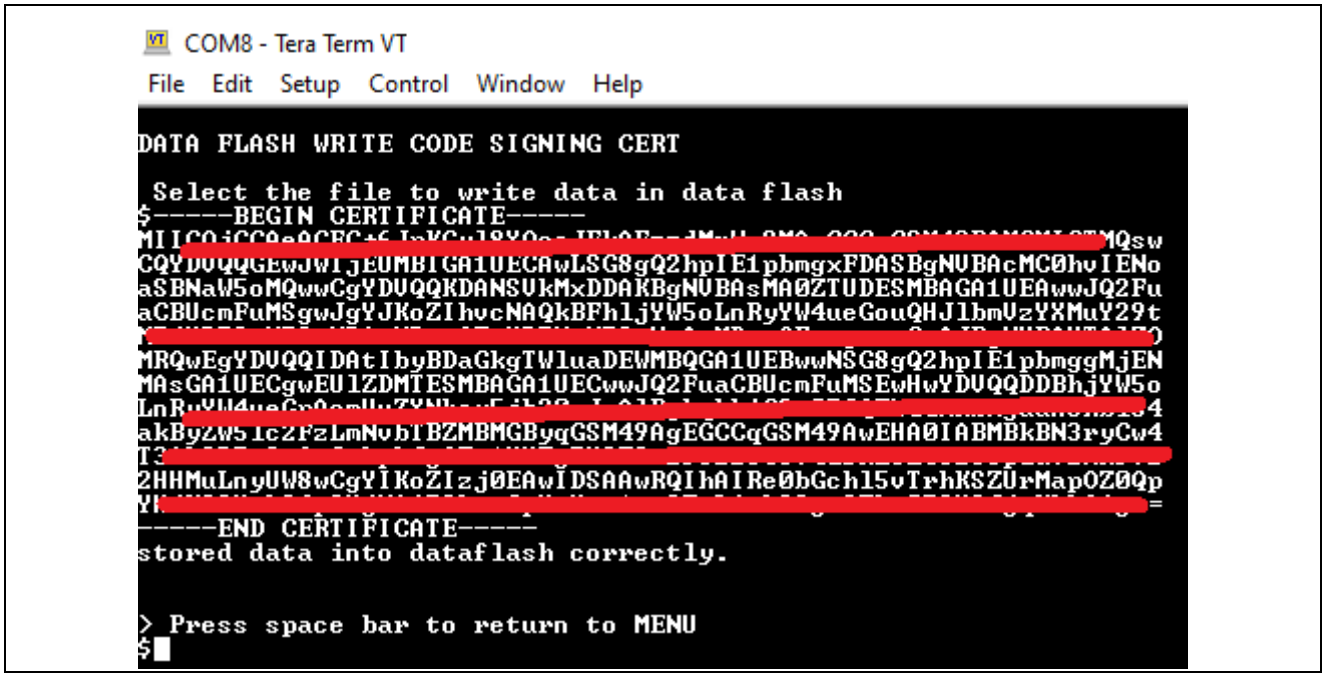

Figure 129. Store code signing certificate into flash (2/2)

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

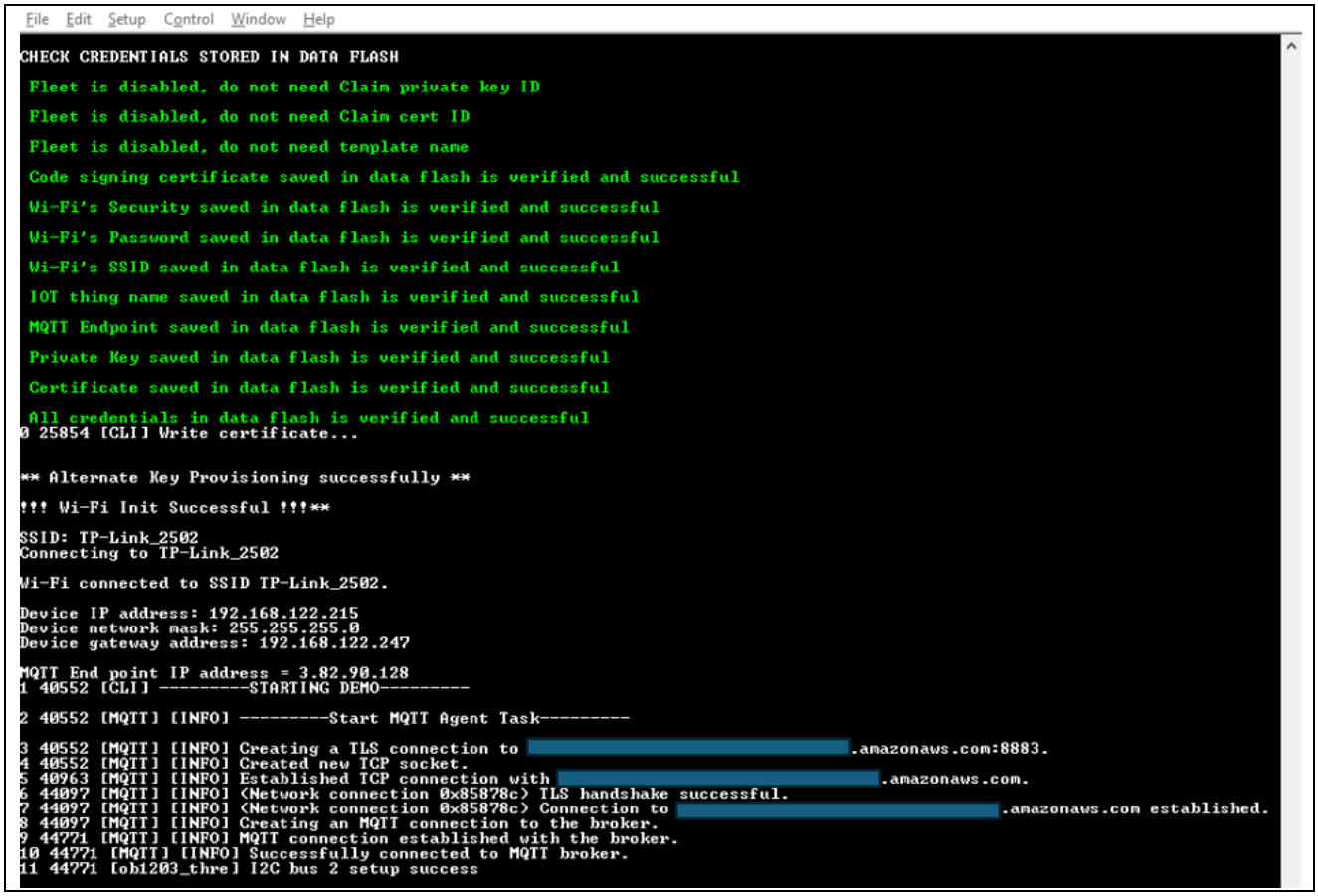

Figure 130. Application with OTA (1/2)

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

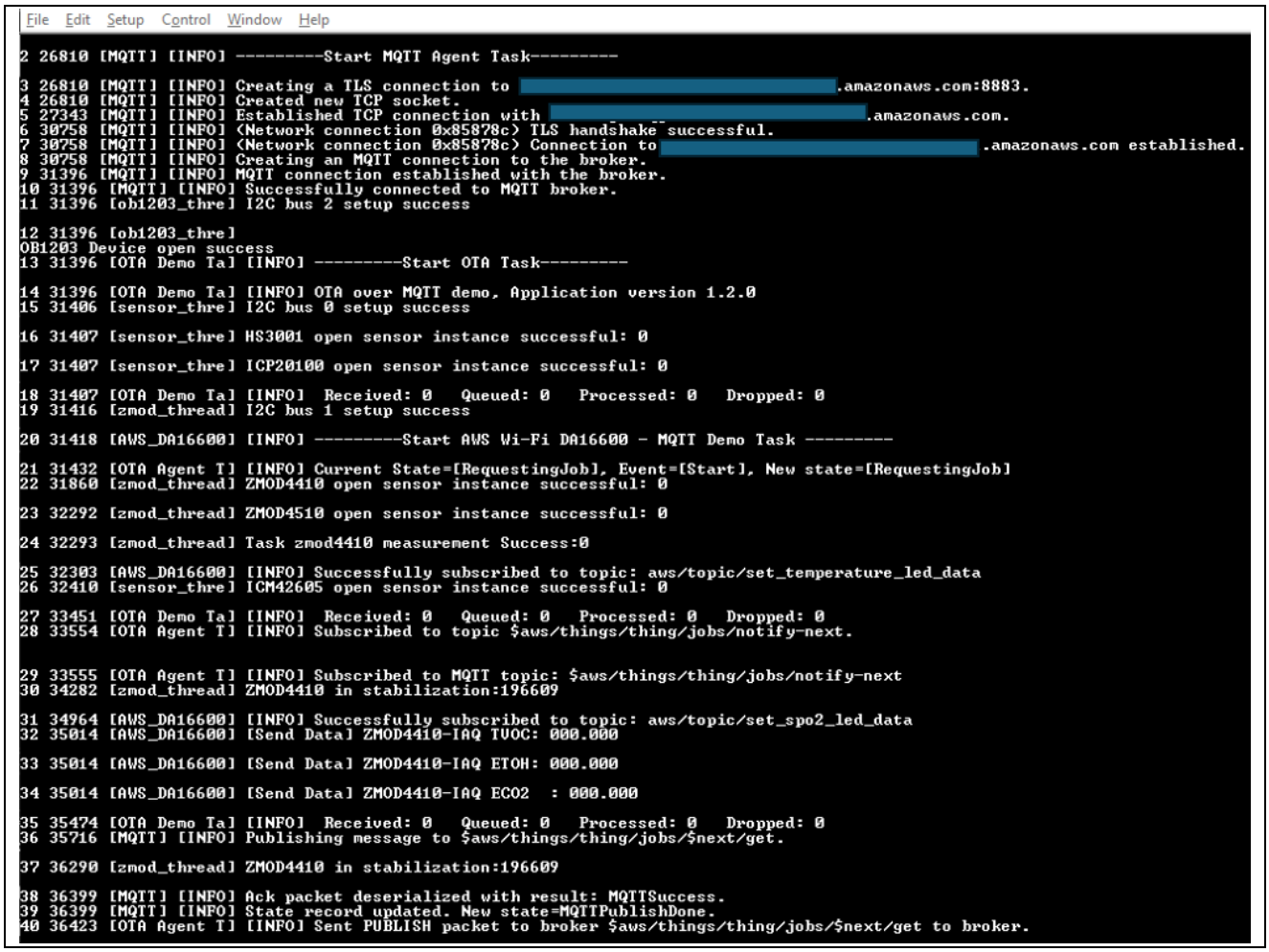

Figure 131. Application with OTA (2/2)

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 1.2.0, so the new version we can choose 1.2.1)

Repeat the build process, this time with 3 specified for the APP_VERSION_BUILD definition in `aws_da16600_ck_rx65n\e2studio_ccrx\src\frtos_config\demo_config.h`.

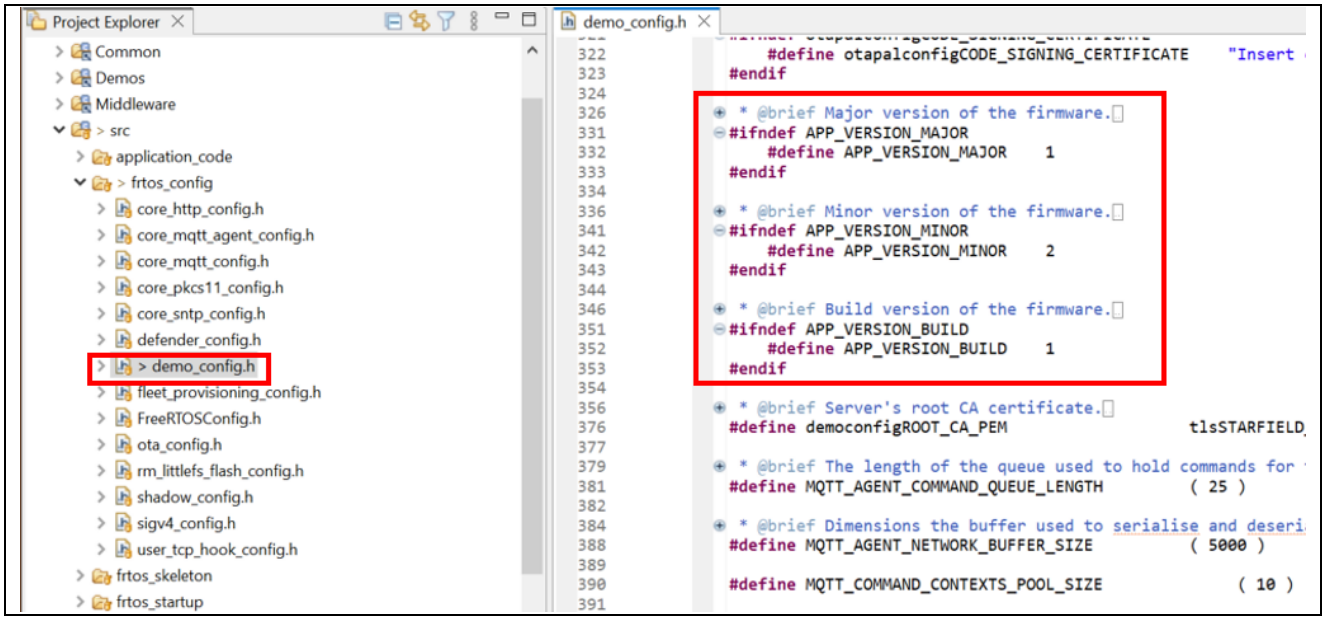

Figure 132. Setting New Version for Firmware

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_121 -key secp256r1.privatekey -vt
ecdsa -ff RTOS
```

This command generates a file named user_121.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** > and **Jobs**, and then click the **Create job** button.

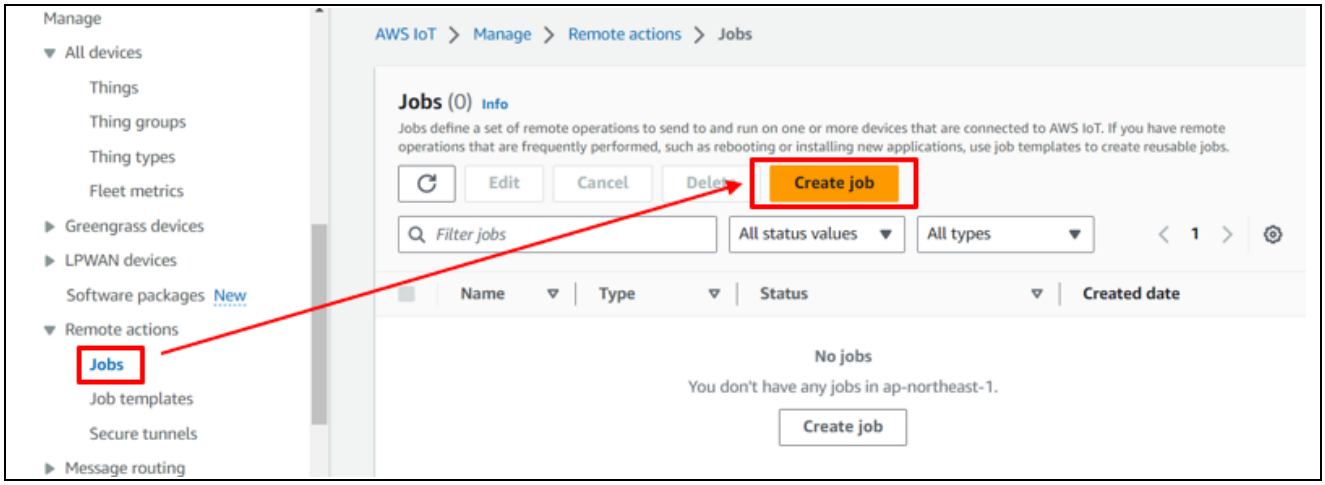

Figure 133. Create New Job for OTA (1/2)

6.5.2.2 Creating FreeRTOS OTA Job Update

Select Create FreeRTOS OTA update job and then click Next.

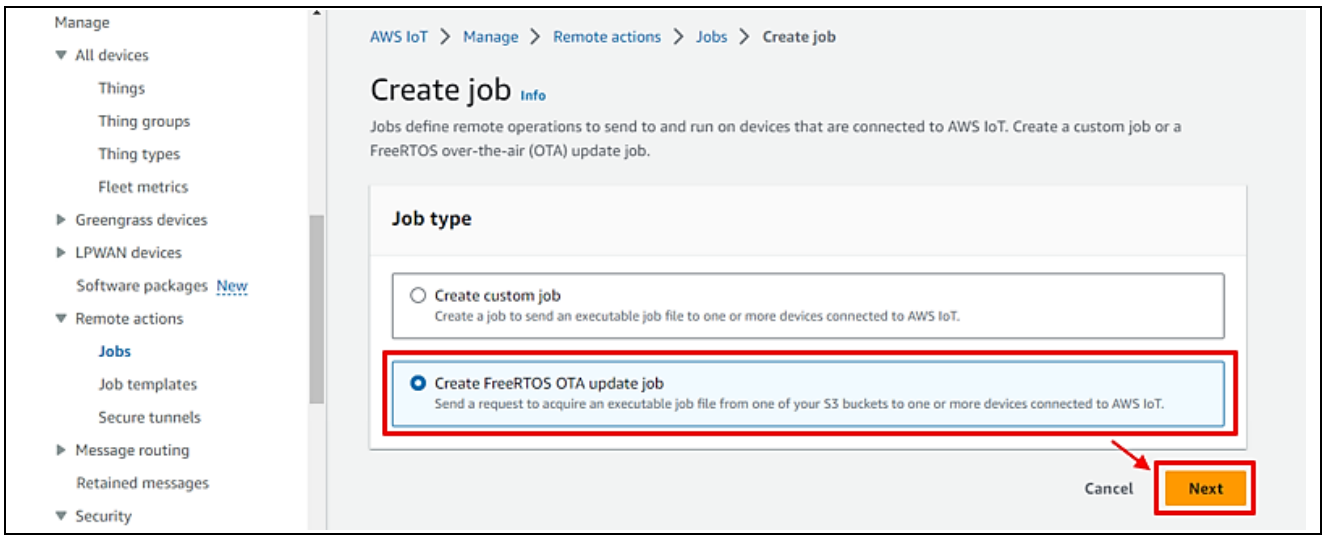

Figure 134. Create New Job for OTA (2/2)

6.5.2.3 Entering a Job Name

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

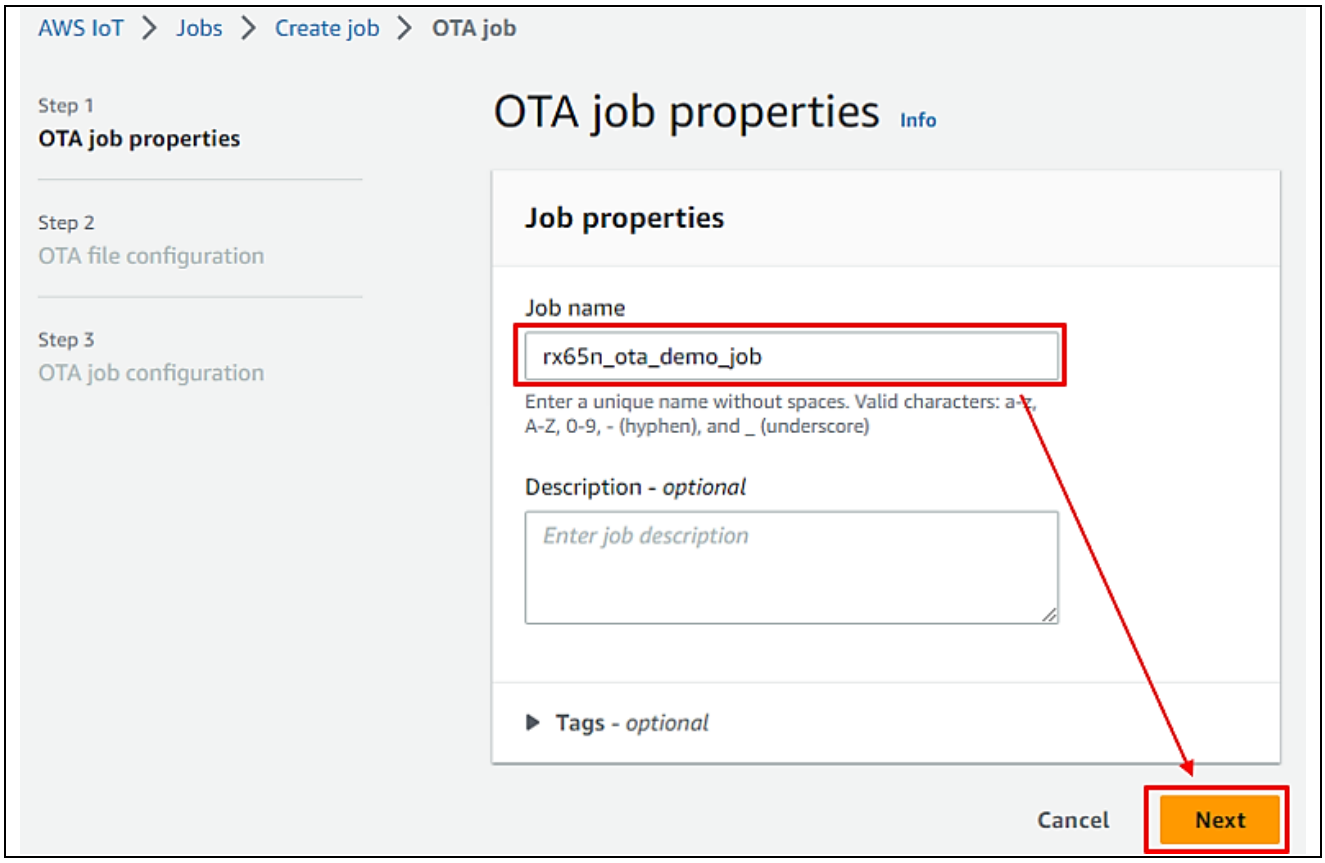

Figure 135. Enter OTA Job Name

6.5.2.4 Updating Devices

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

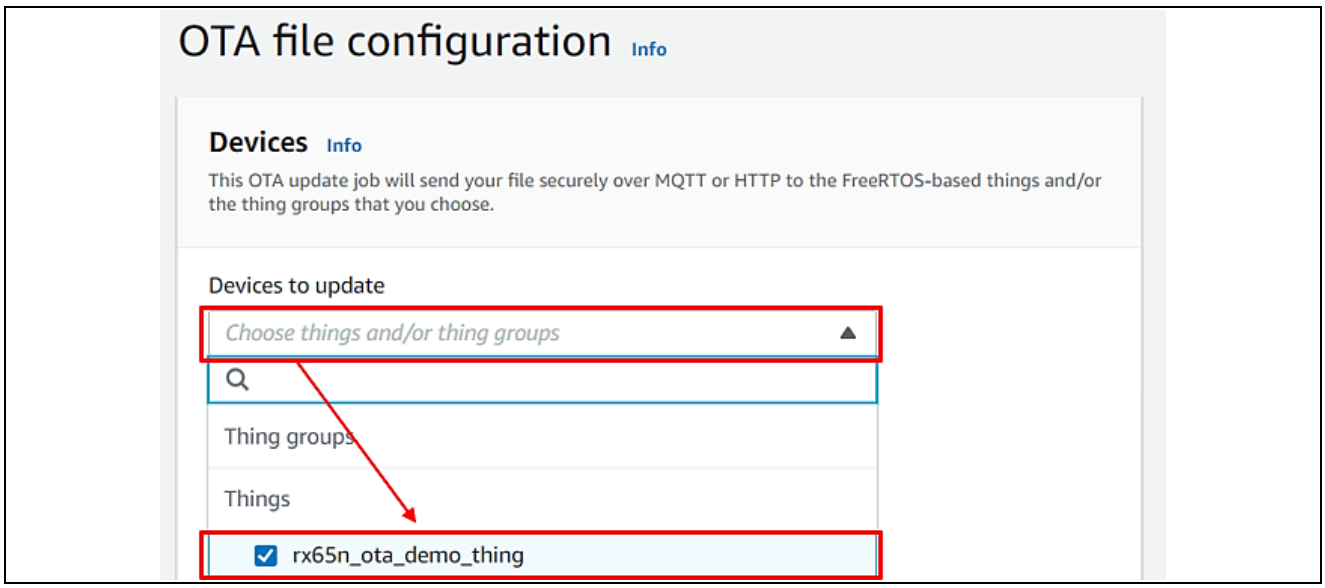

Figure 136. Choose Device to Update

6.5.2.5 Creating New Profile

Click Create new profile.

