CK-RX65N

SIM activation, Creating the trial account and using Dashboard with RYZ014A or Ethernet Application for AWS - Getting Started Guide

Introduction

This document describes a system that uses the CK-RX65N V1 Cloud Kit board from Renesas. This system incorporates the CK-RX65N V1 running Amazon FreeRTOS and via Ethernet/Cellular connection. It visualizes HS3001, ZMOD4410, ZMOD4510, OB1203, ICP10101 and ICM20948 sensor information on Amazon Web Services (AWS) and controls LEDs on the board. In addition, this application note also describes several feature options for users when using CK-RX65N V1 Cloud Kit with AWS: OTA (Over-The-Air) feature **(section 6)** and Fleet Provisioning feature **(section 7).**

This document shows two methods of connectivity for CK-RX65N V1. The first one is the Ethernet, and the second is the Cellular CAT M1 using RYZ014A.

In addition, this document describes the following:

- How to activate the SIM card that is contained with the CK-RX65N V1.
- How to create the 10 USD free trial account for AWS.
- How to operate and install the information of 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.

Note: Renesas announced the discontinuation of the Sequans-sourced LTE module, part number RYZ014A, and will no longer be shipping this product. If you have this in a current design or production, the Sequans part numbers, GM01Q is a pin and functionally compatible replacement for RYZ014A. Below is the cellular driver alternate product.
- RYZ014A Cellular control module: Sequans GM01Q is the compatible module.
Regarding EOL notice of the RYZ014A, please see:
[The link] https://www.renesas.com/document/eln/plc-240004-end-life-eol-process-select-part-numbers
[The product page] https://www.renesas.com/products/wireless-connectivity/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module

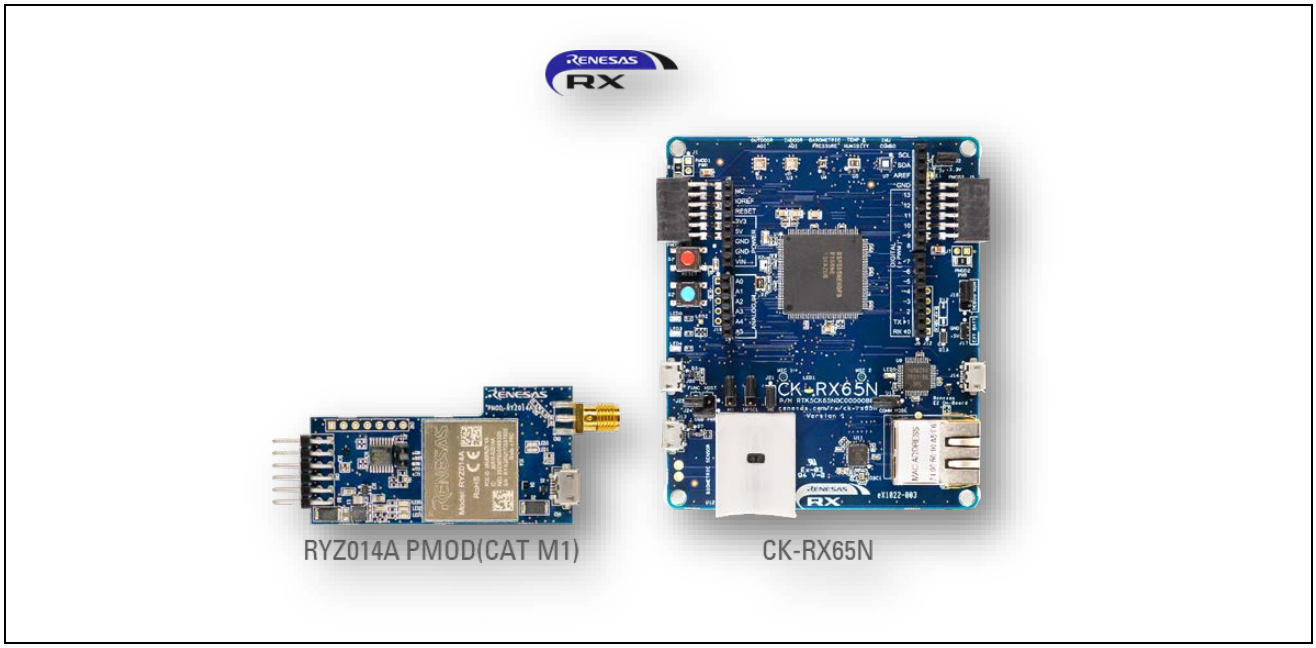

Figure 1. CK-RX65N V1 (with RYZ014A Pmod)

Contents

1. Terms

Terms used in this document are explained below.

Table 1. Terms

Term	Meaning
AWS	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. Preparation

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 V1 Cloud Kit	Target board for CK-RX65N V1	Please see detail at: https://www.renesas.com/rx/ck-rx65n
RYZ014A Cellular Pmod module	SIM card	This Pmod is contained with CK-RX65N V1 kit with SIM card
PC	Windows® 10 Google Chrome / Microsoft Edge	Recommended OS. Web browser used.

2.2 Software Configuration

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

Table 3. Software Configuration

Item	Content	Version
Integrated development environment	e2 studio (e² studio | Renesas)	2024-01
Compiler	CC-RX (CC-RX Compiler)	V3.05
Communication Software	Tera term (Tera Term - Download (softonic.com))	Version 4.99
Emulator	E2 emulator Lite (on-board)	-
RTOS	AWS FreeRTOS	V202210.01
Python	(Please see detail at: **6.2.1**)	V3.11.0 or later
Keygen tool	Win64 OpenSSL (Please see detail at: **6.2.2**)	V3.0.12
Flash programming tool	Renesas Flash Programmer (Renesas Flash Programmer (Programming GUI) | Renesas)	V3.12.00
Renesas Image Generator	Supplied with Firmware Update module Rev.2.01 (Please see detail at: **6.2.3**)	V3.02

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

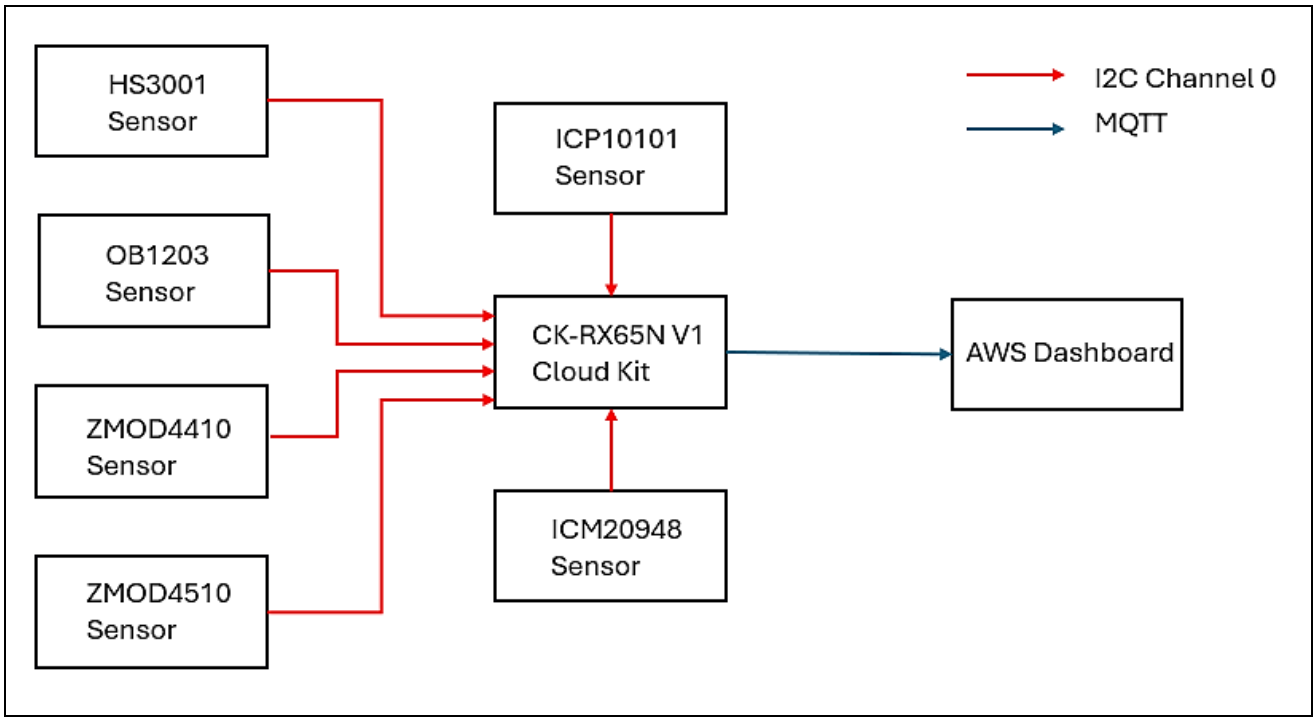

Figure 2. System Diagram

Note: The 9-AXIS MEMS Motion Tracking Sensor TDK ICM-20948 is not fitted on this CK-RX65N board due to shortages in component availability (For more information: CK-RX65N v1 – Release Note (renesas.com))

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 Ethernet/Cellular. It illustrates how the cloud service provider is configured and operated.

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

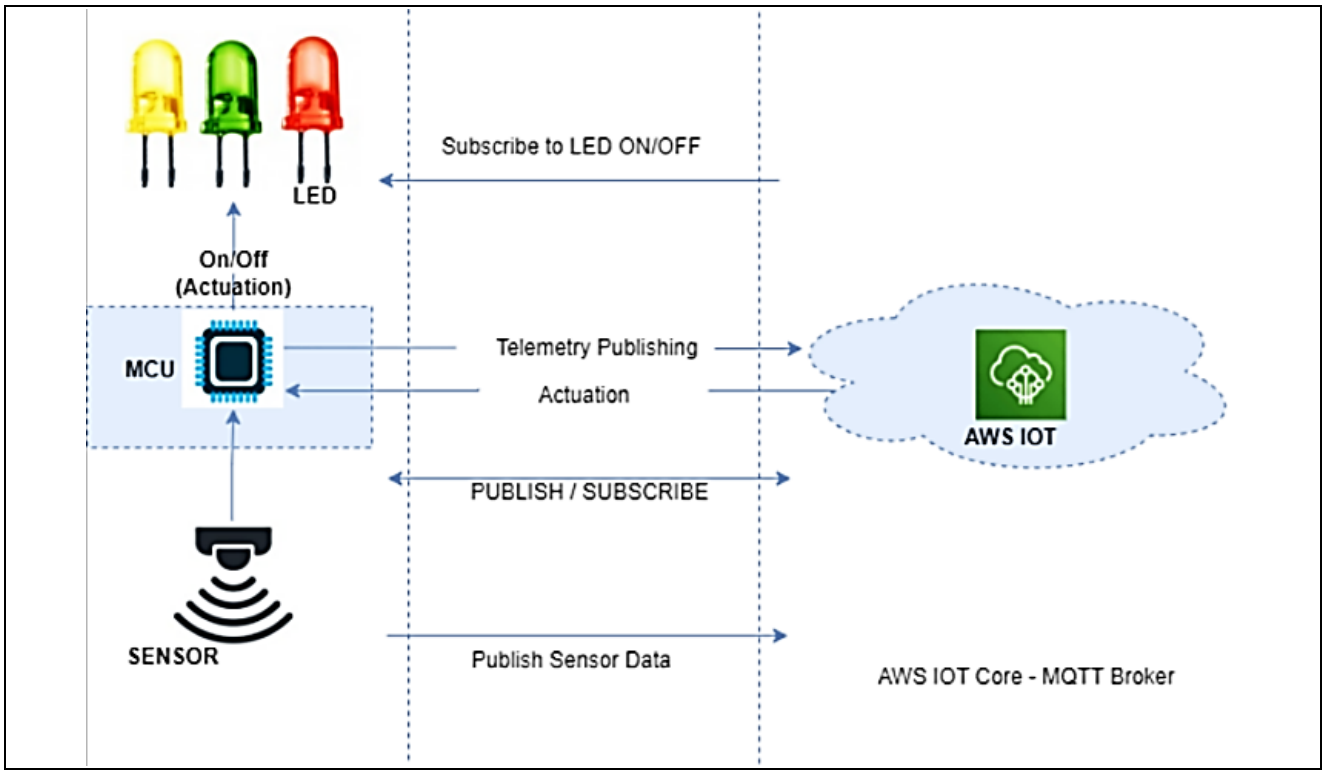

Figure 3. MQTT Publish/Subscribe to/from AWS IoT Core

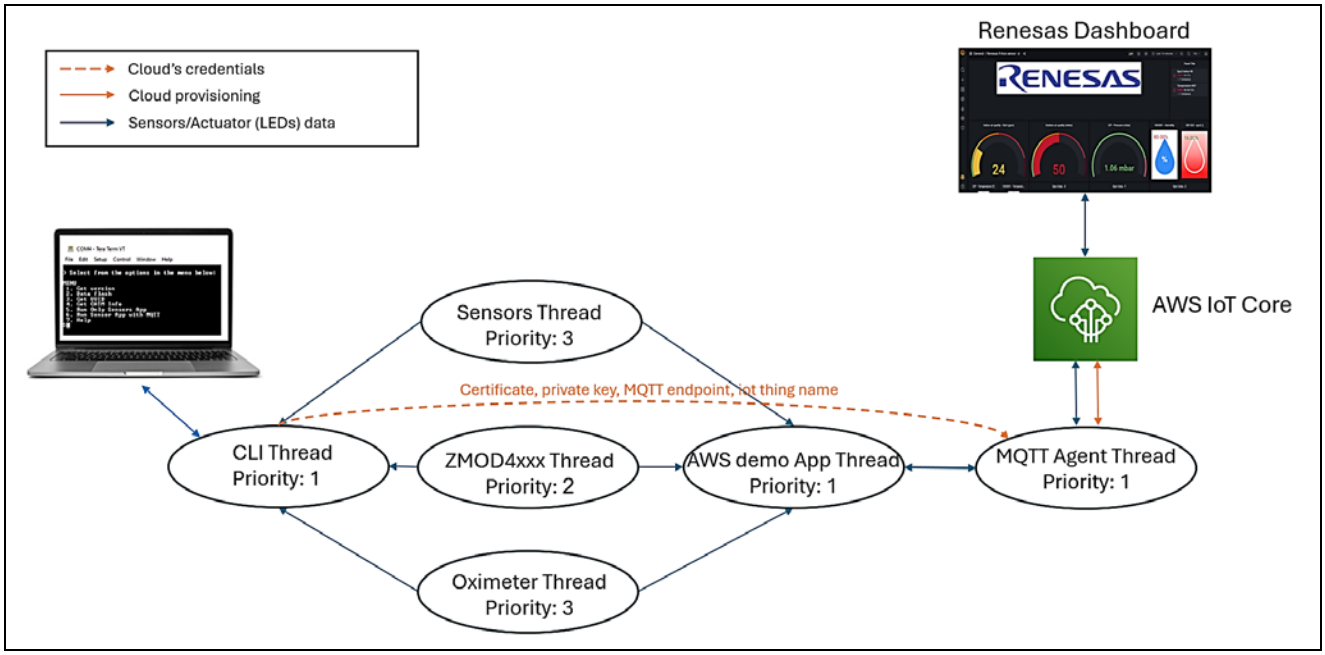

Figure 4. Thread Diagram

Application also supports:

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

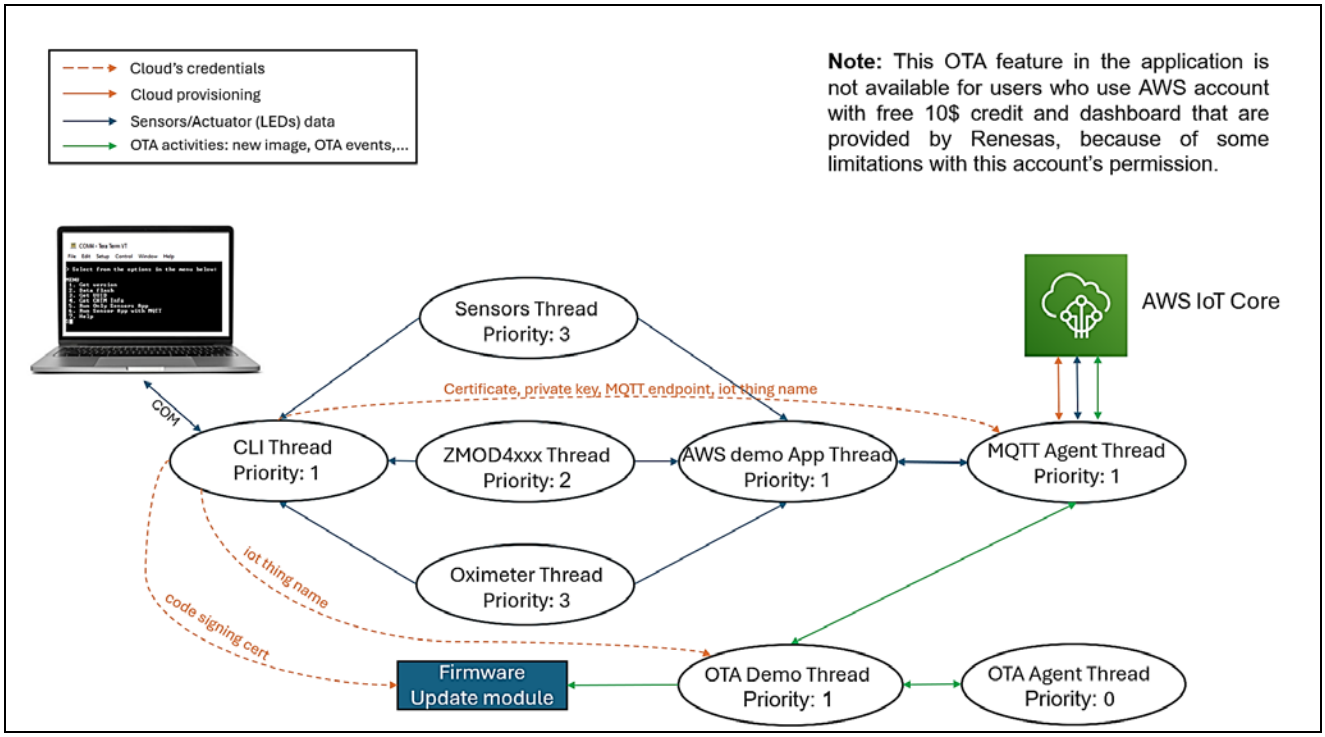

Figure 5. Thread Diagram when enabling OTA feature

- Fleet Provisioning (please refer to the section **7. Fleet Provisioning**):

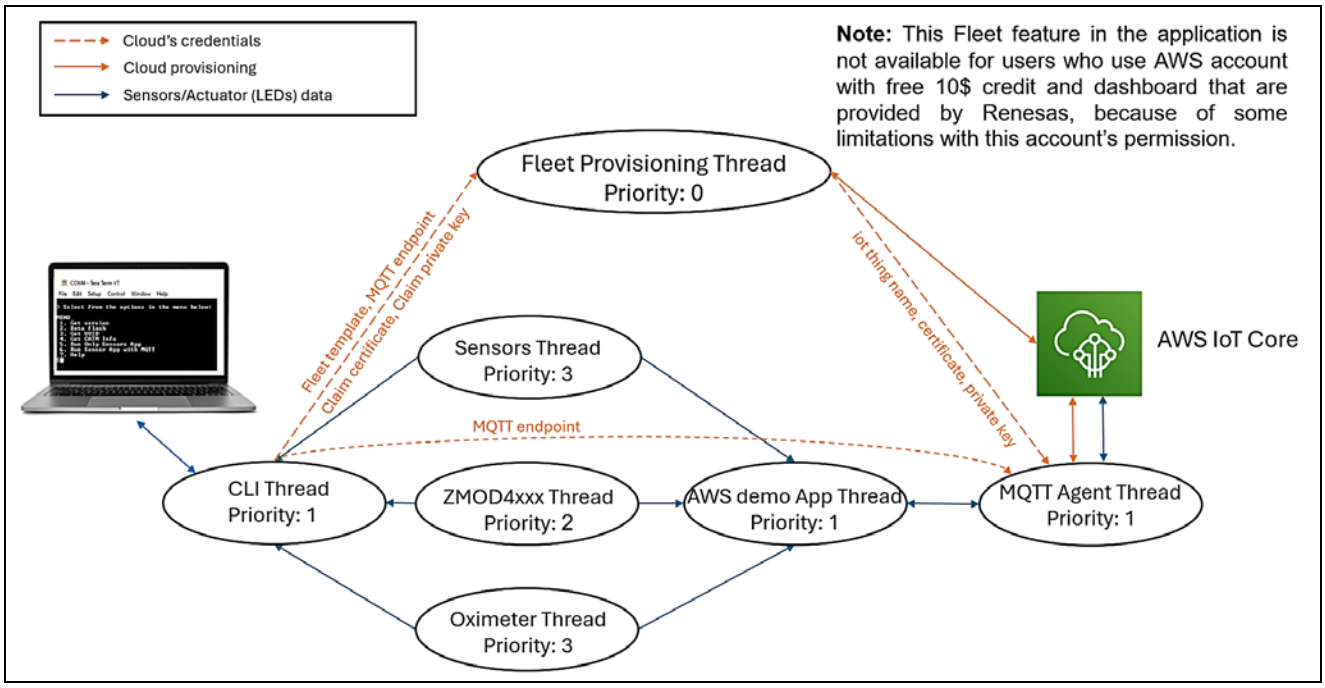

Figure 6. Thread Diagram when enabling Fleet feature

4.2 MQTT/TLS Application Software Overview

The following files from these application projects 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 connection, provisioning cloud credentials used in Cloud Connectivity, main function of application.
2.	`src/application_code/user_init.c`	Contains initialization functions.
3.	`src/application_code/CommandLine/cli_thread_entry.c`	Contains data structures and functions used in CLI thread.
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 typedefs used in application
15.	`src/application_code/sensor_thread_entry.c`	Contains the code for sensor thread (HS3001, ICP10101 and ICM20948)
16.	`src/application_code/ICM20948/icm20948.c`	Contains the code for the 9-Axis MEMS Motion Tracking™ Sensor
17.	`src/application_code/ICM20948/icm20948.h`	Contains the Data structure function prototypes for the 9-Axis MEMS Motion Tracking™ Sensor
18.	`src/application_code/ICM20948/icm_i2c.c`	Contains the I2C code to communicate with 9-Axis MEMS Motion Tracking™ Sensor
19.	`src/application_code/ICM20948/icm_i2c.h`	Contains the I2C function prototypes to communicate with 9-Axis MEMS Motion Tracking™ Sensor
20.	`src/application_code/ICP10101/ICP_20100.c`	Contains the code for Barometric Pressure and Temperature Sensor

No.	Filename	Purpose
21.	src/application_code/ICP10101/ICP_20100.h	Contains the data structure and function prototypes for Barometric Pressure and Temperature Sensor
22.	src/application_code/OB1203/RX_OB1203.c	Contains data structures and functions used for the oximeter sensor
23.	src/application_code/OB1203/ob1203_bio.c	Contains the Data structure for the oximeter sensor
24.	src/application_code/OB1203/ob1203_bio_rx.c	Contains data structures and functions used for the oximeter sensor
25.	src/application_code/OB1203/ob1203_bio.h	Contains the Data structure and function prototypes for the oximeter sensor
26.	src/application_code/OB1203/KALMAN/kalman.c	Contains algorithm for Heart Rate, Blood Oxygen Concentration, Pulse Oximetry, Proximity, Light and Color Sensor sample calculations
27.	src/application_code/OB1203/KALMAN/kalman.h	
28.	src/application_code/OB1203/SAVGOL/SAVGOL.c	
29.	src/application_code/OB1203/SAVGOL/SAVGOL.h	
30.	src/application_code/OB1203/SPO2/SPO2.c	
31.	src/application_code/OB1203/SPO2/SPO2.c	
32.	src/application_code/HS3001/RX_HS3001.c	Contains the code and function for Renesas Relative Humidity and Temperature Sensor.
33.	src/application_code/HS3001/RX_HS3001.h	Contains the common data structure's function prototypes for the Renesas Relative Humidity and Temperature sensors.
34.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_Common.c	Contains the common code for the Renesas ZMOD sensors
35.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_Common.h	Contains the common data structure's function prototypes for the Renesas ZMOD sensors
36.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_IAQ1stGen.c	Contains the common code for the Renesas ZMOD Internal Air Quality sensors
37.	src/application_code/ZMOD4x10/RX_ZMOD4XXX_OAQ1stGen.c	Contains the common code for the Renesas ZMOD Outer Air Quality sensors
38.	src/application_code/frtos_skeleton/ob1203_thread.c	Contains the OB1203 sensor thread (for oximeter sensor)
39.	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 9-Axis MEMS Motion Tracking™ Sensor)
40.	src/application_code/frtos_skeleton/zmod_thread.c	Contains the ZMOD's thread (for Renesas ZMOD Internal Air Quality sensors)
41.	src/application_code/frtos_skeleton/task_function.h	Contains the common data structure's function prototypes for thread
42.	src/application_code/frtos_startup/freertos_object_init.c	Contains the source code for FreeRTOS thread
43.	src/application_code/frtos_startup/freertos_start.c	Contains FreeRTOS user-defined functions

No.	Filename	Purpose
44.	`src/application_code/frtos_startup/freertos_start.h`	FreeRTOS's user-defined functions header file
45.	`src/application_code/frtos_config/*.h`	Contains FreeRTOS configuration header file.
46.	`src/application_code/sensorsData.h`	Contains the common data structure's function prototypes for sensors
47.	`Demos/SimplePubSub/simple_pub_sub_task.c`	Contains code and functions used in MQTT interface for Cloud Connectivity.
48.	`Demos/mqtt_agent/mqtt_agent_task.c`	Contains the code for running the MQTT task
49.	`Demos/OtaOverMqtt/OtaOverMqttDemoExample.c`	Contains function for running OTA over MQTT
50.	`Demos/Fleet_Provisioning_With_CSR_Demo`	Contains function for running Fleet Provisioning
51.	`Demos/cli/serial.c`	Contains function for serial communication.
52.	`Demos/cli/serial.h`	Contains the common data structure's function prototypes for serial.c
53.	`Demos/include/*.h`	Contains the common data structure's function prototypes for demo function.

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

5. Connection to AWS

AWS account is necessary to connect CK-RX65N V1 Cloud Kit to AWS.

Note: Renesas provides 10 USD of AWS account credit to users who buy the CK-RX65N V1 and this 10 USD credit cannot be used for an existing account.

This document covers two ways of connecting to the AWS account:

Case 1: For 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 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 Users Using Their Own AWS Account** to use the account with the application.

5.1 Hardware Preparation and Import the Project

5.1.1 Hardware Preparation

- Connect micro-USB cables to debug port (J14 on the CK-RX65N V1 board)
- Connect micro-USB cables to serial port (J20 on the CK-RX65N V1 board)
- Connect Ethernet cable to the connector (J18) on the board.
- **Set the Jumper of J16 "Debug"**

Note**:** In case user runs the RYZ014A Cellular Application, please skip the Ethernet's setting and connect the RYZ014A Cellular Pmod module which is inserted SIM card to the Pmod 1.

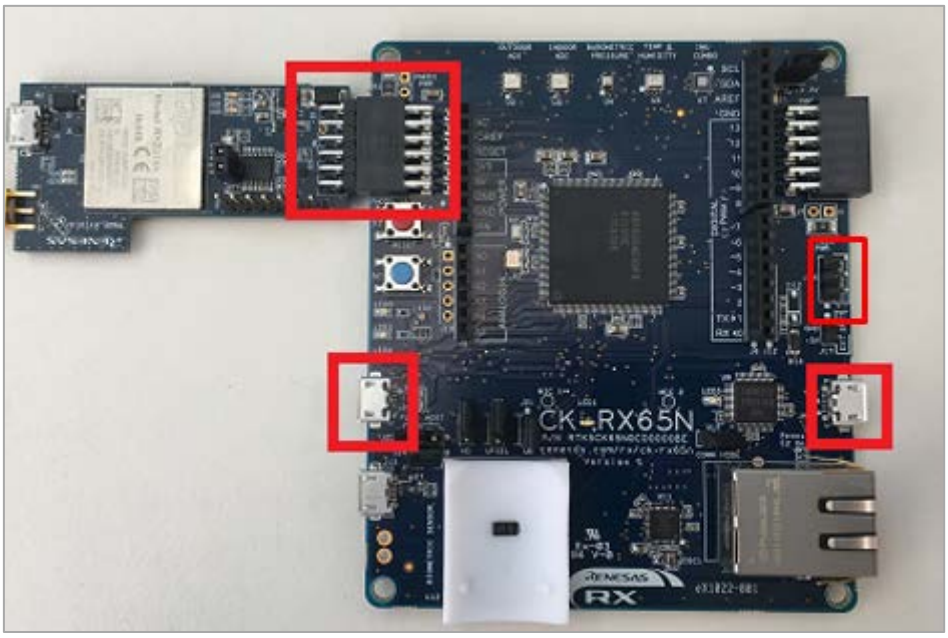

Figure 7. Connecting the USB and RYZ014A Pmod

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 Device (COMxx)** and note down the COM port number for reference in the next step.

Note: USB Serial Device drivers are required to communicate between the CK-RX65N V1 board and the PC.

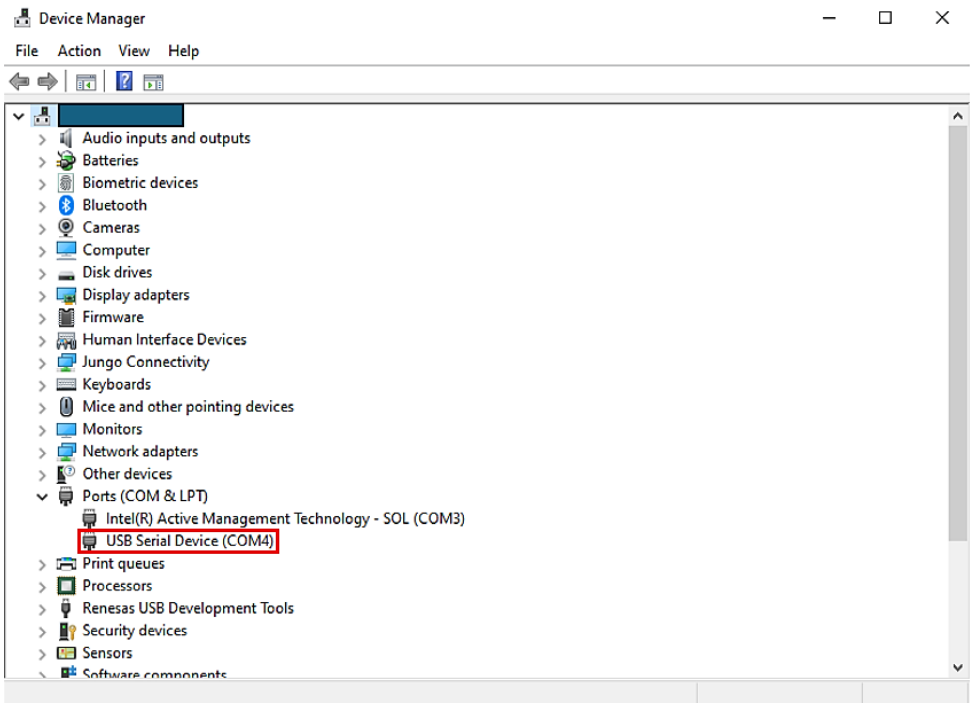

Figure 8. USB Serial Device in Windows Device Manager

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

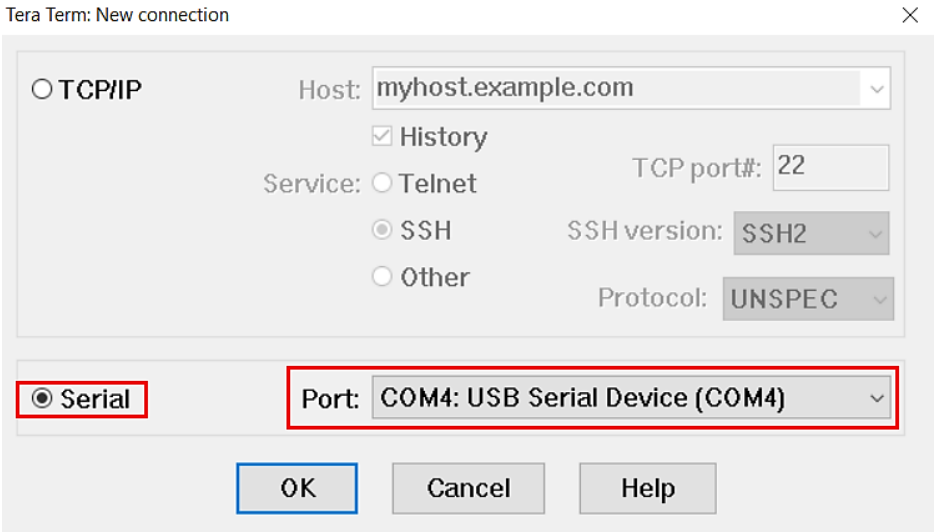

Figure 9. Selecting the Serial Port on Tera Term

3. Using the Setup menu, select **Setup** > **Terminal…** and select "**AUTO**" as Receive, select "**CR**" as Transmit, as shown below.

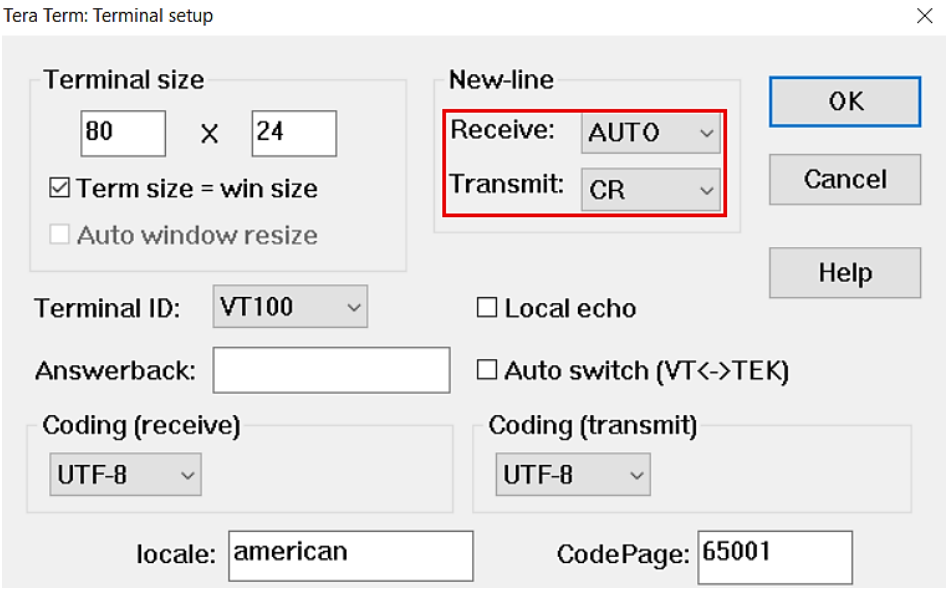

Figure 10. Select Receive: "Auto" and Transmit: "CR" on the Terminal Setting

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

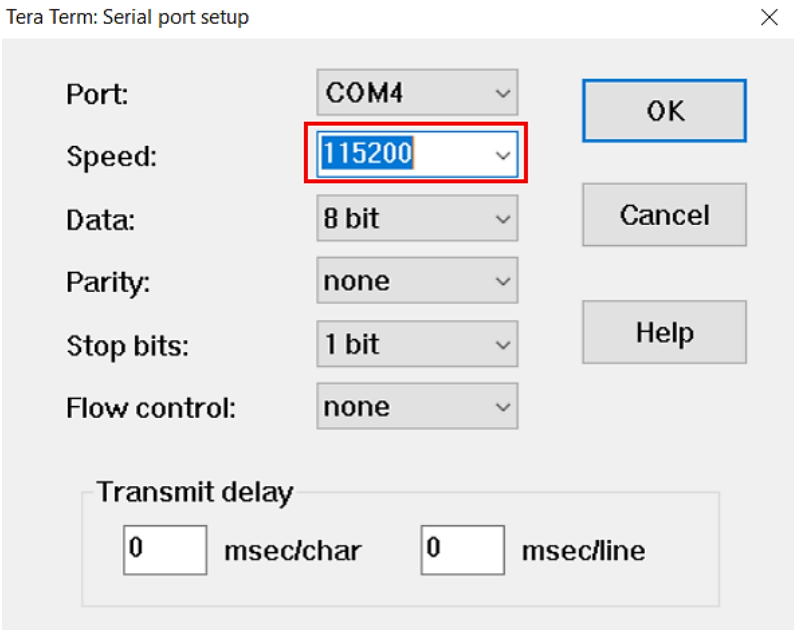

Figure 11. Select 115200 on the Speed Pulldown

5.1.3 Importing 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 **a short 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.**

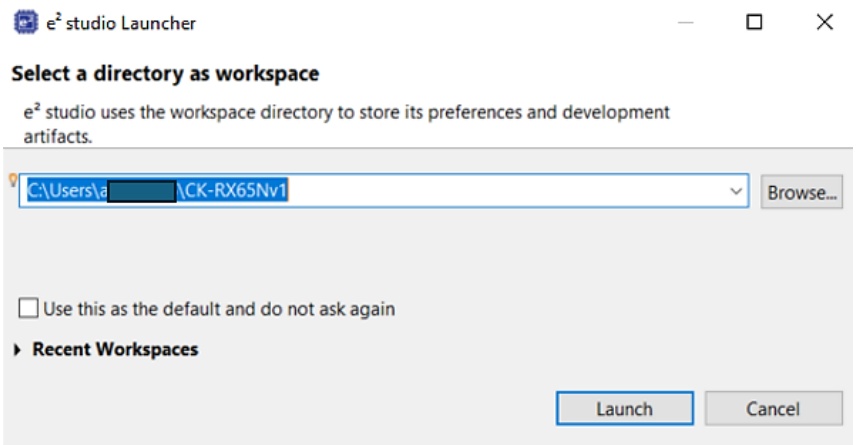

Figure 12. Launch e² studio

3. Select **File** > **Import**….

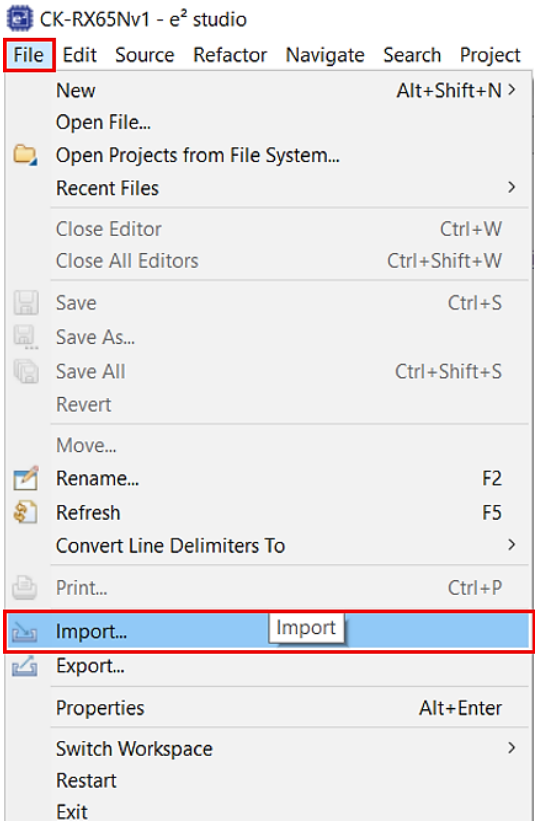

Figure 13. Select Import

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

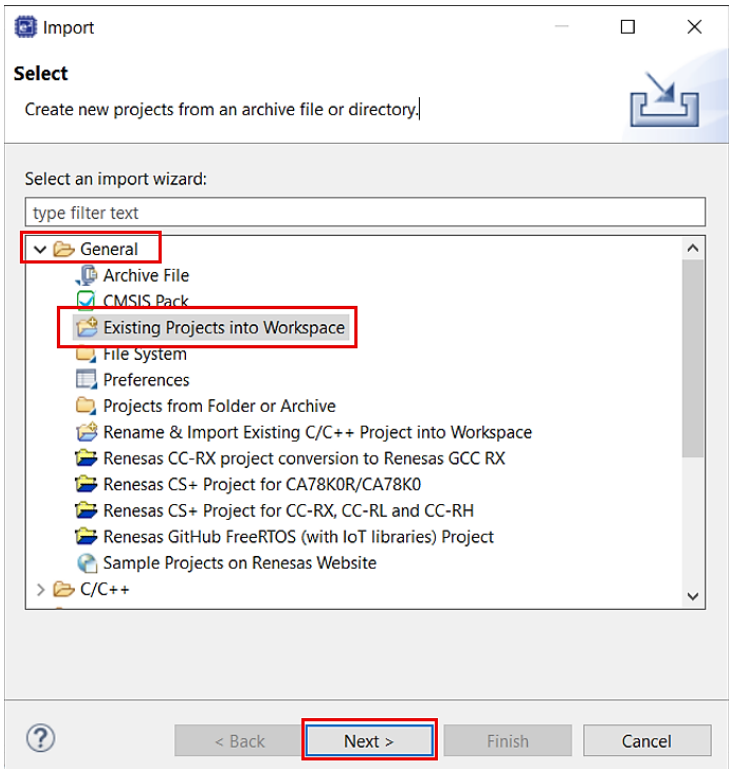

Figure 14. Select Existing Projects into Workspace

5. Click **Browse**…, then specify the root directory as described later in this section.

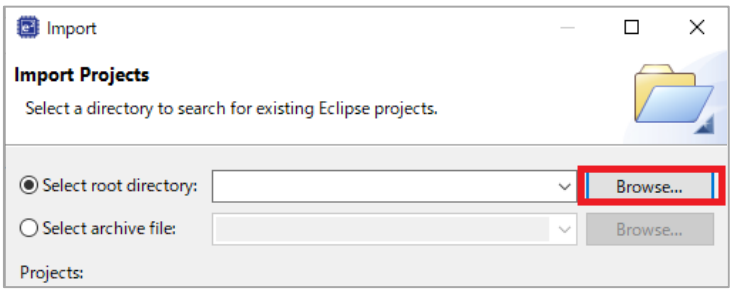

Figure 15. Find the Project

You can choose two types of connectivity when importing the project.
Please go to **"[Project Root folder]\Projects"** folder.

Table 6. Details of Each Project

Project Name	Compiler	Connectivity
aws_ether_ck_rx65n	CC-RX	Ethernet
aws_ryz014a_ck_rx65n		Cellular
boot_loader_ck_rx65n (for OTA)		-

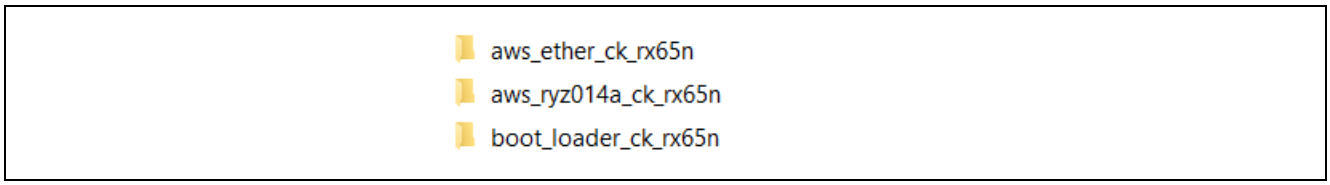

Figure 16. Project Files

This example uses "**aws_ether_ck_rx65n**" as the ethernet project.
Open the **"[Project Root folder]\Projects\aws_ether_ck_rx65n\e2studio_ccrx"** folder.

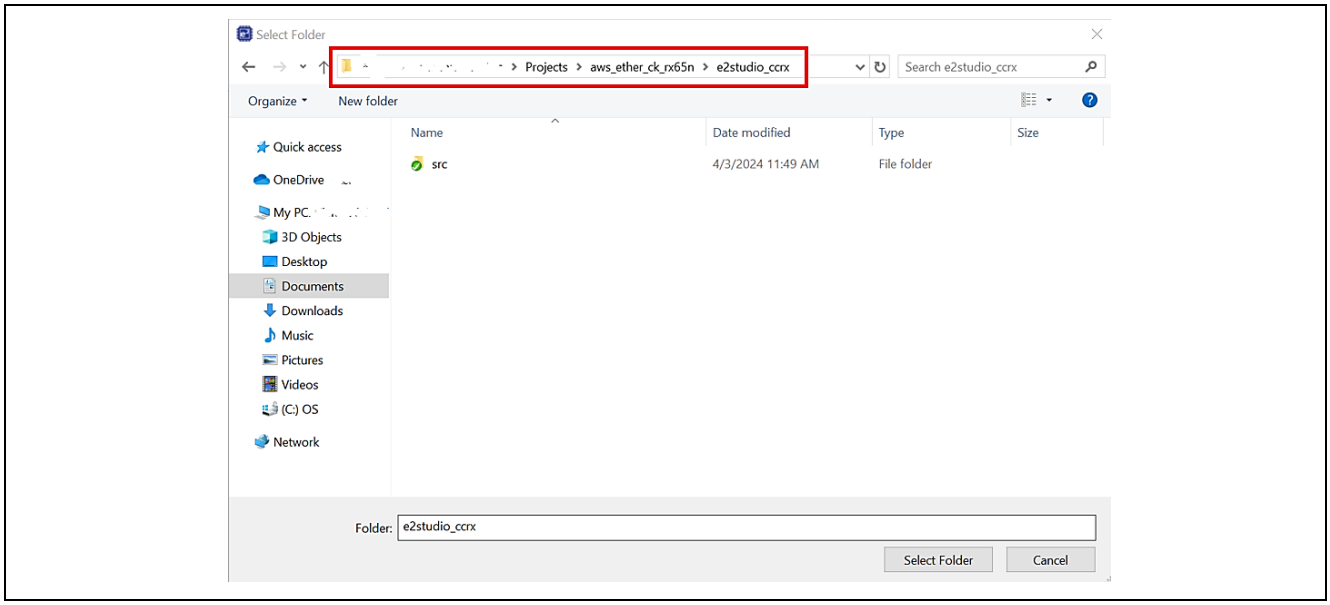

Figure 17. Select the Project Folder

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

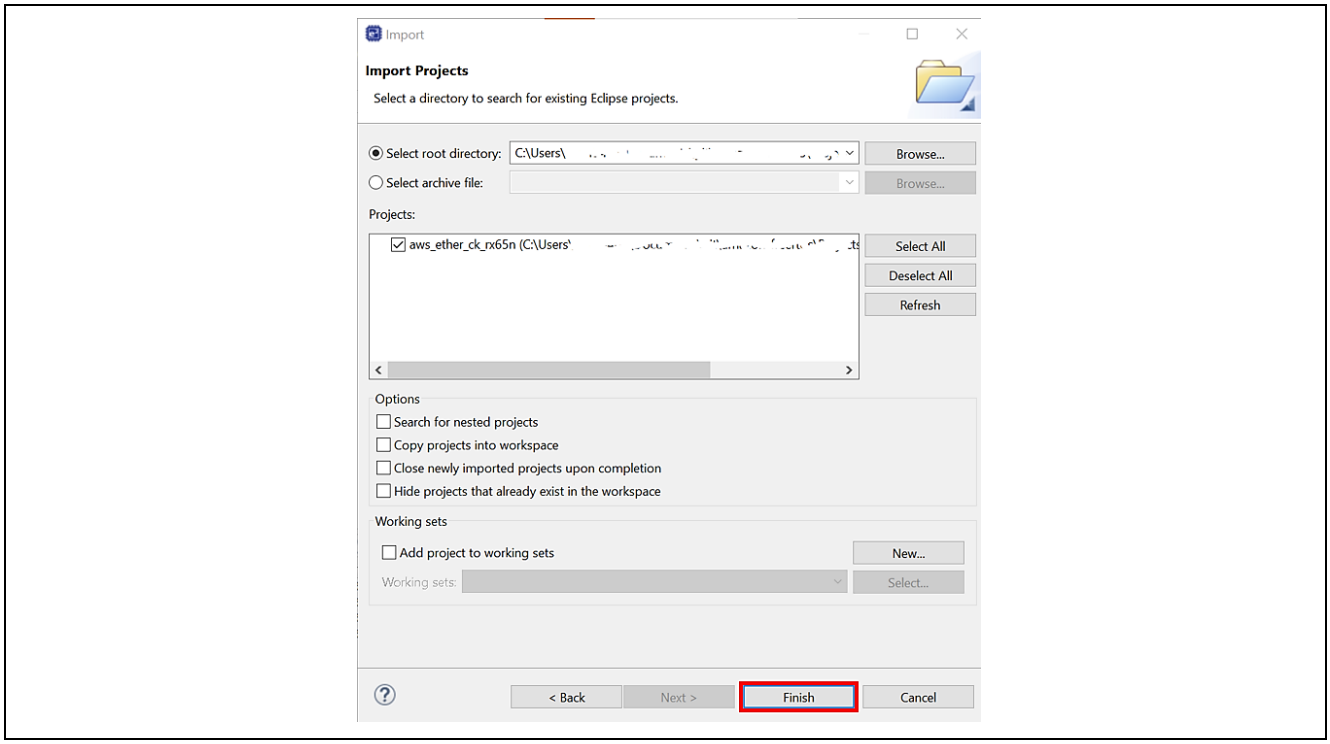

Figure 18. Finish Importing the Project

6. Check and set the SIM card information. **(In case of using cellular application)**
   Double click "**aws_ryz014a_ck_rx65n.scfg"** to open the smart configurator.

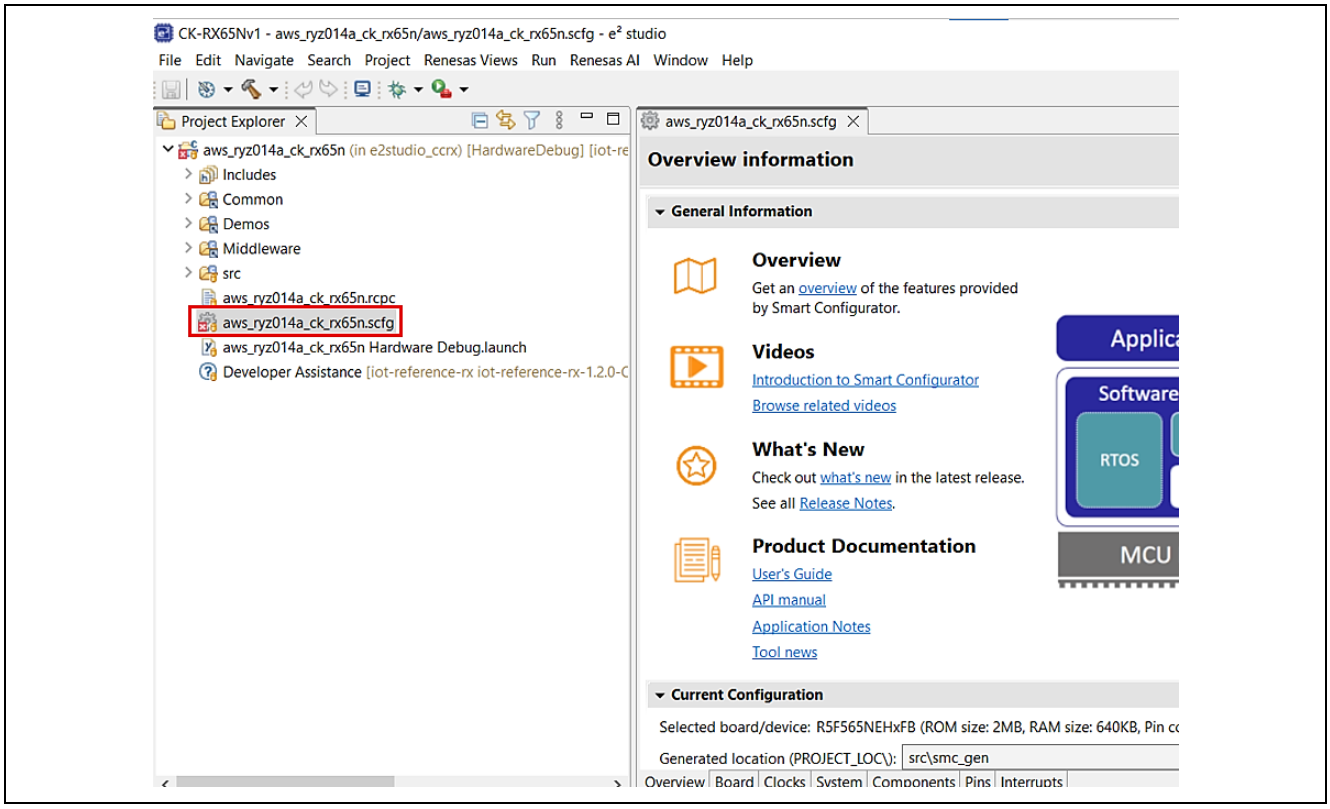

Figure 19. Open the Smart Configurator

If using Cellular application, please check following configurations for SIM card of cellular driver:

Table 7. Information for Setting SIM of Cellular Driver When Using Truphone SIM Card in the Kit

Items	Truphone
APN (Access point name)	iot.truphone.com
UserID (Access point login ID)	(Null)
Password (Access point password)	(Null)
SIM card PIN code	(Null)
Authentication protocol type	1 (PAP)

Note: Please use the configuration of the included SIM card.

Set the above configuration values when using the SIM contained in the kits.

If you are using a different SIM card, please set the value provided by the SIM provider.

Choose "**Components > Middleware > Generic > r_cellular**":

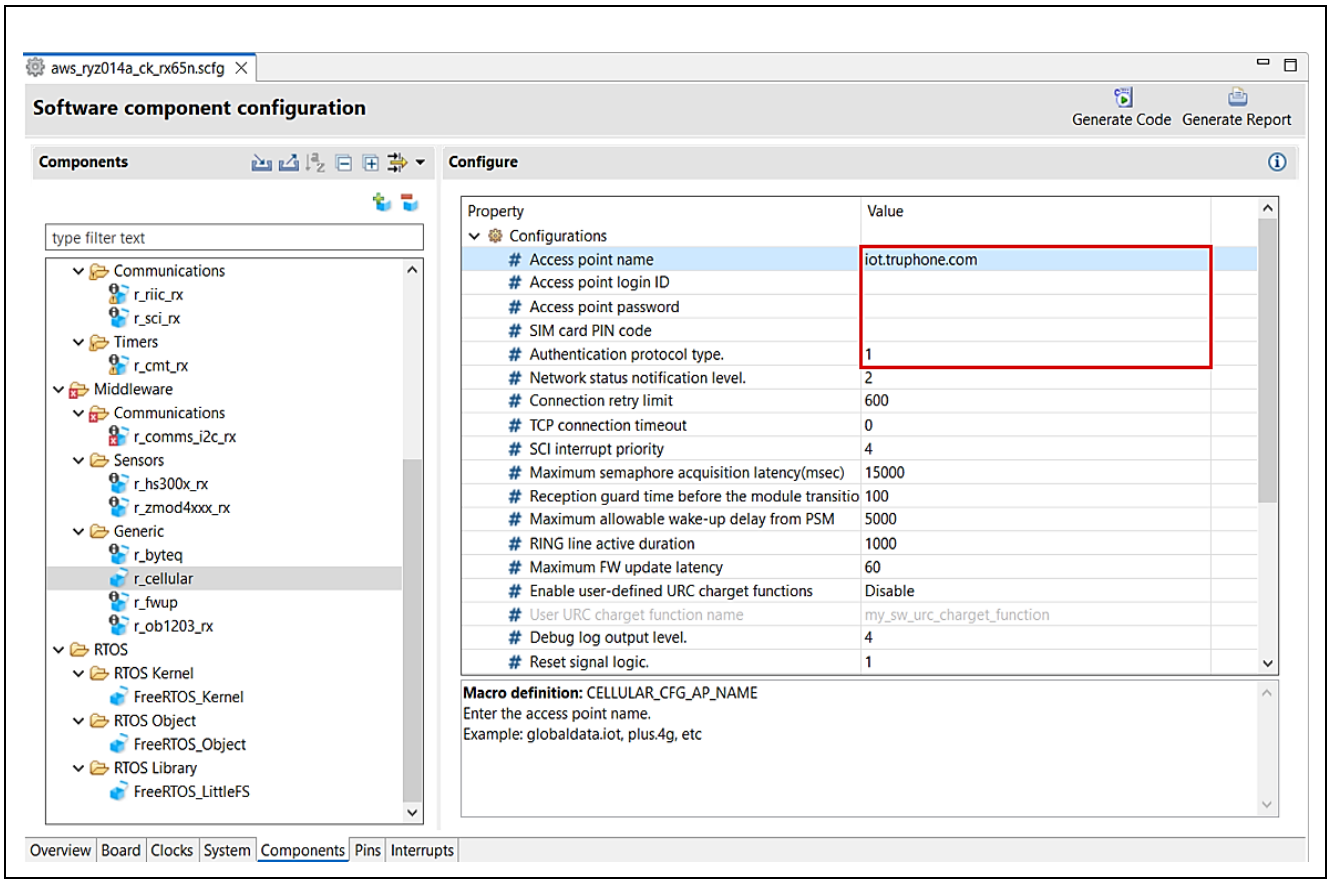

Figure 20. Set the SIM Information when Using Contained SIM Card in the Kit

7. Execute code generation.

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 that matches the application, please download it by choosing **aws_ether_ck_rx65n.scfg** > **Components** > **downloading it**

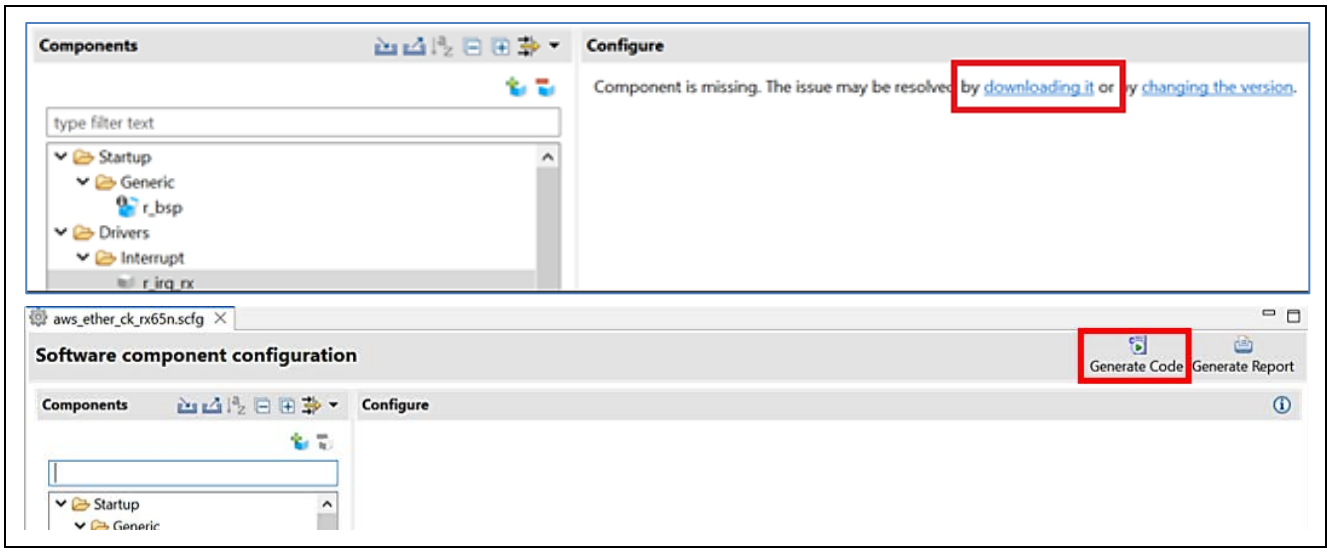

Figure 21. Generate Code

Table 8 shows the common configurations of each component in Projects.

Table 8. Components Configuration

No	Component	Configuration
1	**Startup→Generic→r_bsp (v7.20)**	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**
		Interrupt priority for serial terminal: **Priority level 15 (highest)**
2	**Drivers→Interrupt→r_irq_rx (v4.10)**	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 (v5.00)**	Resources→S12AD→S12AD1: ✓ →AN115 Pin: ✓ →AN117 Pin: ✓
4	**Drivers→Memory→r_flash_rx (v5.00)**	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**

No	Component	Configuration
5	**Drivers→Security→r_tsip_rx (v1.17.l)**	-
6	**Drivers→Communications→r_ether_rx (v1.23)** (For Ethernet project only)	Ethernet interface: **RMII (Reduced Media Independent Interface)**
		PHY-LSI address setting for ETHER0: **5**
		The register bus of PHY0 for ETHER0/: **Use ETHER0**
		The polarity of the link signal output by the PHY-LSI: **Rise -> Fall**
		Use ICS1894032 of the Renesas Electronics Corporation: **Used**
7	**Drivers→Communications→r_riic_rx (v2.50)**	MCU supported channels for CH0: **Supported**
		CH0 RIIC bps(kbps): **400**
		Resources→RIIC RIIC0: ✓ • SCL0 Pin: ✓ **Used** • SDA0 Pin: ✓ **Used**
8	**Drivers→Communications→r_sci_iic_rx (v2.50)**	MCU supported channels for CH0: **Supported**
9	**Drivers→Communications→r_sci_rx (v4.40)**	Include software support for channel 5: **Include**
		Include software support for channel 6: **Include** (For RYZ014A project only)
		ASYNC mode TX queue buffer size for channel 5: **80**
		ASYNC mode TX queue buffer size for channel 6: **2180** (For RYZ014A project only)
		ASYNC mode RX queue buffer size for channel 5: **80**
		ASYNC mode RX queue buffer size for channel 6: **8192** (For RYZ014A project only)
		Transmit end interrupt: **Enable** (For RYZ014A project only)
		GROUPBL0 (ERI, TEI) interrupt priority: **3**
		Resources→SCI →SCI5**:** ✓ • RXD5/SMISO5/SSCL5 Pin: ✓ **Used** • TXD5/SMOSI5/SSDA5 Pin: ✓ **Used** →SCI6: ✓ (For RYZ014A project only) • RXD6/SMISO6/SSCL6 Pin: ✓ **Used** • TXD6/SMOSI6/SSDA6 Pin: ✓ **Used** • CTS6#/RTS6#/SS6# Pin: ✓ **Used**
10	**Drivers→Timers→r_cmt_rx (v5.40)**	CMT interrupts priority level: **4**
11	**Middleware→ Communications→r_comms_i2c_rx (v1.21)**	Number of I2C Share Buses: **1**
		Number of I2C Communication Devices: **7**
		I2C Driver Type for I2C Shared Bus0: **RIIC** Channel No. for I2C Shared Bus0: **0**
		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**

No	Component	Configuration
		I2C Shared Bus No. for I2C Communication Device1: **I2C Shared Bus0** 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 Bus0** 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 Bus0** Slave address for I2C Communication Device3: **0x53** Callback function for I2C Communication Device3: **rm_ob1203_callback0**
		I2C Shared Bus No. for I2C Communication Device4: **I2C Shared Bus0** 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**
12	**Middleware→Sensors→r_hs300x_rx (v1.22)**	Number of HS300x Sensors: **1**
		Data types from HS300x Sensor: **Humidity and Temperature**
		I2C Communication device No. for HS300x sensor device0: **I2C Communication Device0**
		Callback function for HS300x sensor device0: **hs300x_callback**
13	**Middleware→Sensors→r_zmod4xxx_rx (v1.20)**	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.**

No	Component	Configuration
		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**
14	**Middleware→Generic→r_byteq (v2.00)**	Memory allocation for queue control blocks: **Static memory allocation**
		Number of static queue control block: **32**
15	**Middleware→Generic→r_cellular (v1.11)** (For RYZ014A project only)	Access point name: **iot.truphone.com** Note**:** Depending on SIM Card information
		Access point login ID: - Note**:** Depending on SIM Card information
		Access point password: **-** Note**:** Depending on SIM Card information
		SIM card PIN code: **-** Note**:** Depending on SIM Card information
		Authentication protocol type: **1**
		Network status notification level: **2**
		Reset signal logic: **1**
		SCI Channel: **6**
		UART hardware flow control: **CTS(Hardware), RTS(Software)**
		RTS port number: **PORT0**
		RTS pin number: **BIT2**
		Reset port number: **PORT5**
		Reset pin number: **BIT5**
		IRQ Number: **4**
16	**Middleware→Generic→r_fwup (v2.01)**	Select the update mode: **Dual bank**
		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 (v1.01)**	Number of OB1203 Sensors: **2**
		Sensor mode of OB1203 Sensor device0: **Proximity sensor mode**
		I2C Communication device No. for OB1203 sensor device0: **I2C Communication Device 3**
		I2C callback function for OB1203 sensor device0: 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**

<table>
<tr><th>No</th><th>Component</th><th>Configuration</th></tr>
<tr><td></td><td></td><td>I2C Communication device No. for OB1203 sensor device1: I2C Communication Device 4</td></tr>
<tr><td></td><td></td><td>I2C callback function for OB1203 sensor device1: ob1203_comms_i2c_callback</td></tr>
<tr><td></td><td></td><td>Enable IRQ from OB1203 sensor device1: Enabled</td></tr>
<tr><td></td><td></td><td>IRQ callback function for OB1203 sensor device1: ob1203_irq_callback</td></tr>
<tr><td></td><td></td><td>IRQ number for OB1203 sensor device1: IRQ15</td></tr>
<tr><td></td><td></td><td>IRQ trigger for OB1203 sensor device1: Falling</td></tr>
<tr><td></td><td></td><td>IRQ interrupt priority for OB1203 sensor device1: Priority 14</td></tr>
<tr><td>18</td><td>RTOS→RTOS Kernel→FreeRTOS_Kernel (V202210.01)</td><td>RTOS scheduler: Preemptive</td></tr>
<tr><td></td><td></td><td>Maximum number of priorities to the application task: 7</td></tr>
<tr><td></td><td></td><td>The frequency of the RTOS tick interrupt: (TickType_t) 1000</td></tr>
<tr><td></td><td></td><td>The size of the stack used by the idle task: 768</td></tr>
<tr><td></td><td></td><td>The configTOTAL_HEAP_SIZE_N: 256</td></tr>
<tr><td></td><td></td><td>The maximum permissible length of name: 12</td></tr>
<tr><td></td><td></td><td>Idle should yield: ✓ Enabled</td></tr>
<tr><td></td><td></td><td>Mutex functionality: ✓ Enable</td></tr>
<tr><td></td><td></td><td>Counting semaphore functionality: ✓ Enable</td></tr>
<tr><td></td><td></td><td>Software timer functionality: ✓ Enable</td></tr>
<tr><td></td><td></td><td>Priority of the software timer task: 6</td></tr>
<tr><td></td><td></td><td>The length of the software timer command queue: 5</td></tr>
<tr><td></td><td></td><td>Kernel interrupt priority: 1</td></tr>
<tr><td></td><td></td><td>Maximum syscall interrupt priority: 4</td></tr>
<tr><td></td><td></td><td>Tick vector: _CMT0_CMI0</td></tr>
<tr><td></td><td></td><td>Record stack high address: ✓ Enable</td></tr>
<tr><td></td><td></td><td>Dynamic allocation: ✓ Enable</td></tr>
<tr><td></td><td></td><td>Static allocation: ✓ Enable</td></tr>
<tr><td>19</td><td>RTOS→RTOS Object→FreeRTOS_Object (V202210.01)</td><td>
<table>
<tr><th>Initialize</th><th>Task Code</th><th>Task Handler</th><th>Priority</th><th>Stack Size</th></tr>
<tr><td>kernel start</td><td>zmod_thread</td><td>handle_zmod_thread</td><td>2</td><td>1024</td></tr>
<tr><td>kernel start</td><td>ob1203_thread</td><td>handle_ob1203_thread</td><td>3</td><td>1024</td></tr>
<tr><td>kernel start</td><td>sensor_thread</td><td>handle_sensor_thread</td><td>3</td><td>2048</td></tr>
</table>
</td></tr>
<tr><td>20</td><td>RTOS→RTOS Library→FreeRTOS_LittleFS (V202210.01)</td><td>.block_count: 70</td></tr>
</table>

8. Data Publishing Interval Settings (Optional)

Data publish interval can be set by the user. The default publishing interval time is 2 seconds. “Demos/SimplePubSub/simple_pub_sub_task.c” file has the macro to change the publish time interval.