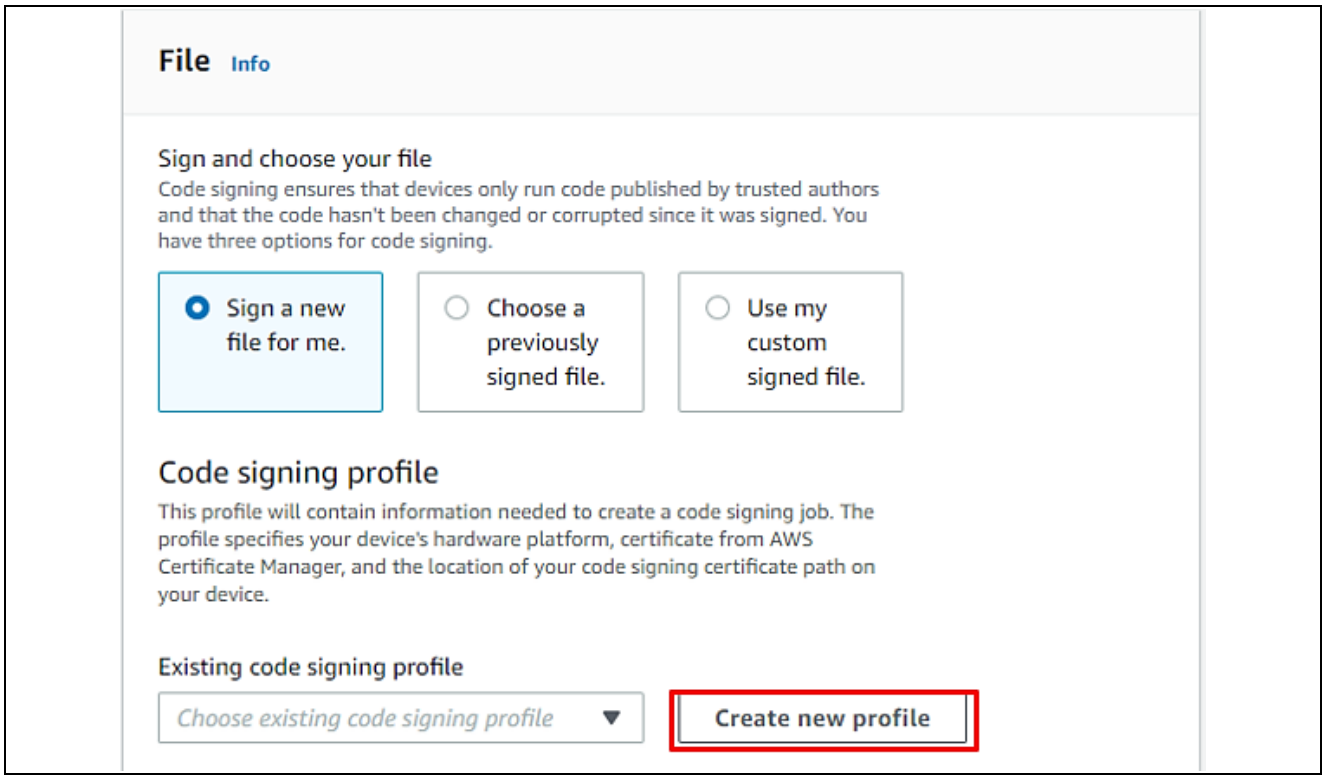

Figure 137. Create New Code Signing Profile

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.

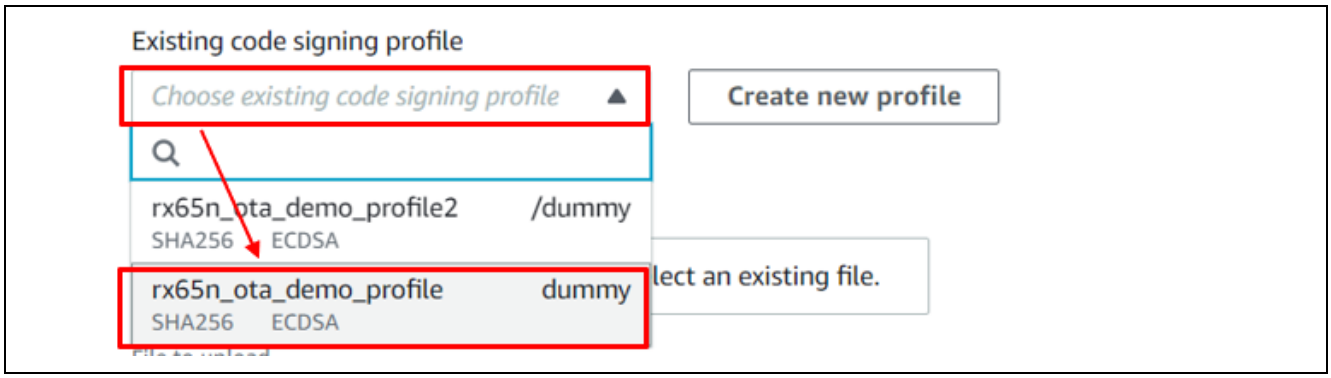

Figure 138. Choose Existing Code Signing Profile

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!

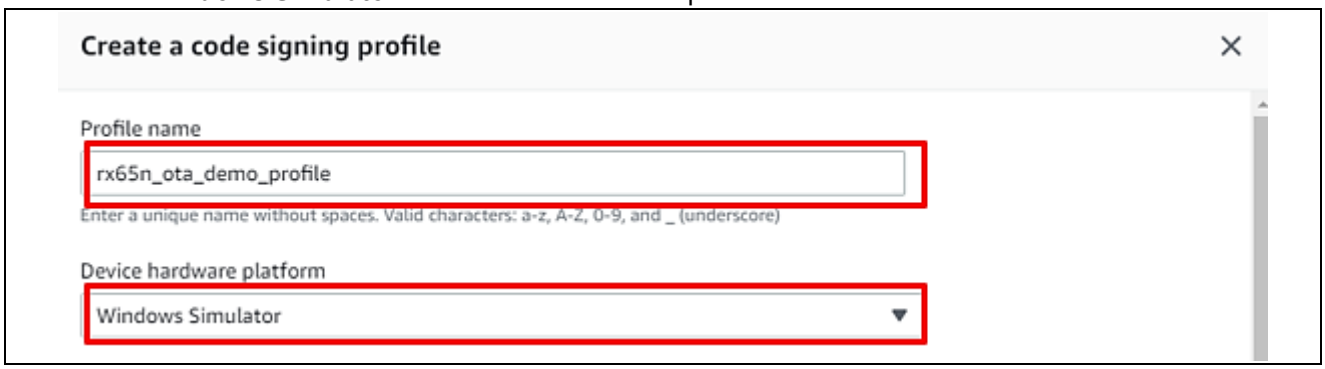

Figure 139. Create New Code Signing (1/2)

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**

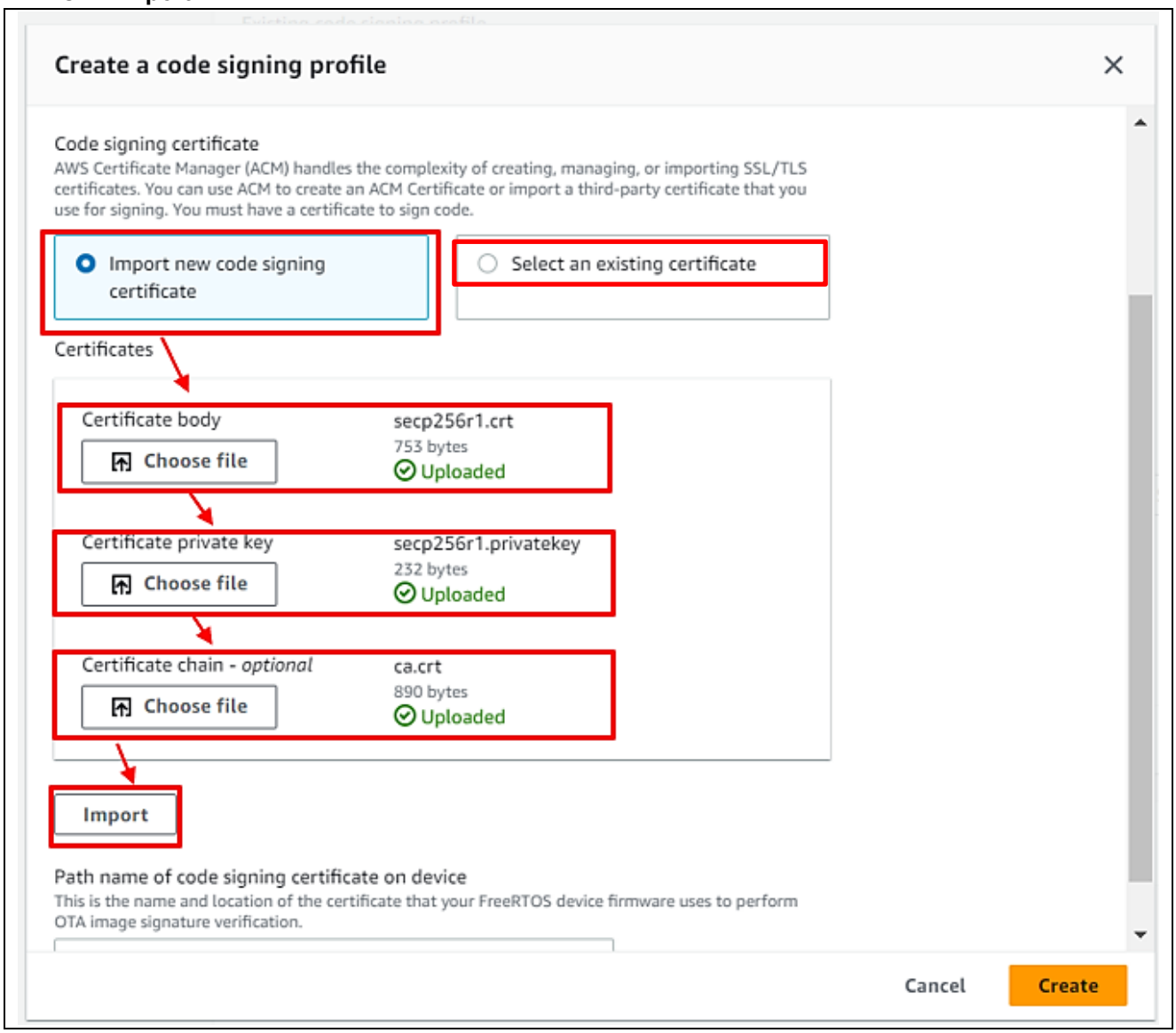

Figure 140. Create New Code Signing (2/2)

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)

Figure 141. 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.

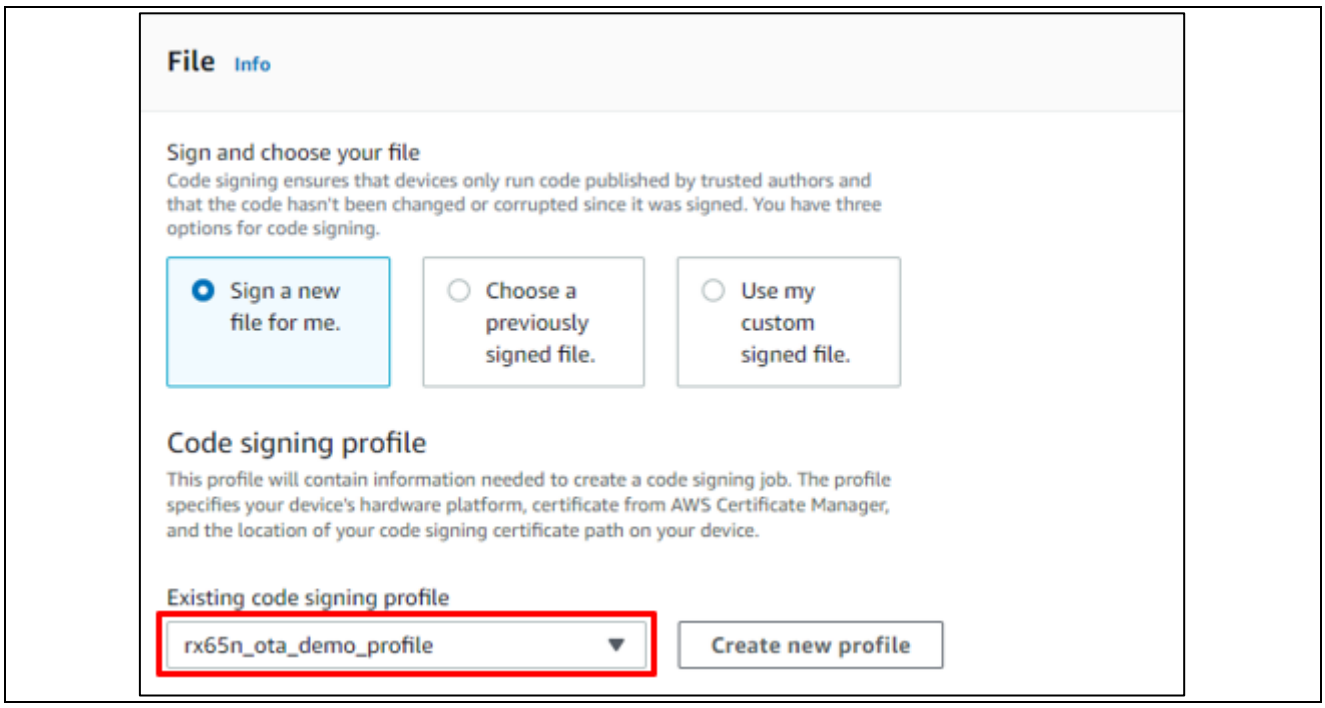

Figure 142. Choose Existing Code Signing Profile

6.5.2.10 Updating the Firmware

- Select **Upload a new file.**
- In **File to upload**, select the file user_121.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).

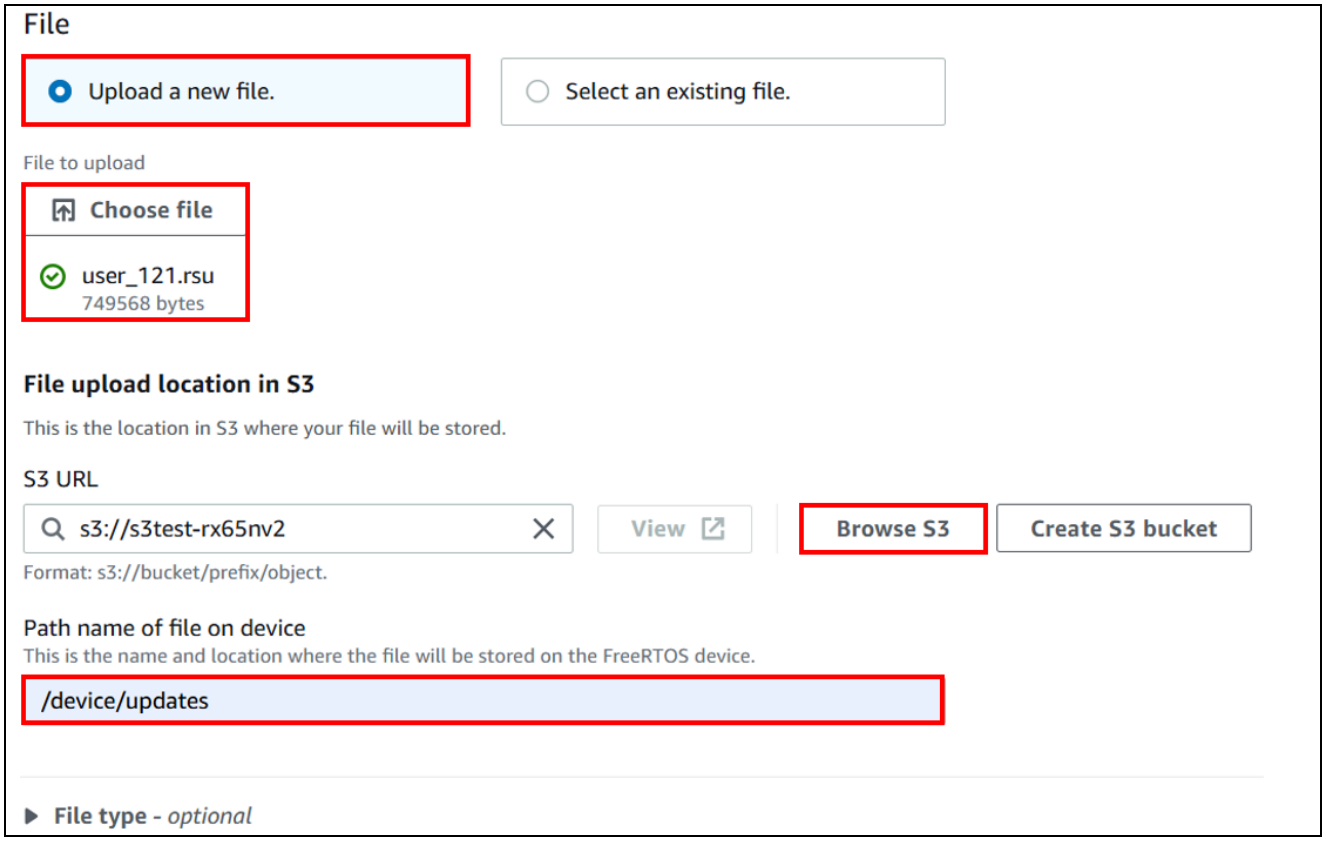

Figure 143. Choose Firmware to Update

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**.

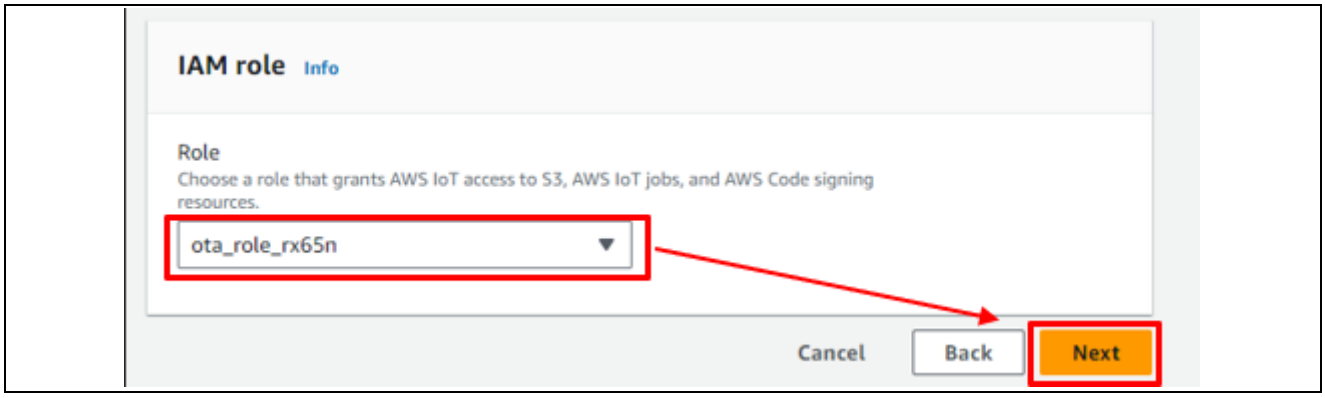

Figure 144. Choose Role for Creating Job

Click **Create job**.

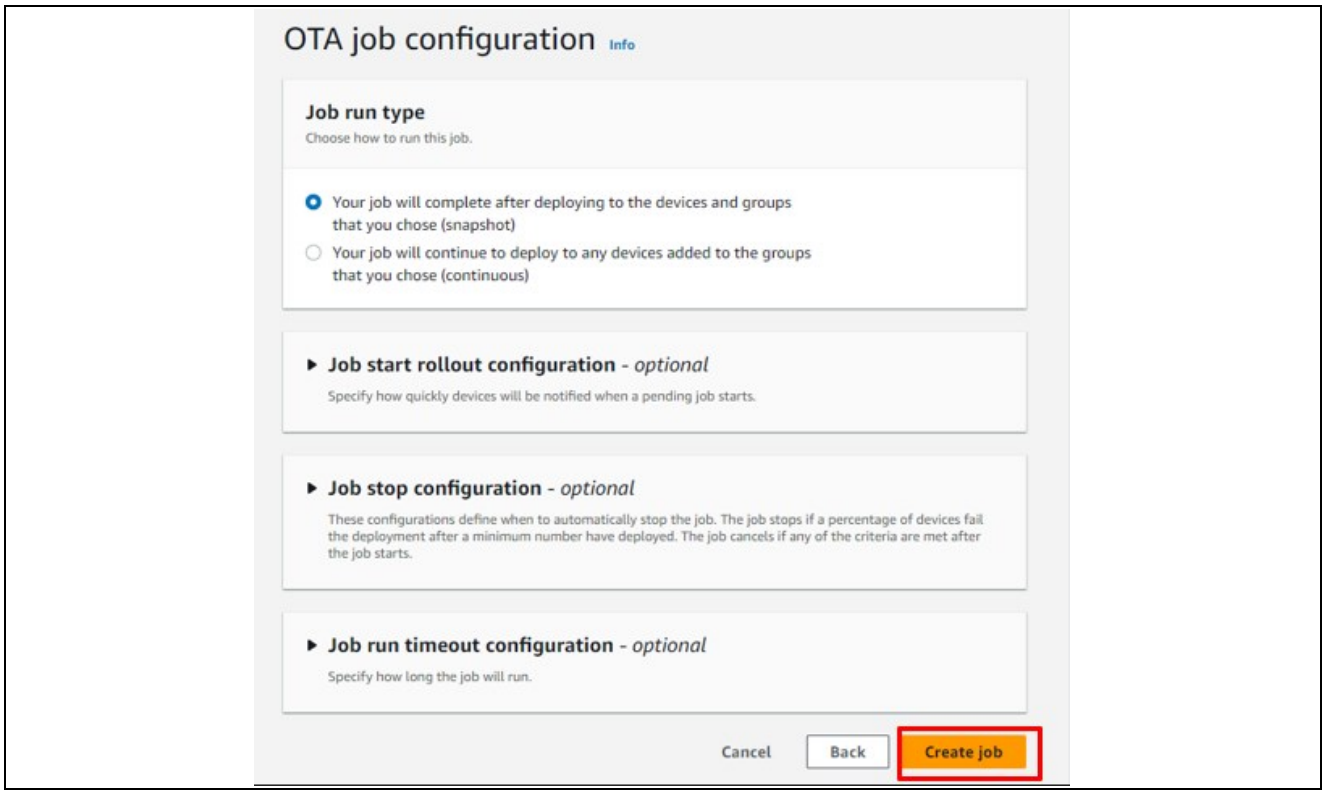

Figure 145. Create Job

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 reception starts.

Figure 146. OTA Job is Processed

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

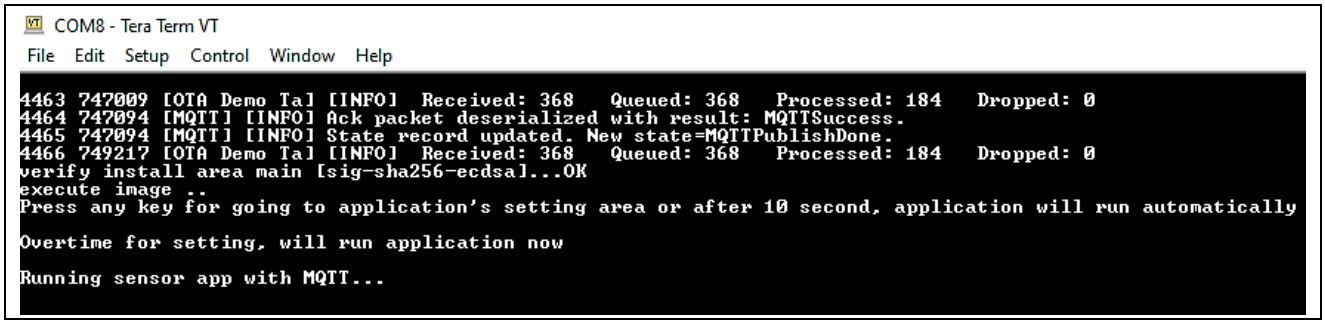

Figure 147. Device Reset and Update New Firmware

6.5.2.13 Confirming that the Firmware Version is a New Version

Example: 1.2.1 (updated at **6.5.1.1**)

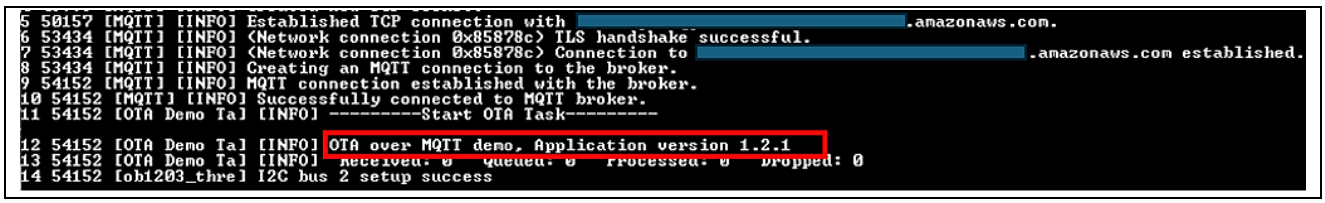

Figure 148. Firmware with New Version is Updated

When OTA updated firmware successfully, status will be “Succeeded” like below:

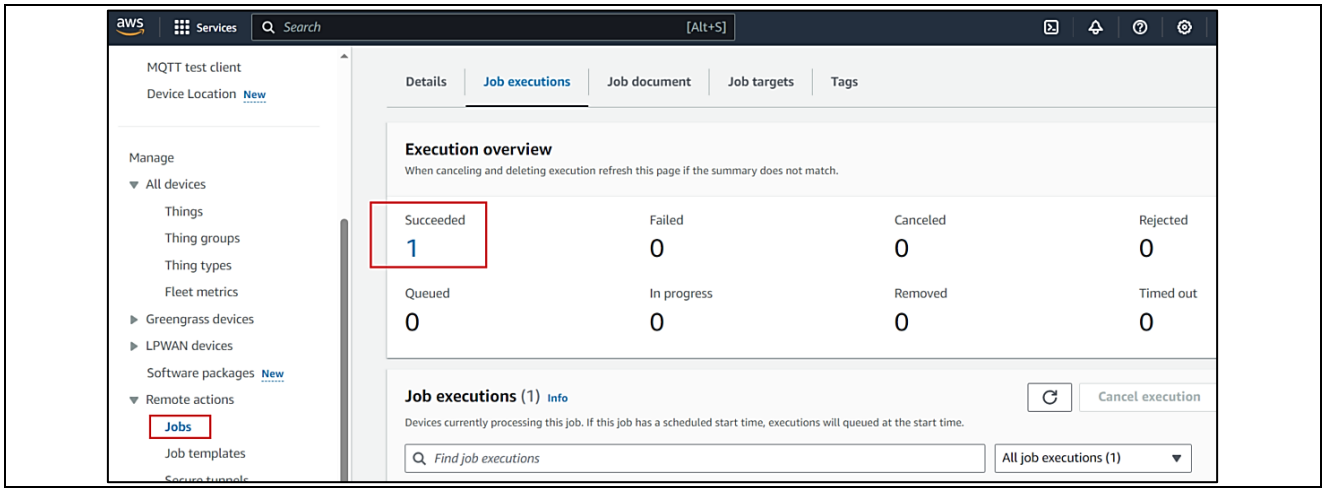

Figure 149. Job’s Status in AWS Portal

7. Fleet Provisioning

7.1 Overview Fleet Provisioning

This section describes the steps on 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 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 chapter 3, 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 in order 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.

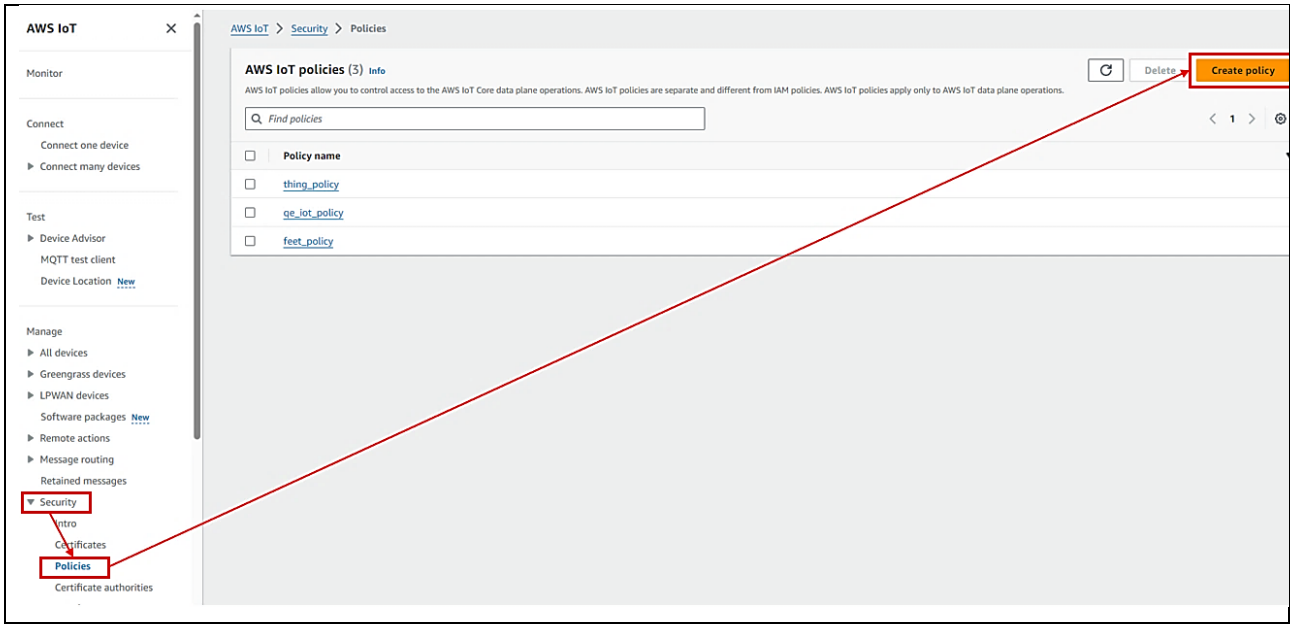

Figure 150. Creating an AWS IoT Policy (1/2)

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

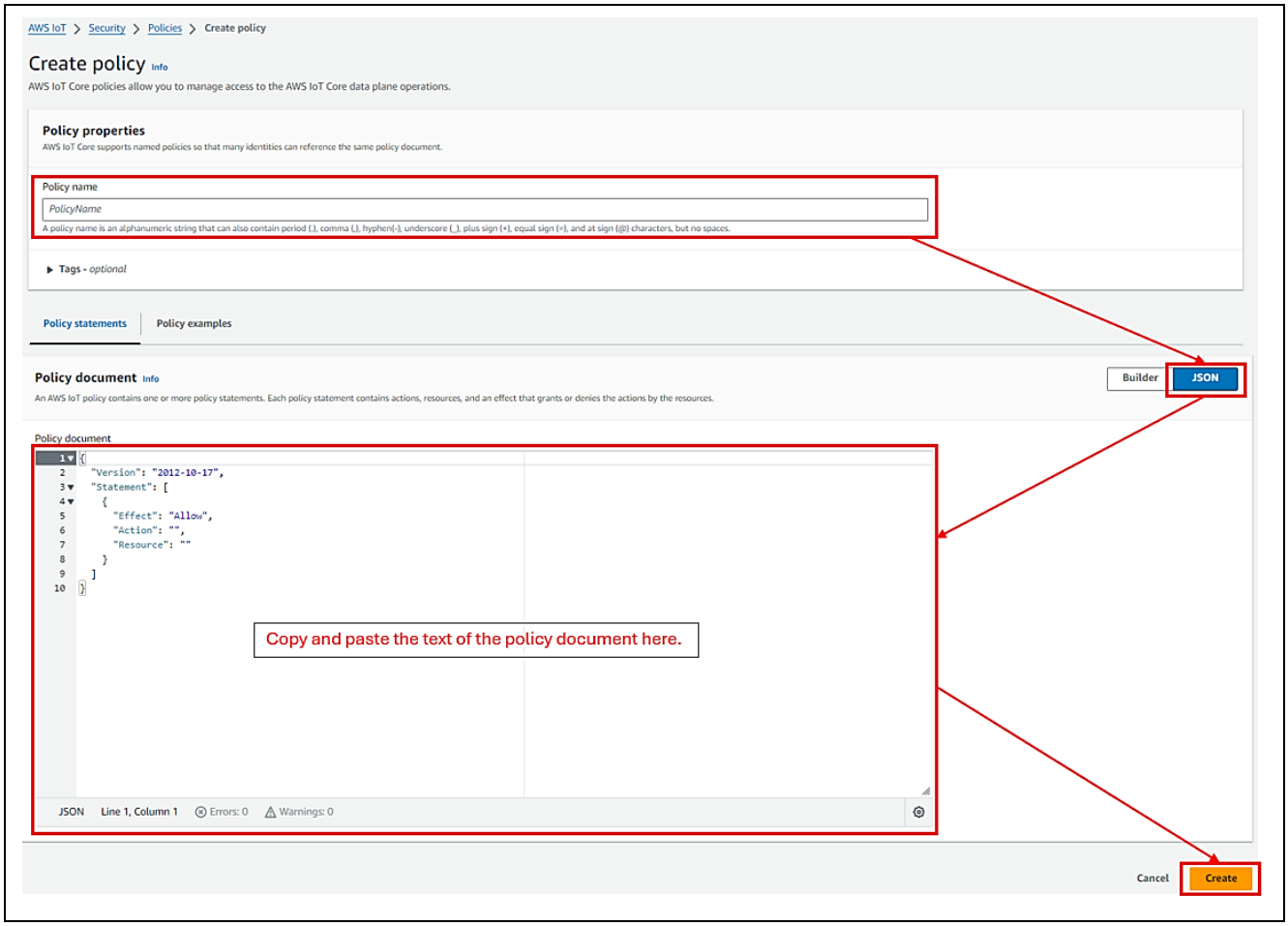

Figure 151 Creating an AWS IoT Policy (2/2)

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 own 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**.

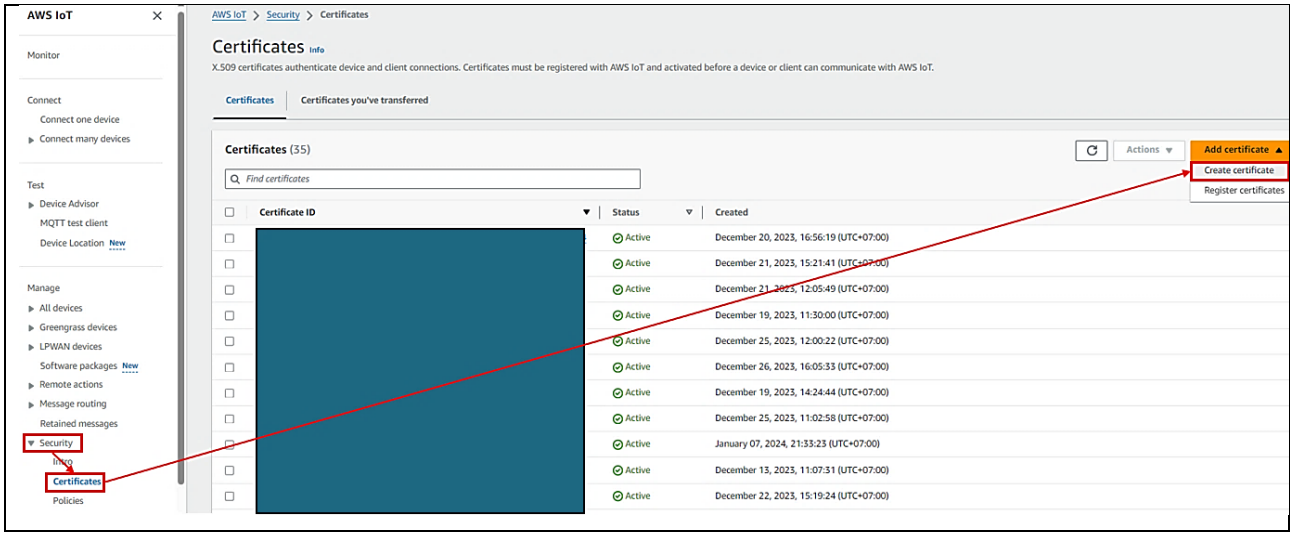

Figure 152. Create a Certificate

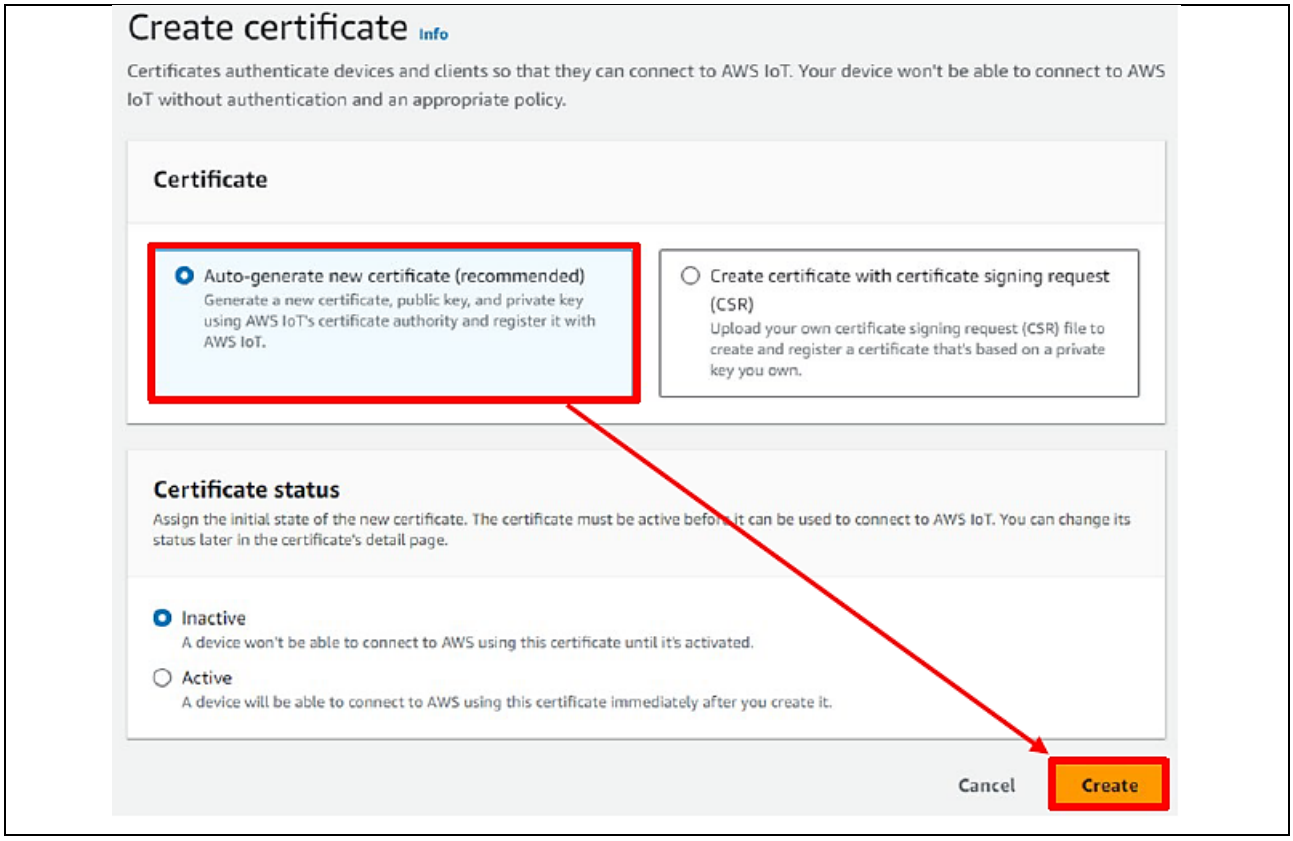

Figure 153. Creating a Certificate Automatically

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

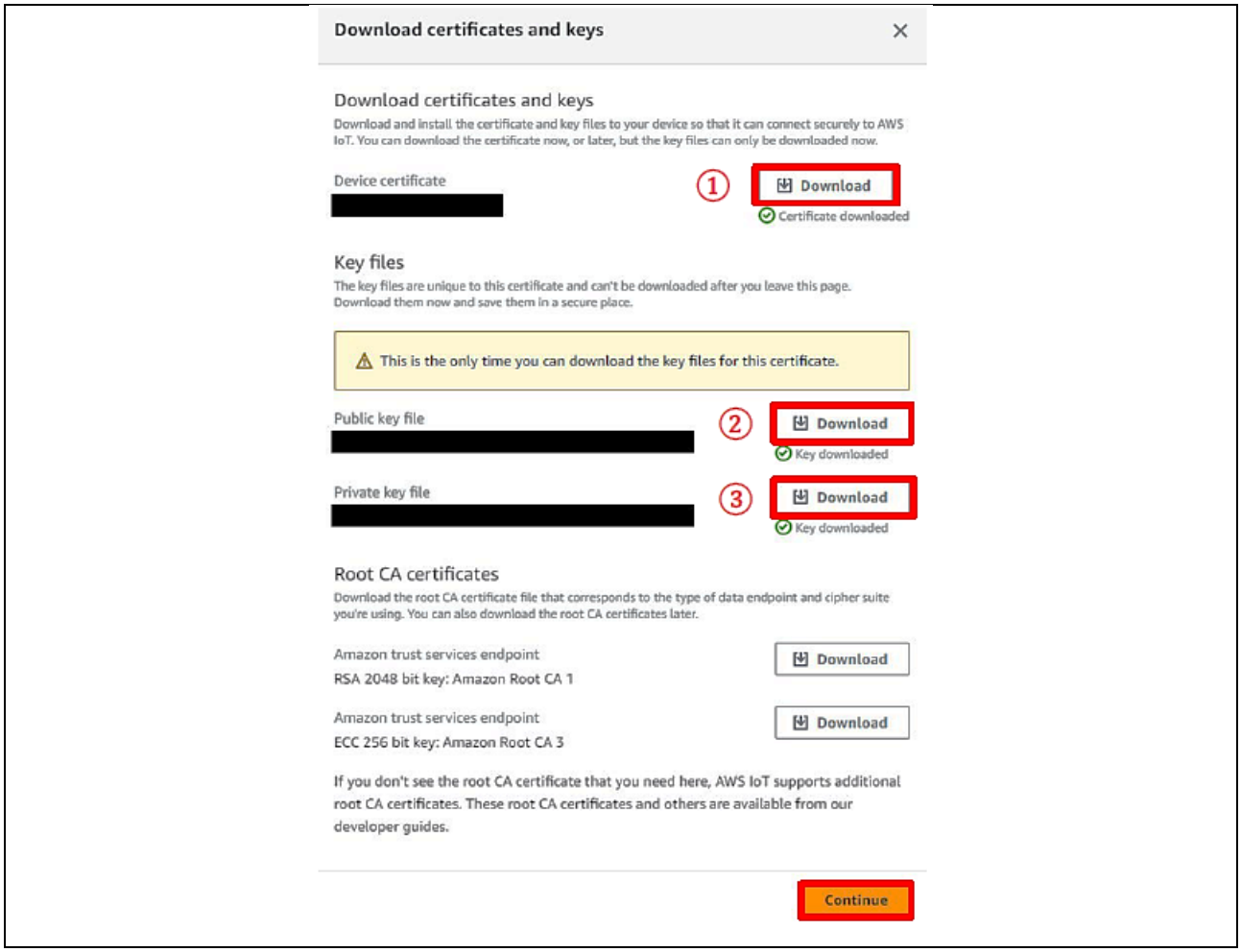

Figure 154. Downloading the Certificate and Key Pair

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

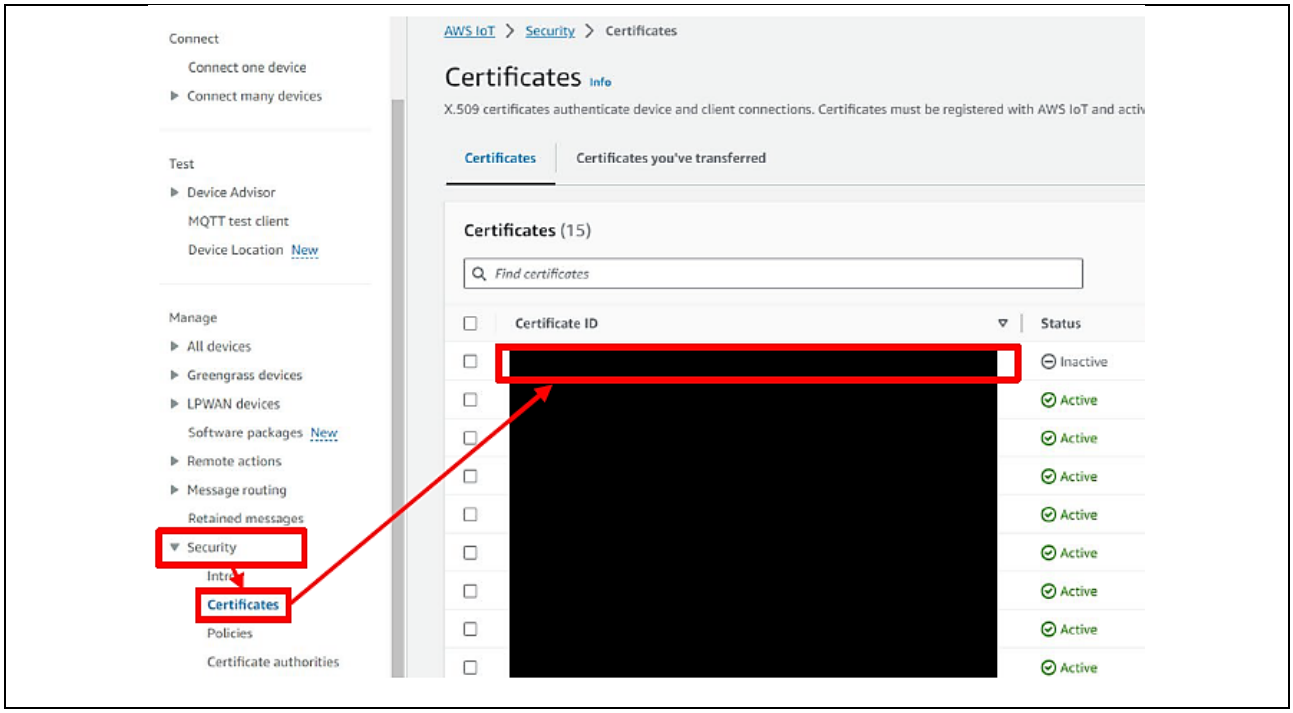

Figure 155. Certificate Settings

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

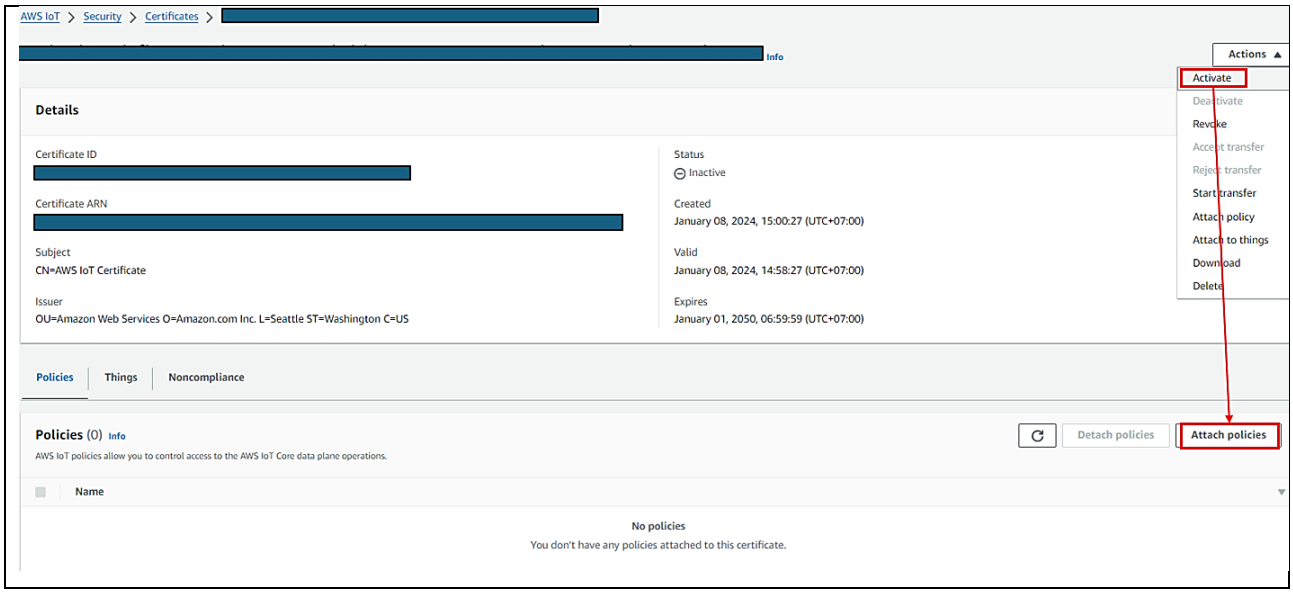

Figure 156. Certificate Settings: Attach Policies (1/2)

Clicking the Attach policies button opens the dialog box shown in **Figure 157**.