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

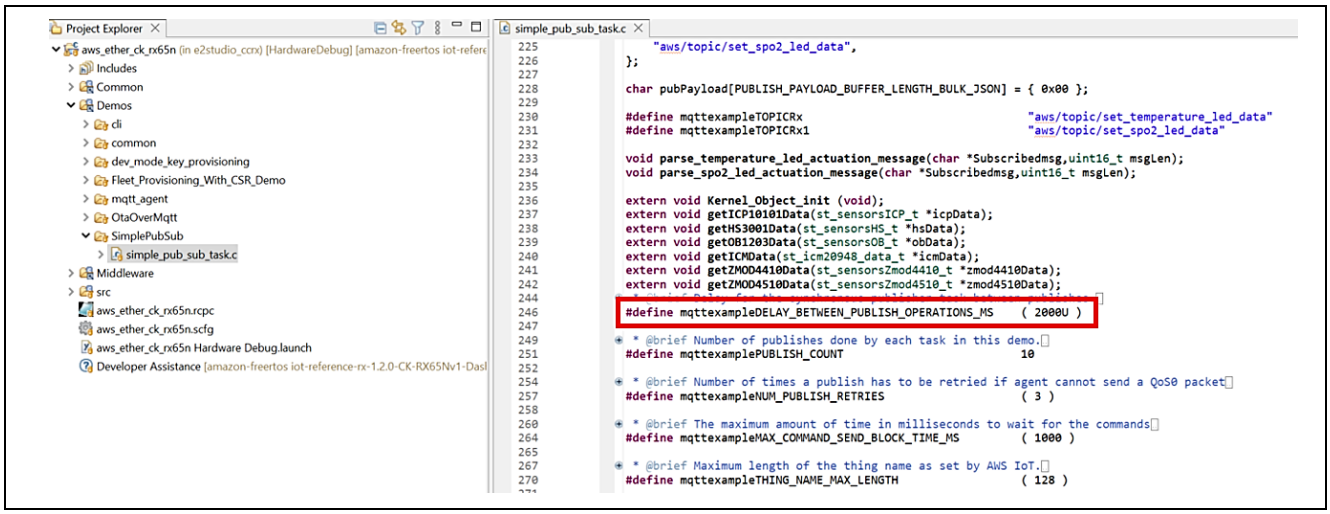

Figure 22. Data Publishing Interval Settings (Optional)

9. Select **Project** > **Build All** and confirm that 0 errors are reported.

Note: Check the compiler setting before building project:

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**

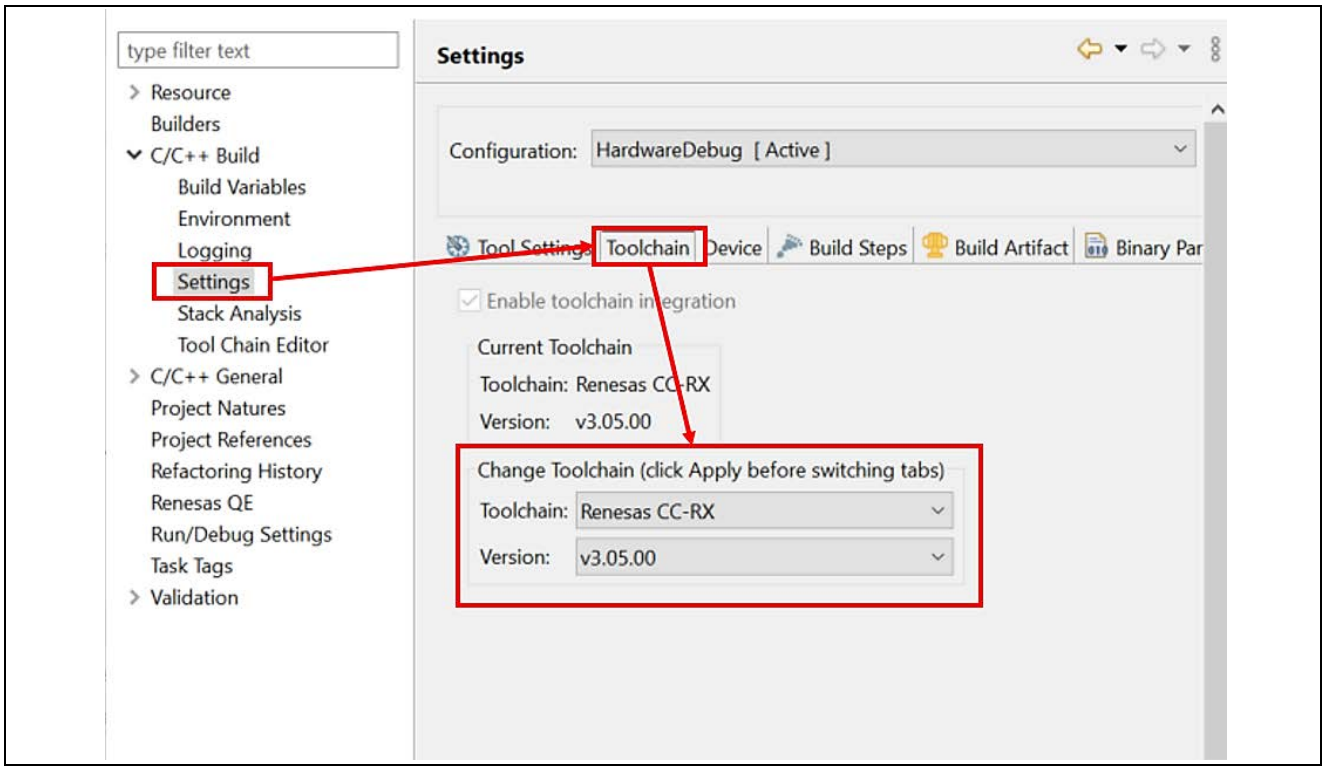

Figure 23. Project toolchain

Note: Make sure to clean the project before building it for the first time. If a demo build error occurs after the initial build, clean the project again and then rebuild it.

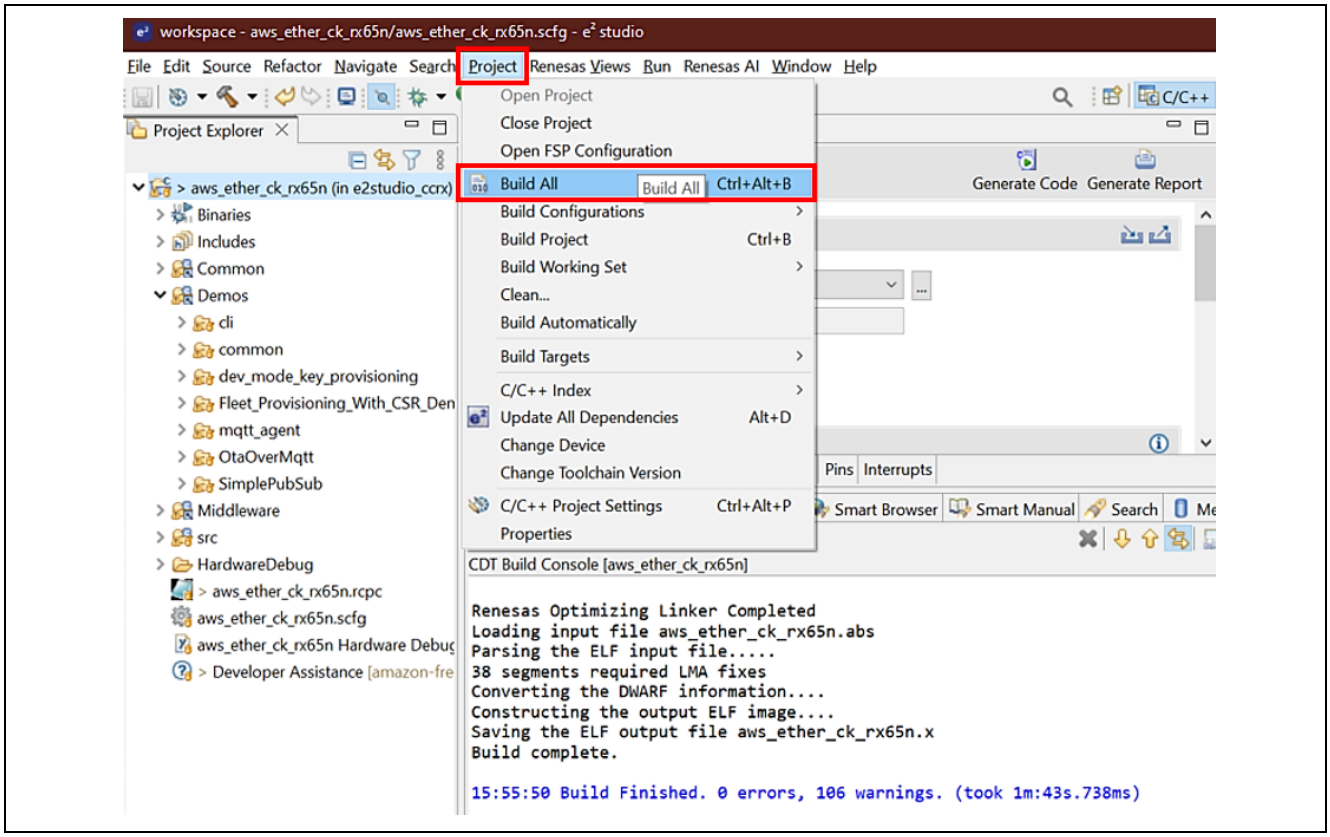

Figure 24. Build the Project

10. Debug Configuration

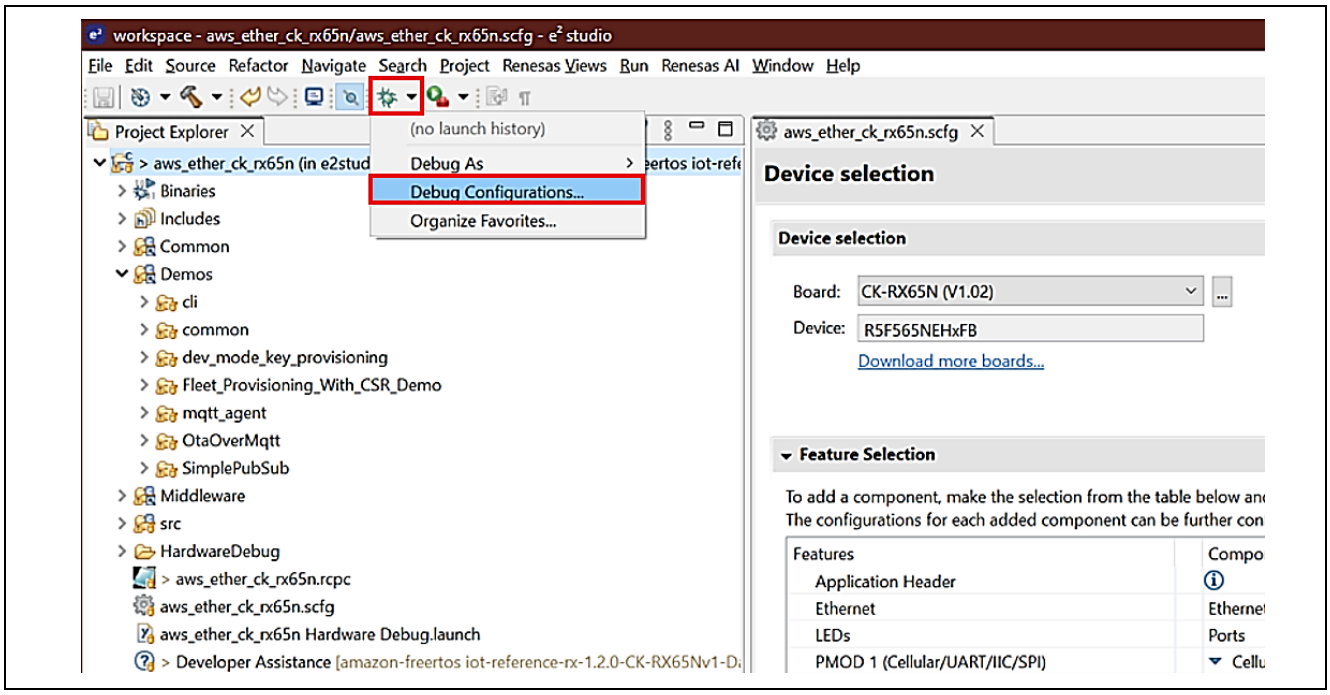

Figure 25. Configuration Debugger (1/2)

Go to **aws_ether_ck_rx65n HardwareDebug > Debugger > Connection Settings** tab then configure for "Main Clock Source: **EXTAL**" and "Connection Type: **Fine**".

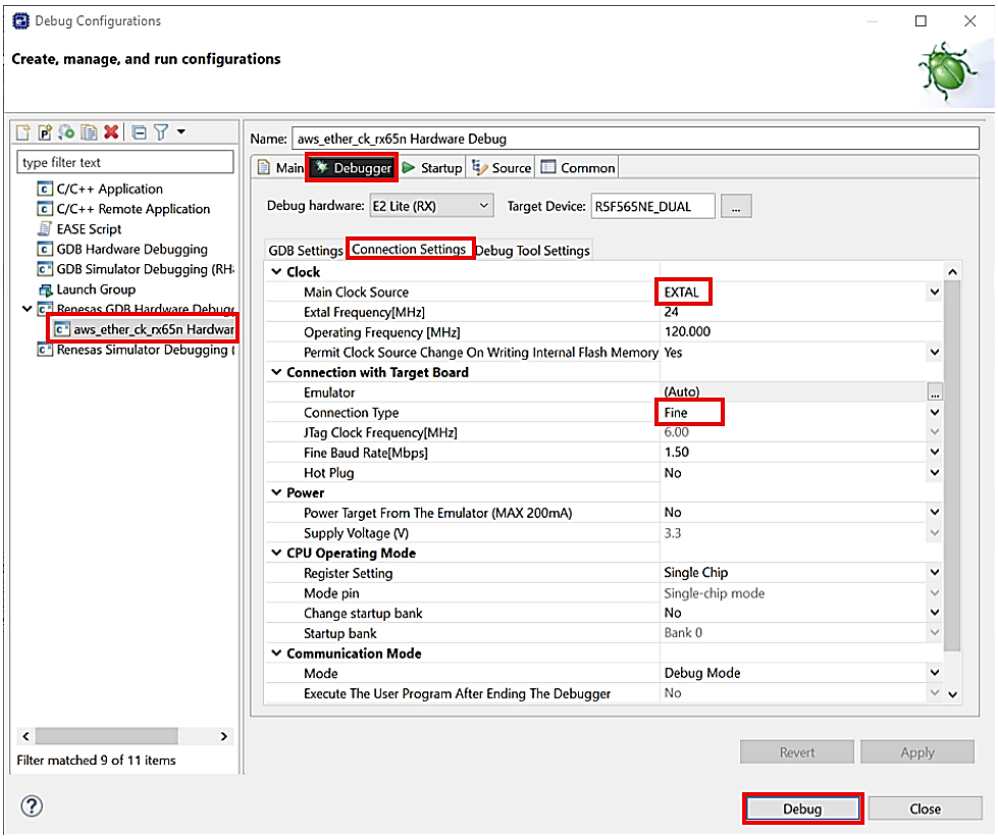

Figure 26. Configuration Debugger (2/2)

5.1.4 Running the Application Project

To run the Application project, use the following instructions.

The serial port console (Tera Term) of the PC set up in section **5.1.2** will display as below.

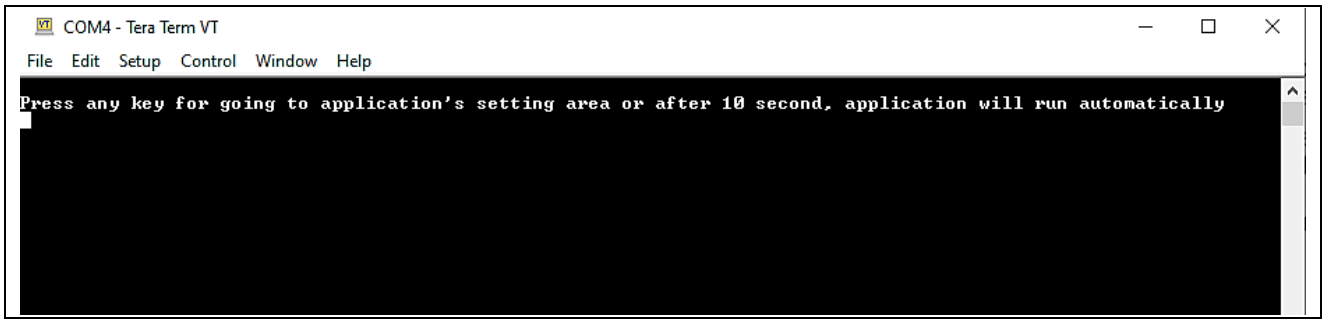

Figure 27. Start the Application

For the first-time users running the application, or for users who want to change the configuration, please press any key to set the necessary credentials for the application.

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

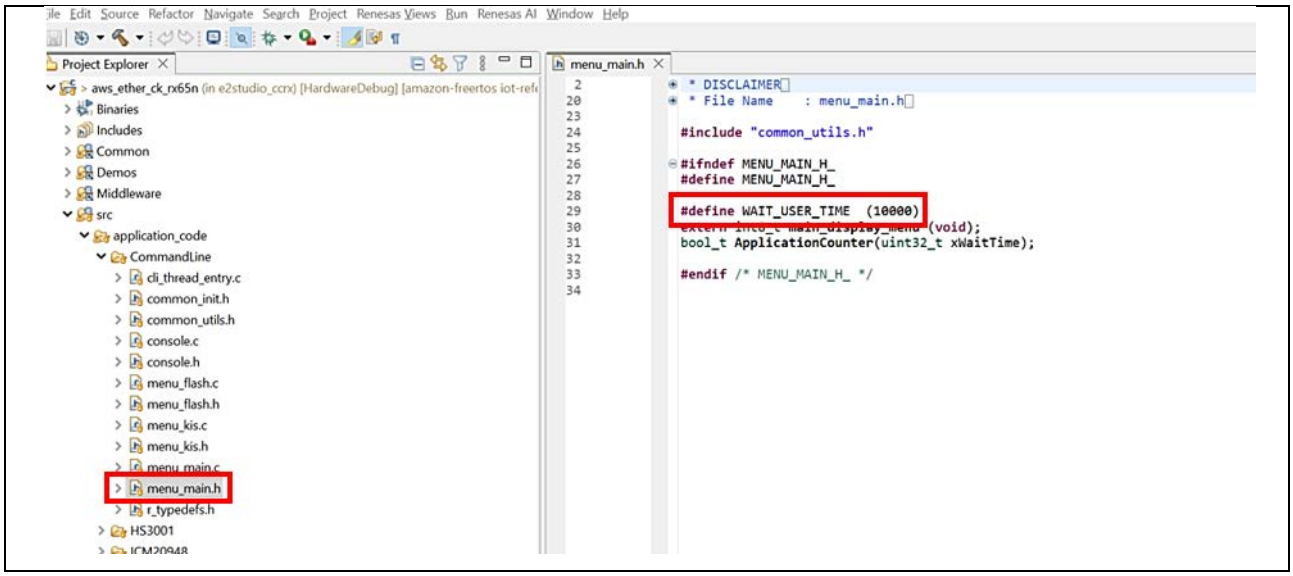

Figure 28. Modify macro to Wait for User Input

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

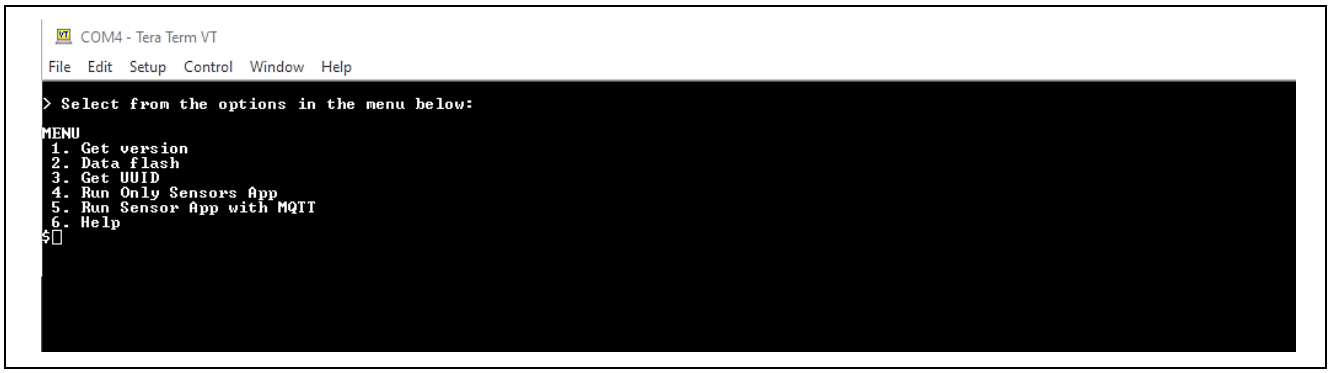

Figure 29. Main Menu

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

Figure 30. Get Version Information

5.1.5 Activating SIM Card

This section explains how to activate the SIM card that is contained in CK-RX65N V1 for using cellular application. If you use ethernet application, you can skip this step.

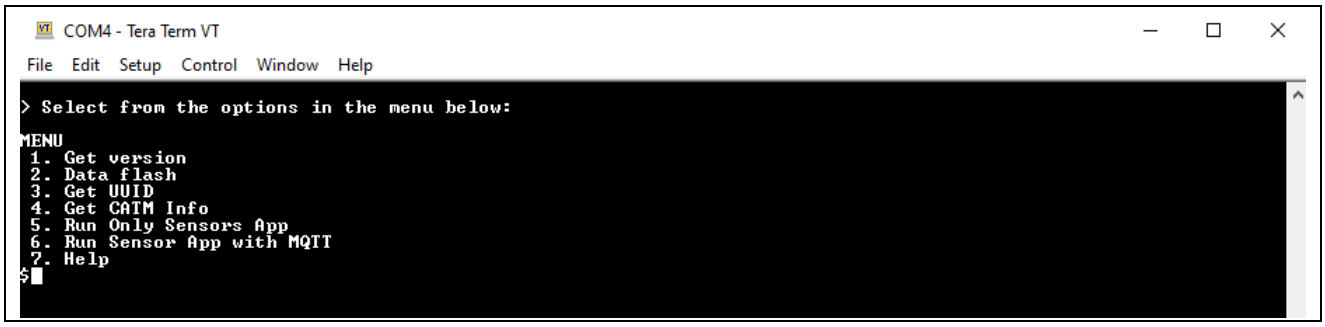

Figure 31. Main menu of cellular application

From the main menu of cellular application shown in the above figure, press '**4**' to display CAT-M Information (IMEI & ICCID). This menu will communicate with the CAT-M module to obtain the IMEI and ICCID values needed for activating the SIM card. Upon success, the IMEI and ICCID values will be displayed on the terminal screen. The program will continue to attempt to communicate with the CAT-M module until it has successfully connected or timed out. The IMEI and ICCID values are used to activate the SIM card.

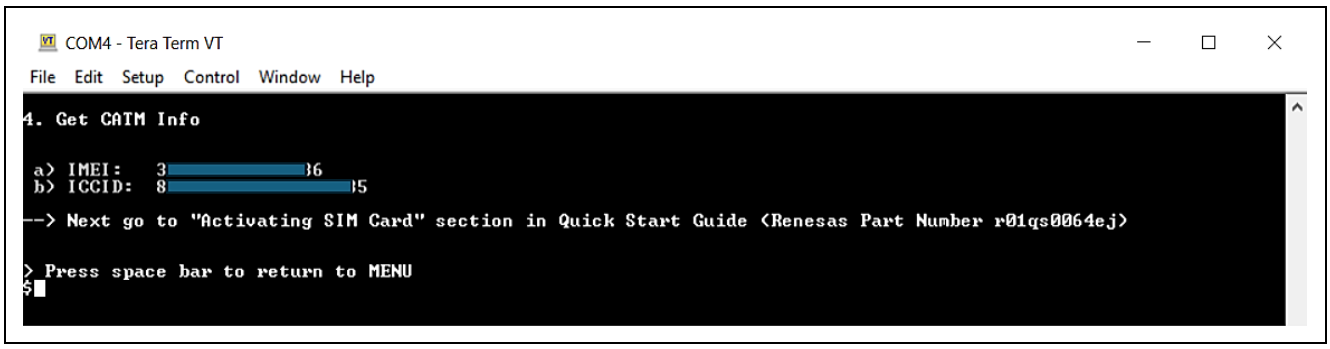

Figure 32. CAT-M Information

After completing the above steps, use the information to activate the SIM card.

Note: A Truphone or MicroAI SIM card is included in this kit.

Note: The MicroAI SIM card has been discontinued to support CK-RX65N. If a MicroAI SIM card was included in the kit, please contact Renesas support to request a replacement to a Truphone SIM card. A MicroAI SIM card can be identified by observing that the manufacturer's name is not printed on the card.

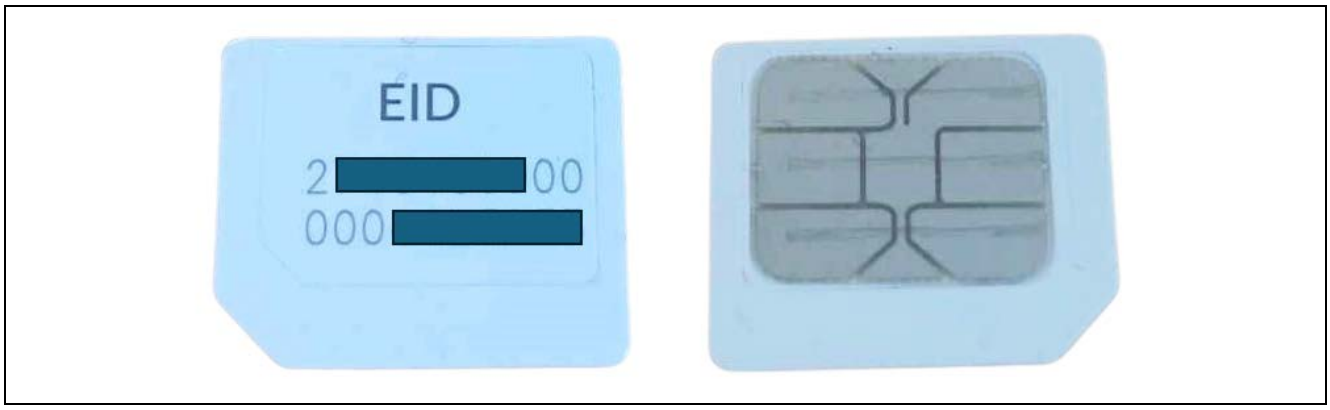

Figure 33. MicroAI SIM Card

Please activate the SIM card using the following steps:

To activate the included SIM card, please visit the Truphone SIM Activation platform at truphone.com/connectit and use the following steps:

1. On the Business page, click **Start activation** button under **IoT SIM Activation.**

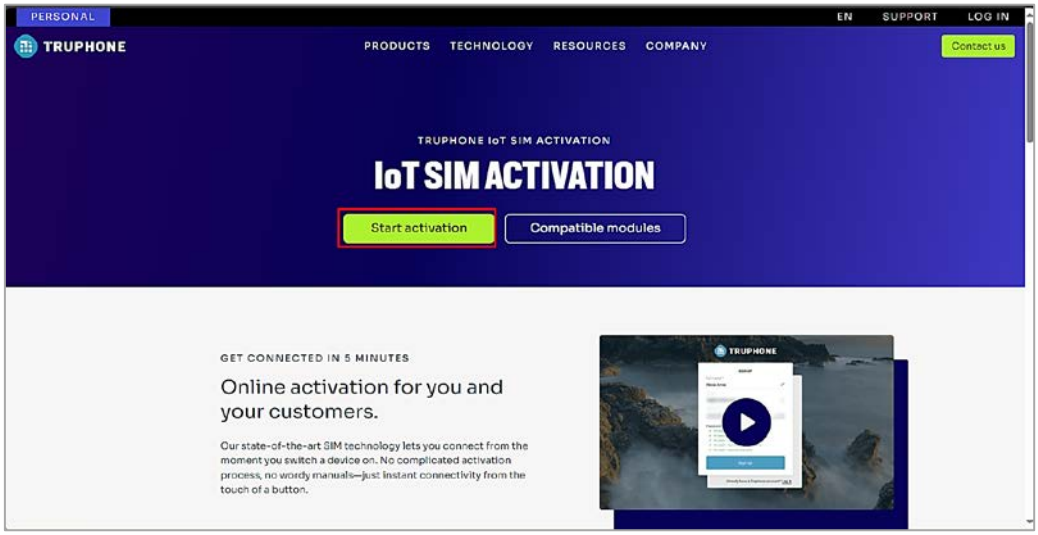

Figure 34. Activating the SIM Card

2. Create a new Truphone Account by selecting **Sign up** (next to **Don't have an account yet?**) and fill-in your full name, Email, and a password. Then click **Sign up** to create a new account.
3. Select **Personal** as the account type and press **Get Started**.
4. Verify your email by entering the activation code sent to your email account. (Note: check your Junk folder if the email is not received to your Inbox).
5. Complete the **Profile information** form – then select **Create account**.
6. Select **Activate SIMs** to activate your individual SIM by **ICCID** and **PUK** found on the SIM Card packaging. Note: The **ICCID** value can also be obtained when running cellular project. See the **ICCID** value in **Figure 32. CAT-M Information.** Fill other fields as needed.
7. You can open the page using the link https://account.truphone.com/login and select Home > SIM Cards > ICCID#xxx", and "Activate" the SIM from the status page as shown in **Figure 35. Activating the SIM Card on Truphone**, the status changes from Pre-Active to Active.