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. 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.

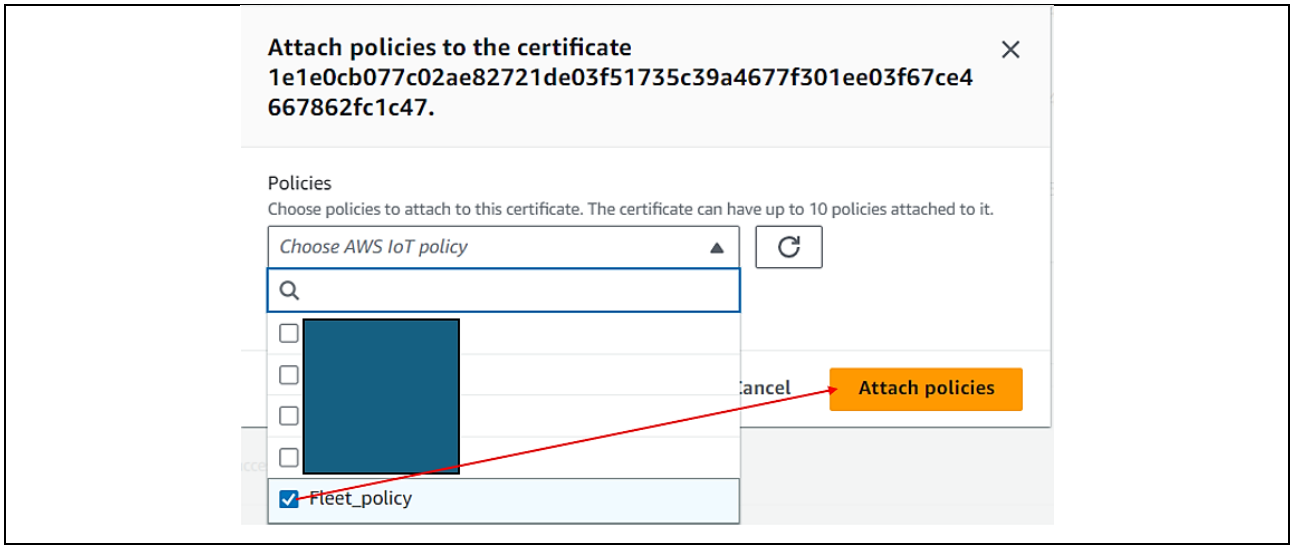

Figure 157. Certificate Settings: Attach Policies (2/2)

7.2.3 Creating a Fleet Provisioning Template

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

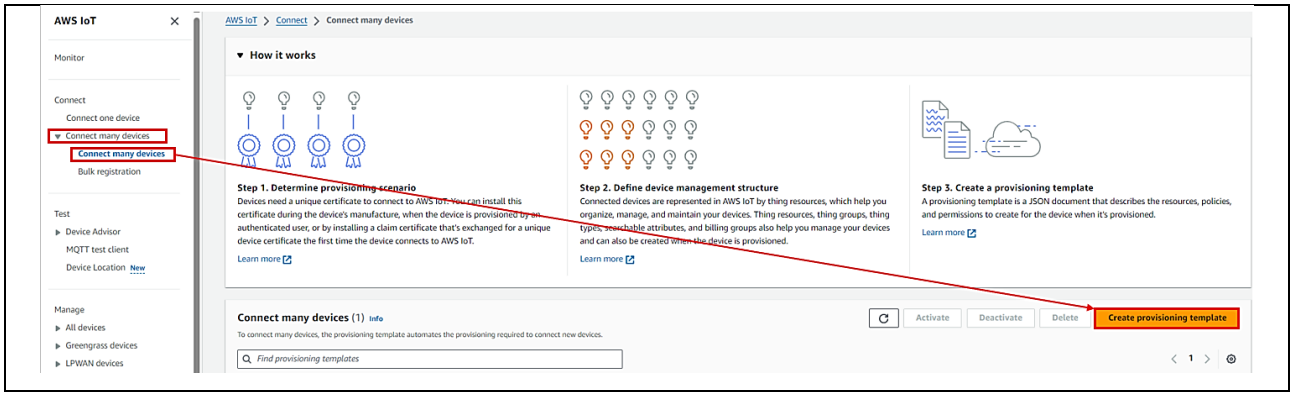

Figure 158. Creating a Provisioning Template (1/7)

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

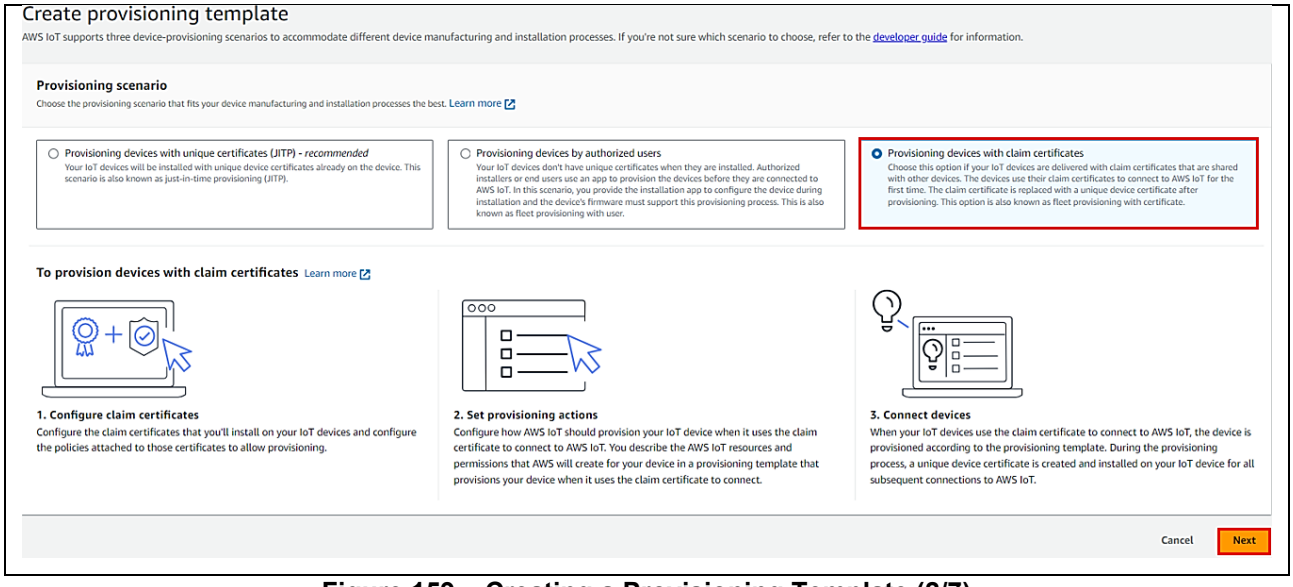

Figure 159. Creating a Provisioning Template (2/7)

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.

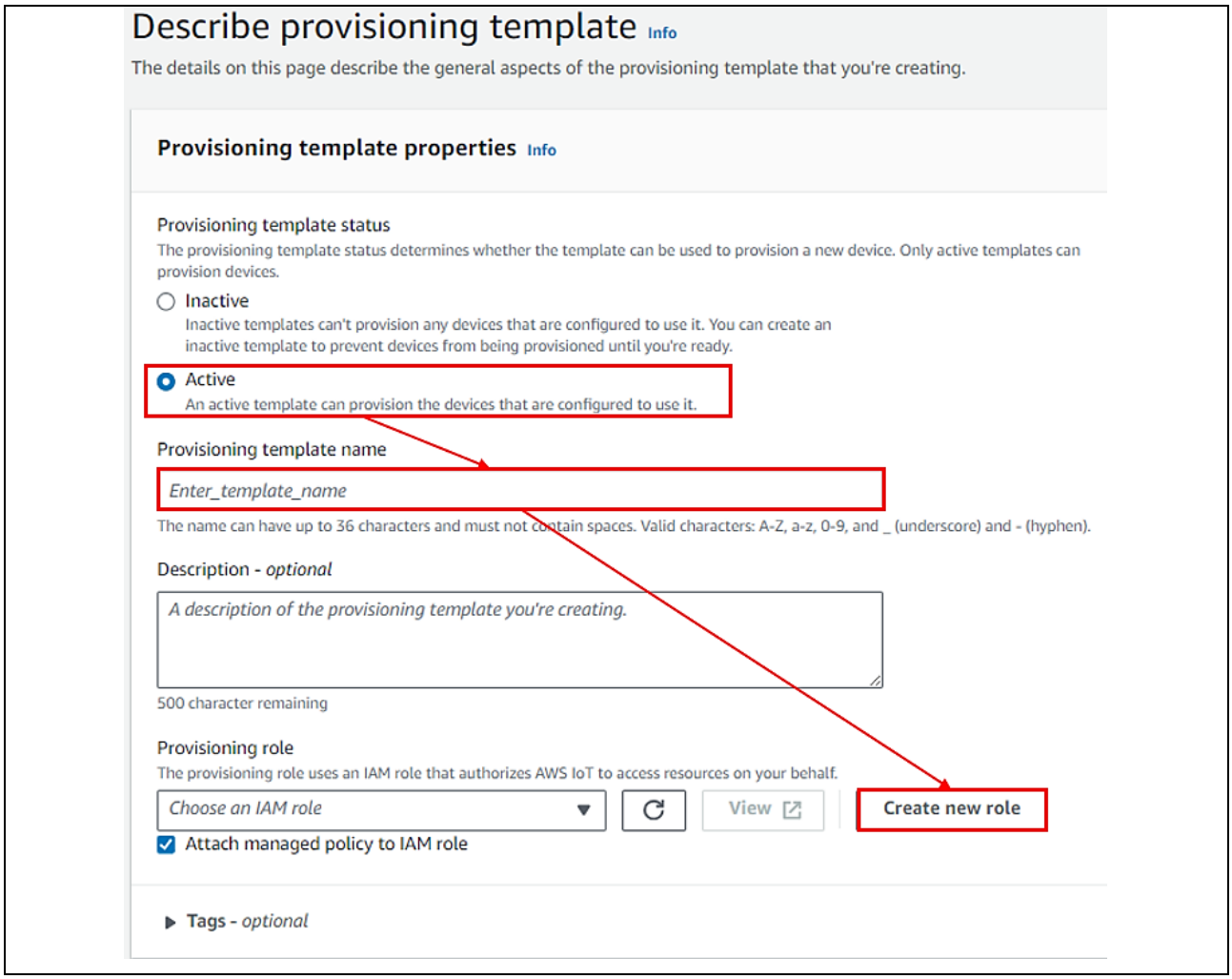

Figure 160. Creating a Provisioning Template (3/7)

For **Claim certificate policy**, select the policy to be used when fleet provisioning is run, created in **7.2.1**. For **Claim certificate**, select the certificate created in **7.2.3**, and click the **Next** button.

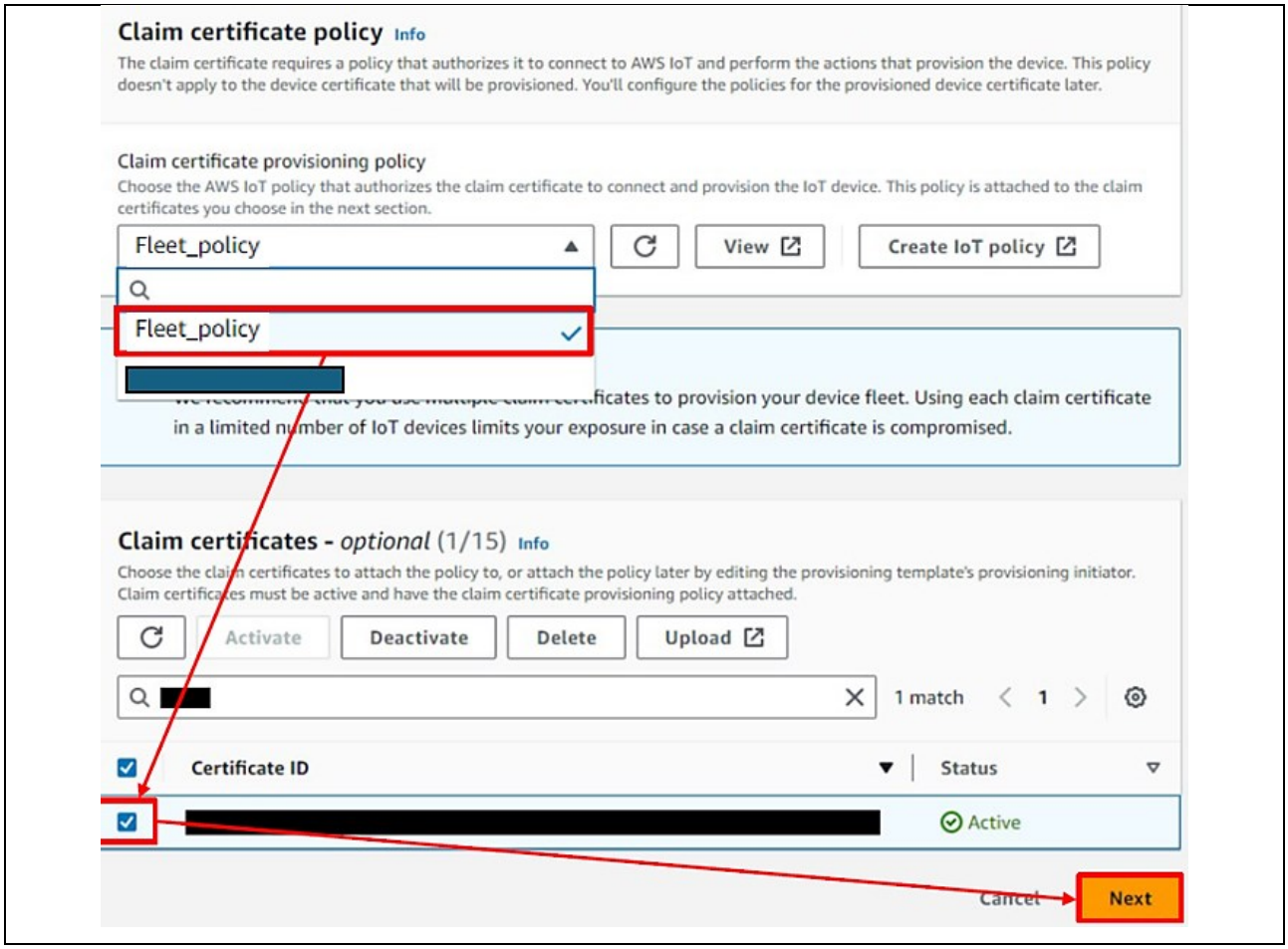

Figure 161. Creating a Provisioning Template (4/7)

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:

Provisioning devices that don't have device certificates using fleet provisioning – AWS IoT Core (amazon.com)

“Using pre-provisioning hooks with the AWS CLI”

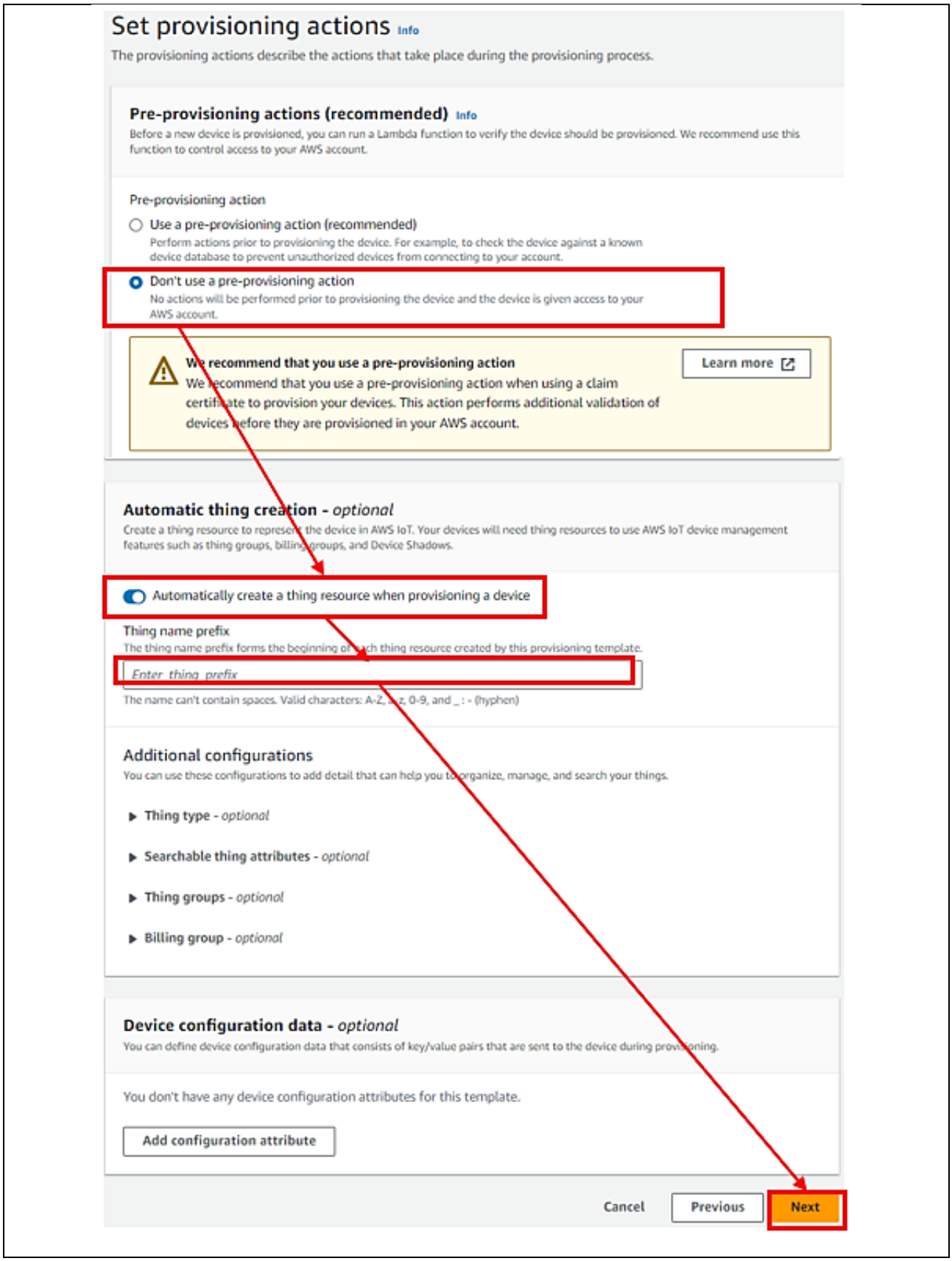

Figure 162. Creating a Provisioning Template (5/7)

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

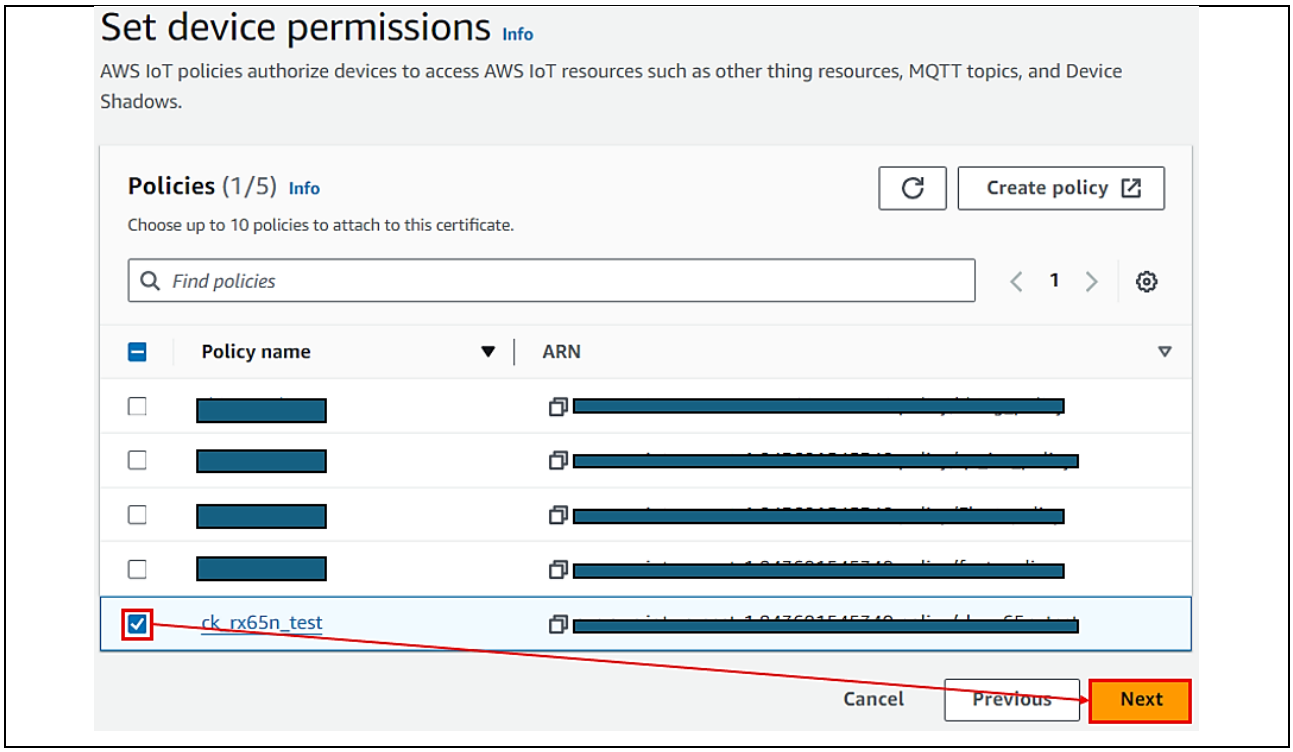

Figure 163. Creating a Provisioning Template (6/7)

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

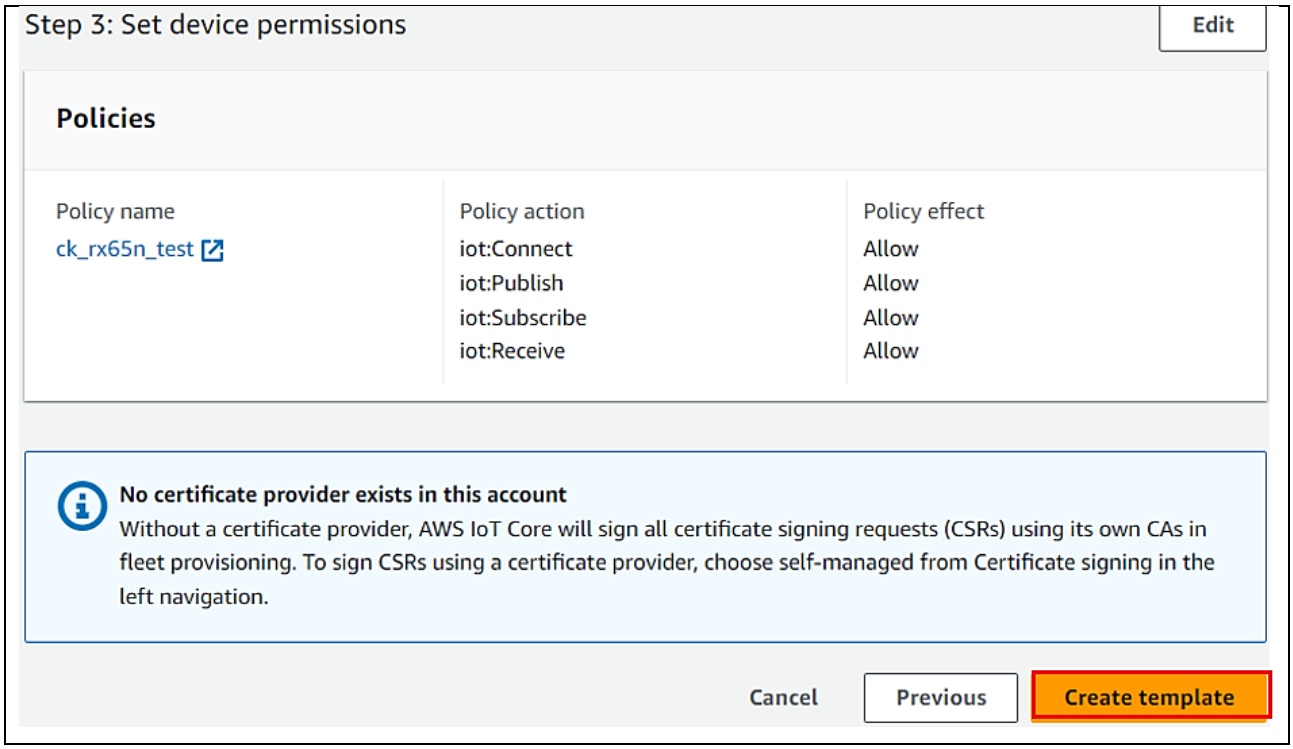

Figure 164. Creating a Provisioning Template (7/7)

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:**

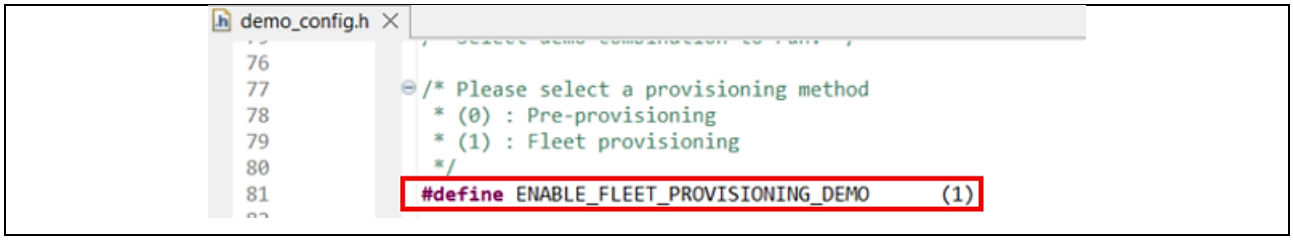

Figure 165. Enable Fleet Provisioning macro

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

7.4 Running Fleet Provisioning

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

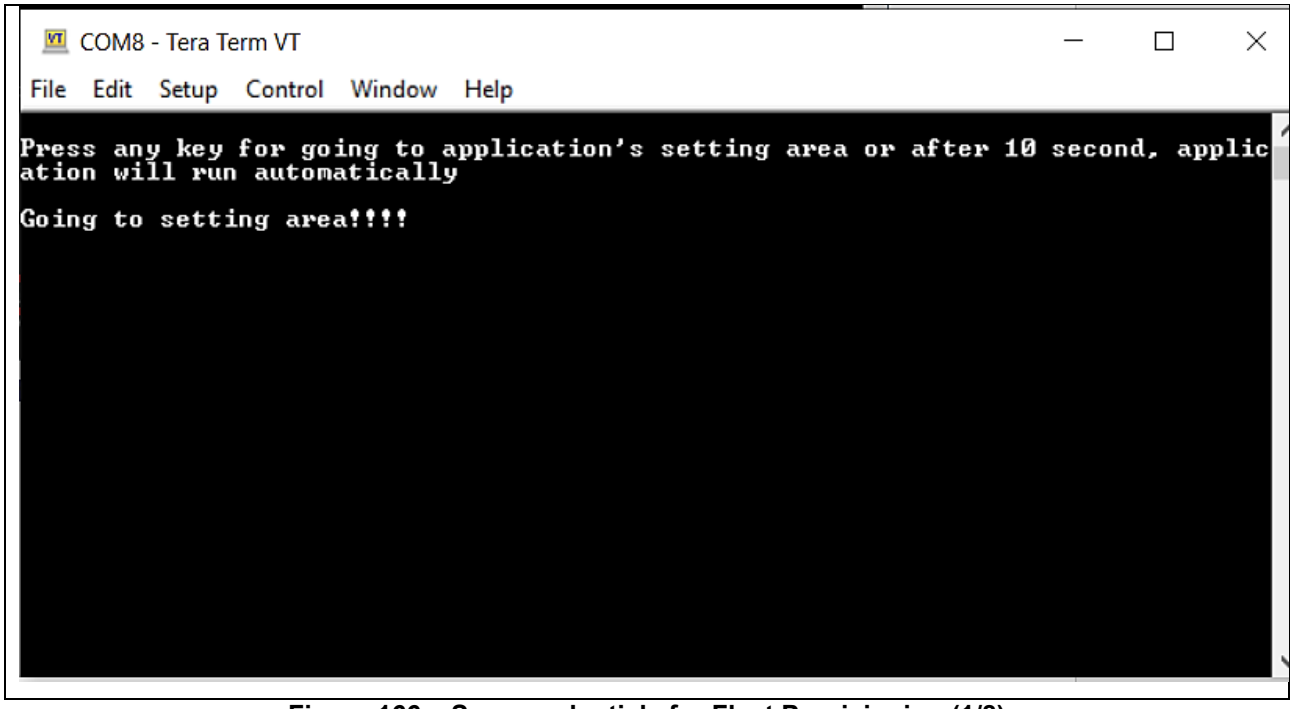

Figure 166. Save credentials for Fleet Provisioning (1/8)

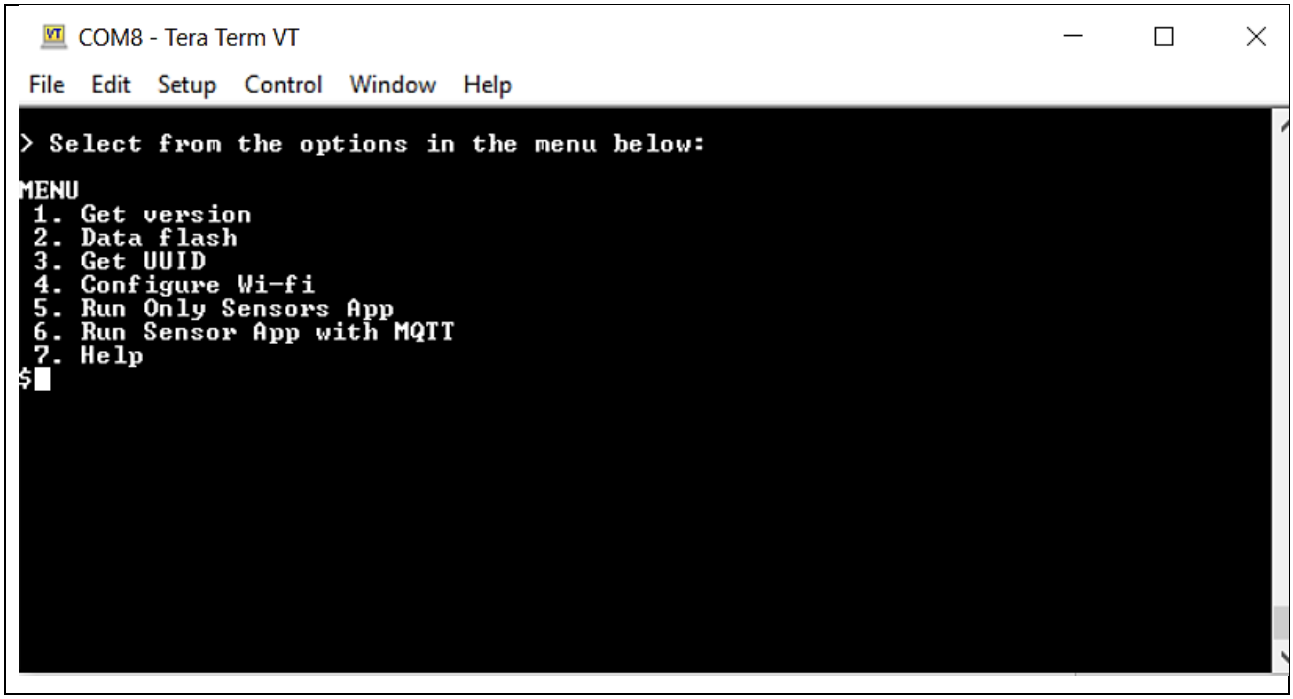

Figure 167. Save credentials for Fleet Provisioning (2/8)

2. 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)**

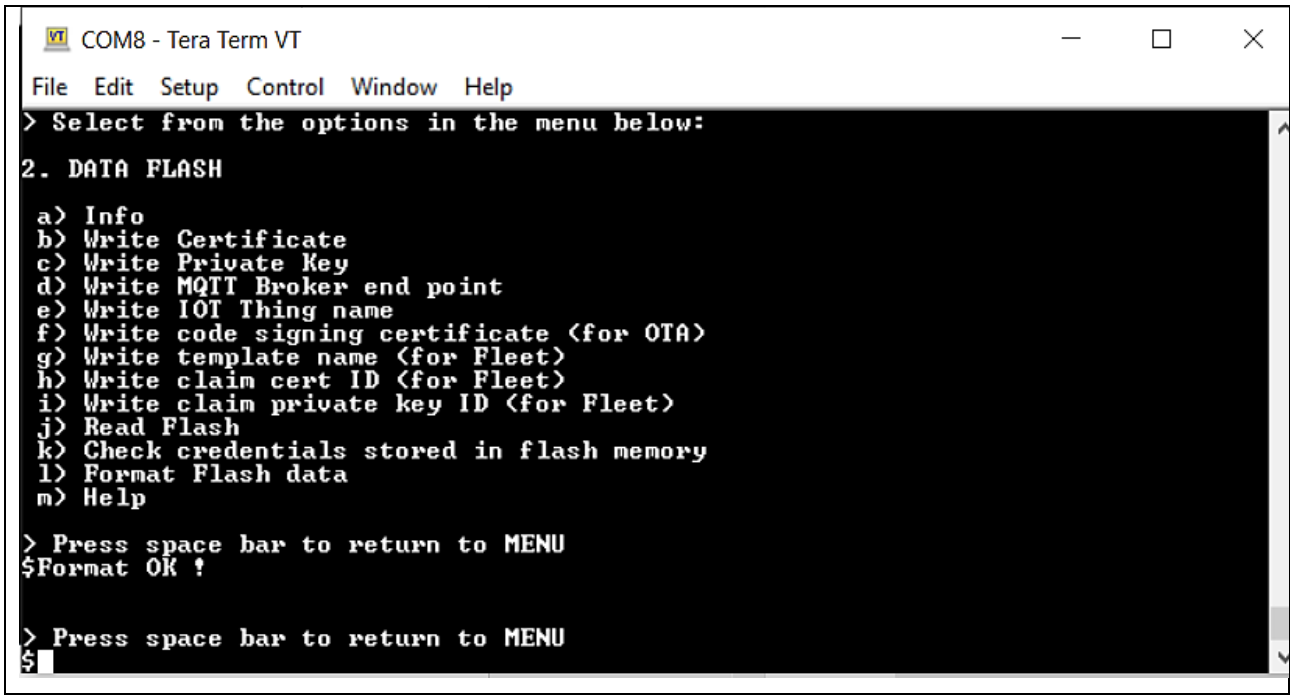

Figure 168. Save credentials for Fleet Provisioning (3/8)

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

3. Get MQTT endpoint and save to flash like topic **5.4.6 and 5.3.1**:

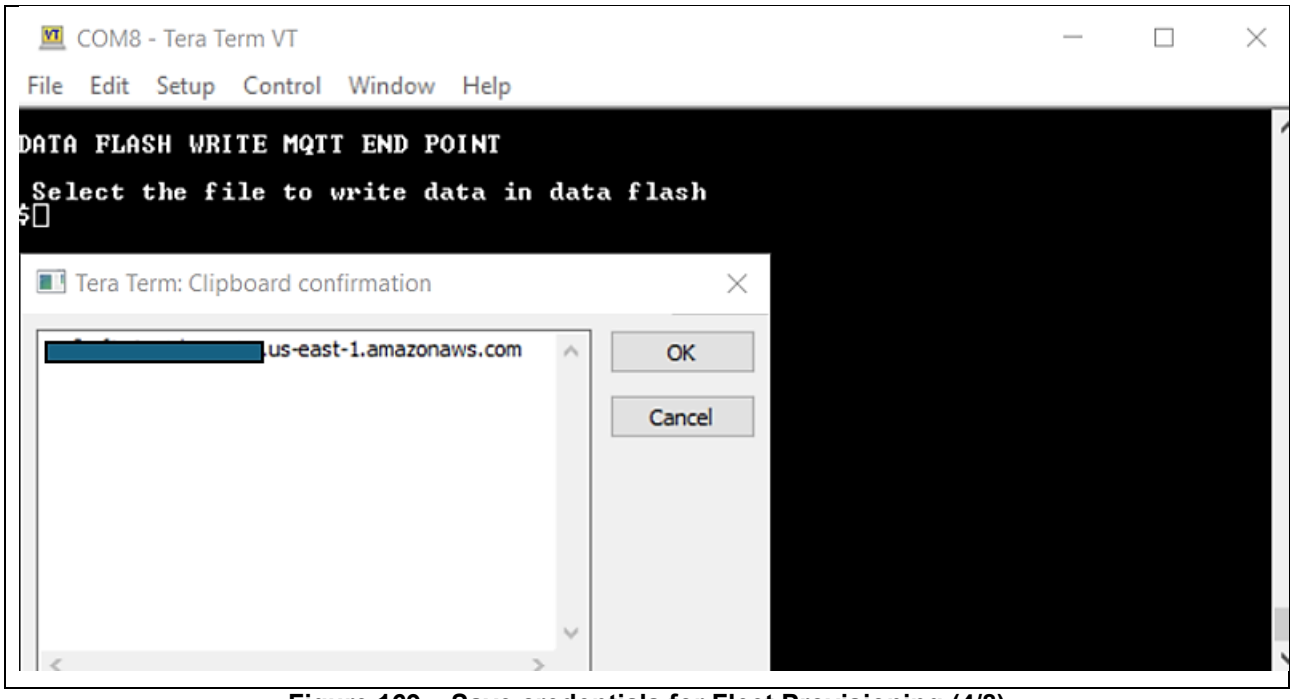

Figure 169. Save credentials for Fleet Provisioning (4/8)

4. 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.

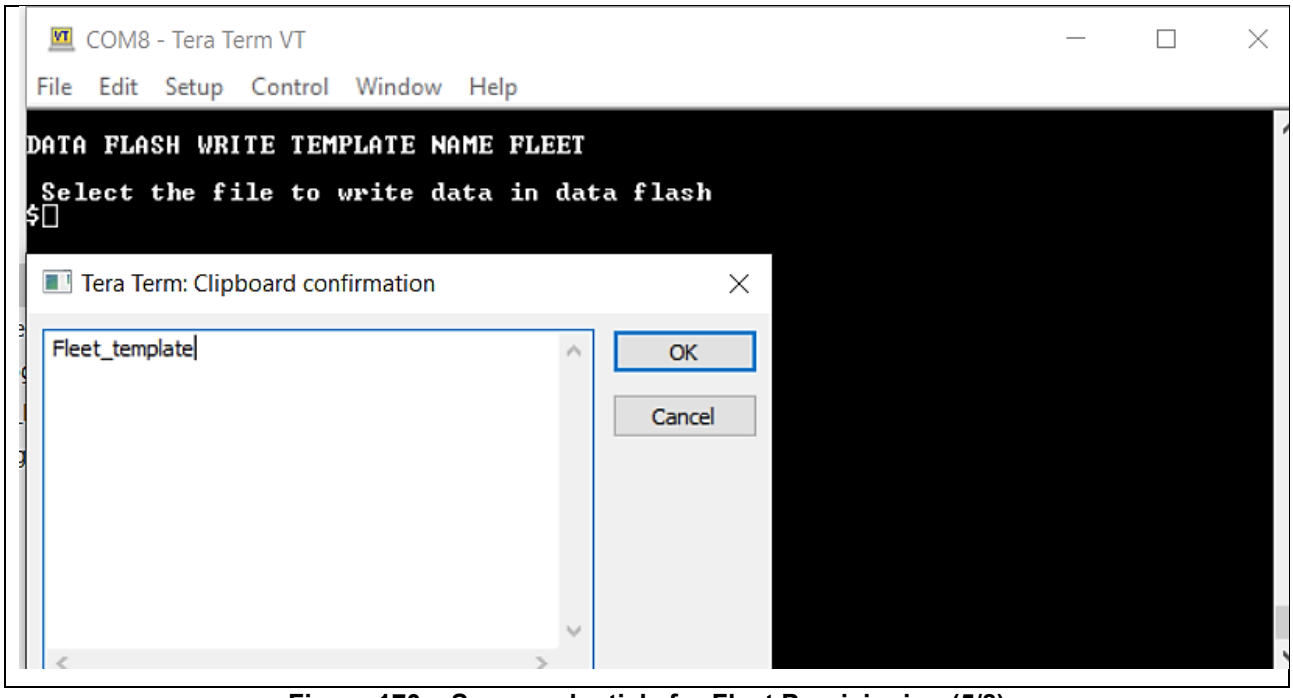

Figure 170. Save credentials for Fleet Provisioning (5/8)

5. Next, storing fleet claim cert ID, claim private key ID that users created at **0,** users downloaded (**Figure 154. Downloading the Certificate and Key Pair**):

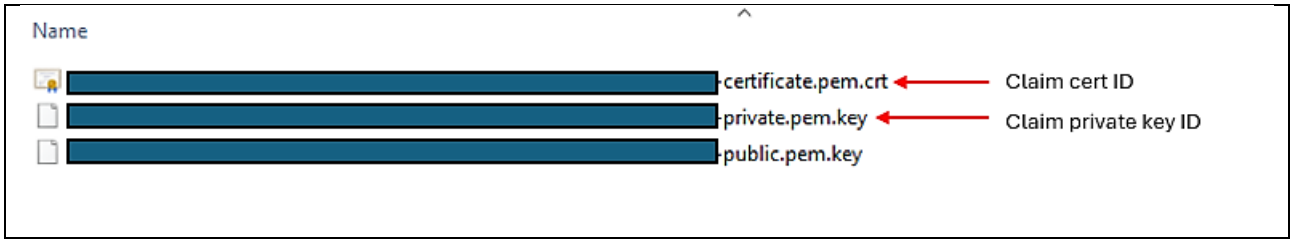

Figure 171. Save credentials for Fleet Provisioning (6/8)

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**".