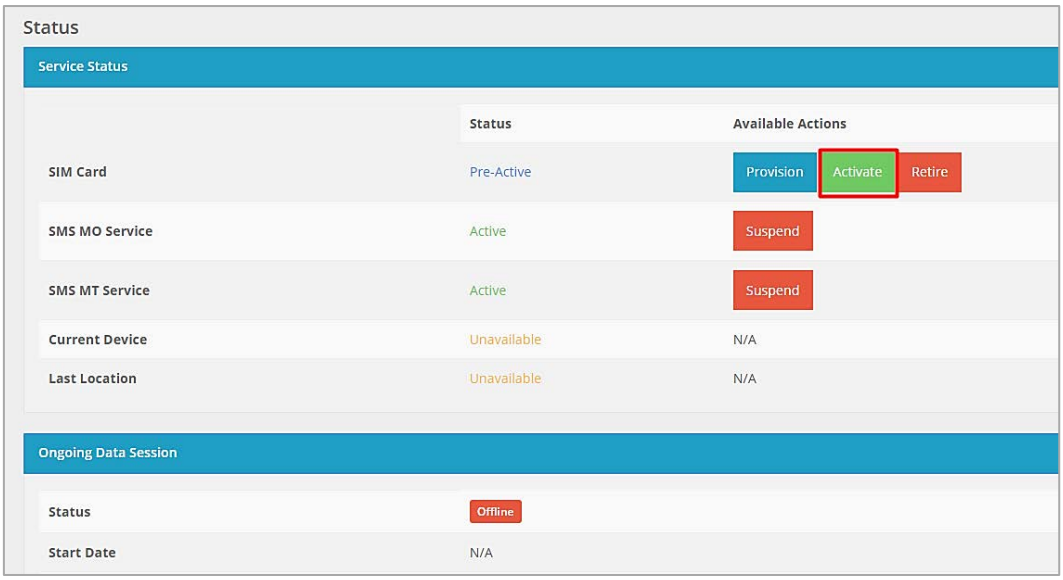

Figure 35. Activating the SIM Card on Truphone

8. You will receive email confirmation when the SIM card activation is complete.
   The **CK-RX65N V1** kit and SIM card should be activated and can be validated on the Tera Term terminal.

   Note: The SIM card includes free credit for the first 90 days / 50 MB. After the free data charge is used up, communication charges will be incurred.

Disclaimer

The activation steps above are provided by the SIM provider, Truphone. They are the most current at the time of publishing this application note. If you need help activating your SIM card, contact Truphone support iot.truphone.com or Contact Support | Truphone.

If you have a SIM card from any other provider then contact the technical support for that provider.

For any other issue that cannot be resolved please contact Renesas Support at Technical Support.

Note**:** The SIM card provider for the Quick Start Guide example project is Truphone. If you use any other SIM card provider, you must change the Access Point Name required for the SIM card provider in your global region (Please refer to the **6 Check and set the SIM card information** to re-configure). Failure to do so could result in the RYZ014A not connecting to the cellular network.

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

This section explains getting the board "UUID" information, account registration, and accessing the dashboard.

5.2.1 Getting the Board UUID Information

Press '**3'** from the **Main Menu** to display board UUID. This displays the board UUID information on the console as shown in the screenshot below. You will need this information to register on the Cloud Dashboard.

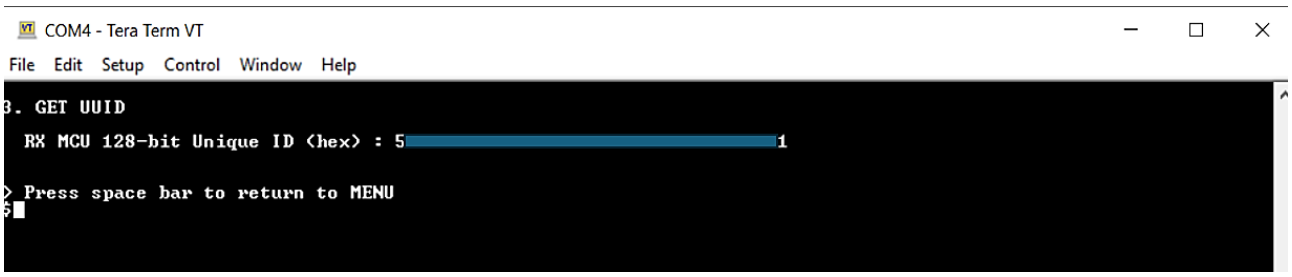

Figure 36. Getting Board UUID Information

5.2.2 Getting the Account 10 USD for 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 being used when users register their email and UUID on this system. Renesas recommends disabling the AWS EC2 service when users do not use this system. Please refer to section **8.8** How to Enable/Disable EC2 Instance**.**

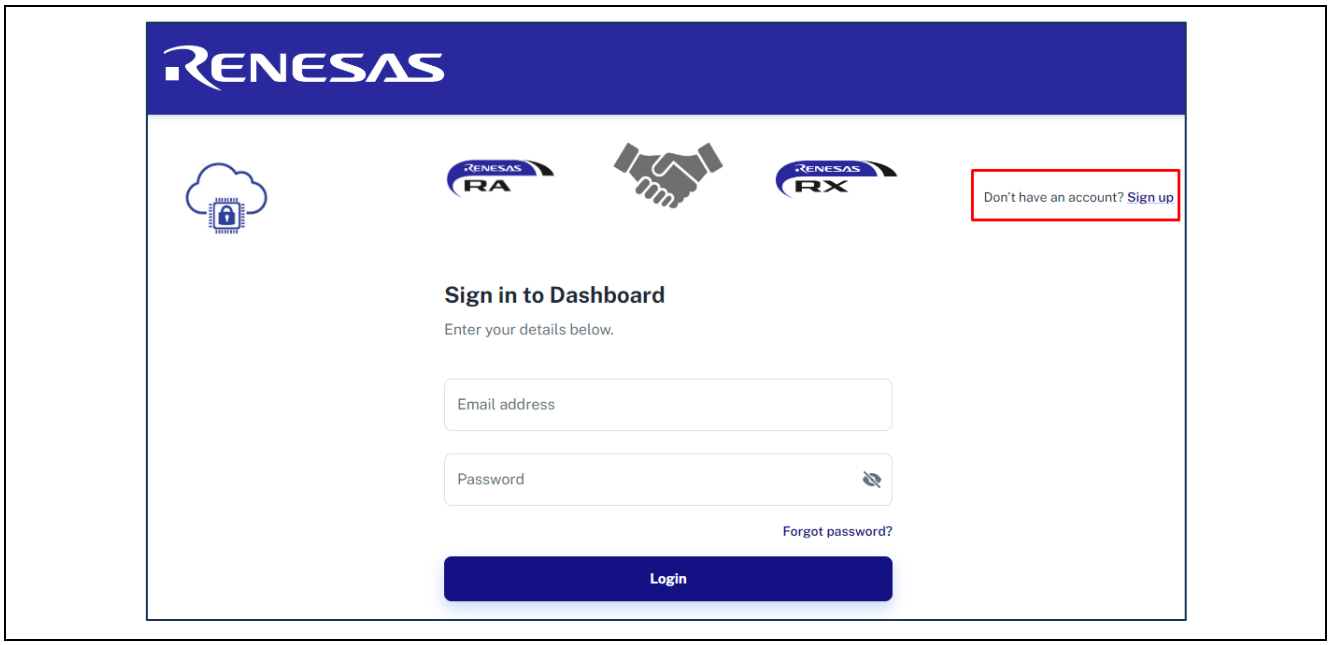

Figure 37. Get the Account 10 USD for Trial of AWS (1/2)

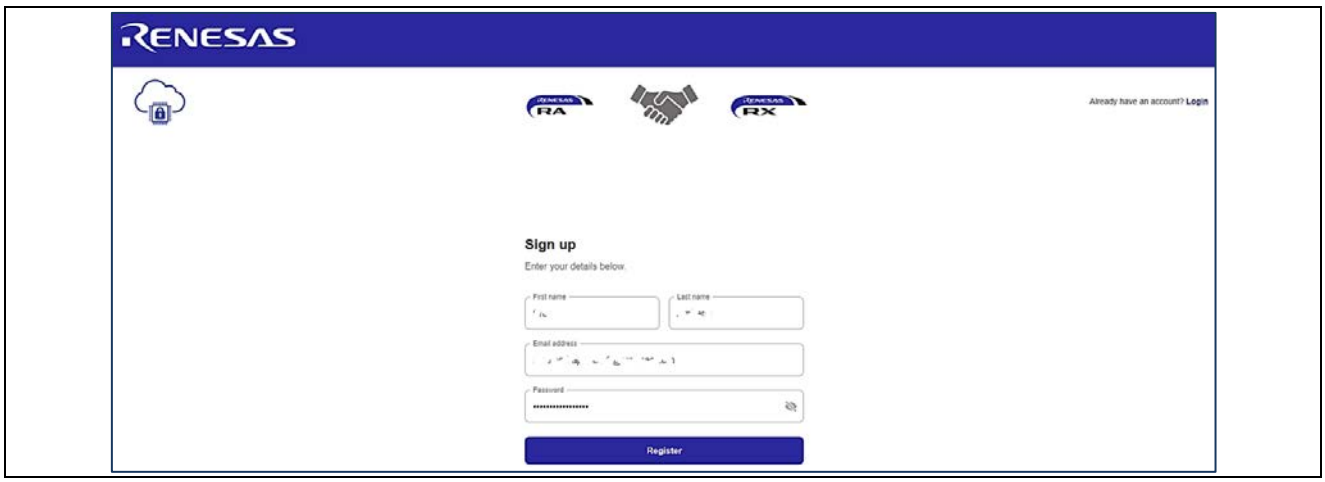

Figure 38. Get the Account 10 USD for Trial of AWS (2/2)

2. Wait for AWS verification email (it may take up to 10 minutes). Then, enter the email and UUID to register the kit as shown in the following window. You can get the UUID from section **5.2.1 Getting the Board UUID Information**

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

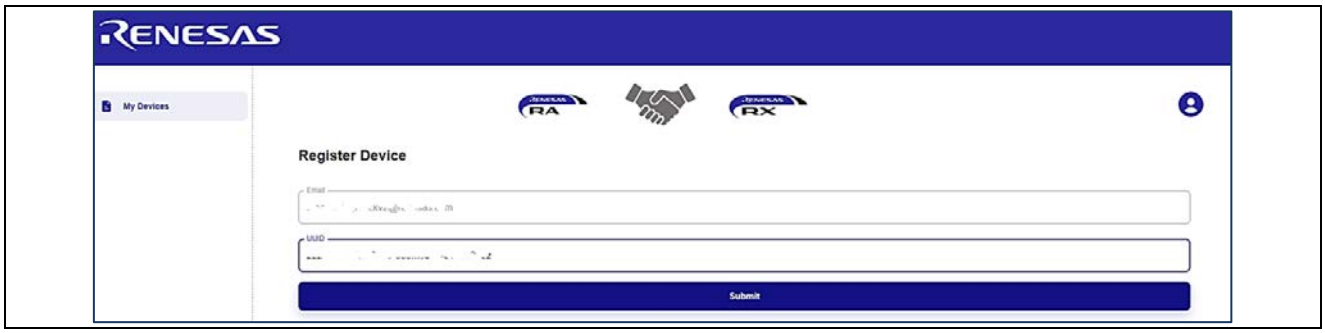

Figure 39. Register Device

3. **Verify the AWS account** in your email that you registered.
4. Wait for the status change on the registration page or wait for provisioning to complete. Please refresh the page in case the “Registration in progress” screen still shows up.

Figure 40. Dashboard Build In Progress

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

Figure 41. Active Device

6. After finishing the progress, you can get the connection certificate using the “Download Certificate” button. This is used for installation on the application demo of kits that you got in the previous step.

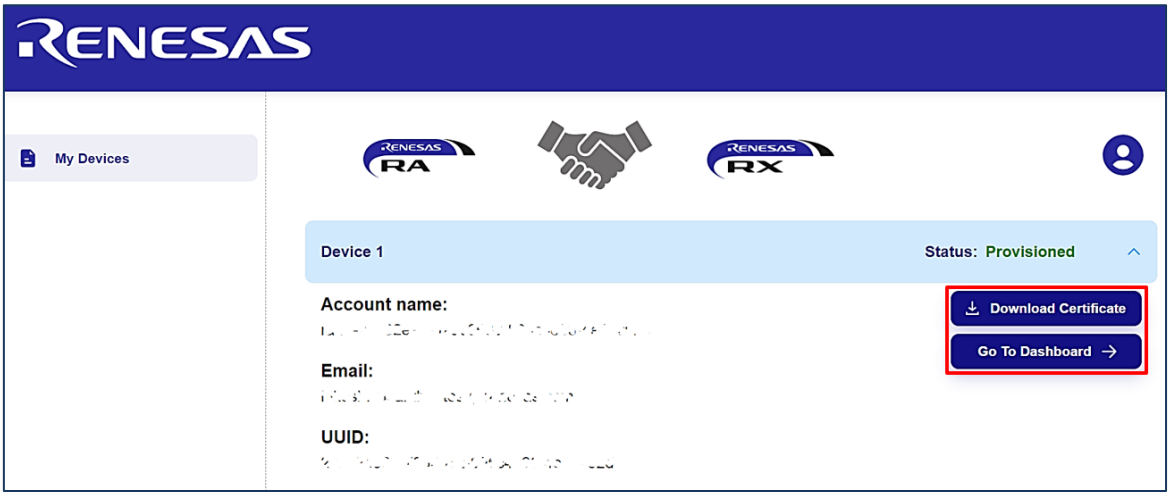

Figure 42. Dashboard Build Complete

7. Click "**Go To Dashboard**" to access the dashboard. First time users will access the dashboard with credentials "**admin**" **for both username and password** and will be directed to change the password. Once completed, users can access the dashboard.

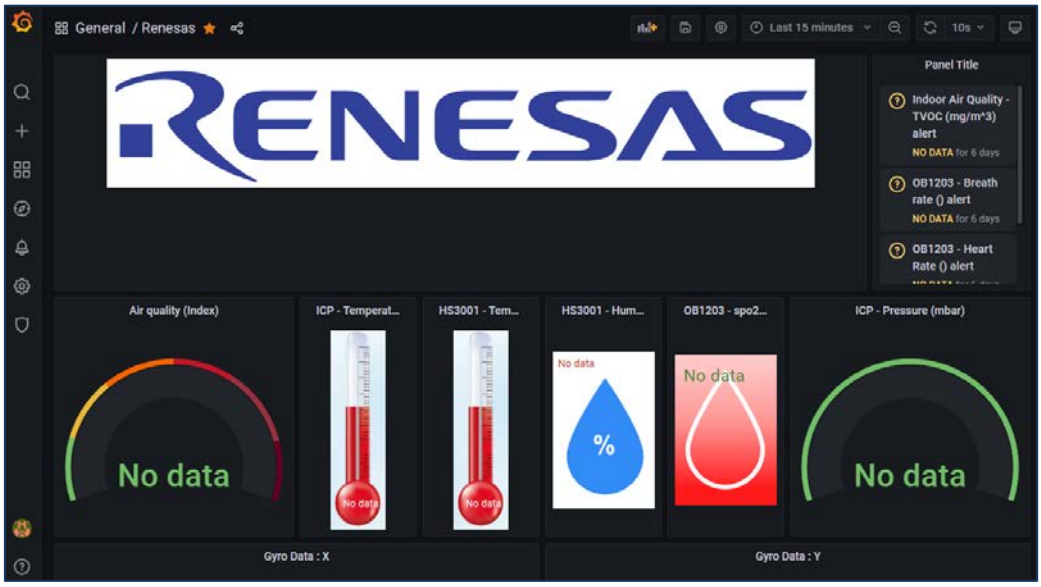

Figure 43. Dashboard for this Application

5.3 Software Preparation - Running 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 screenshot. This sub menu has commands to store, read, and validate the data.

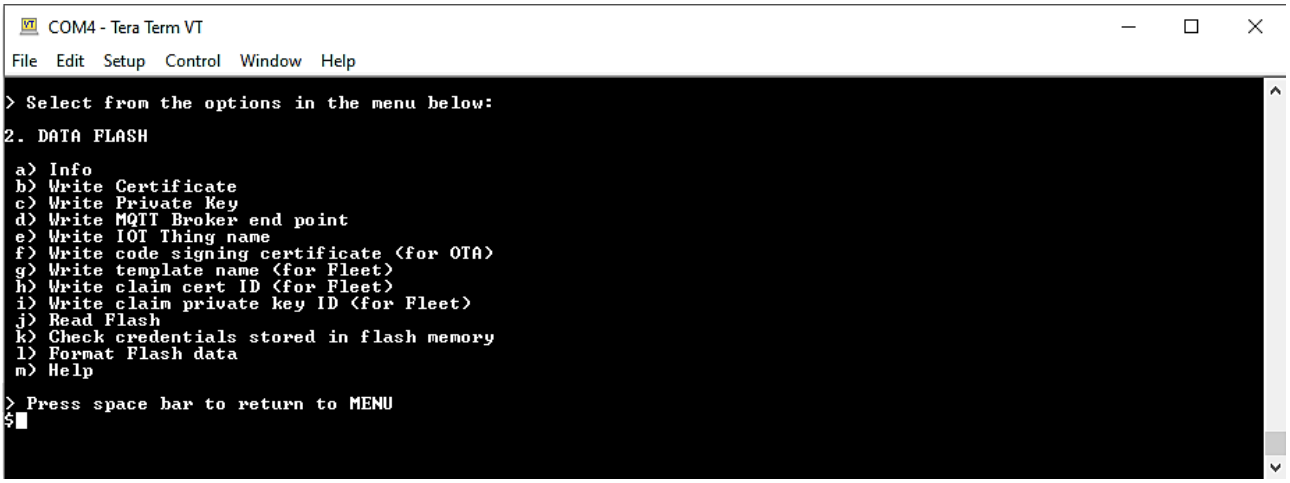

Figure 44. Data Flash related Menu and Commands

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

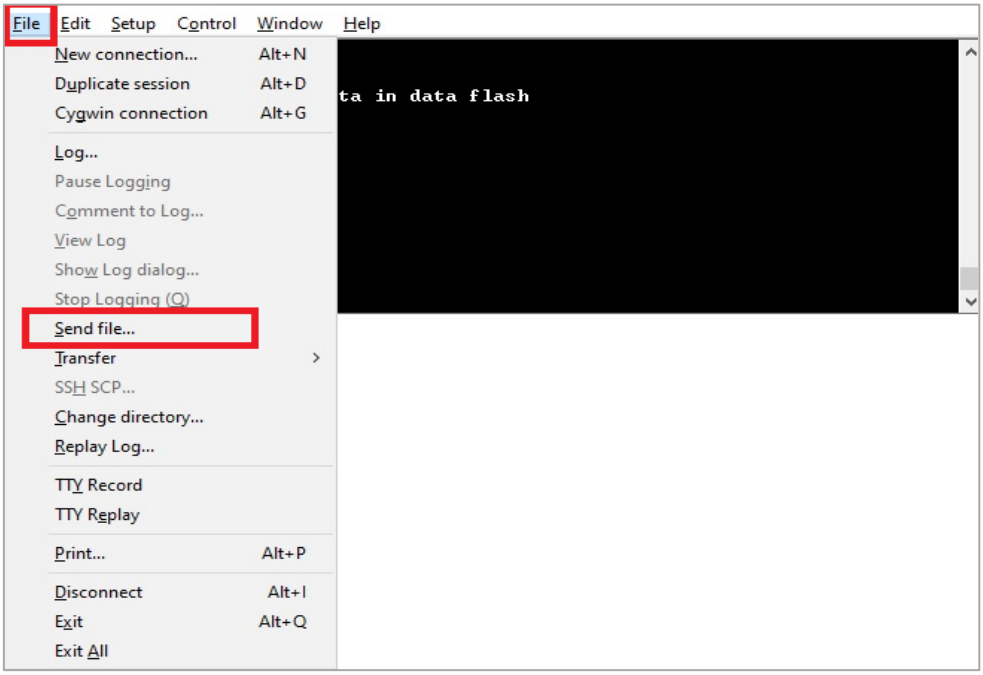

Figure 45. Accessing the Device Certificate

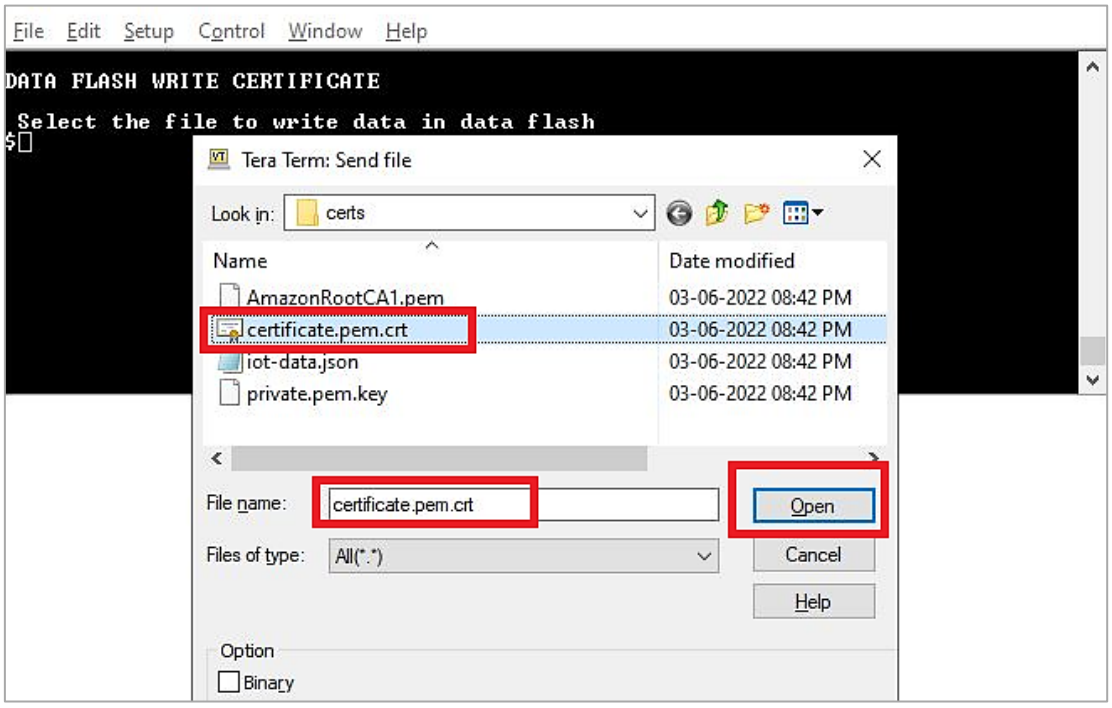

Figure 46. Downloading the Device Certificate into the Data Flash

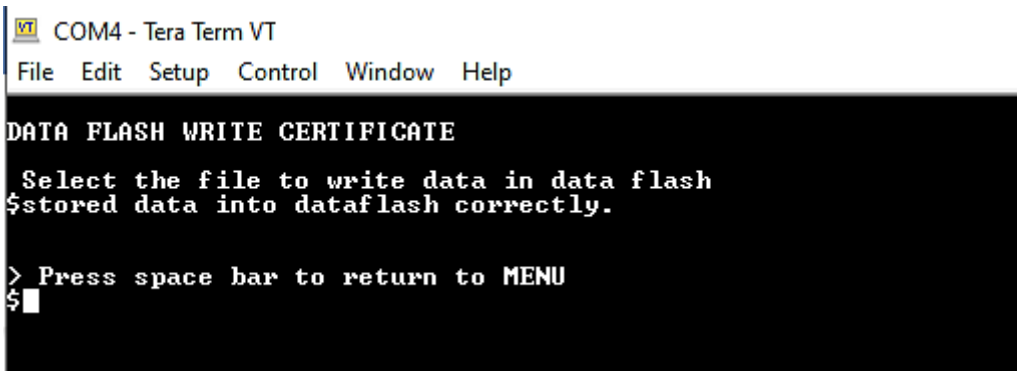

Figure 47. Status of the Downloaded Device Certificate into the Data Flash

4. To store the **Device Private Key,** press the option '**c**', click the **File** tab in the Tera Term and select the **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.

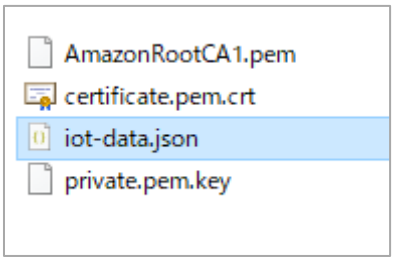

Figure 48. Getting the IoT Things Name and IoT Endpoint Information

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 in the Tera Term and "**Paste<CR>**" and verify and confirm the valid string and press **OK**.
   Note: Please copy the IOTEndpoint without "".

Figure 49. Copy the IOTEndpoint

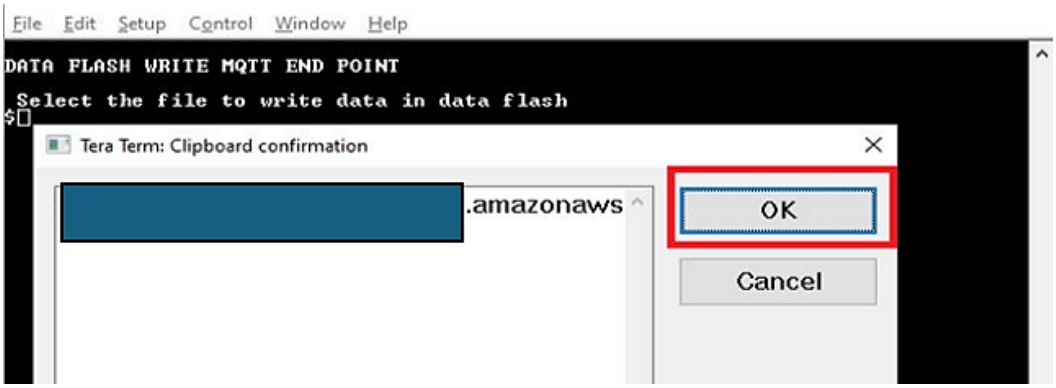

Figure 50. Storing the MQTT IoT endpoint into the Data Flash

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**', click the **Edit** tab in the Tera Term and "**Paste<CR>**", verify and confirm the valid string, and press **OK**.
   Note: Please copy the IOTThingName without "".

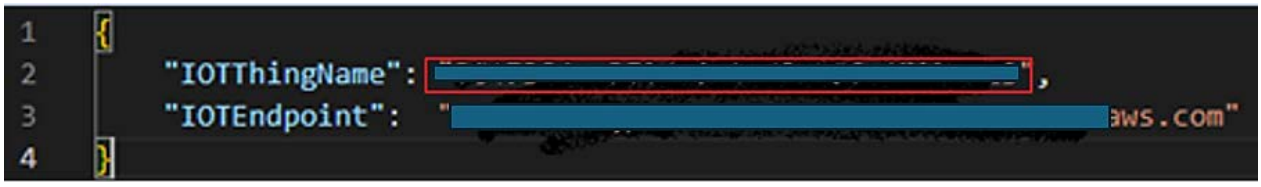

Figure 51. Copy the IOTThingName

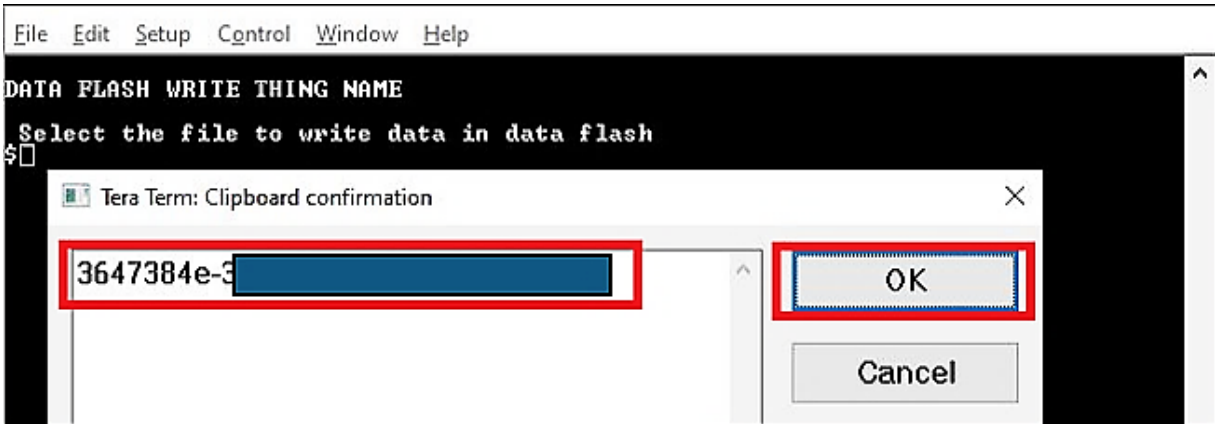

Figure 52. Storing the Thing Name into the Data Flash

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 **'l) Format Flash data'** will erase all saved value in data flash. Please be careful when using this option in application.

5.3.2 Starting the Application

After activating the SIM card (if using Cellular project), registering to the Dashboard and configuring the required Cloud credentials via the CLI, the application is ready to run. Press option **'Run Sensor App with MQTT'** to start the application. The application prints a welcome screen along with the status of validating the Cloud credentials data present in the data flash as shown below.