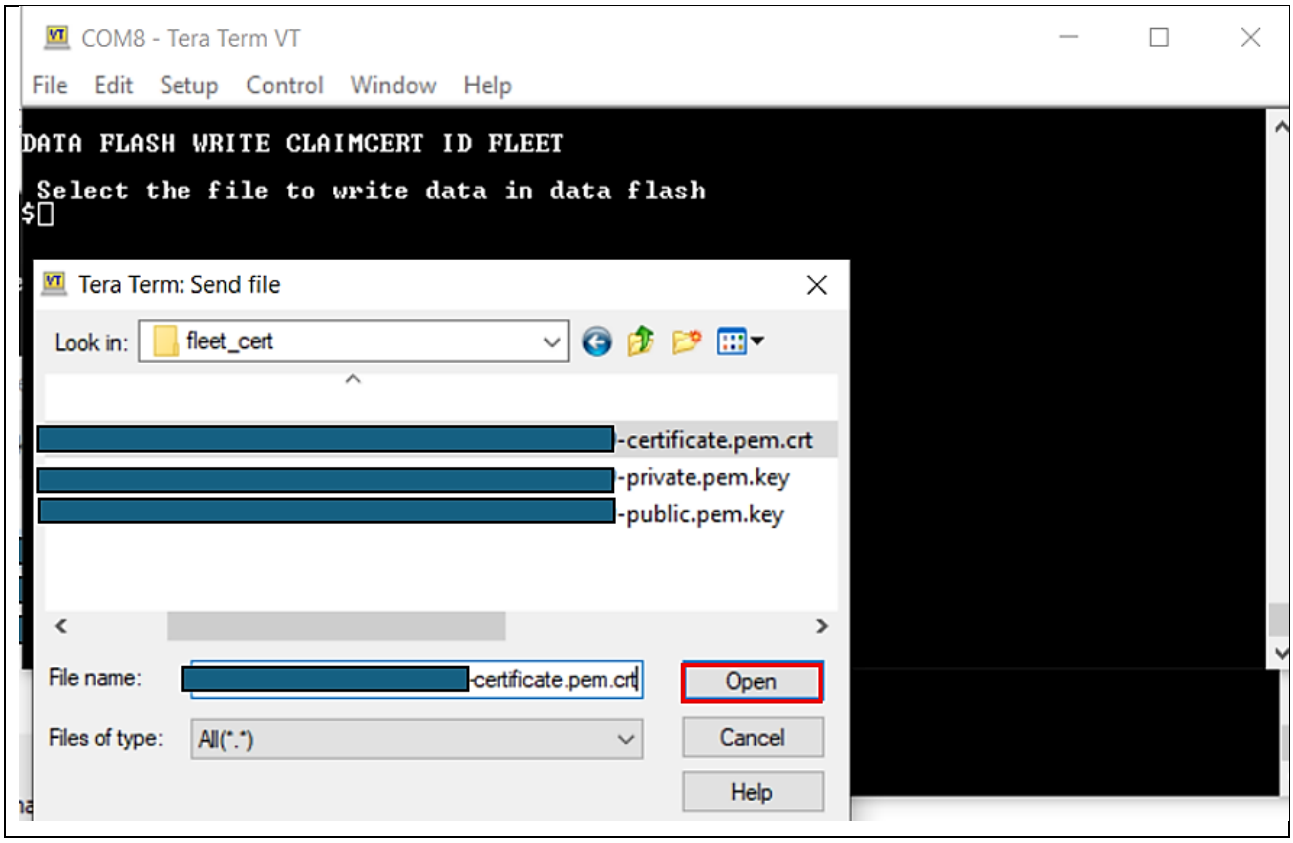

Figure 172. Save Credentials for Fleet Provisioning (7/8)

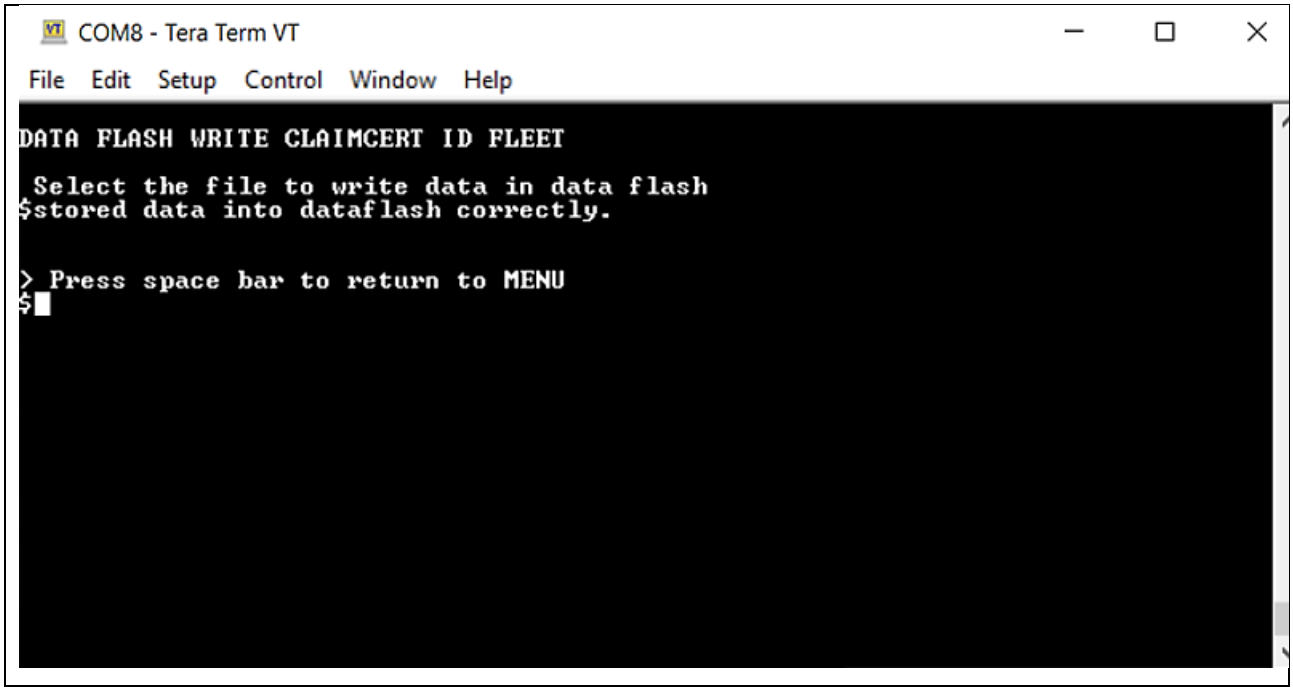

Figure 173. Save Credentials for Fleet Provisioning (8/8)

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:

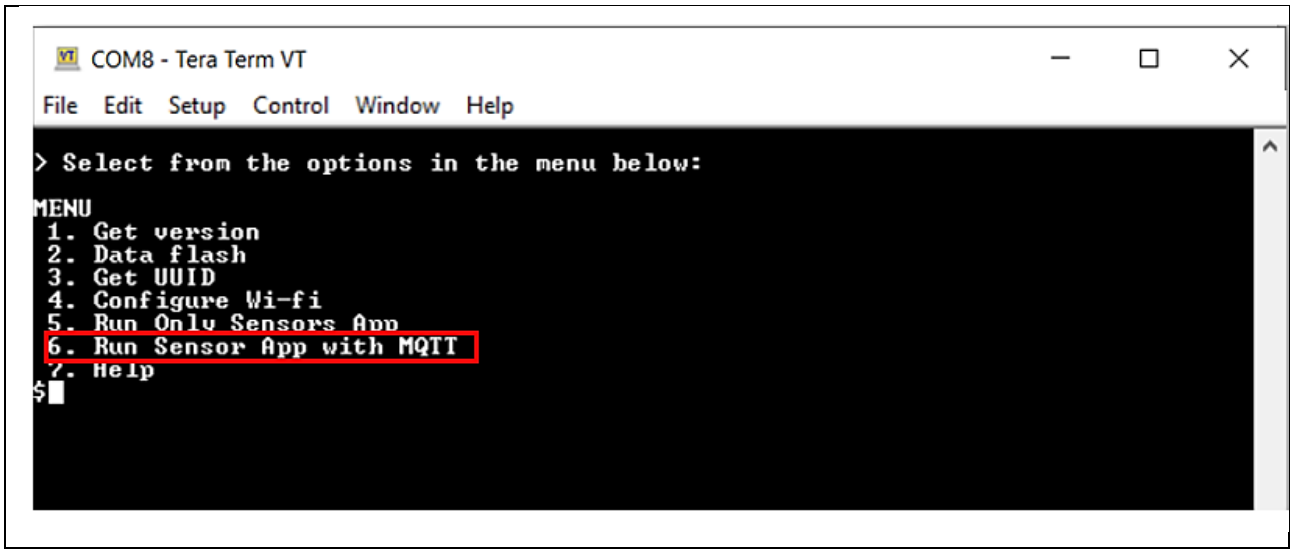

Figure 174. Running Application with Fleet (1/8)

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.

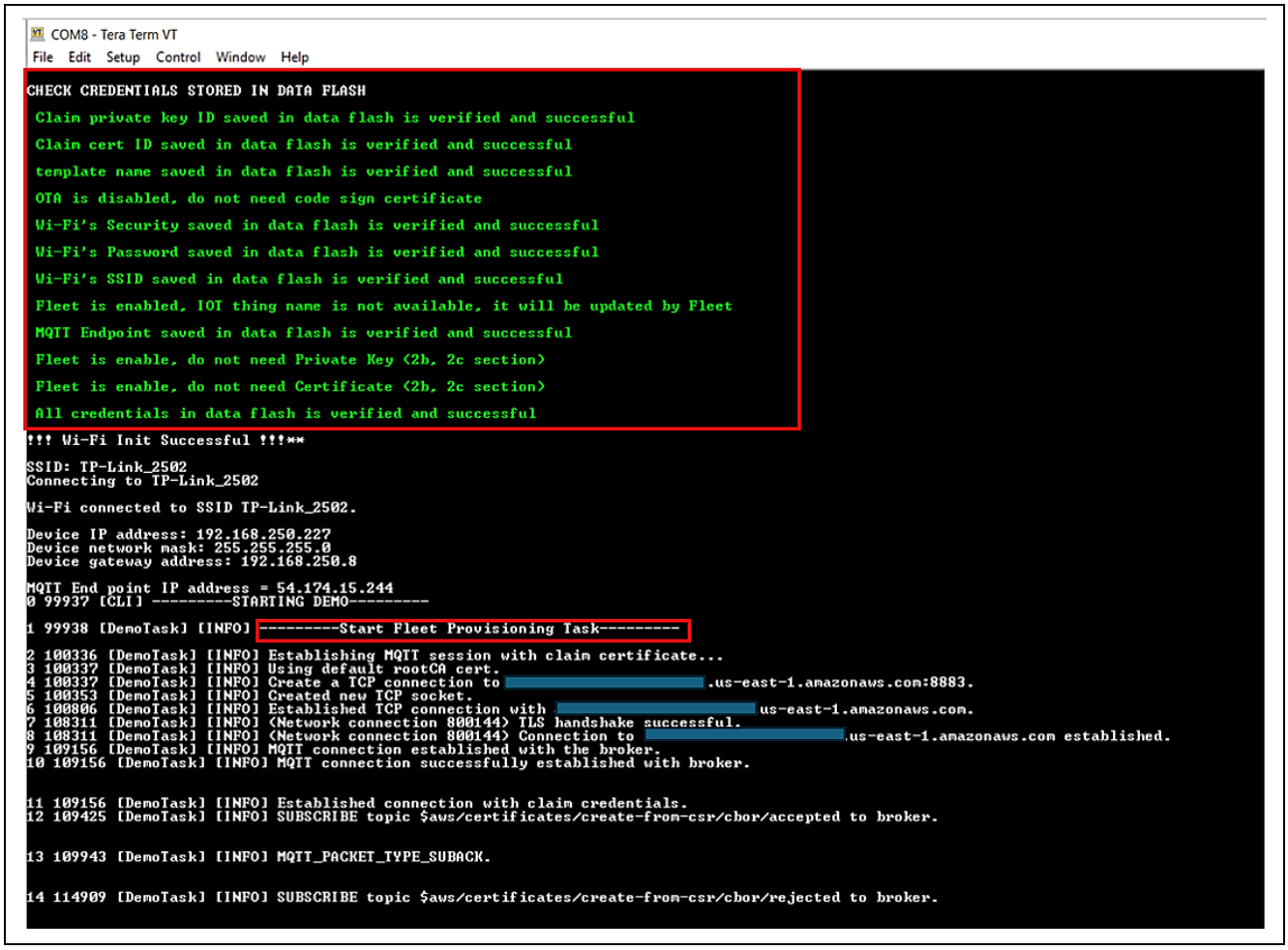

Figure 175. Running Application with Fleet (2/8)

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.

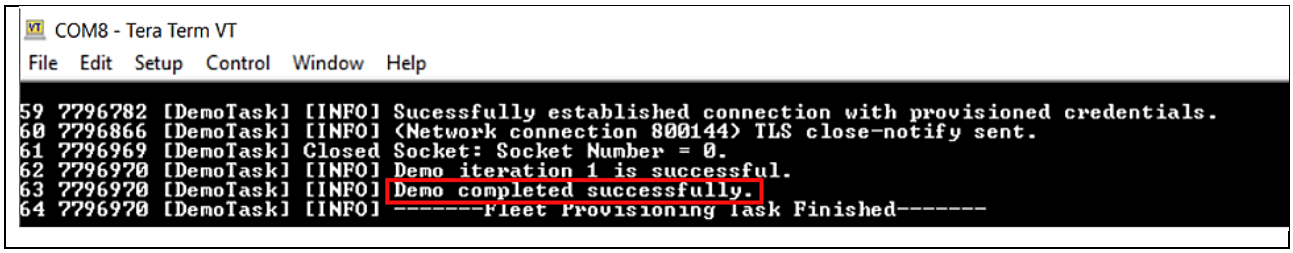

Figure 176. Running Application with Fleet (3/8)

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:

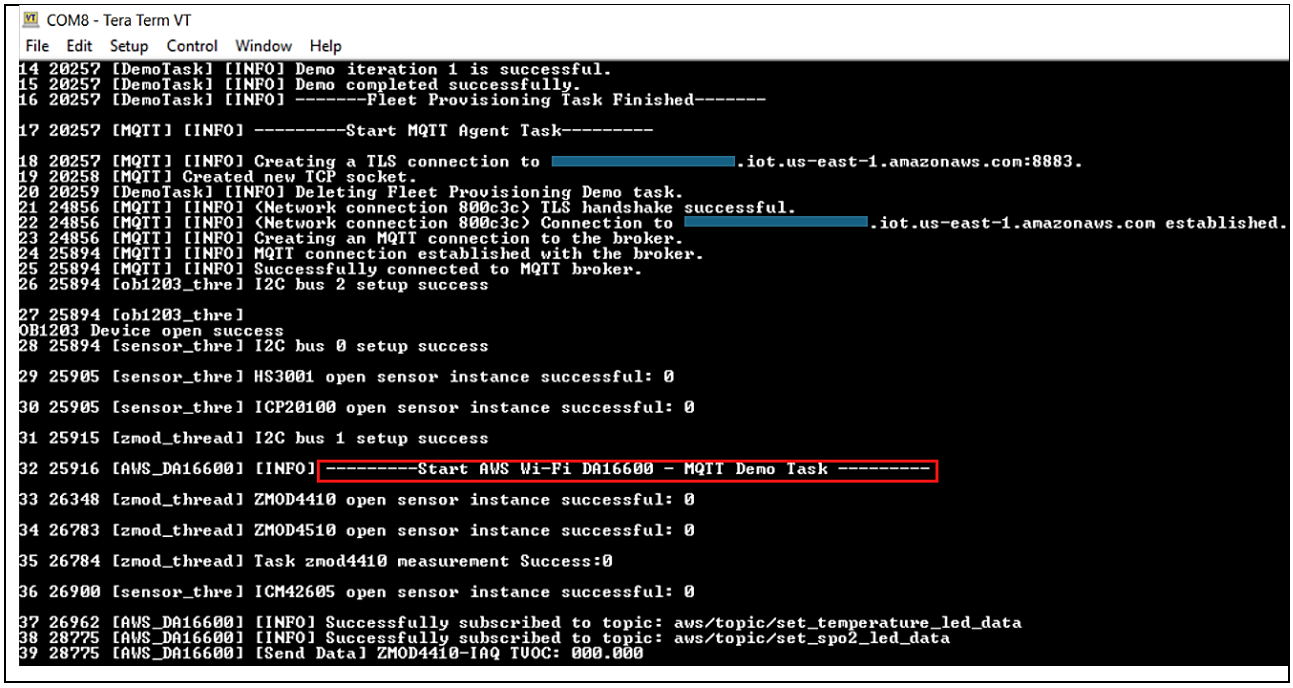

Figure 177. Running Application with Fleet (4/8)

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**"

Figure 178. Running Application with Fleet (5/8)

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

Under **All devices**, select **Things**. The registered thing is below (it likes thing's name above):

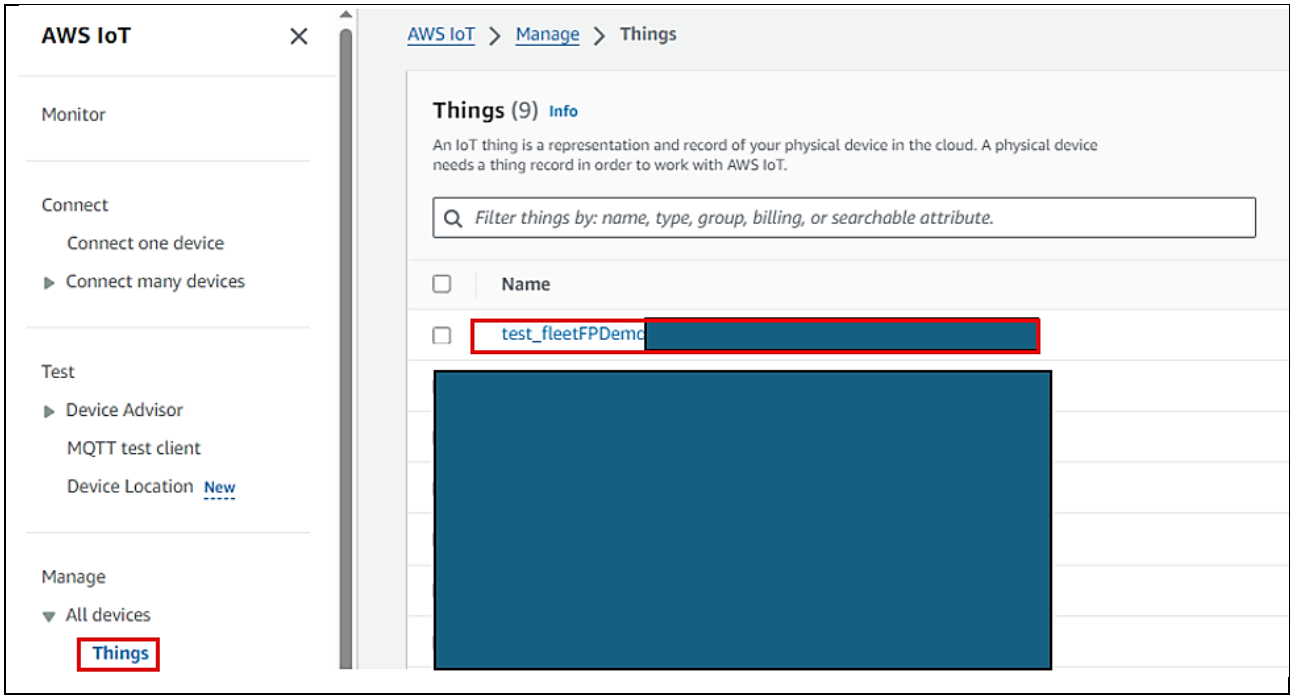

Figure 179. Running Application with Fleet (6/8)

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

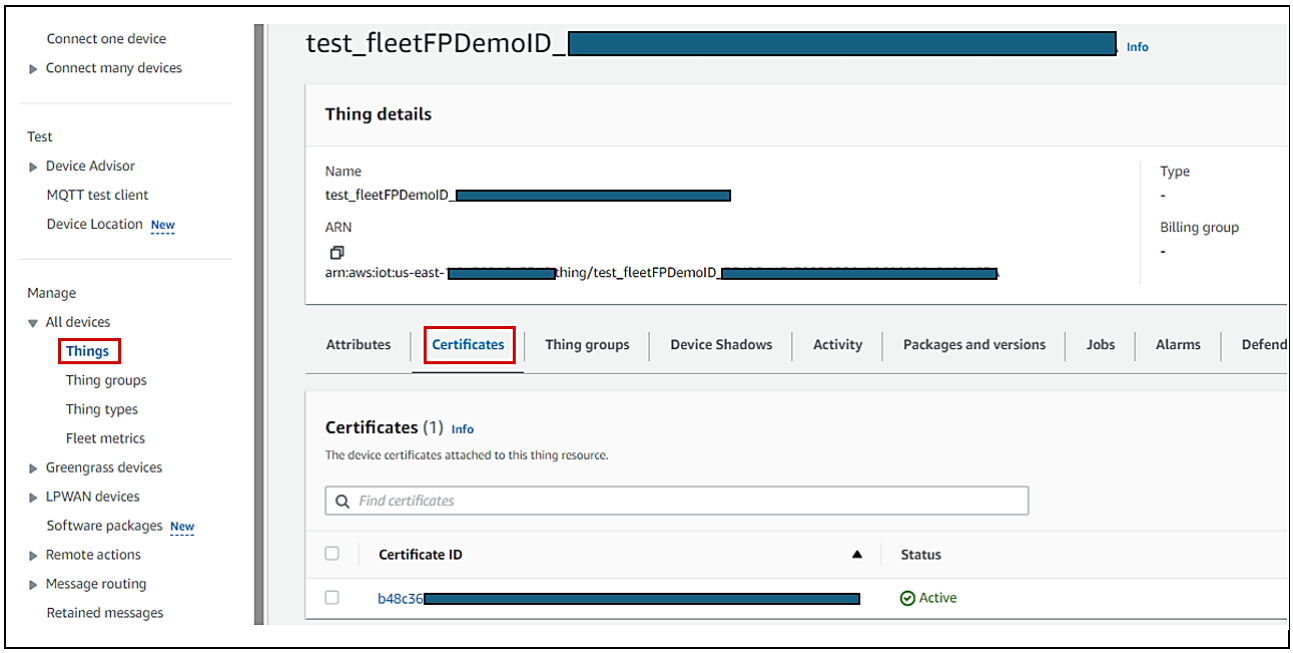

Figure 180. Running Application with Fleet (7/8)

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

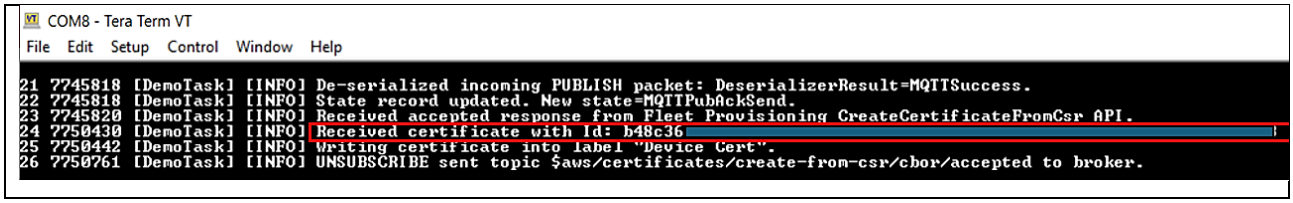

Figure 181. Running Application with Fleet (8/8)

8. Note and Trouble Shooting

8.1 About Stabilization Time for Sensor

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

The following table gives details of 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 min	Up to 1 Min
ZMOD4510 OAQ	Up to 1.5 hours	Up to 1 hours
OB1203	Up to 20 min (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

1. 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:\).
2. If a **'Cannot find "PFRAM2' specified in option "rom"'** error occurs as below:

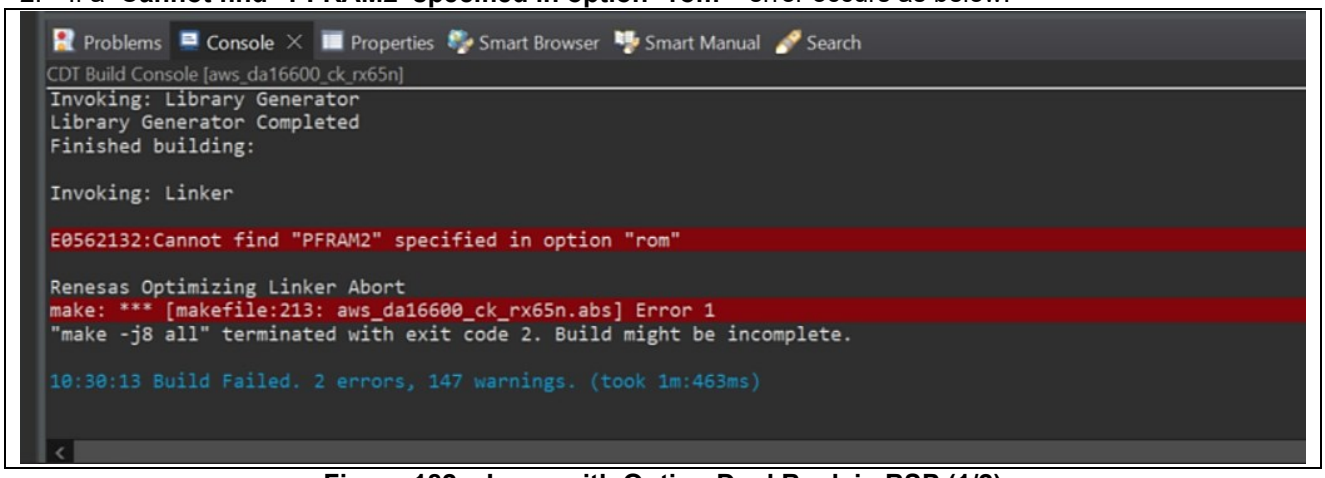

Figure 182. Issue with Option Dual Bank in BSP (1/2)

This issue happens after re-generating the Smart Configuration (file "aws_da16600_ck_rx65n.scfg") as shown in **Figure 16. Generate Code.** Please modify the value of macro **BSP_CFG_CODE_FLASH_BANK_MODE** to 0 (in file `src/smc_gen/r_config/r_bsp_config.h`) to run with dual mode of dual-bank function with the code flash memory and re-build again.

r_bsp_config.h

```
523         b31:b0  ROM Code - 0000 0000h: ROM code protection enabled (ROM code protection 1).
524                            0000 0001h: ROM code protection enabled (ROM code protection 2).
525                            Other than above: ROM code protection disabled.
526         Note: The ROMCODE register should be set in 32-bit units.
527       Default value is 0xFFFFFFFF.
528     */
529     #define BSP_CFG_ROMCODE_REG_VALUE  (0xFFFFFFFF)
530
531     /* Select the bank mode of dual-bank function of the code flash memory.
532       0 = Dual mode.
533       1 = Linear mode. (default)
534       NOTE: If the dual bank function has been incorporated in a device, select the bank mode in this macro.
535             Default setting of the bank mode is linear mode.
536             If the dual bank function has not been incorporated in a device, this macro should be 1.
537     */
538     #define BSP_CFG_CODE_FLASH_BANK_MODE    (0)
539
```

Figure 183. Issue with Option Dual Bank in BSP (2/2)

8.3 When Run Errors Occur

In case of 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 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.

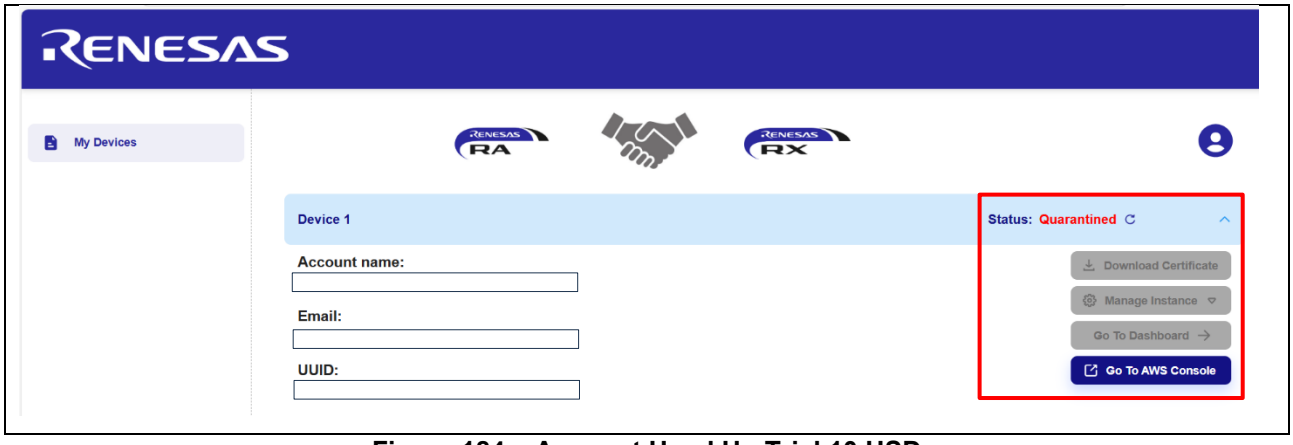

Figure 184. Account Used Up Trial 10 USD

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**.

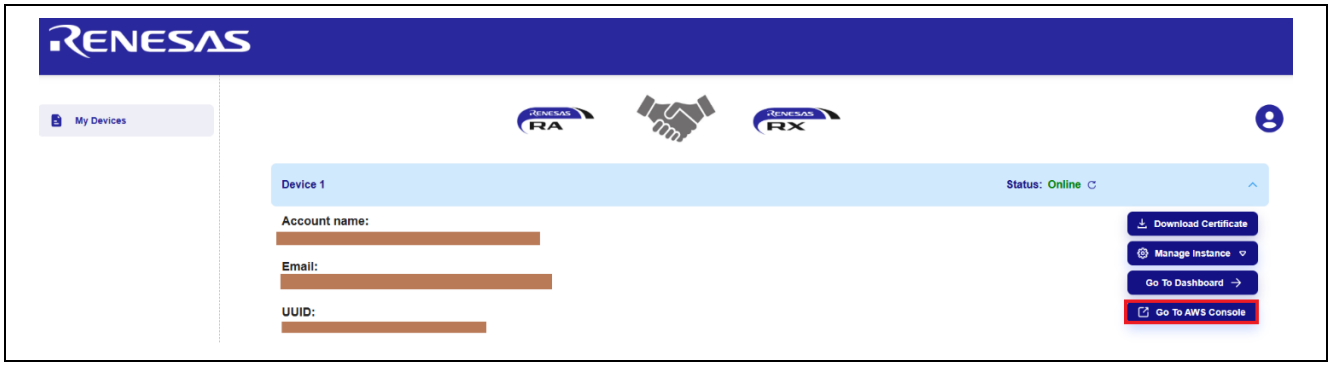

Figure 185. Accessing AWS Account from the Dashboard

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

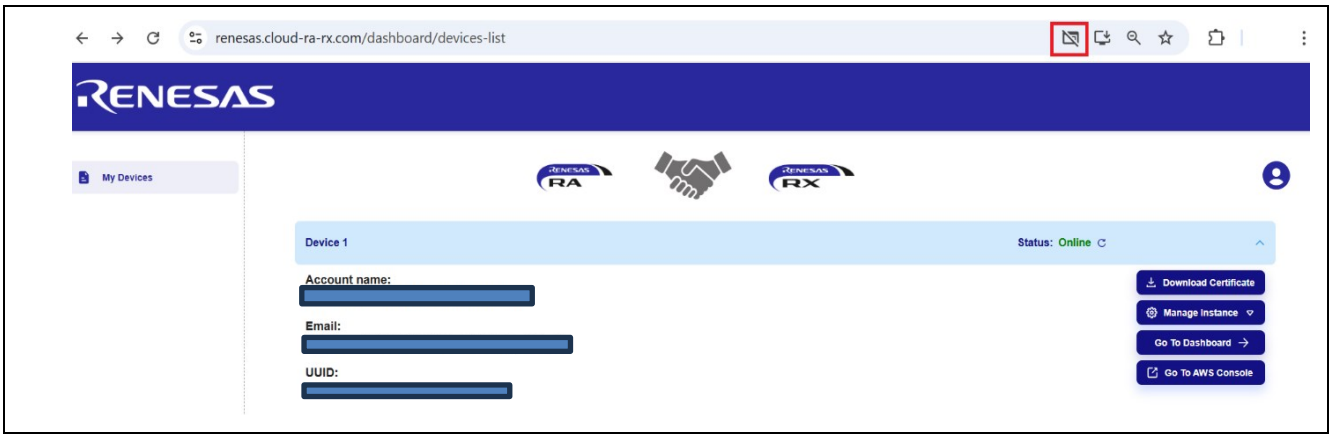

Figure 186. Pop Ups on the browser

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

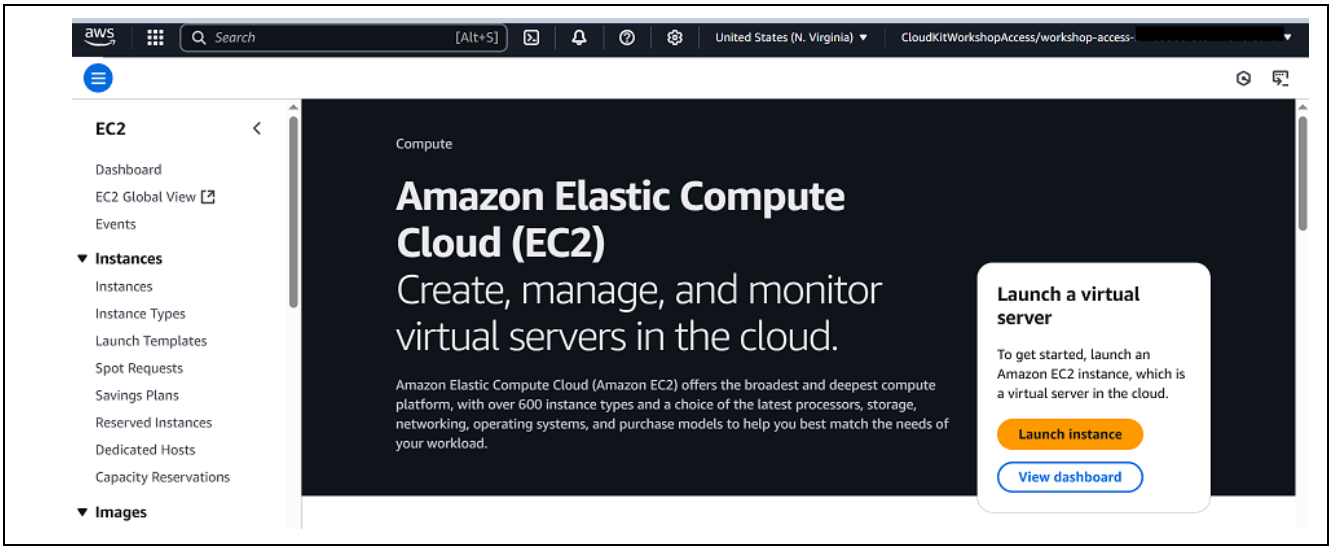

Figure 187. AWS account

4. Payment options can be accessed from the ‘Billing and Cost Management’ section of the console.

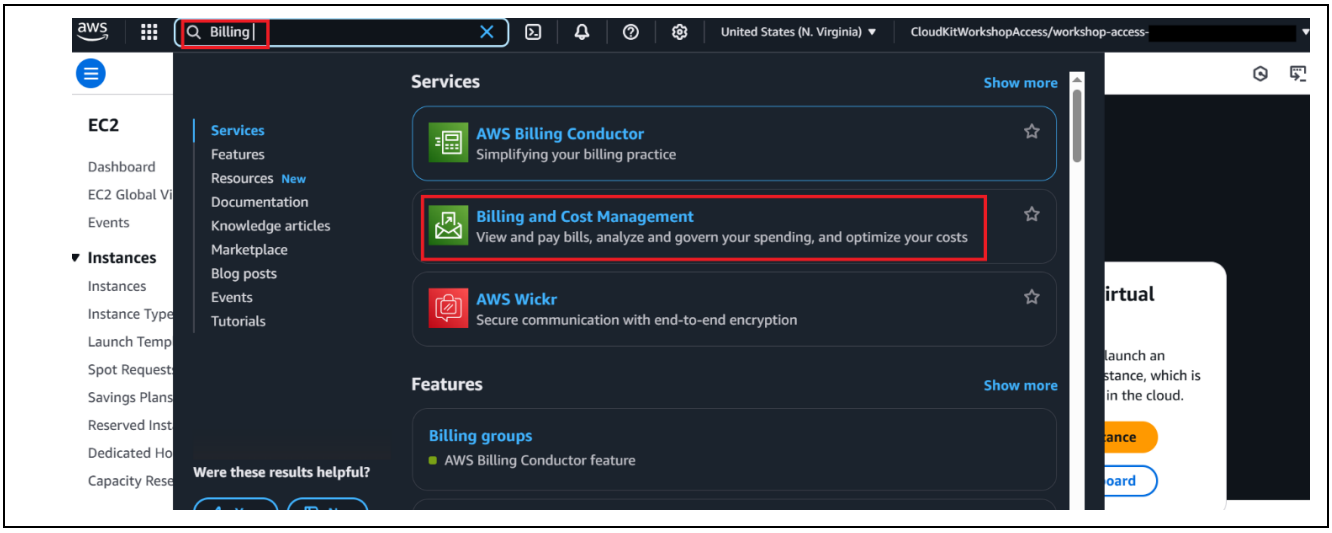

Figure 188. Billing and Cost Management

5. Scroll to the ‘Preferences and Settings’ section and click on ‘Payment Preferences’.

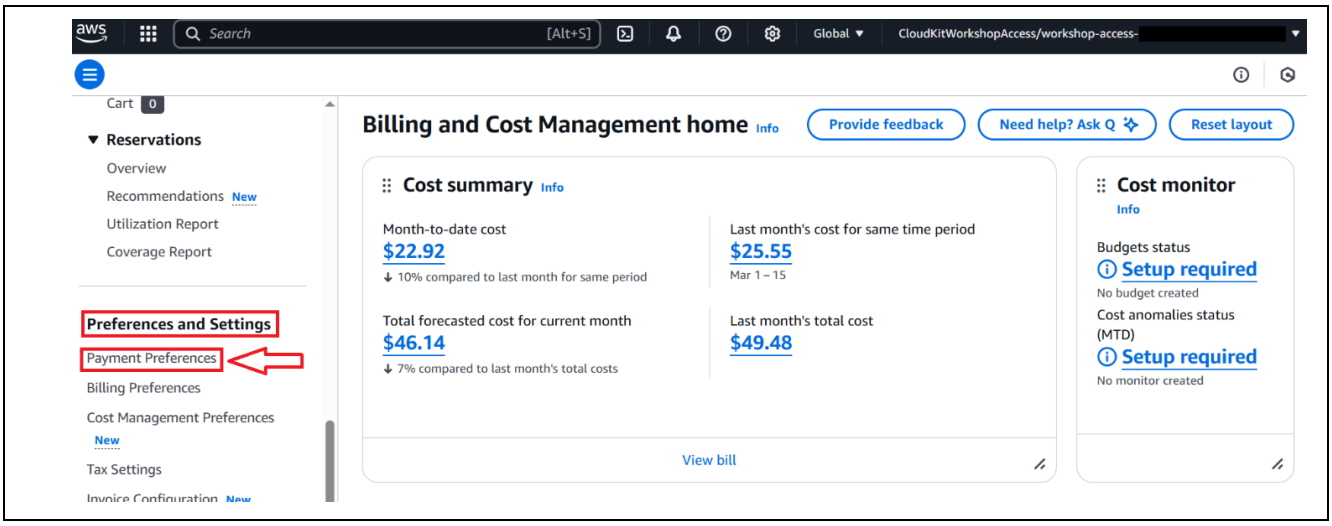

Figure 189. Payment Preferences

6. Add payment method.

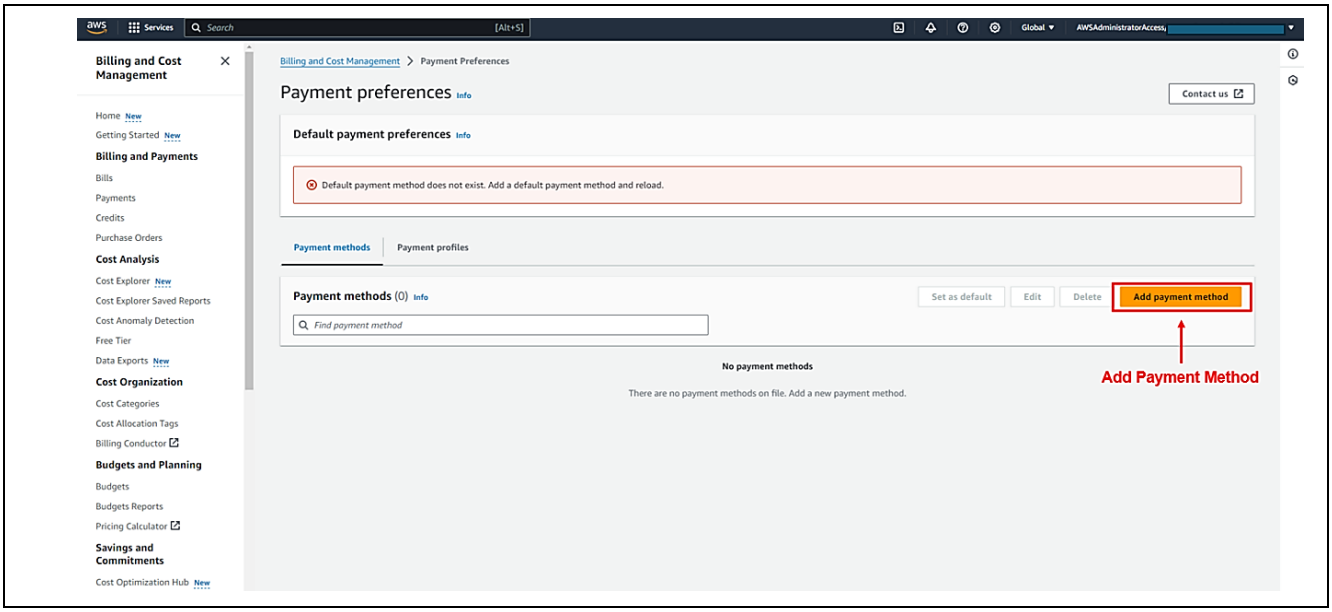

Figure 190. Add Payment Method

7. Go to ‘AWS Organizations’.

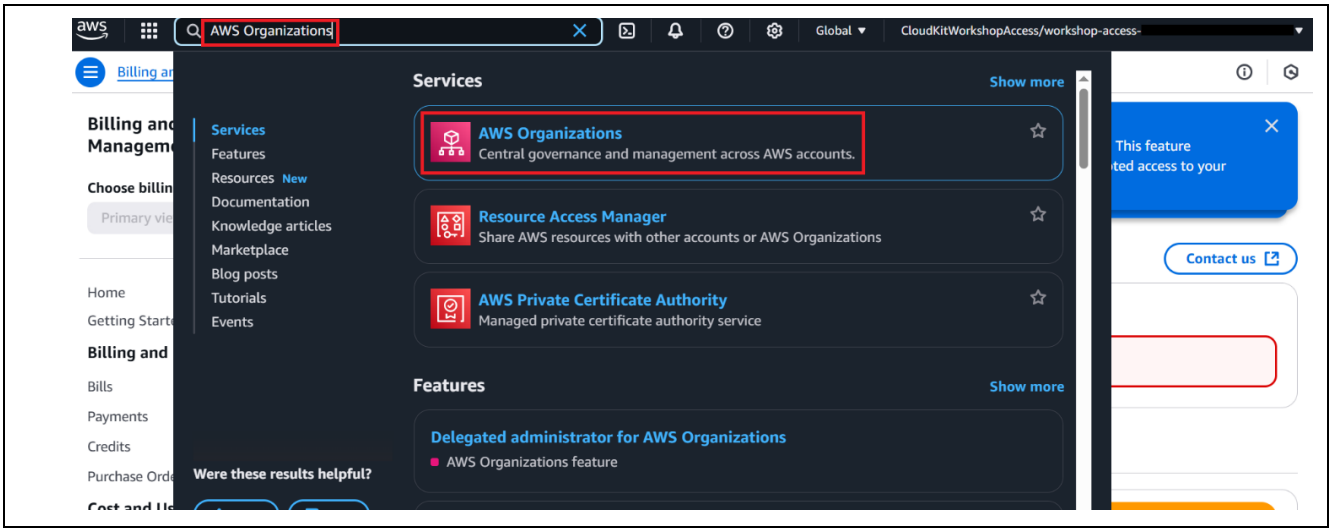

Figure 191. AWS Organizations

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

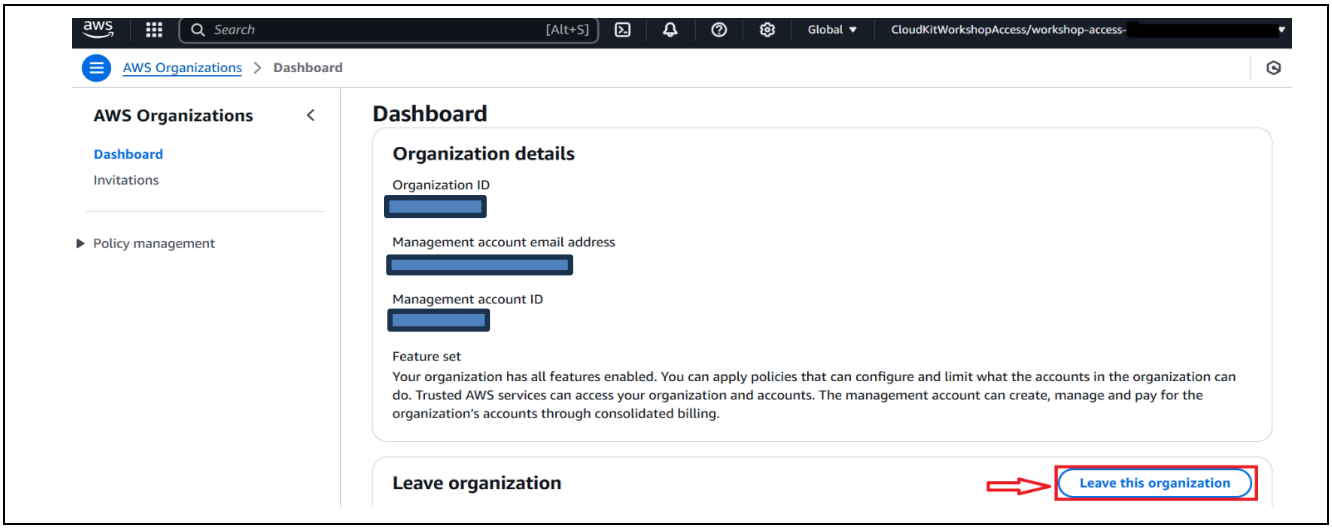

Figure 192. Leaving the organization

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'.

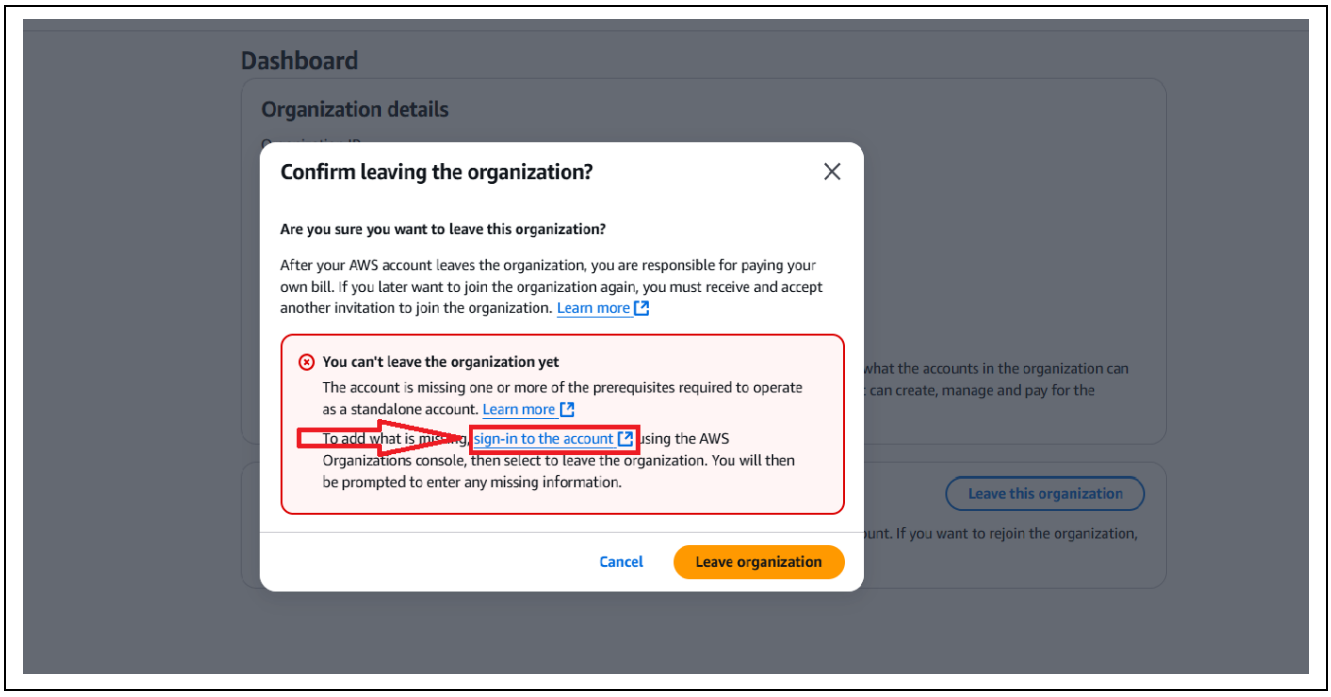

Figure 193. AWS Sign up process

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

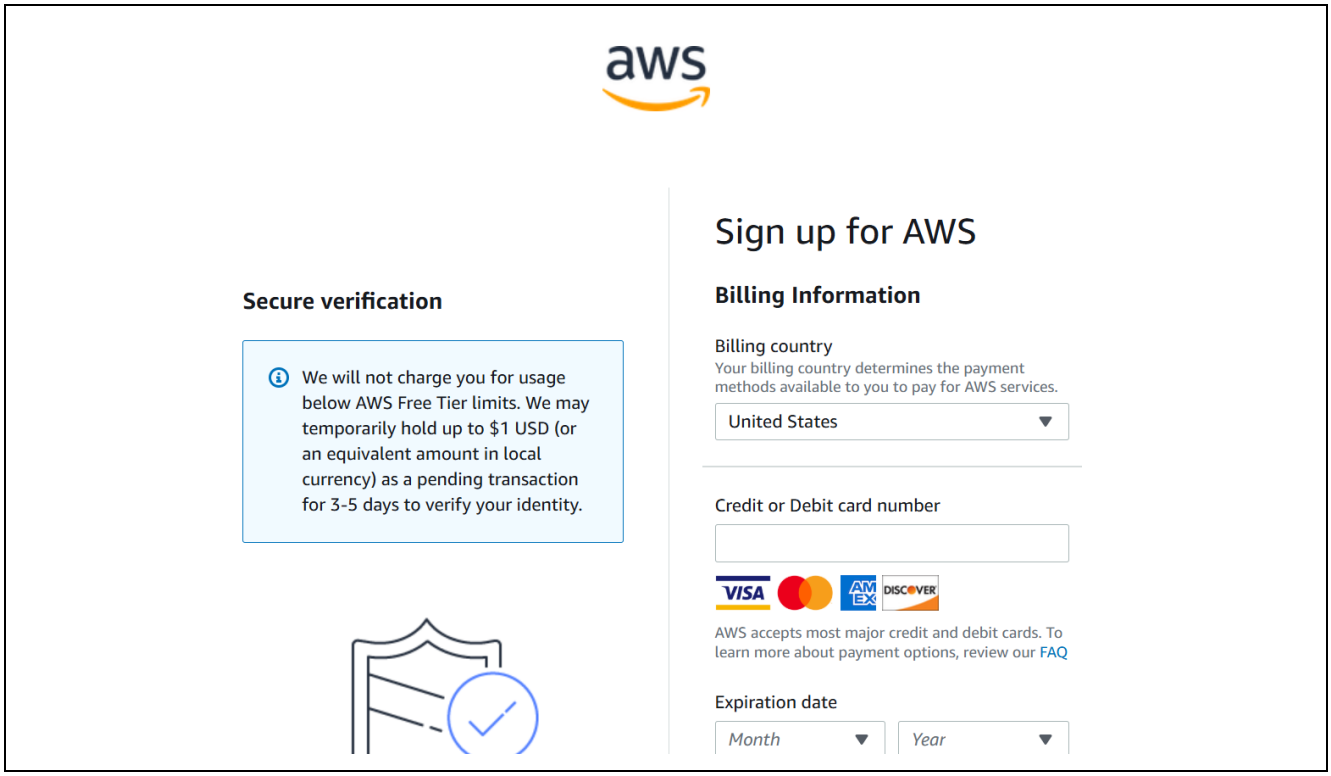

Figure 194. Prerequisites for leaving the organization

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 193.