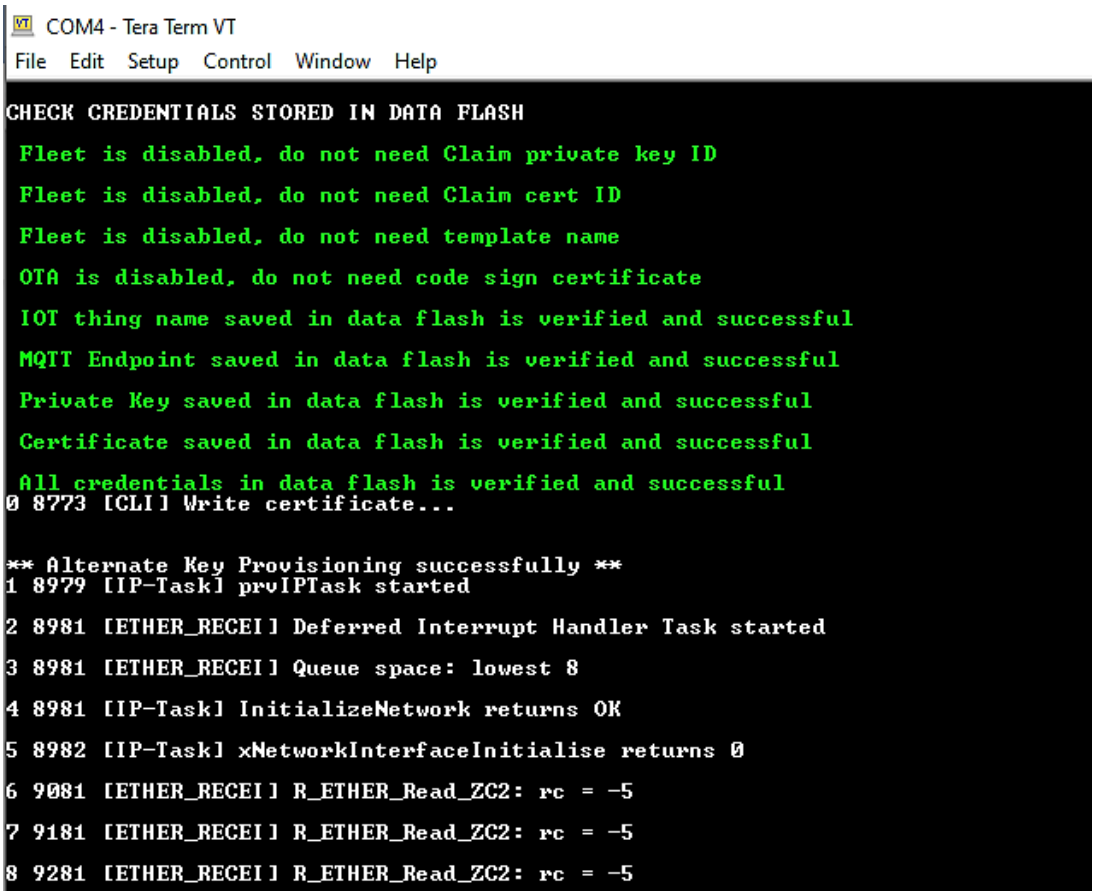

Figure 53. Welcome Screen on the Console

When the connection is successful, the data is shown.

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help
25 10981 [ETHER_RECEI] R_ETHER_Read_ZC2: re = -5
26 10982 [ETHER_RECEI] prvLinkStatusChange( 1 )
27 10982 [ETHER_RECEI] prvEMACHandlerTask: PHY LS now 1
28 11982 [IP-Task] xNetworkInterfaceInitialise returns 1
29 11982 [IP-Task] prvInitialiseDHCP: start after 250 ticks
30 12232 [IP-Task] vDHCPProcess: discover
31 12234 [ETHER_RECEI] Network buffers: 3 lowest 2
32 17482 [IP-Task] vDHCPProcess: discover
33 17482 [IP-Task] vDHCPProcess: timeout 10000 ticks
34 27732 [IP-Task] vDHCPProcess: discover
35 27732 [IP-Task] vDHCPProcess: timeout 20000 ticks
36 28517 [IP-Task] vDHCPProcess: offer c0a80395ip
37 28517 [IP-Task] vDHCPProcess: reply c0a80395ip
38 28521 [IP-Task] vDHCPProcess: offer c0a80395ip
39 28521 [IP-Task] vDHCPProcess: acked c0a80395ip
40 28521 [IP-Task] IP Address: 192.168.3.149
41 28521 [IP-Task] Subnet Mask: 255.255.255.0
42 28522 [IP-Task] Gateway Address: 192.168.3.1
43 28522 [IP-Task] DNS Server Address: 8.8.8.8
44 28800 [CLI] Initialize the RTOS's TCP/IP stack
45 28800 [CLI] ---------STARTING DEMO---------
46 28804 [MQTT] [INFO] ---------Start MQTT Agent Task---------
47 28804 [MQTT] [INFO] Creating a TLS connection to [redacted]:-east-1.amazonaws.com:8883.
48 28905 [ETHER_RECEI] Heap: current 199048 lowest 194816
49 33838 [MQTT] FreeRTOS_ProcessDNSCache: add: '[redacted]-east-1.amazonaws.com' @ 36a288c1ip (TTL 60)
50 33838 [MQTT] DNS[0xAA23]: The answer to '[redacted]-east-1.amazonaws.com' (54.162.136.193) will be stored
51 33840 [MQTT] FreeRTOS_connect: 61069 to 36a288c1ip:8883
52 33840 [IP-Task] prvSocketSetMSS: 1400 bytes for 36a288c1ip:8883
53 33840 [IP-Task] prvWinScaleFactor: uxRxWinSize 2 MSS 1400 Factor 0
54 33840 [IP-Task] Connect[36a288c1ip:8883]: next timeout 1: 3000 ms
55 34074 [IP-Task] MSS change 1400 -> 1452
56 34074 [IP-Task] TCP: active 61069 => 36a288c1ip:8883 set ESTAB (scaling 1)
57 34075 [MQTT] [INFO] Established TCP connection with a[redacted]east-1.amazonaws.com.
58 34174 [ETHER_RECEI] Heap: current 138816 lowest 138344
59 34895 [ETHER_RECEI] Heap: current 124720 lowest 122568
60 36888 [ETHER_RECEI] Network buffers: 1 lowest 1
61 36889 [MQTT] [INFO] (Network connection 805230) TLS handshake successful.
62 36891 [MQTT] [INFO] (Network connection 805230) Connection to a[redacted]-east-1.amazonaws.com established.
63 36891 [MQTT] [INFO] Creating an MQTT connection to the broker.
64 37171 [IP-Task] TCP: No active socket on port 55473 (34363dddip:8883)
65 37184 [MQTT] [INFO] MQTT connection established with the broker.
66 37184 [MQTT] [INFO] Successfully connected to MQTT broker.
67 37190 [sensor_thre] I2C bus 0 setup success
68 37190 [sensor_thre] HS3001 open sensor instance successful: 0
69 37190 [sensor_thre] ICP10101 open sensor instance successful: 0
70 37190 [ob1203_thre]
OB1203 Device open success
71 37200 [sensor_thre] ICM20948 open sensor instance successful: 0
```

Figure 54. Application with MQTT

Note: Above is the log of the Ethernet application running.

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

Note: With OB1203 sensor, besides the stabilization time, OB1203 sensor data which is sent to the MQTT (shown 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 slightly longer than the stabilization time.

Note: For Ethernet applications, firewalls in the network may prevent connectivity to AWS IoT. Configure the network to allow access to the MQTT Port 8883.

About the detail of stabilization time, please see **Table 12. Sensor Stabilization Time**

5.4 For Users Using Their Own AWS Account

Note: Complete the steps up to "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**.

5.4.2 Log in to the AWS Management Console

Amazon Web Services > My Account > AWS Management Console

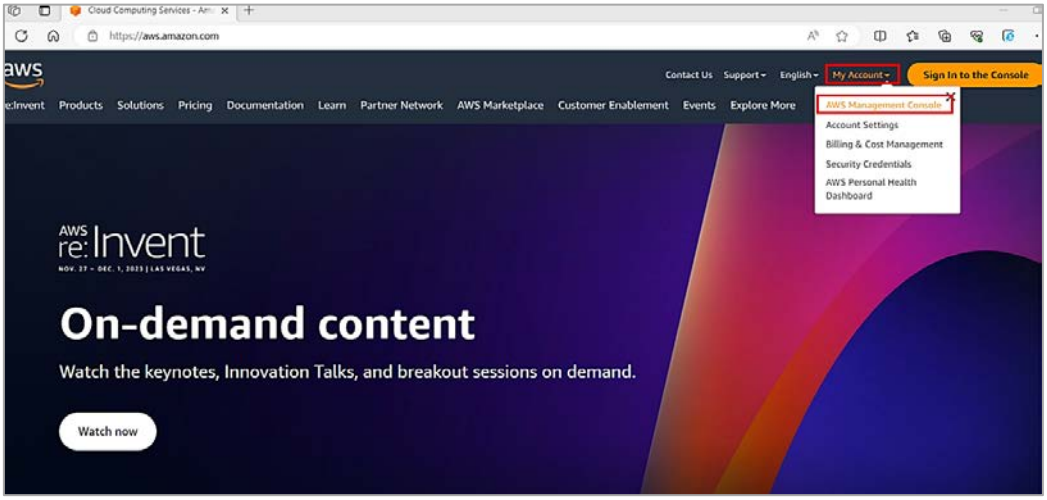

Figure 55. Login the AWS

5.4.3 Move to IoT Core Control Panel

AWS services > All services > IoT Core

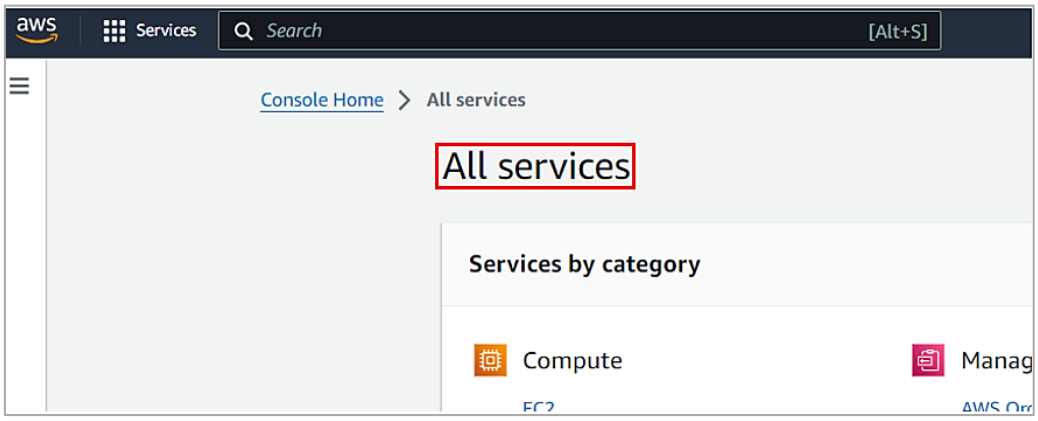

Figure 56. Search the IoT Core (1/2)

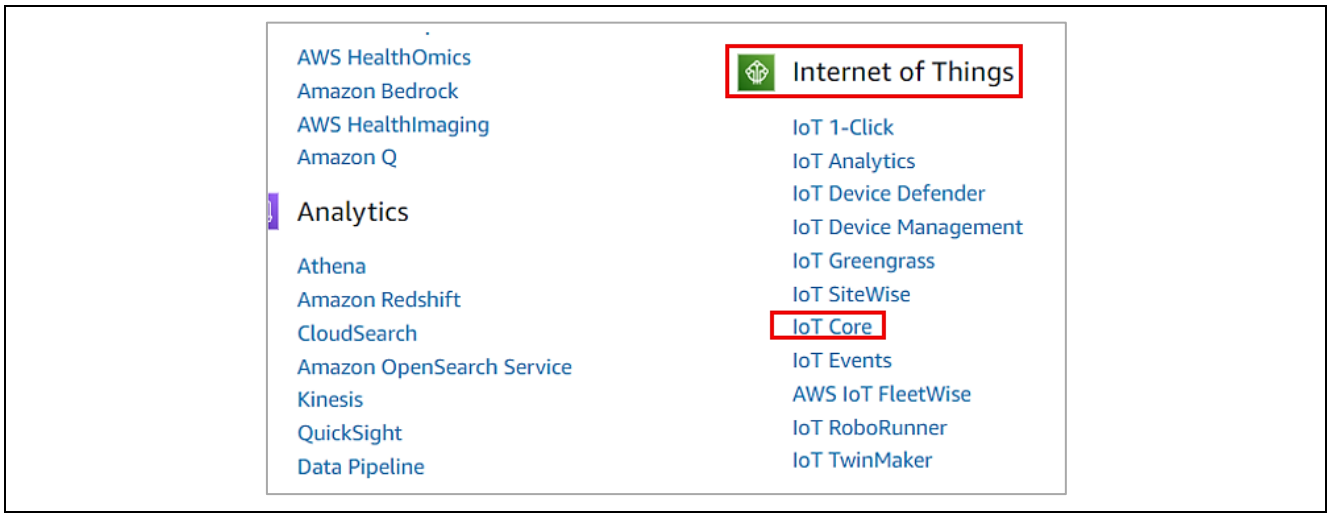

Figure 57. Search the IoT Core (2/2)

5.4.4 Create a Security Policy

Security > **Policies** > **Create policy.**

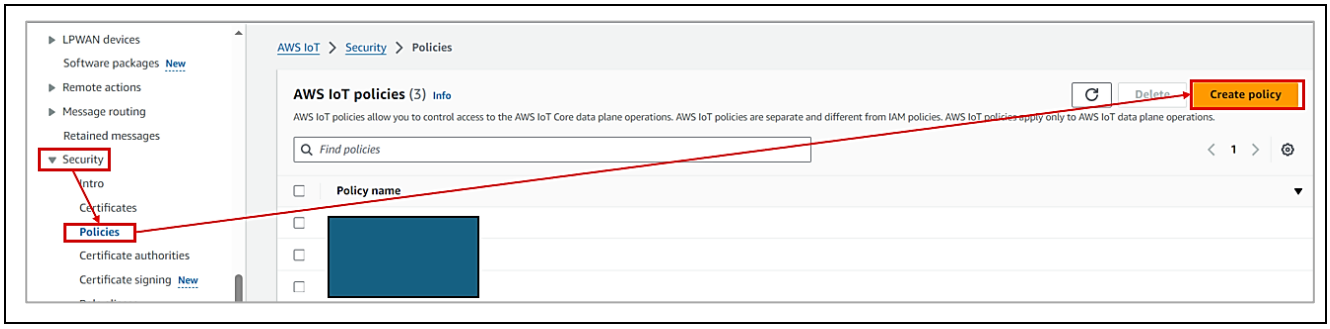

Figure 58. Create the Policy (1/3)

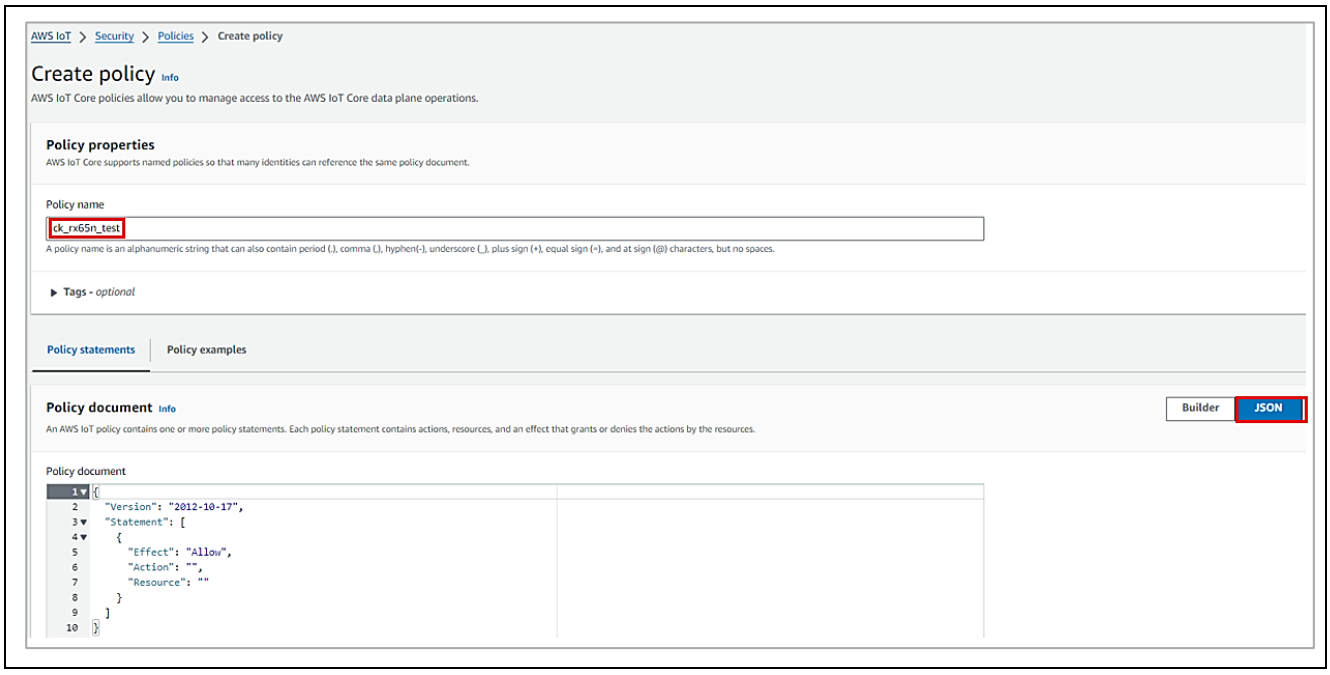

Figure 59. Create the Policy (2/3)

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 document > Create

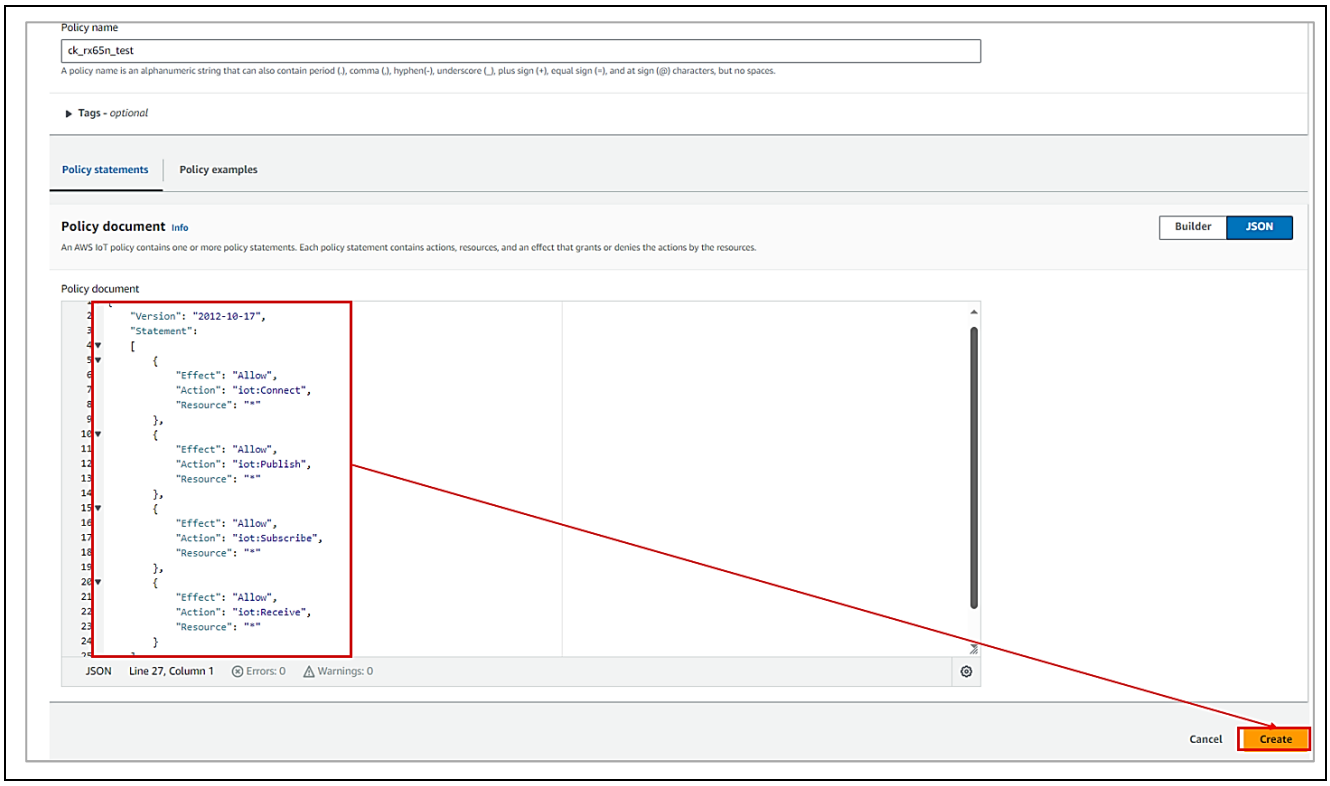

Figure 60. Create the Policy (3/3)

5.4.5 Register your device (thing) with AWS IoT

Manage > **Things** > **Create things**

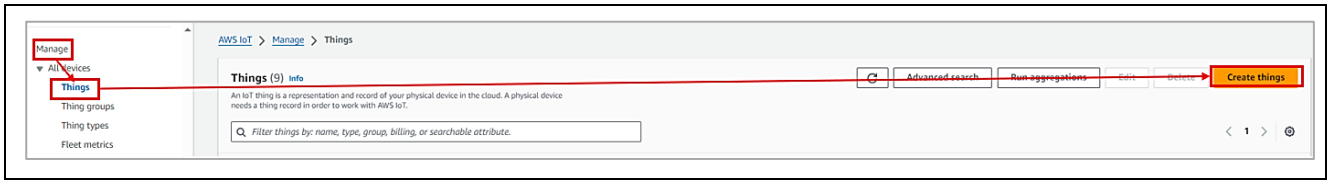

Figure 61. Creating the Things (1/5)

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

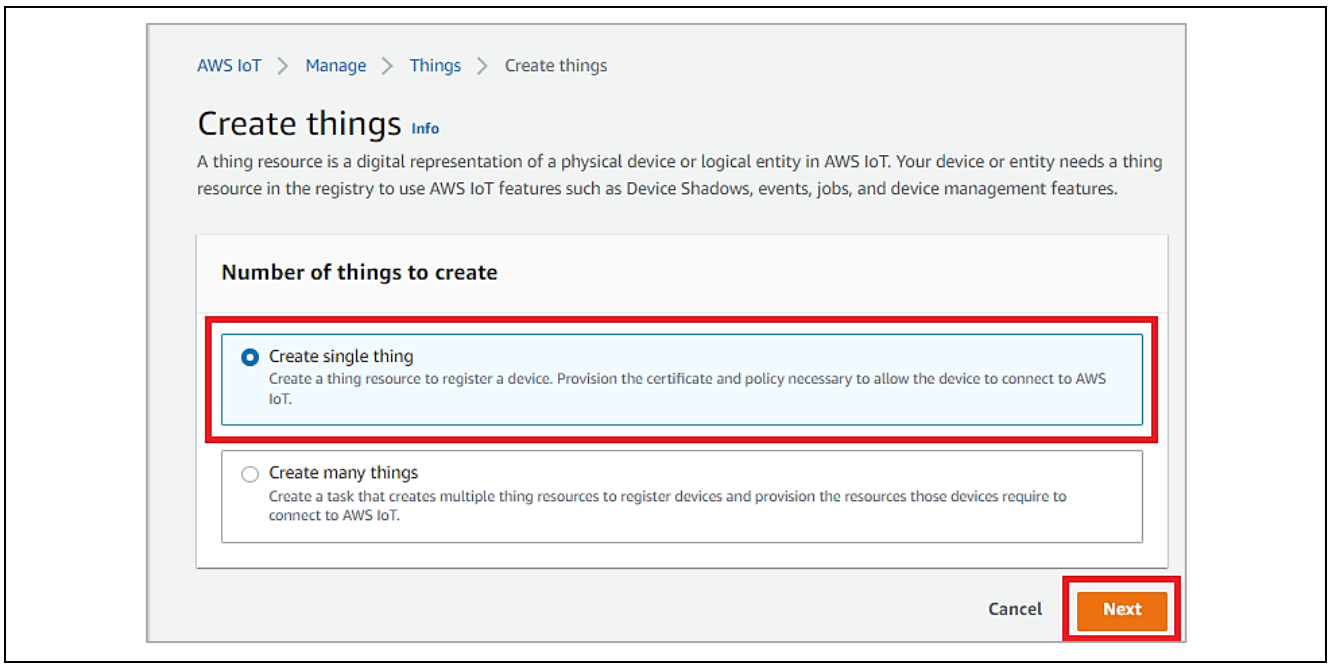

Figure 62. Creating the Things (2/5)

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

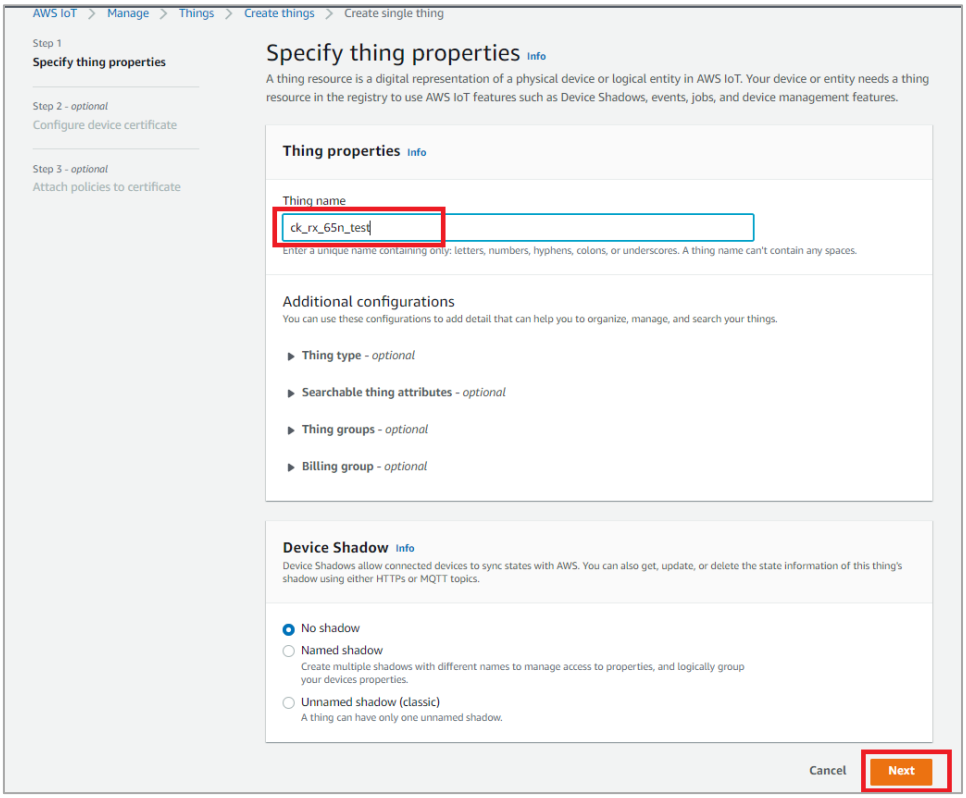

Figure 63. Creating the Things (3/5)

Auto-generate a new certificate.

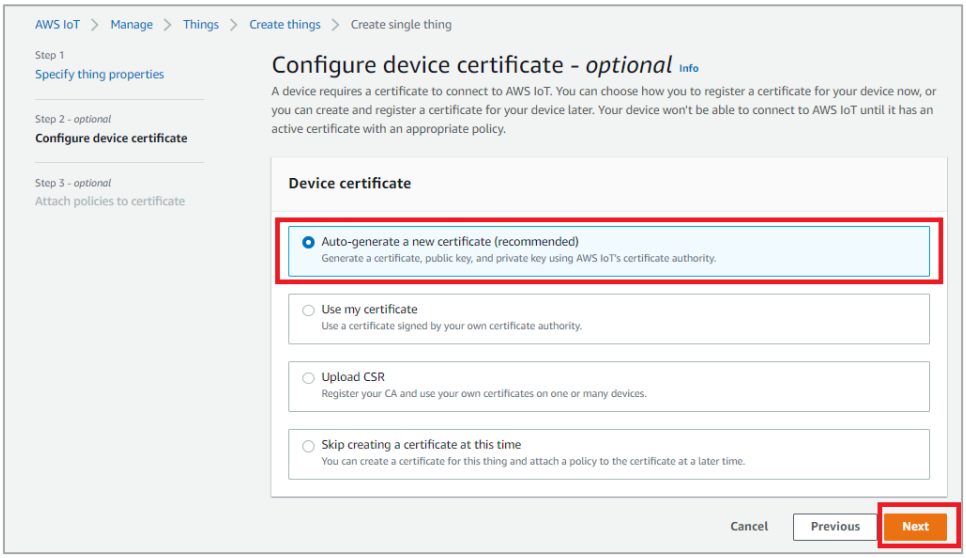

Figure 64. Creating the Things (4/5)

Add a policy for your thing.

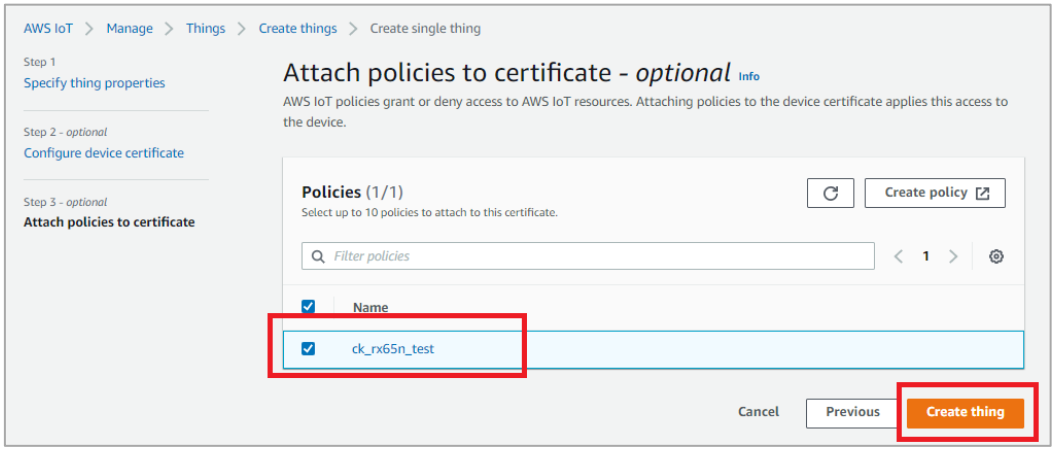

Figure 65. Creating the Things (5/5)

Download certificate, public key and private key for this thing.

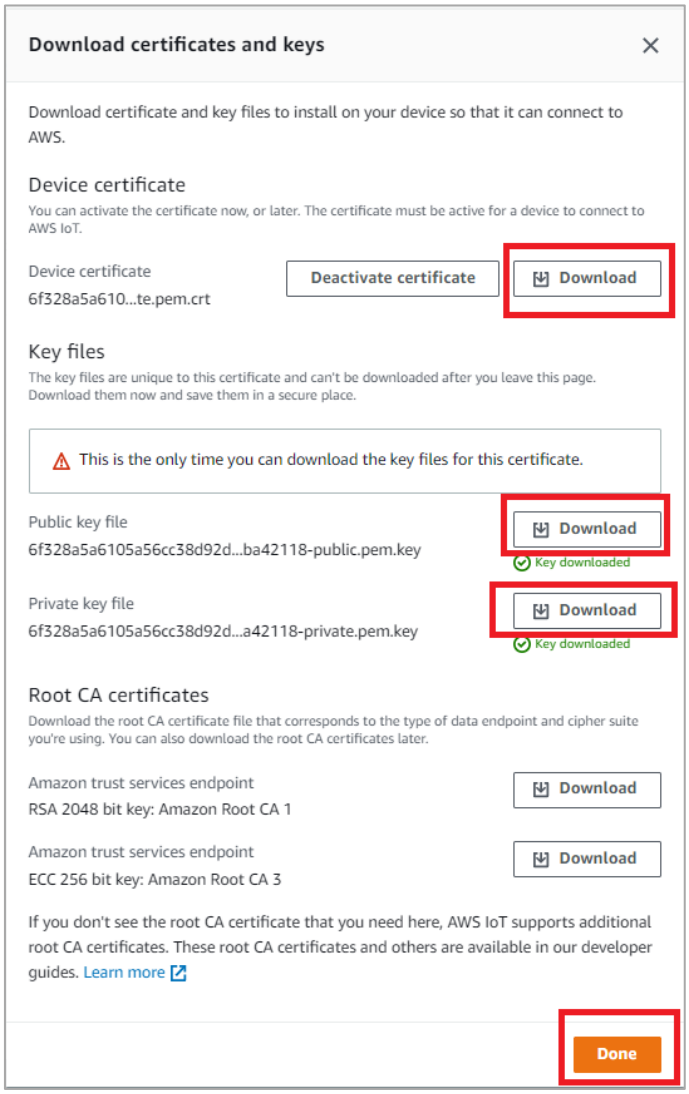

Figure 66. Download certificate, public key and private key

5.4.6 Check AWS IoT Endpoint

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

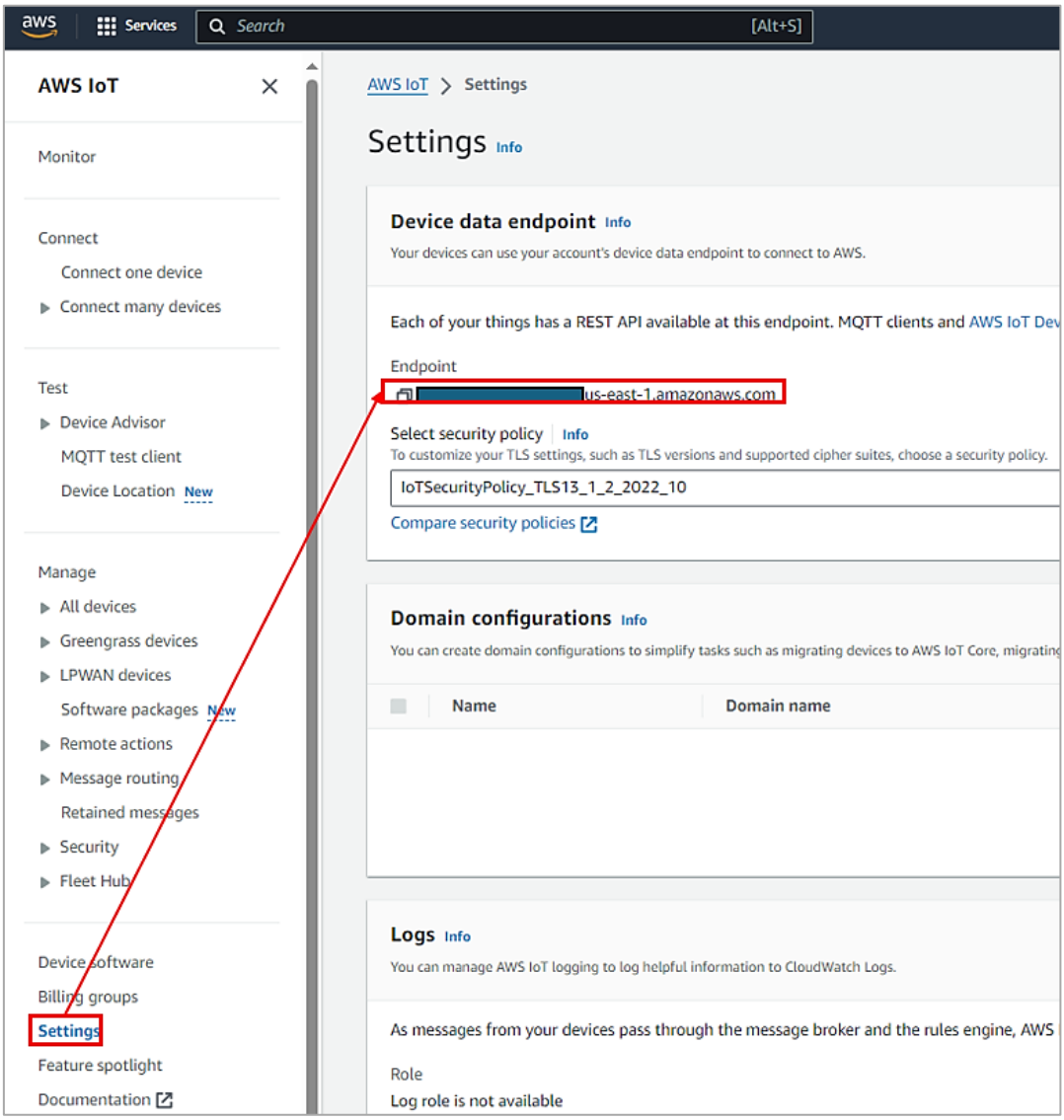

Figure 67. Check AWS IoT Endpoints

You can also access the tutorial to register a device to AWS IoT tutorial on GitHub at https://github.com/renesas/amazon-freertos/wiki/Register-device-to-AWS-IoT

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 - Running 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 how to verify this application example's functions.

Note: Wait for the board to get the IP address from the service provider upon successful cellular/ethernet initialization, and the board to resolve the DNS lookup for the endpoint. After the successful MQTT connection message on the Console, "***Successfully connected to MQTT broker***", the device is ready for Publishing and Subscribing of Messages.

Note: This Application involves AWS MQTT IOT Core, user has an option to use the AWS IOT Dashboard for the Validation purpose, in addition to using the Renesas GUI based Dashboard for customized view of all the Sensor Data.

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

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

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

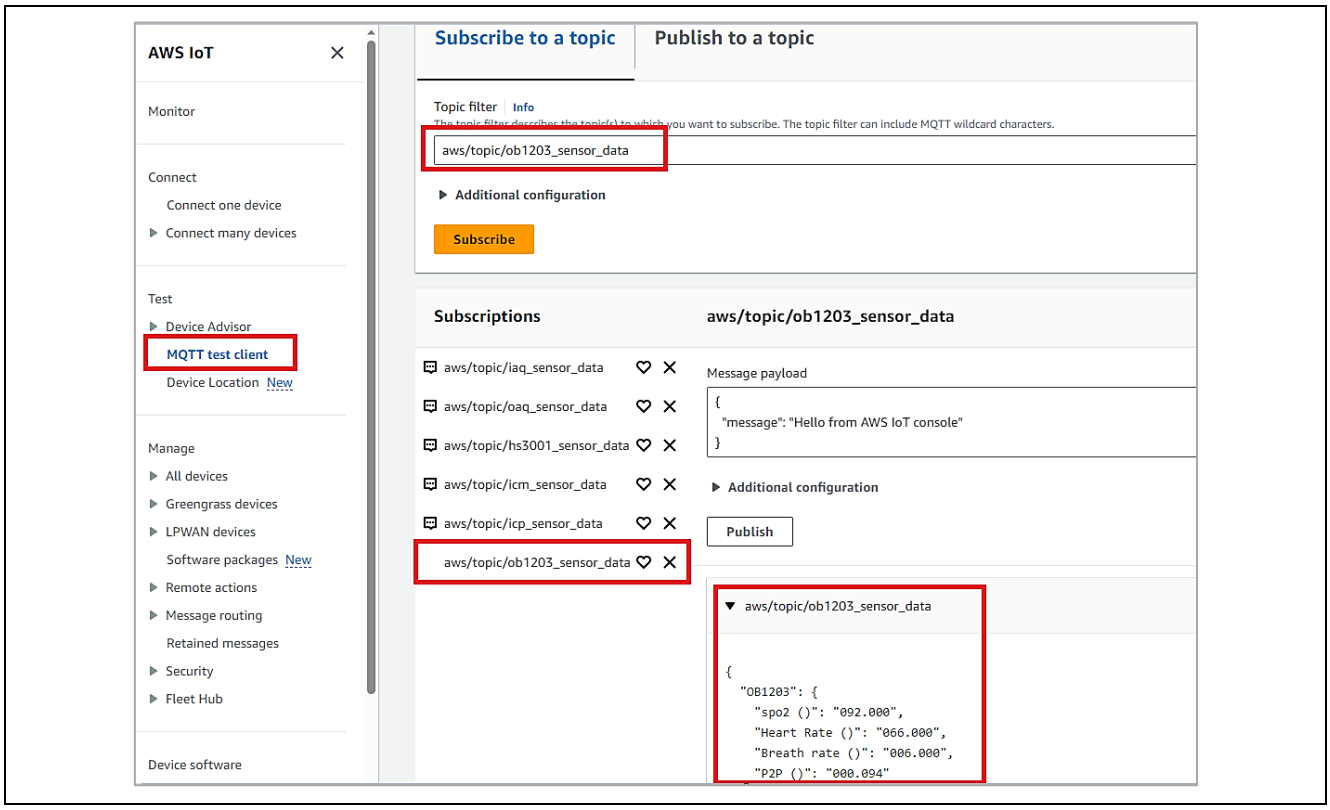

Figure 68. Subscribe to a Topic Messages on the AWS IoT Screen

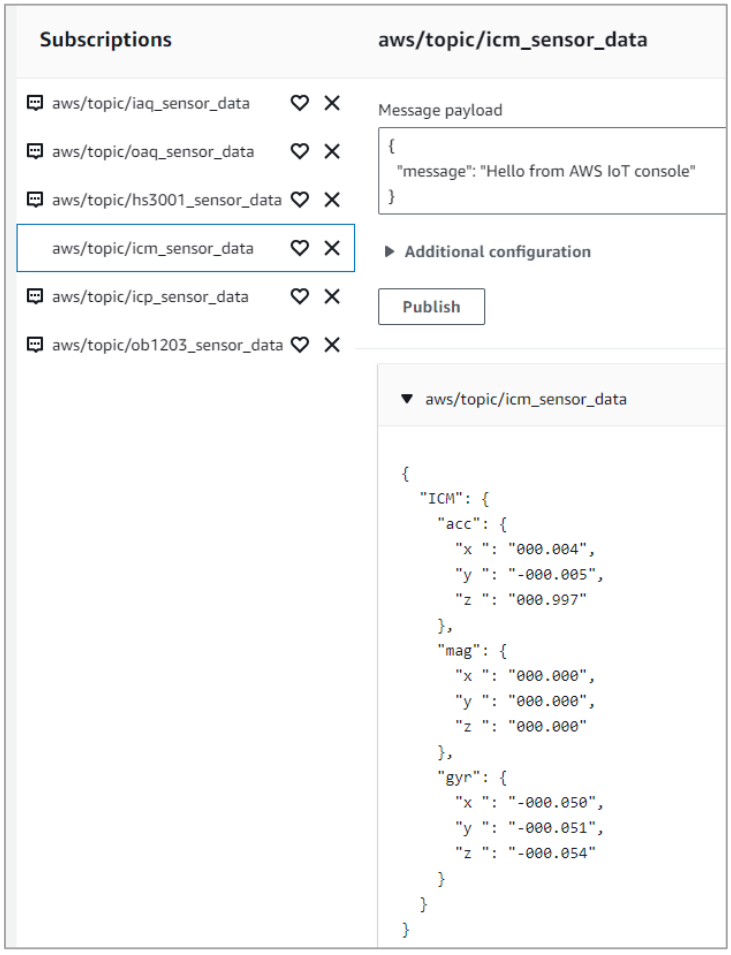

Figure 69. Subscribed Messages on the AWS IoT Screen

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 V1 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**

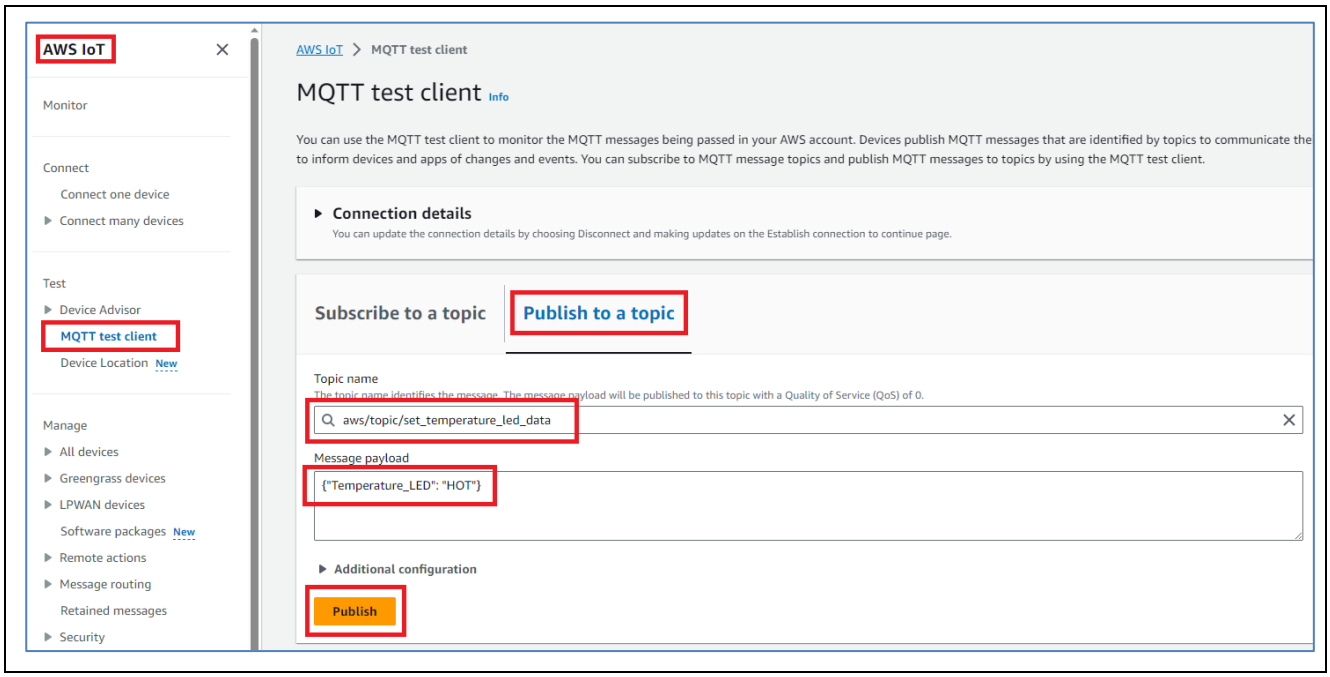

Figure 70. Publish the MQTT Message (1/2)

OB1203 SPO2 alerts:

Based on SPO2 value, dashboard will send the alert messages to CK-RX65N V1 kit via below topic

Topic: `aws/topic/set_spo2_led_data`

```
Message: {"Spo_LED": "ON"} Will turn on BLUE LED in CK-RX65N V1
Message: {"Spo_LED": "OFF"}Will turn off BLUE LED in CK-RX65N V1
```

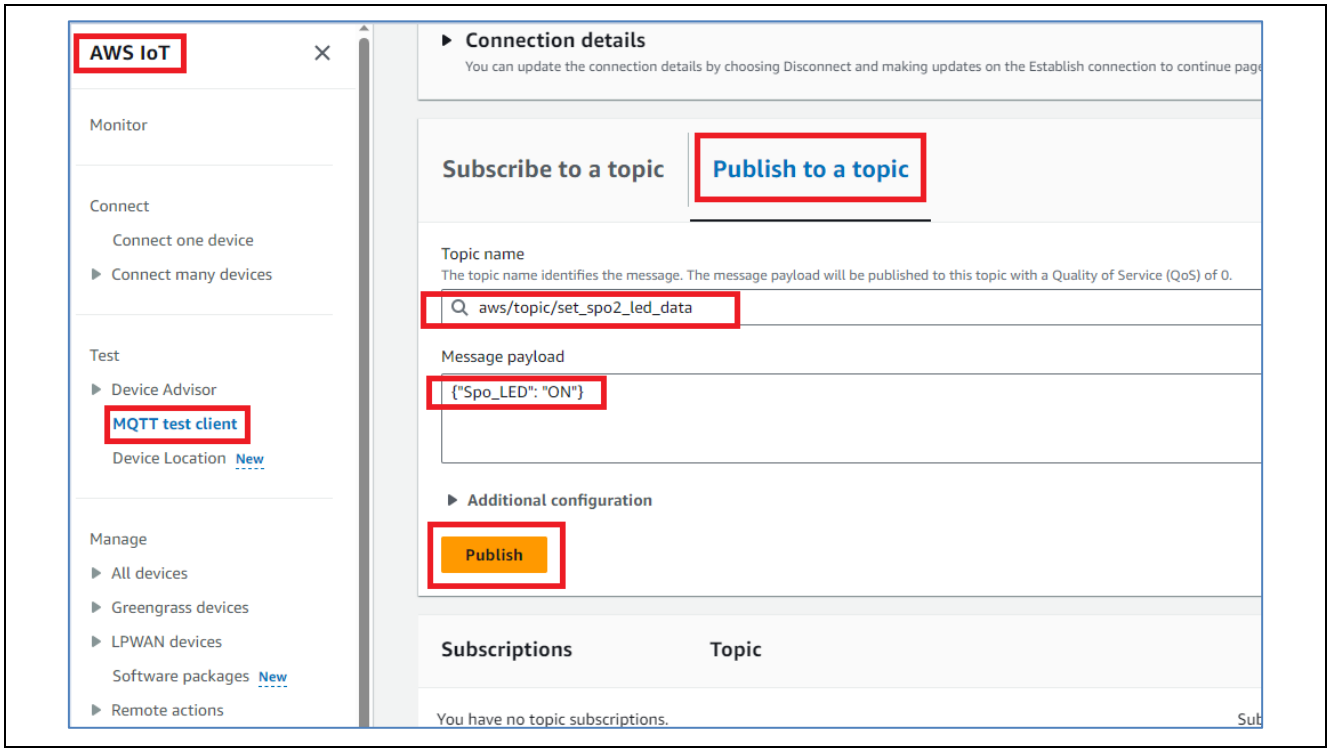

Figure 71. Publish the MQTT Message (2/2)

5.5.2.2 With Renesas Dashboard

(1) Dashboard Types

Depending on the sensors, you can choose one of the dashboard types: Renesas 9-Axis sensor or Renesas. Click on **Renesas** option.

Note: The 9-AXIS MEMS Motion Tracking Sensor TDK ICM-20948 is not fitted on this CK-RX65N board due to shortages in component availability (For more information: CK-RX65N v1 – Release Note (renesas.com))

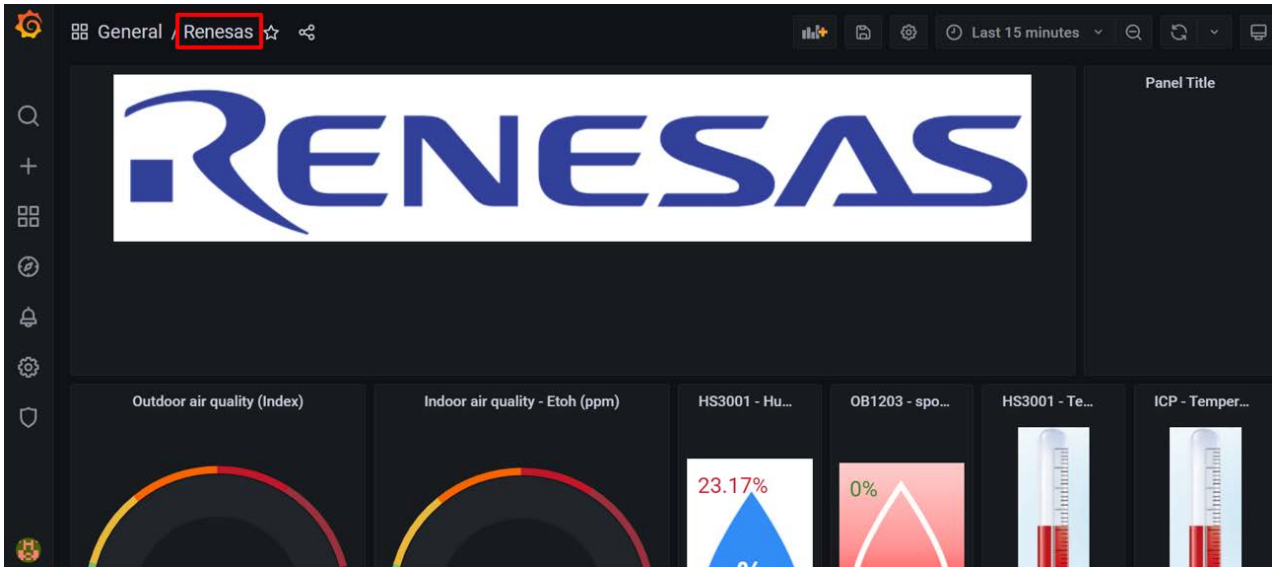

Figure 72. Renesas AWS Cloud Dashboard Types

Choose Renesas 9-Axis sensor.

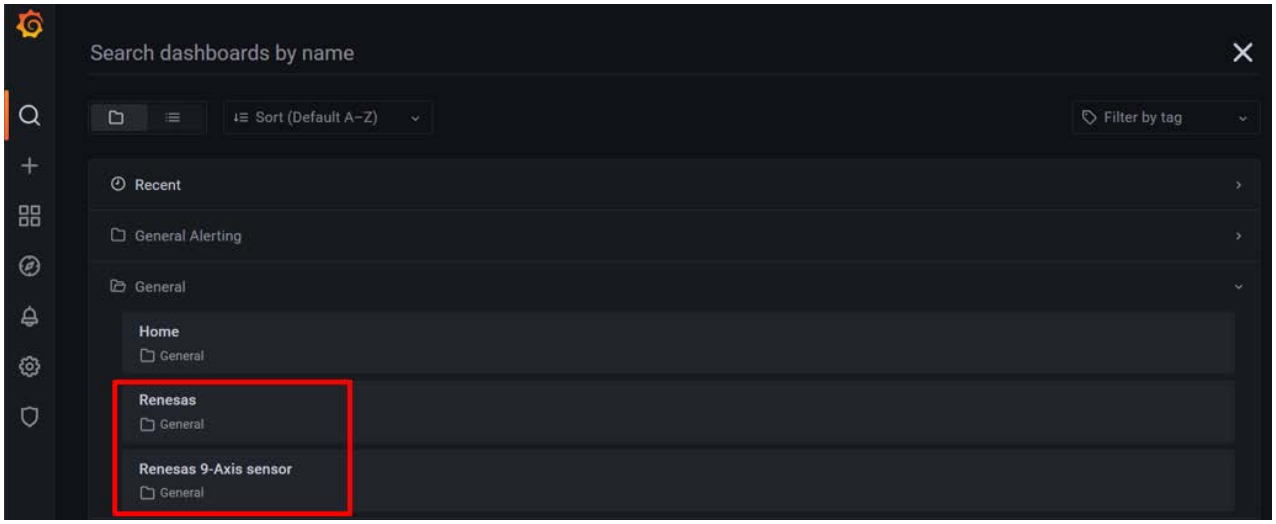

Figure 73. Choosing Renesas 9-Axis Sensor

Figure 74. Renesas Dashboard displays 9-Axis Sensor

(2) Sensor Data for Cloud Kits

The Grafana dashboard displays the following Data from sensors.

Table 9. Sensor Data from Grafana Dashboard

Sensor	Data
HS3001- Humidity and Temperature Sensor	Temperature, F
	Humidity, %
ZMOD4410- Indoor Air Quality Sensor	Etoh, ppm
	ECO2- Estimated Carbon dioxide, ppm
	TVOC - Total Organic Compounds, mg/m^3
OB1203 - Heart Rate, Blood Oxygen Concentration, Pulse Oximetry, Proximity, Light and Color Sensor	SPO2, %
	HR (Heart Rate), bpm (beats per minute)
	RR (Respiration Rate), breaths per minute
	P2P
ICP-10101 - Barometric Pressure and Temperature Sensor	Temperature, F
	Barometric Pressure, mbar
ICM-20948 Motion Tracking Sensor	Acc values, unit: g
	Gyro Data, unit: dps (degrees per sec)
	Mag Data, unit: mT
OAQ – Outdoor Air Quality	OAQ, ppm

(3) Alerting and Anomaly Detection

Grafana's alerts are ways 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

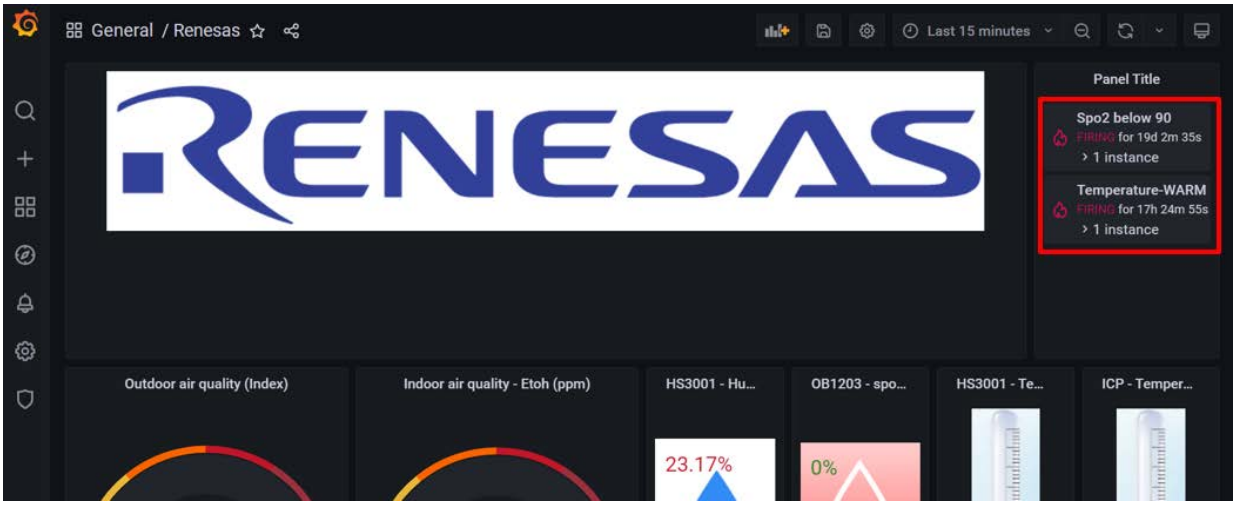

Figure 75. Sensor Status Feedback

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 limitations with this account's permission.

6.2 Prerequisites

6.2.1 Installing Python

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

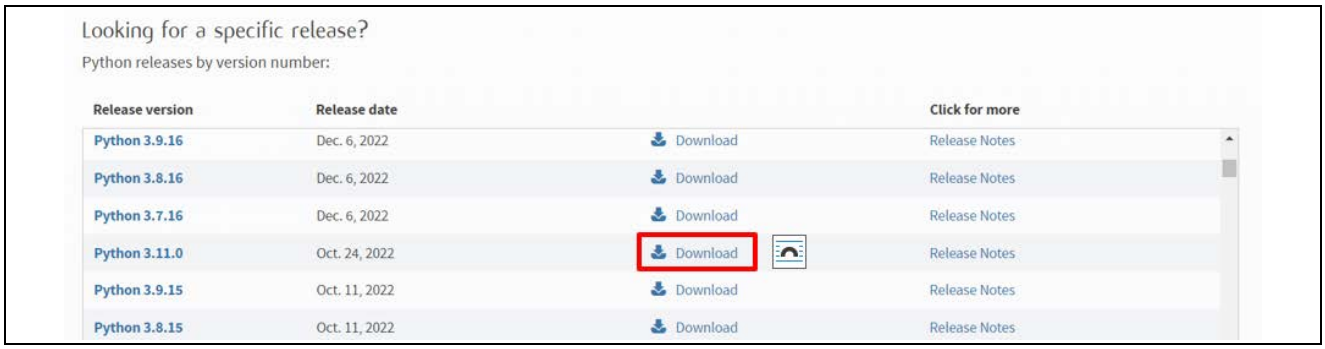

Figure 76. Python Download Website

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

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

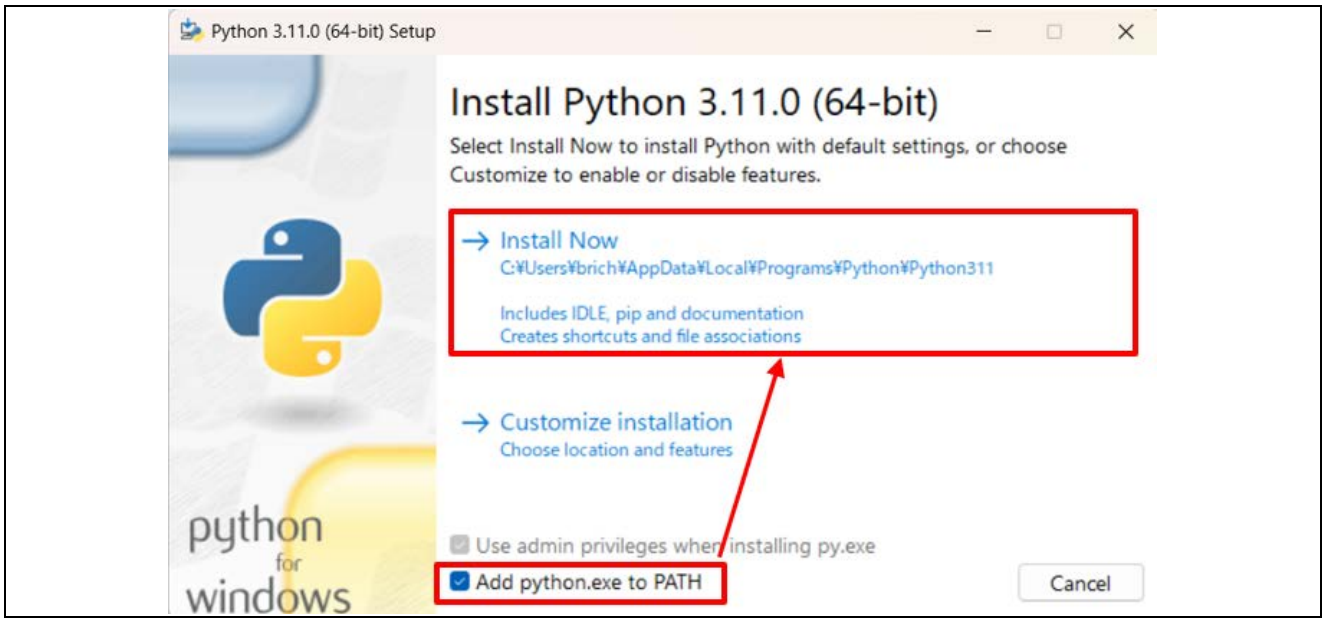

Figure 77. Python 3.11.0 installer

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 78. Checking version of Python

4. Install the Python encryption library (pycryptodome)

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

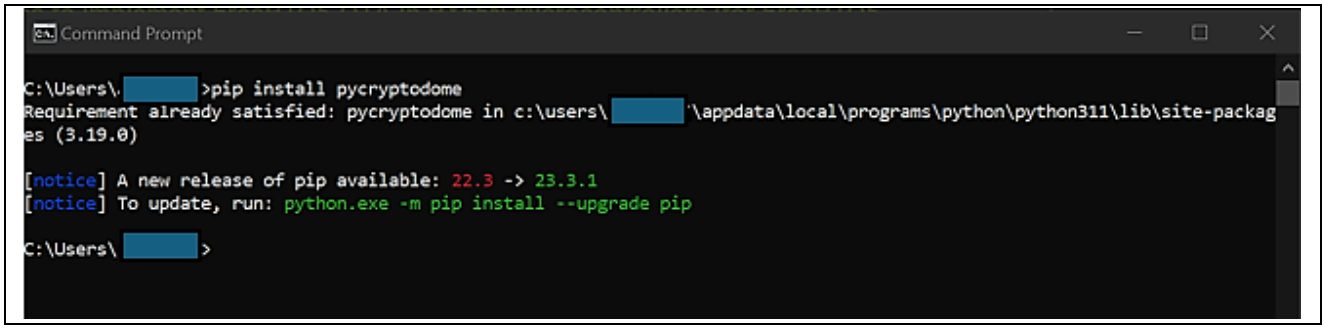

Figure 79. Installing Python encryption library

6.2.2 Installing OpenSSL

1. Access the Win32/Win64 download website for OpenSSL: OpenSSL Download Website and download the installer for the operating system you are using.