2. Update the payment preference.

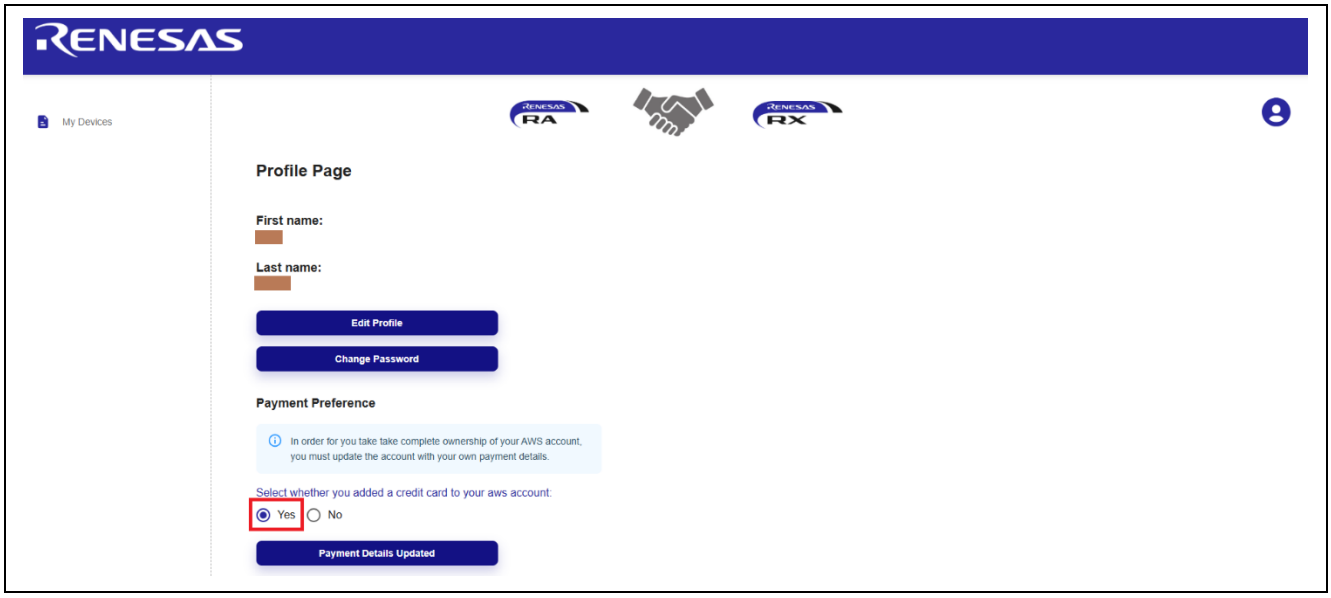

Figure 195. Payment Preference Update

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 6 of 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 your 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.

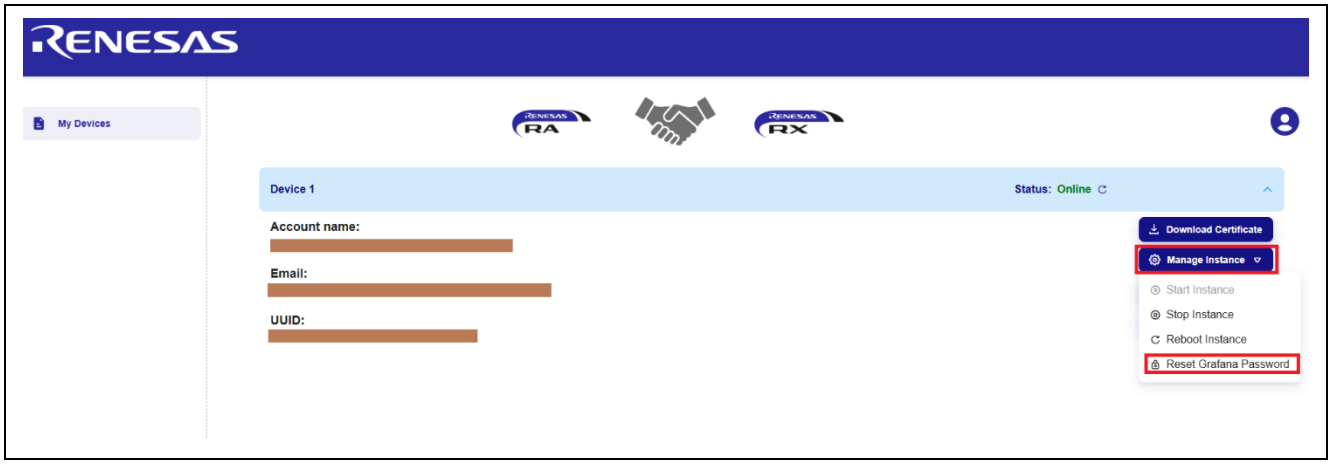

Figure 196. Grafana Password Reset

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.

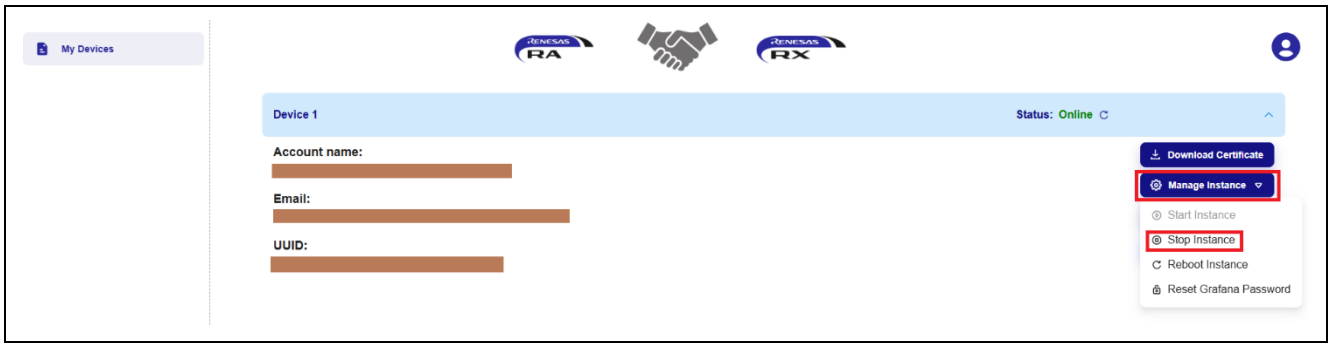

Figure 197. Stop EC2 instance

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

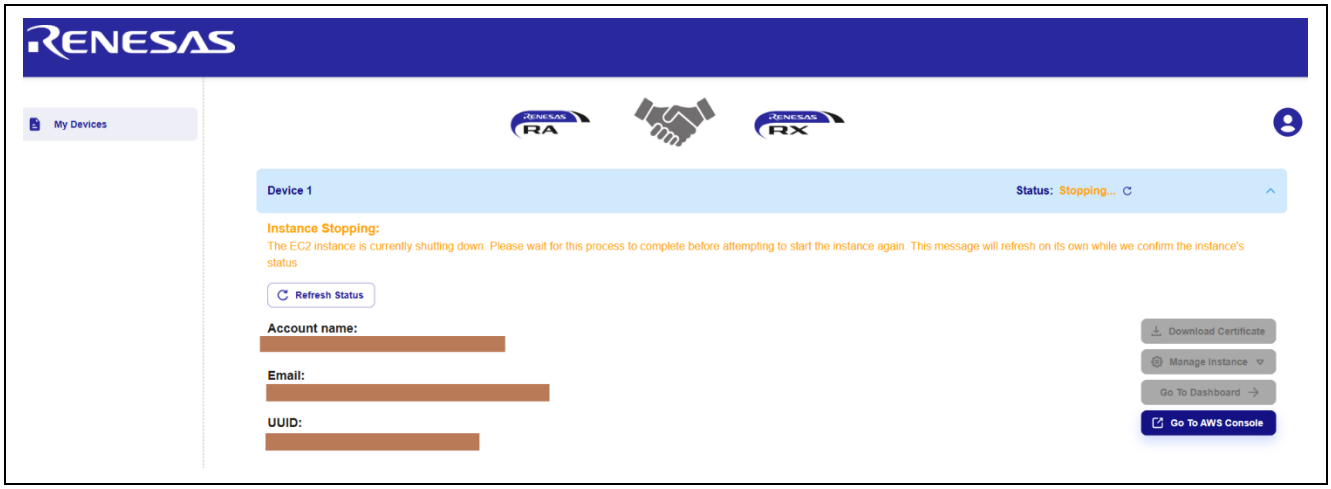

Figure 198. EC2 instance status

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

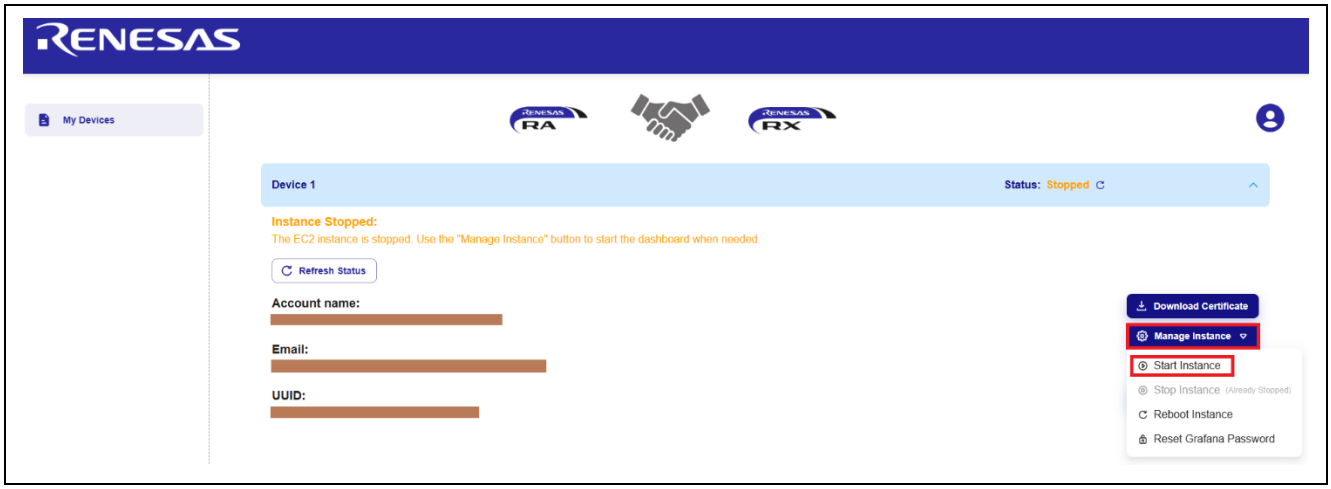

Figure 199. Restarting the EC2 Instance State

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

1. Access AWS account (Refer to Error! Reference source not found.)

2. From the **Services** menu, select **Compute** and then choose **EC2.**

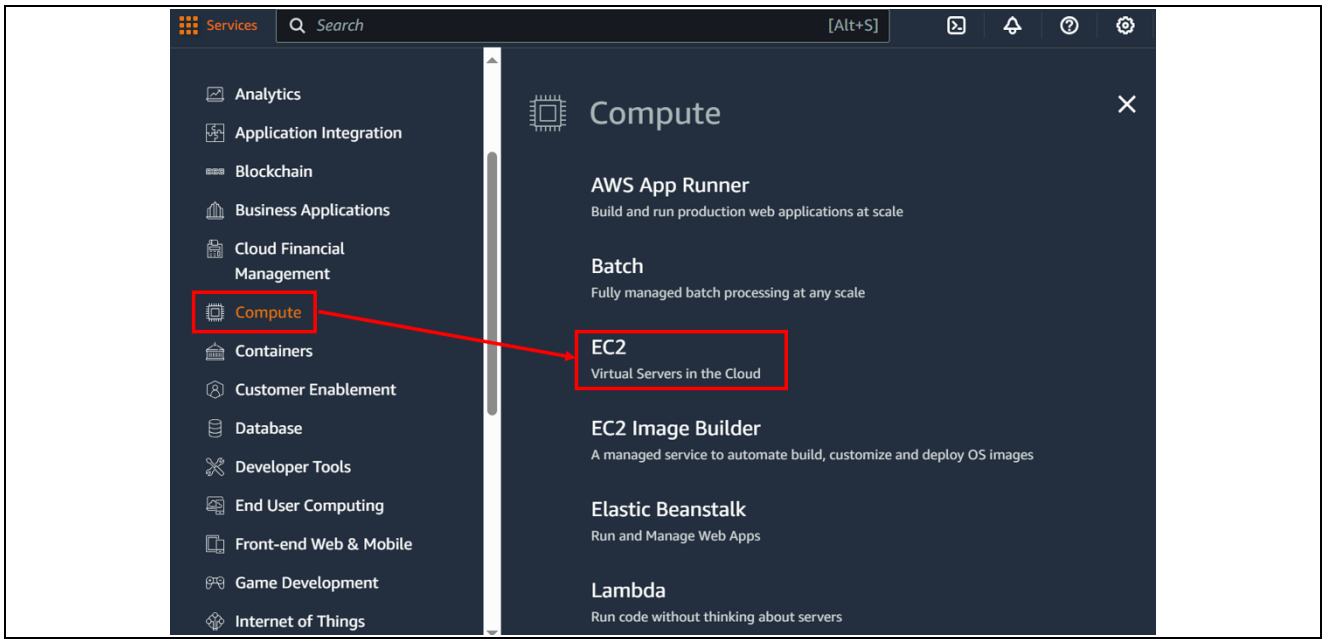

Figure 200. EC2 AWS Service

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

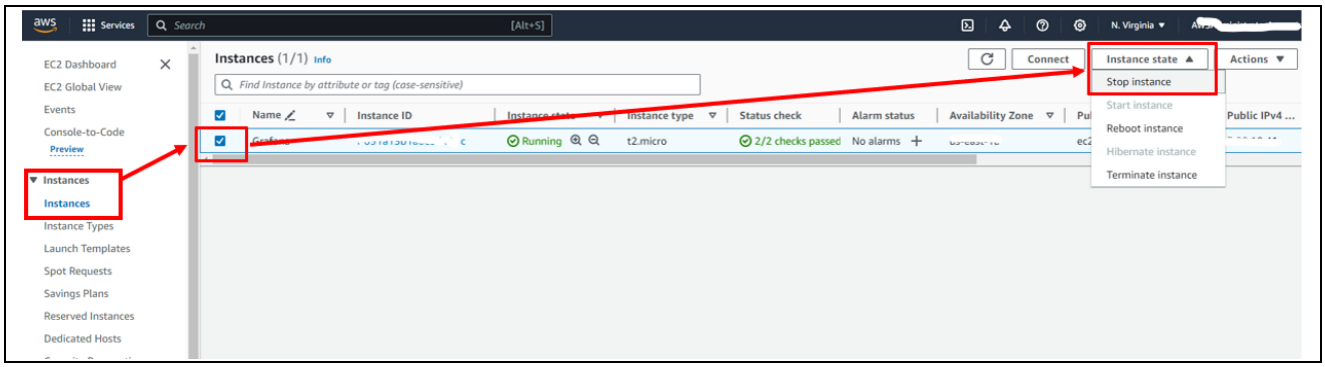

Figure 201. Disable Instance

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 202. Renesas AWS Cloud Dashboard Types

Choose Renesas CK v2.0.

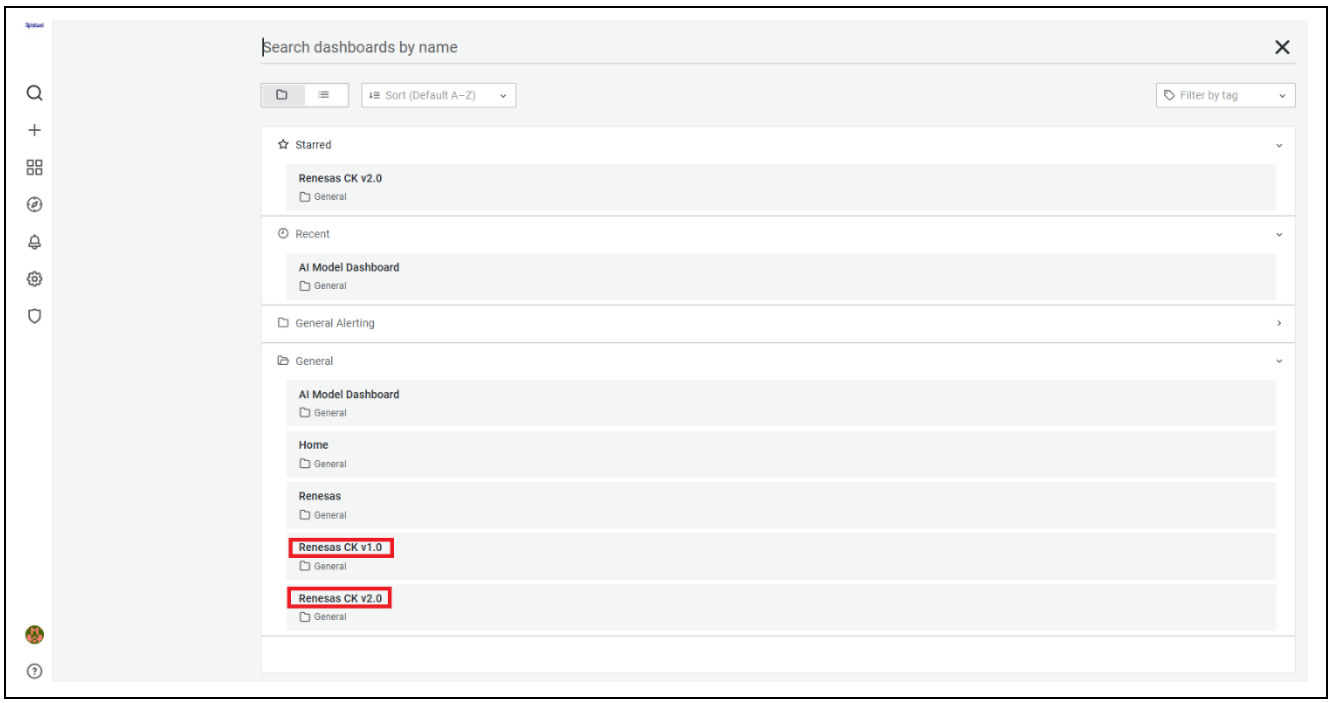

Figure 203. Choosing Renesas dashboards based on the cloud kit version

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.

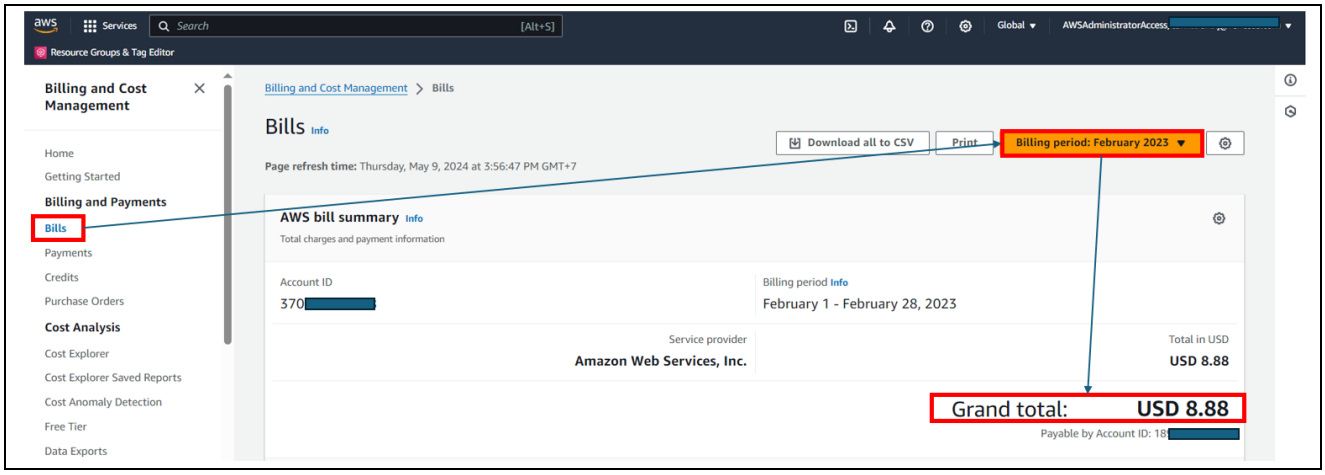

Figure 204. Check the amount spent in AWS Account

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 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 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 boot loader (OTA)

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

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 setting 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	Oct.20.23	—	Initial release
1.10	Dec.10.23	—	Updated e^2 studio version (2023-10) and FreeRTOS version (V202210.01)
		5 – 8	Table 5: Updated file structures following LTS#2
		26, 27	5.1.4: Updated starting application flow
		52, 53	Updated note and trouble shooting in building, running and using up 10 USD
		56, 57	Added note “6.7 How to enable/disable EC2 instance”
1.20	Apr.10.24	—	Added OTA feature (section 6), Fleet Provisioning feature (section 7)
1.21	Jul.21.25	17	Emphashize the word “unchecked” (for “copy projects into workspace” option) when importing project
		42, 51, 63, 92, 93	Update and fix issues in JSON format
		29, 81, 82, 88, 90	Update firmware version description to match source code
		—	Update for Dashboard descriptions

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/.