Win32 OpenSSL v3.0.12 EXE | MSI	120MB Installer

Figure 80. 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*

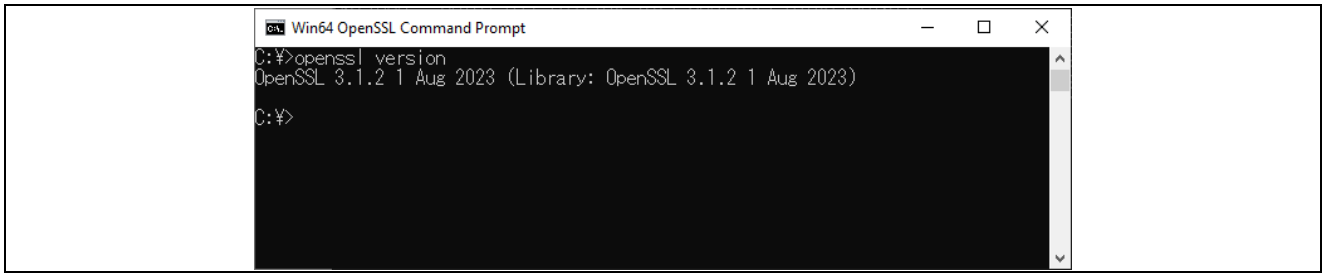

Figure 81. Checking version of OpenSSL

6.2.3 Installing Renesas Image Generator

Renesas Image Generator is a tool that generates the firmware images used by the firmware update module. This tool 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.

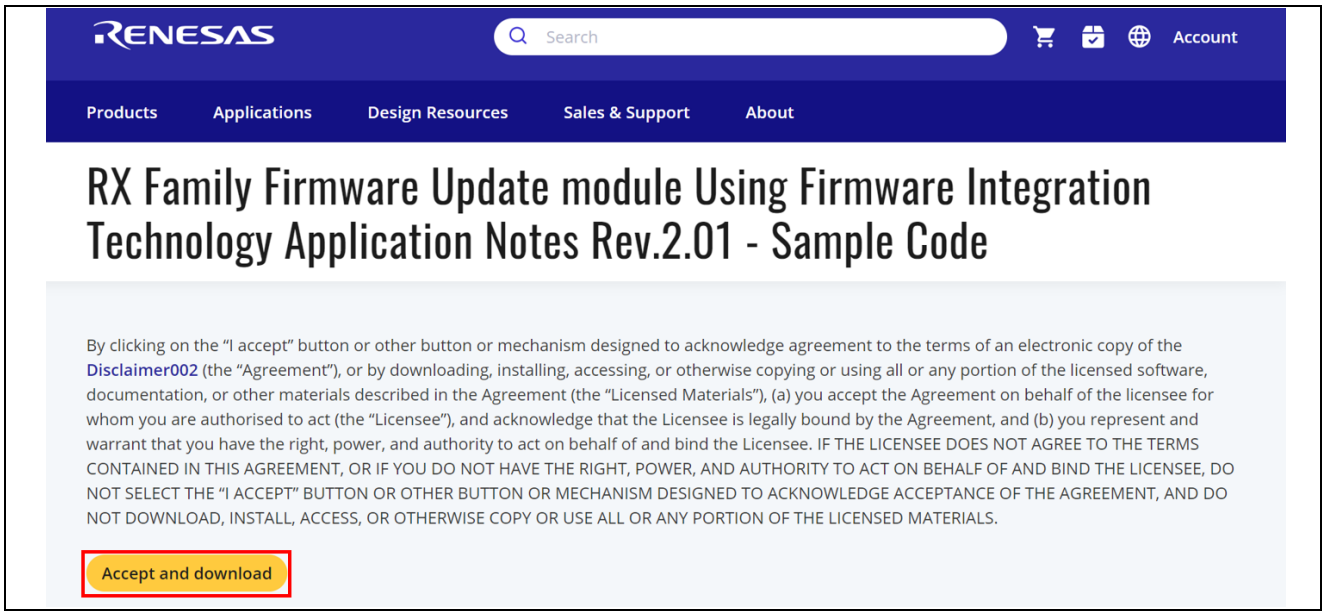

Figure 82. Renesas Image Generator Downloading

2. Extract the downloaded firmware update module.

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).

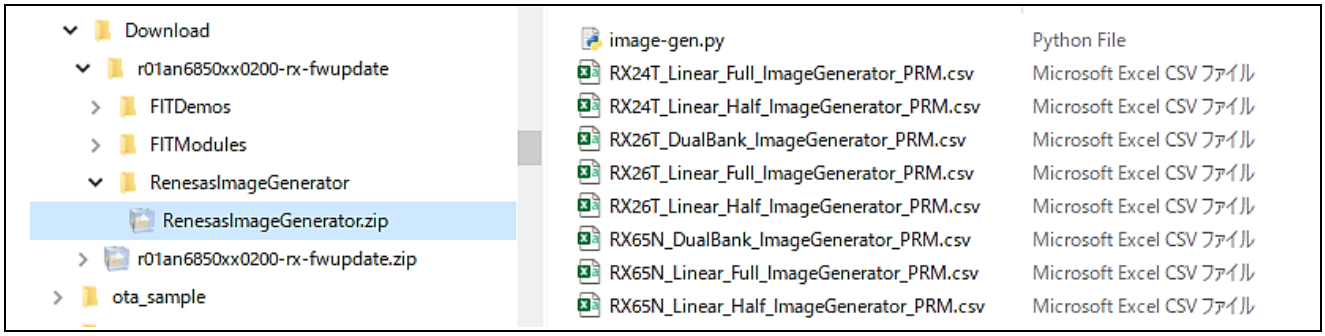

Figure 83. Renesas Image Generator package

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 the device will be updated.

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

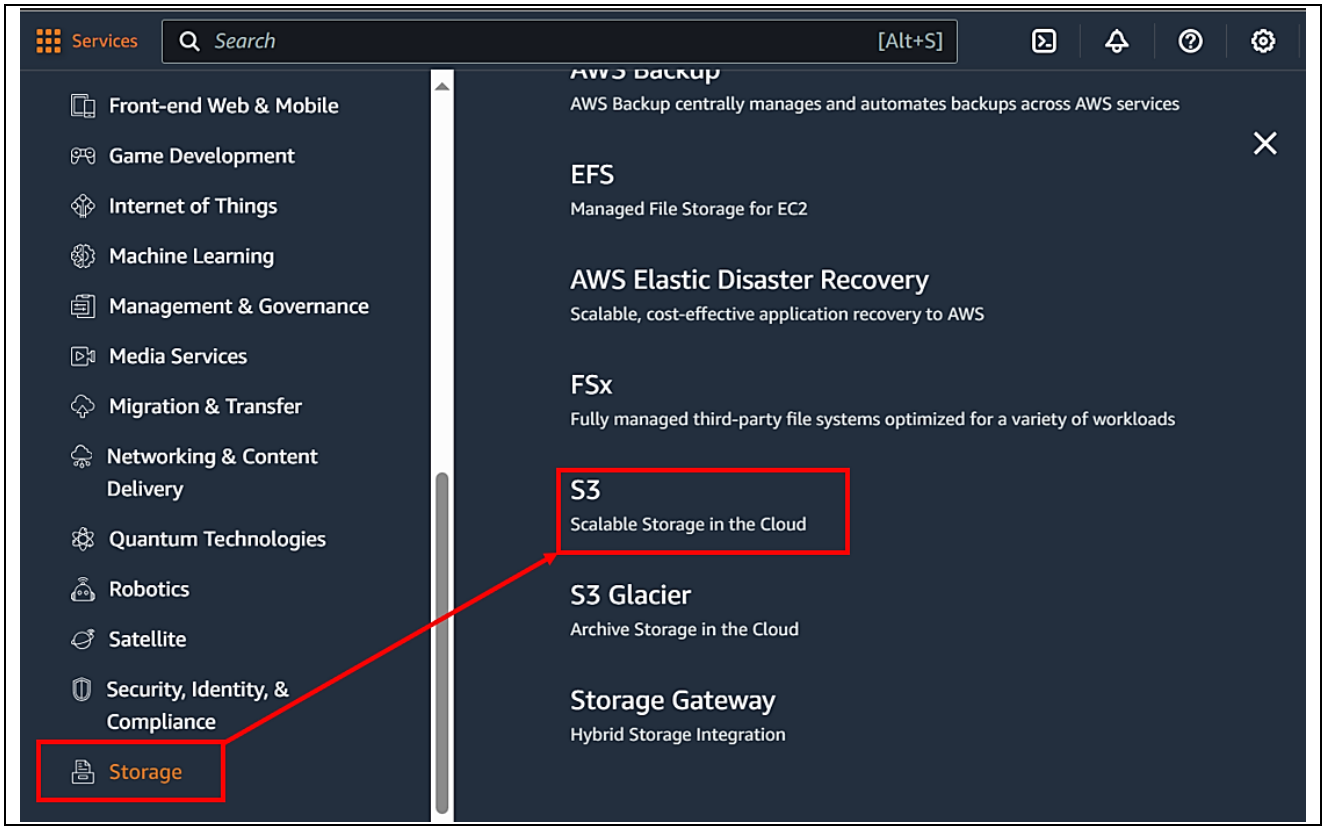

Figure 84. Create AWS S3 bucket (1/2)

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

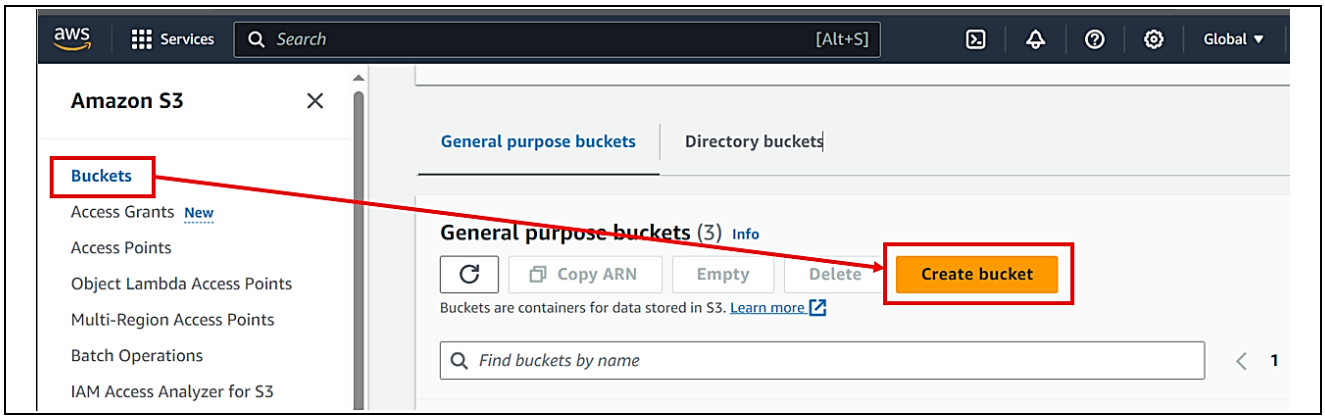

Figure 85. Create AWS S3 bucket (2/2)

3. Enter a bucket name (example: s3test-rx65n)

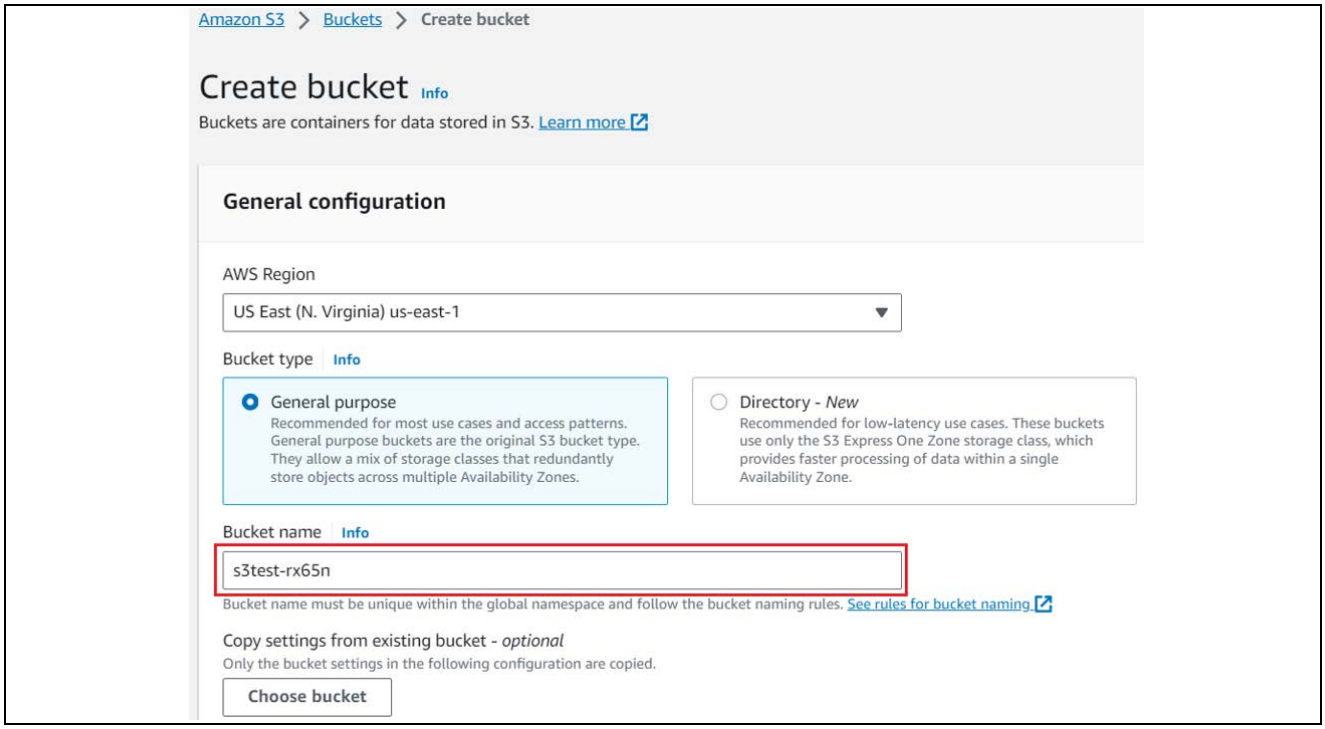

Figure 86. Naming for bucket

Note: The bucket name must be globally unique. And it follows the bucket naming rules.

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**

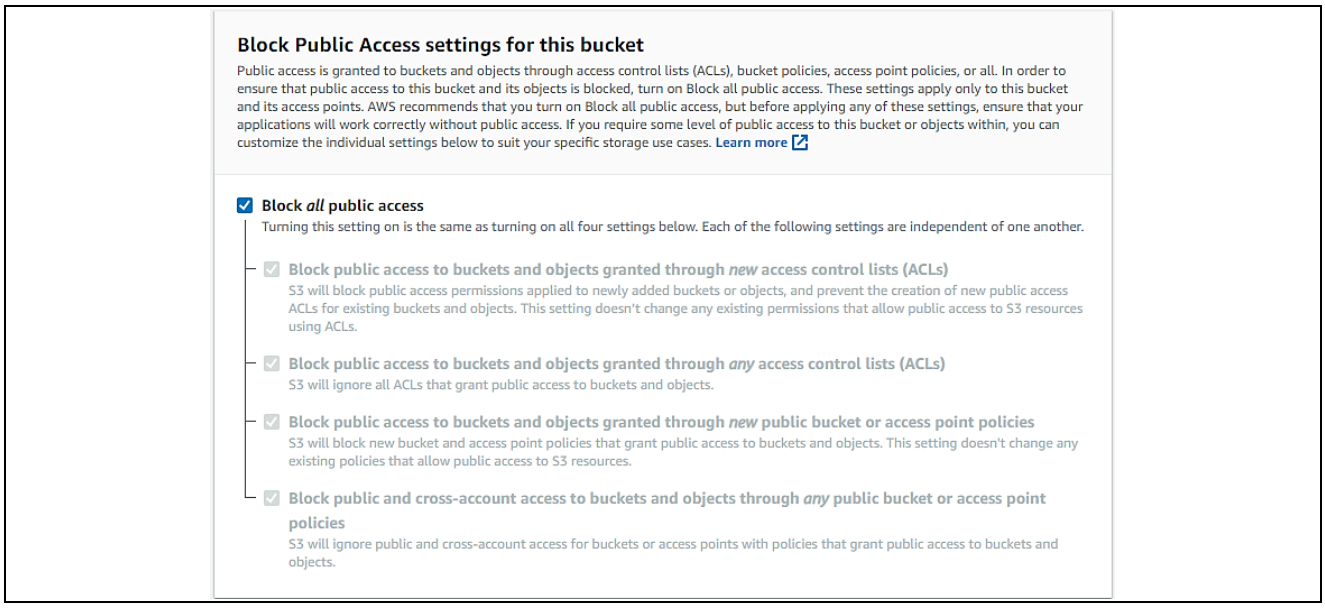

Figure 87. Bucket setting (1/2)

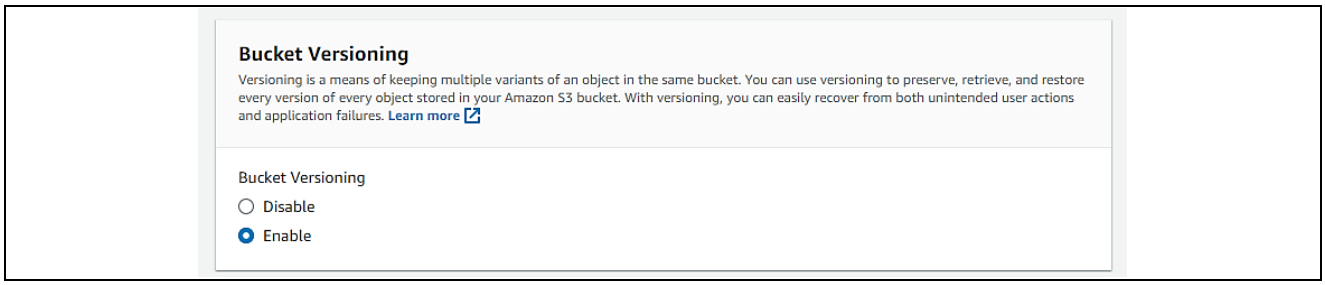

Figure 88. Bucket setting (2/2)

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.

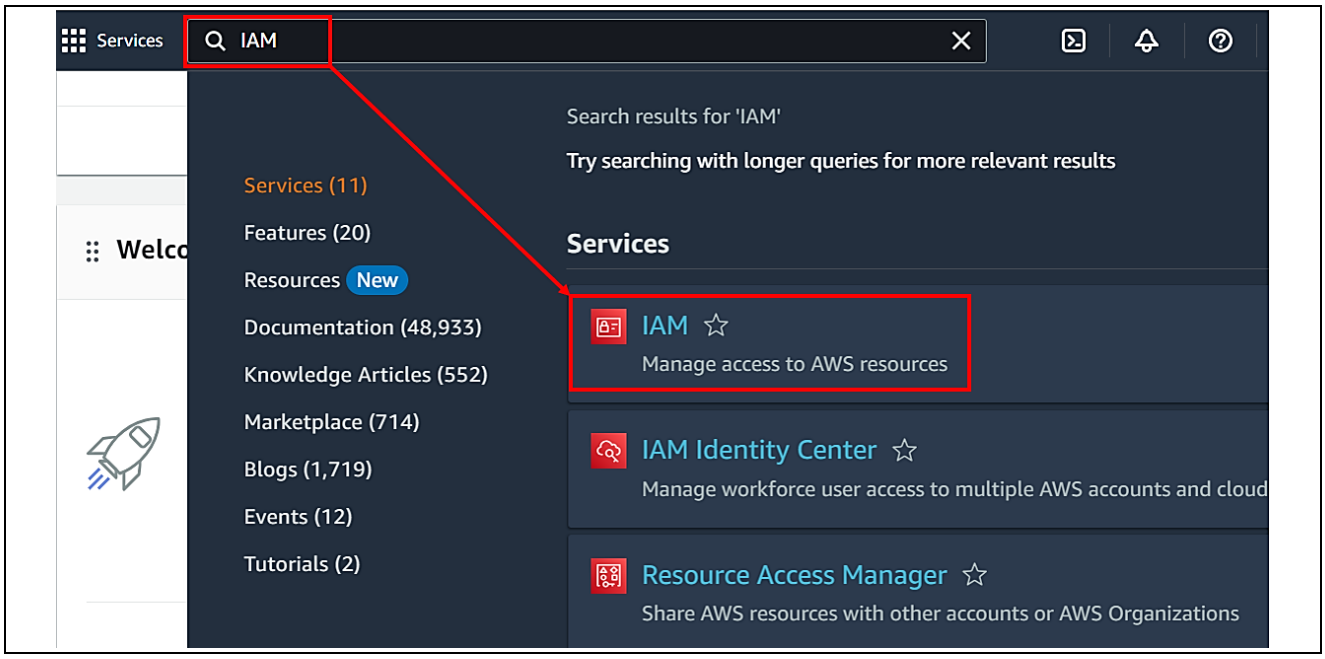

Figure 89. IAM search box

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

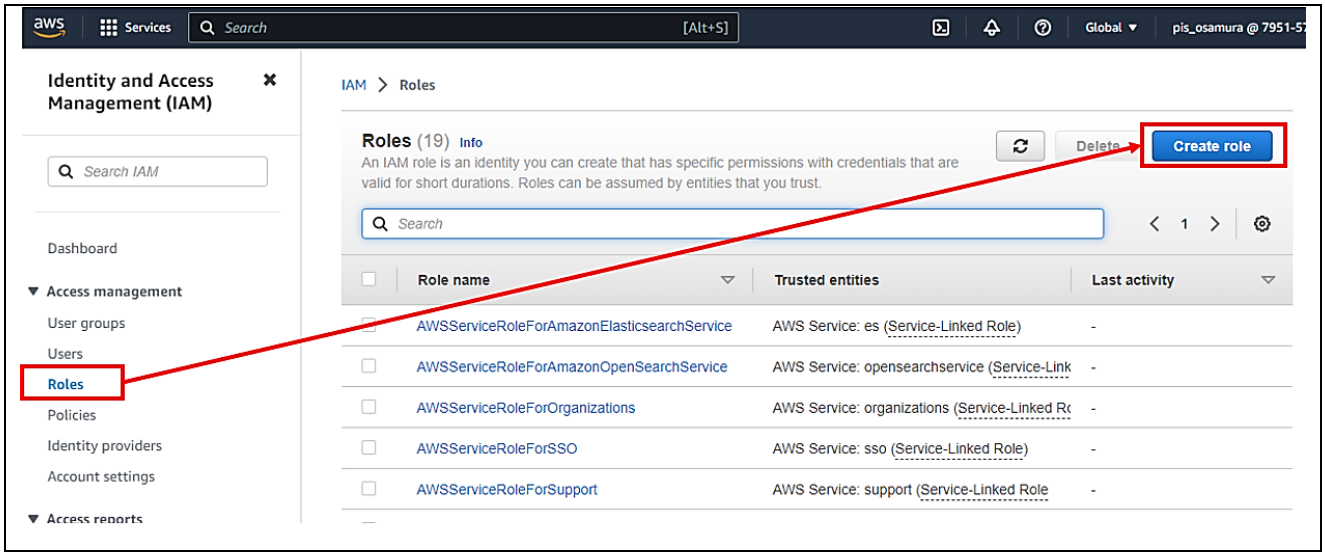

Figure 90. Creating a role

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

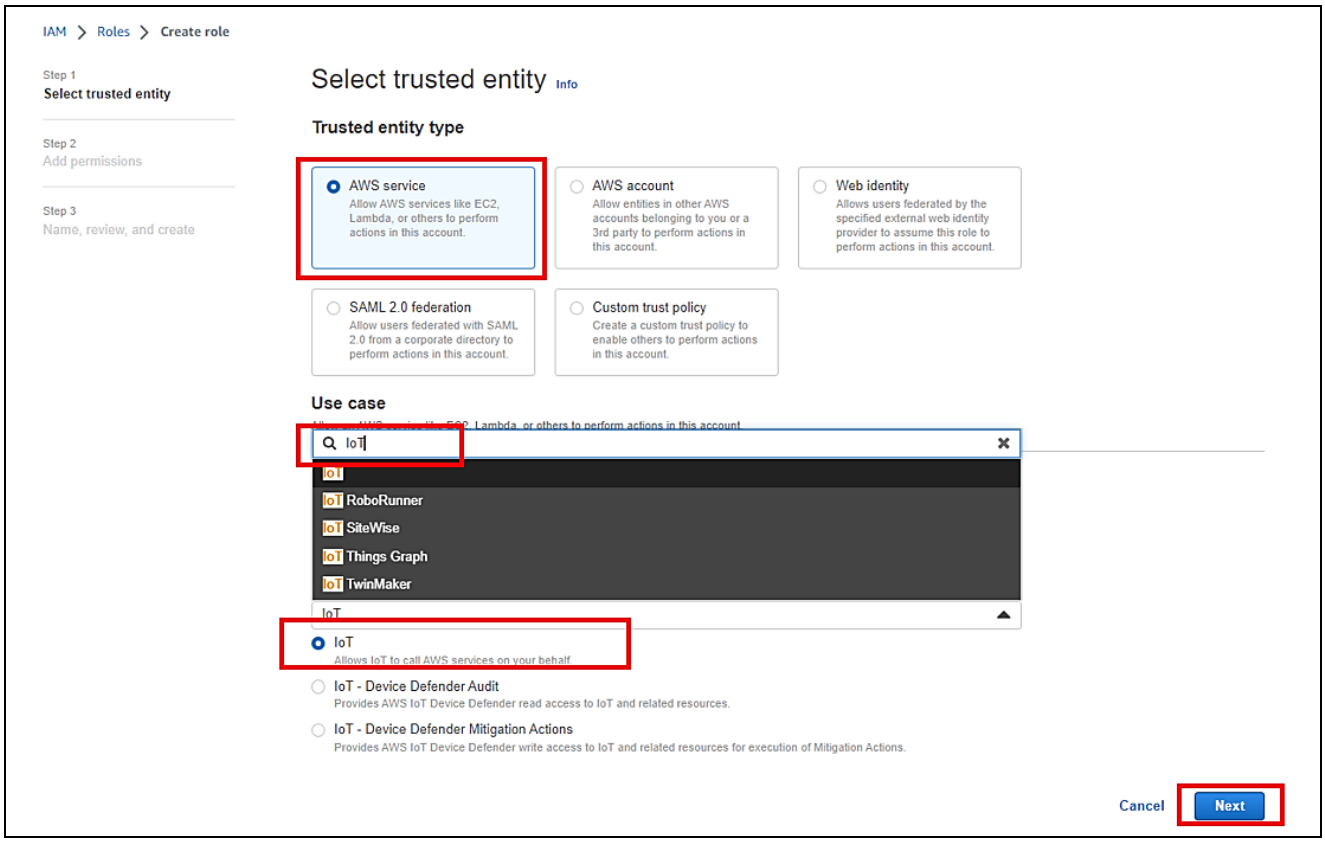

Figure 91. Selecting trusted entity

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

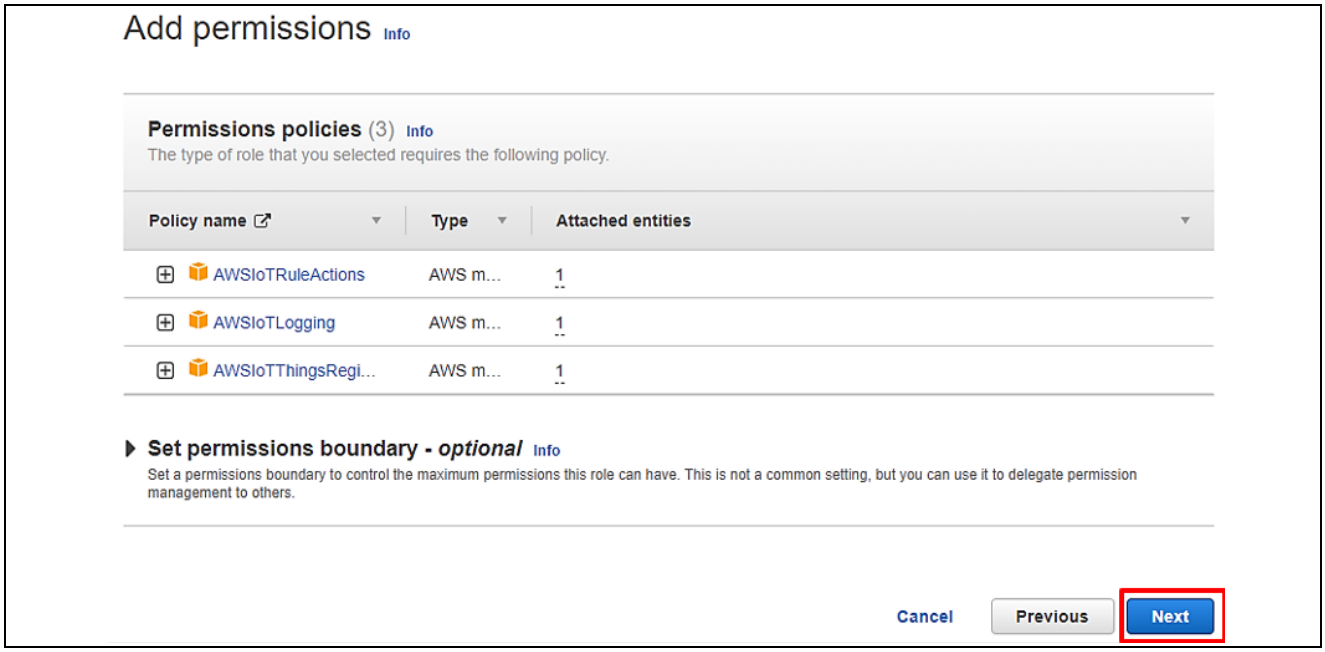

Figure 92. Add Permissions for Role

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

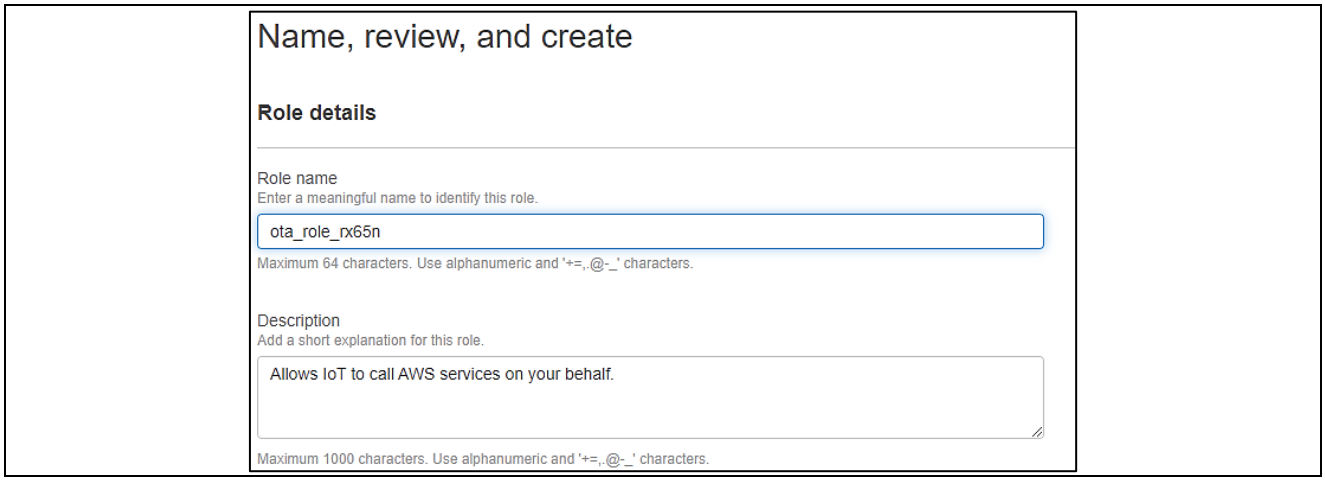

Figure 93. Naming for Created Role

6. Click on the role you created.

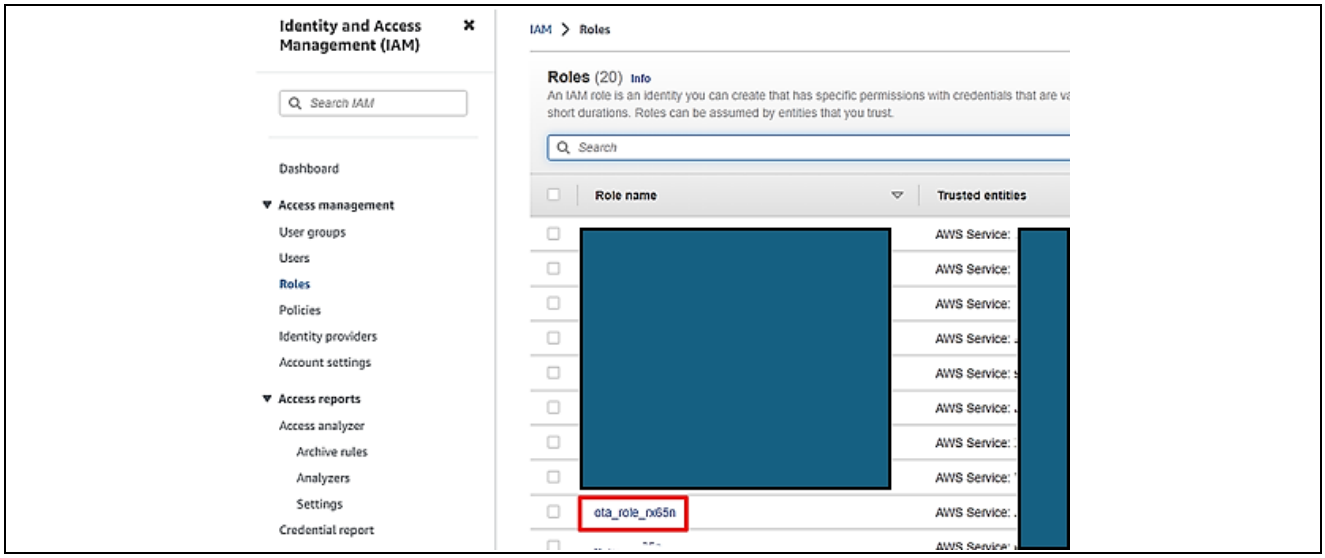

Figure 94. Created Role

7. Select **Attach policies**.

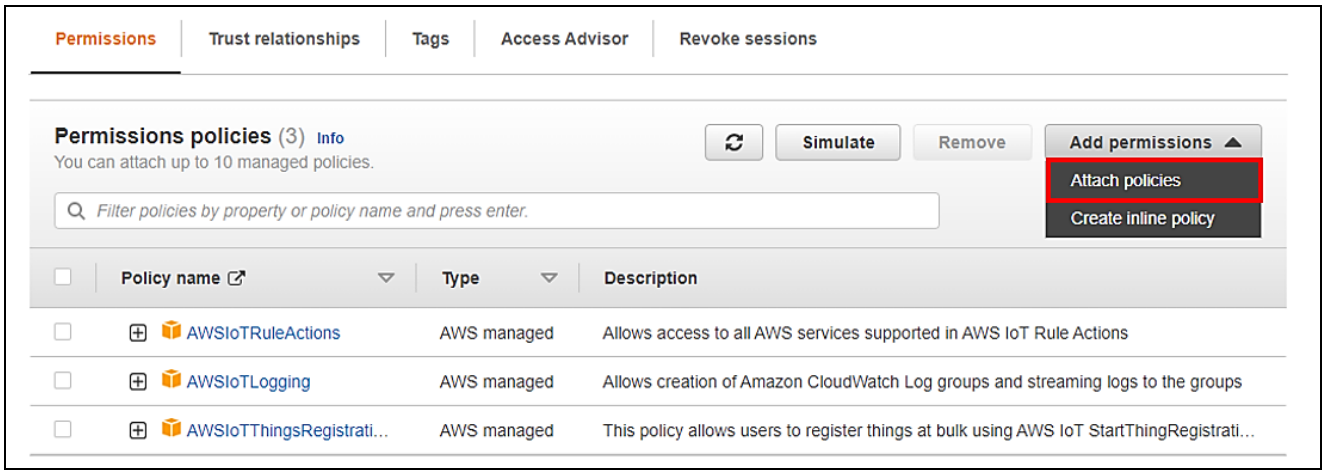

Figure 95. Add permission (1/3)

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

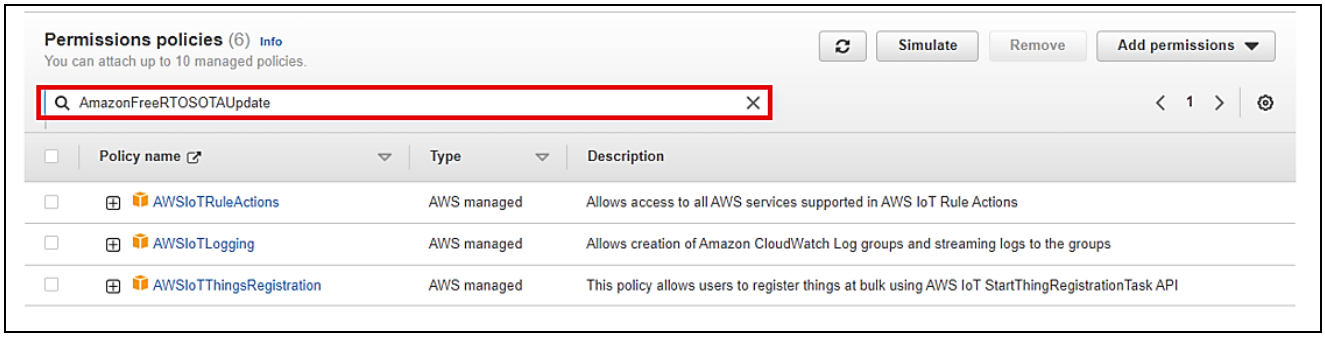

Figure 96. Add permission (2/3)

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

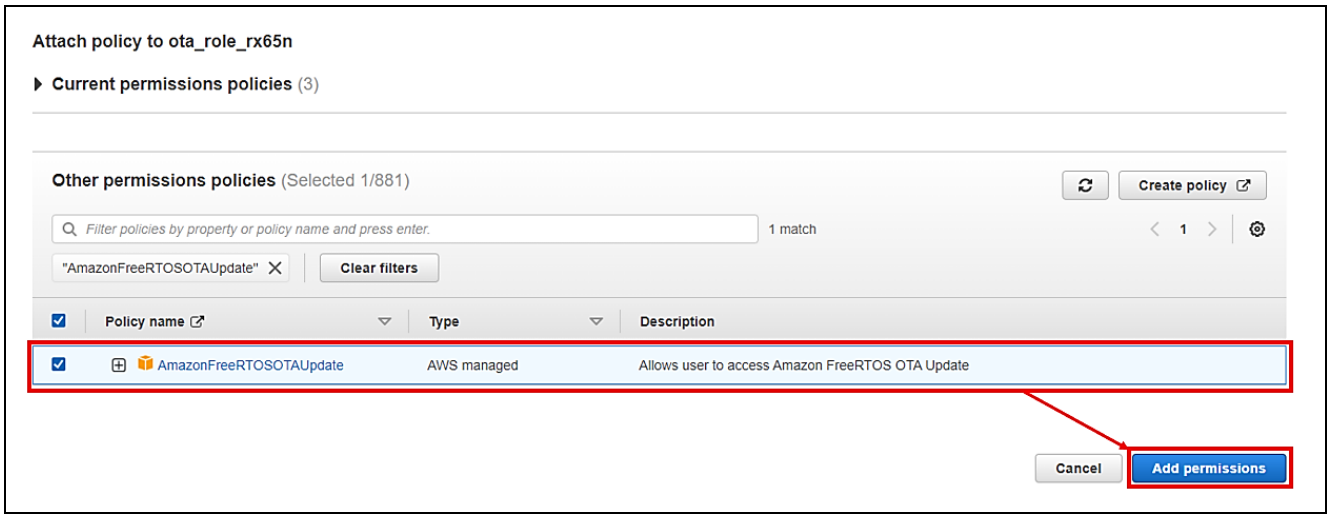

Figure 97. Add permission (3/3)

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

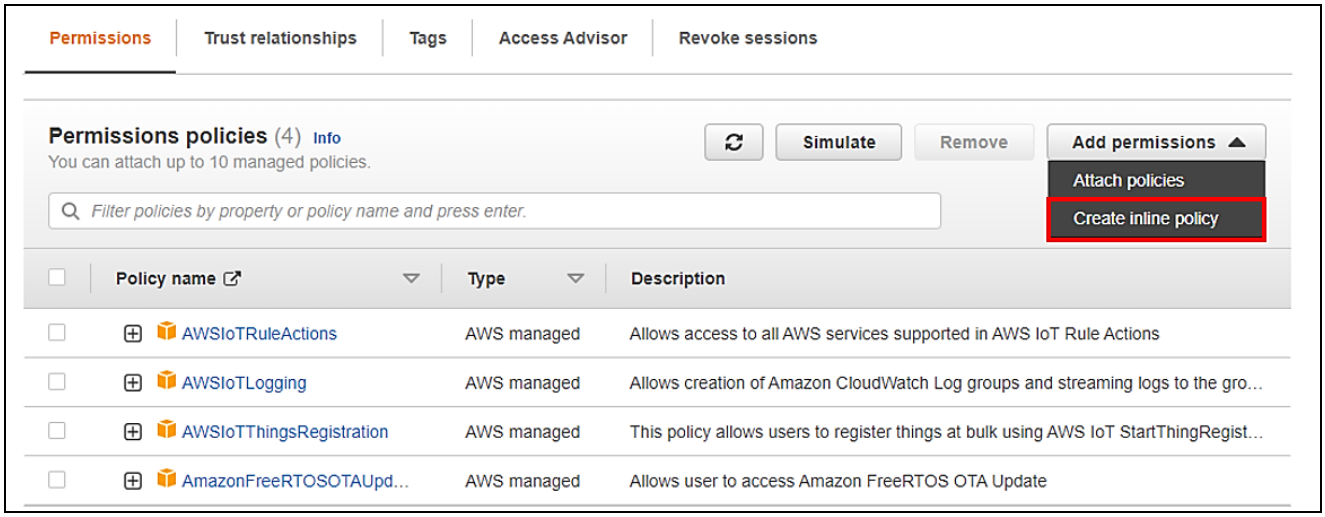

Figure 98. Create inline policy (1/3)

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”: “*”
     }
   ]
}
```

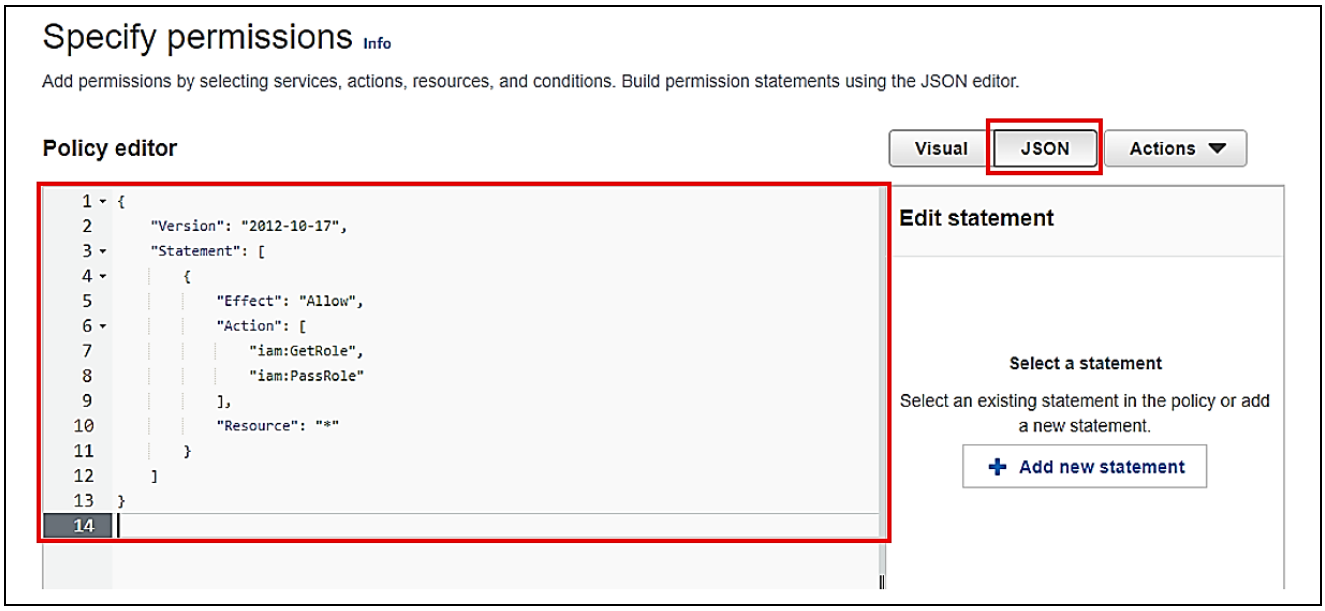

Figure 99. Create inline policy (2/3)

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

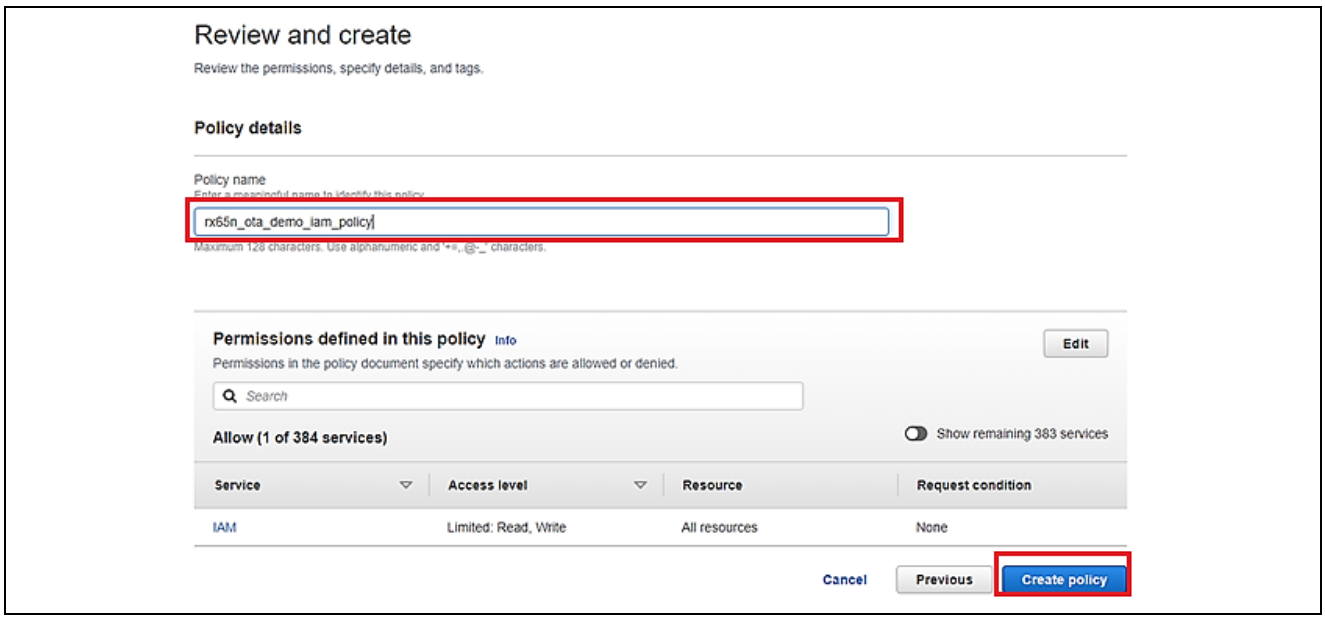

Figure 100. Create inline policy (3/3)

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

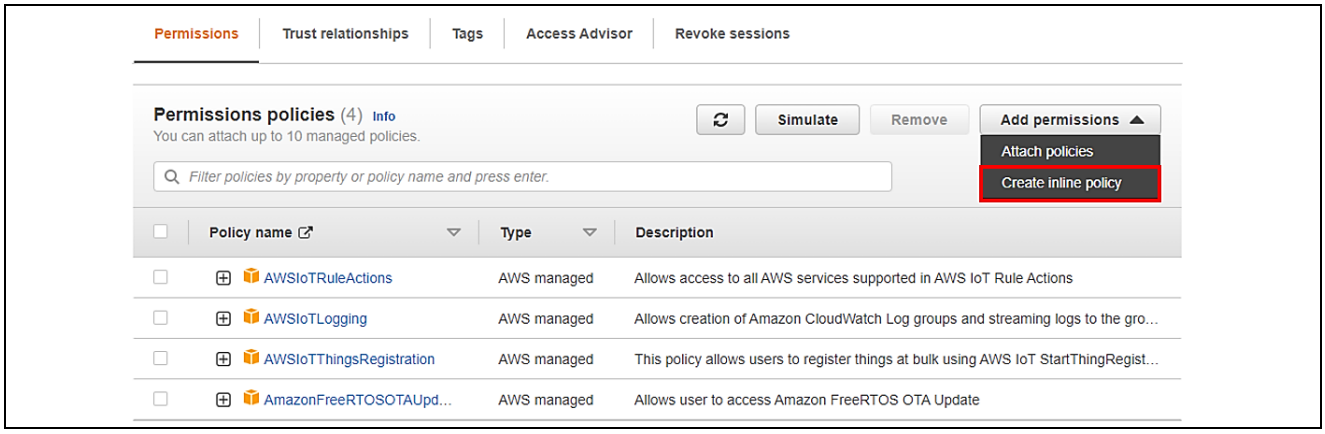

Figure 101. Create inline policy (1/3)

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”: [
            “*”
        ]
    }
  ]
}
```

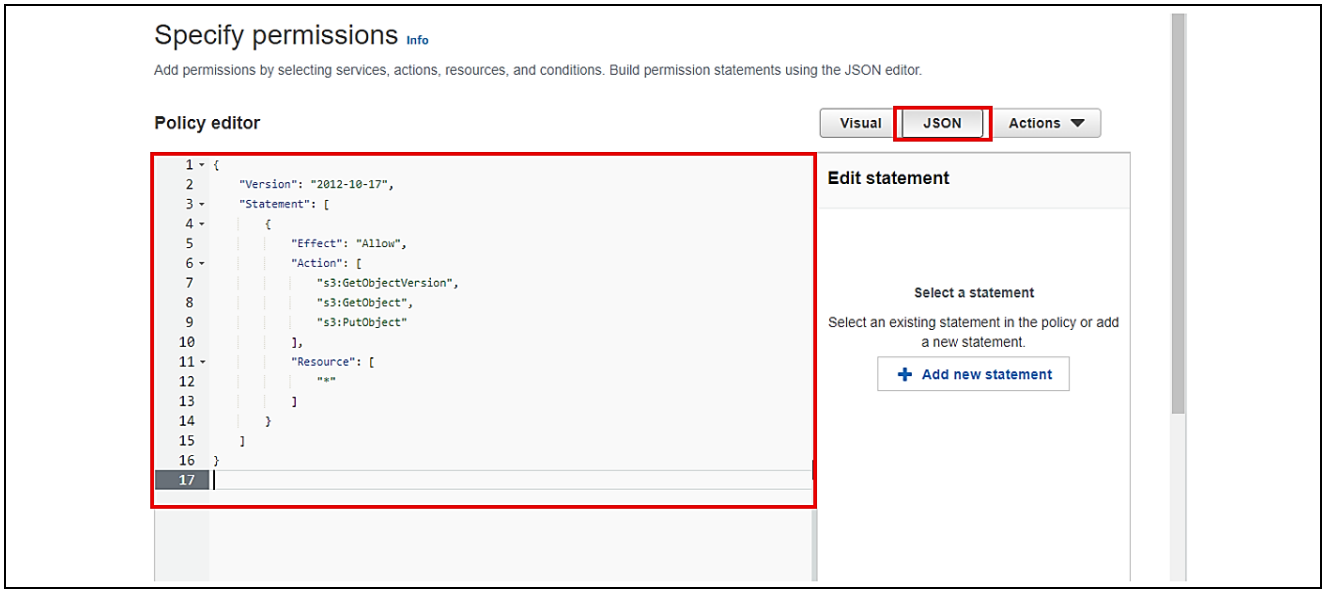

Figure 102. Create inline policy (2/3)

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

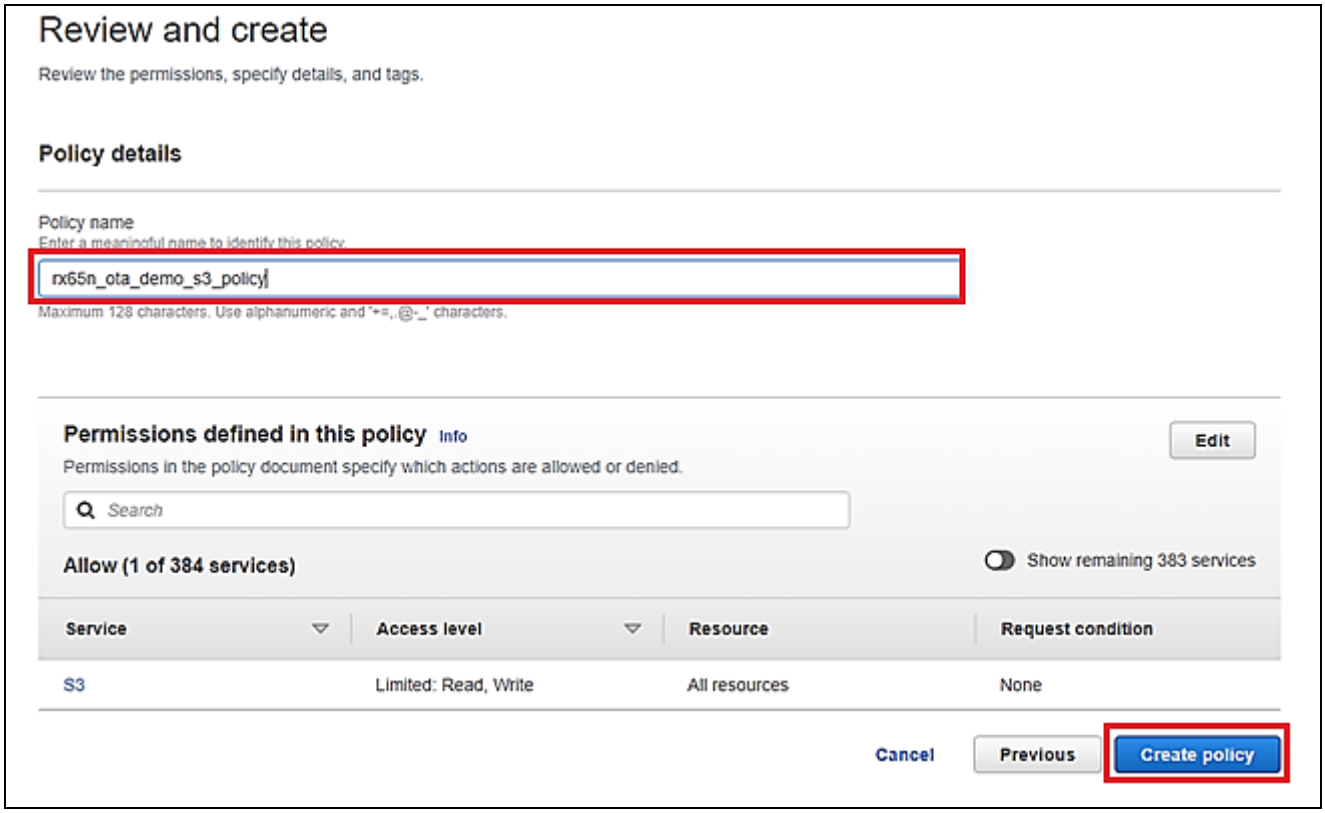

Figure 103. Create inline policy (3/3)

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 104. 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 105. 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 106. 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*

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

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

Figure 107. 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*

```
C:¥openssl>openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out secp256r1.crt
Signature ok
subject=C = [redacted], ST = [redacted], L = [redacted], O = [redacted], OU = [redacted], CN = [redacted], emai
lAddress = [redacted]
Getting CA Private Key
```

Figure 108. 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

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

Figure 109. 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

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

Figure 110. Create public key (secp256r1.publickey)

8. After creating, please check the result:

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

Figure 111. Total created files

6.4.2 Setting up the project

6.4.2.1 Creating initial firmware

The following steps explain how to create the initial firmware that combines the boot loader (boot_loader_ck_rx65n) and the firmware (aws_ryz014a_ck_rx65n (when using Cellular project), or aws_ether_ck_rx65n (when using Ethernet project).

1. Additional import **boot_loader_ck_rx65n** project:

Make sure that the configuration in **boot_loader_ck_rx65n.scfg** file of boot_loader_ck_rx65n project is as below:

Table 10. Components Configuration of boot loader project

No	Component	Configuration
1	**Startup→Generic→r_bsp (v7.20)**	Enable user stdio charput function: **Use user charput() function**
2	**Drivers→Memory→r_flash_rx (v5.00)**	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 (v4.40)**	Include software support for channel 5: **Include**
4	**Middleware→Generic→r_byteq (v2.00)**	Memory allocation for queue control blocks: **Static memory allocation**
		Number of static queue control blocks: **32**
5	**Middleware→Generic→r_fwup (v2.01)**	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.

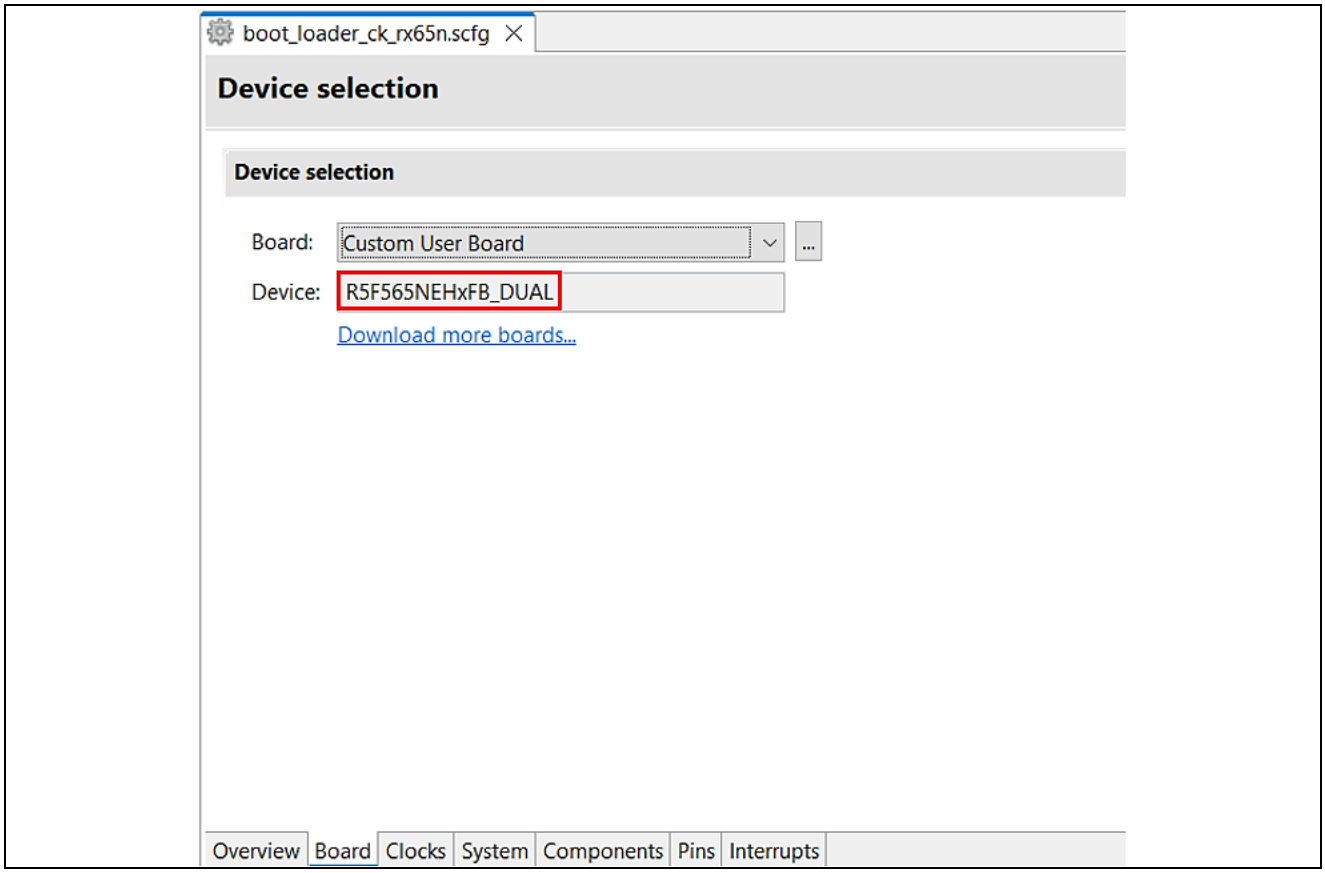

Figure 112. bootloader's device selection

Assign public key: Copy the contents of the secp256r1.publickey file you created in **"Extract the public key from the ECDSA key pair."**, and paste the contents into **CODE_SIGNER_PUBLIC_KEY_PEM** defined in the following files: boot_loader_ck_rx65n\src\key\code_signer_public_key.h

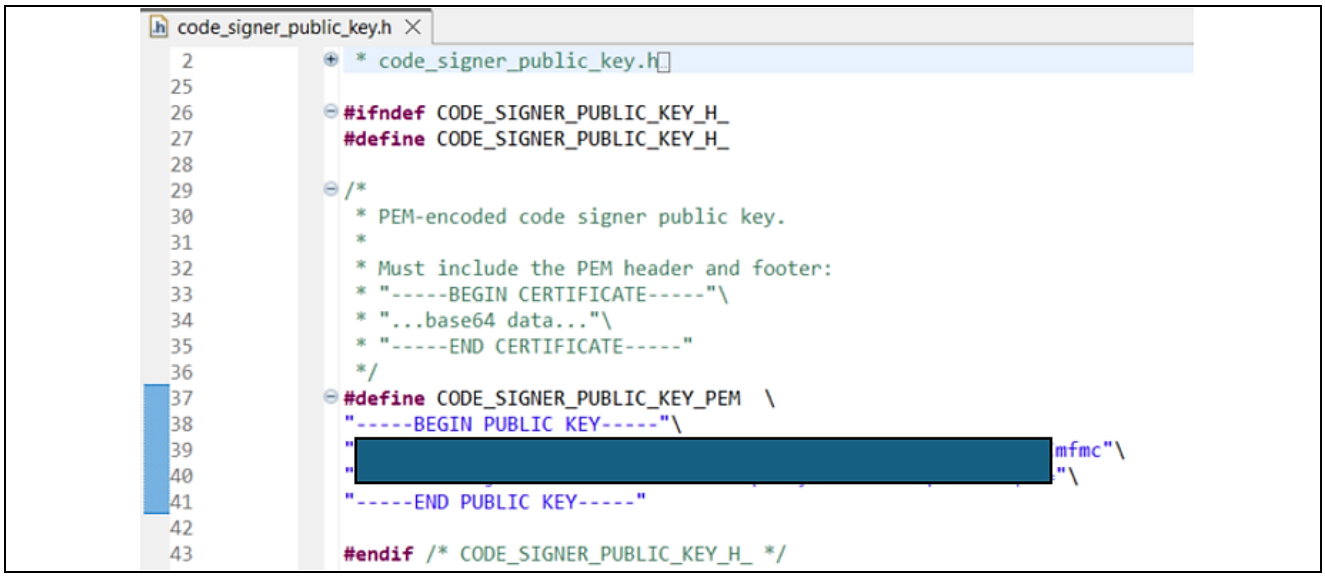

Figure 113. Add Code Signing Public Key to Boot Loader Project

2. Set **ENABLE_OTA_UPDATE_DEMO** to 1 (Enable) in [Project_Loc]\e2studio_ccrx\src\frtos_config\demo_config.h. (The default is 0)

```
demo_config.h
77   /* Please select a provisioning method
78    * (0) : Pre-provisioning
79    * (1) : Fleet provisioning
80    */
81   #define ENABLE_FLEET_PROVISIONING_DEMO      (0)
82
83   /* Please select whether to enable or disable the OTA demo
84    * (0) : OTA demo is disabled
85    * (1) : OTA over MQTT demo is enabled
86    */
87   #define ENABLE_OTA_UPDATE_DEMO              (1)
88
89   #define democonfigROOT_CA_PEM               tlsSTARFIELD_ROOT_CERTIFICATE_PEM
90
92    * @brief Path of the file containing the provisioning claim certificate. This
109  #define democonfigCLAIM_CERT_PEM            "...insert here..."
110
112   * @brief Path of the file containing the provisioning claim private key. This
122  #define democonfigCLAIM_PRIVATE_KEY_PEM     "...insert here..."
123
125   * @brief An option to disable Server Name Indication.
131  #define democonfigDISABLE_SNI               ( pdFALSE )
```

Figure 114. Enable OTA Demo

3. Confirm the initial project version:

   Confirm the version definitions in [Project_Loc]\src\frtos_config\demo_config.h:

- APP_VERSION_MAJOR
- APP_VERSION_MINOR
- APP_VERSION_BUILD

Example: The initial project version is 0.9.2

```
demo_config.h
330  /**
331   * @brief Major version of the firmware.
332   *
333   * This is used in the OTA demo to set the appFirmwareVersion variable that is
334   * declared in the ota_appversion32.h file in the OTA library.
335   */
336  #ifndef APP_VERSION_MAJOR
337      #define APP_VERSION_MAJOR    0
338  #endif
339
341   * @brief Minor version of the firmware.
346  #ifndef APP_VERSION_MINOR
347      #define APP_VERSION_MINOR    9
348  #endif
349
351   * @brief Build version of the firmware.
356  #ifndef APP_VERSION_BUILD
357      #define APP_VERSION_BUILD    2
358  #endif
```

Figure 115. The initial project version 0.9.2

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

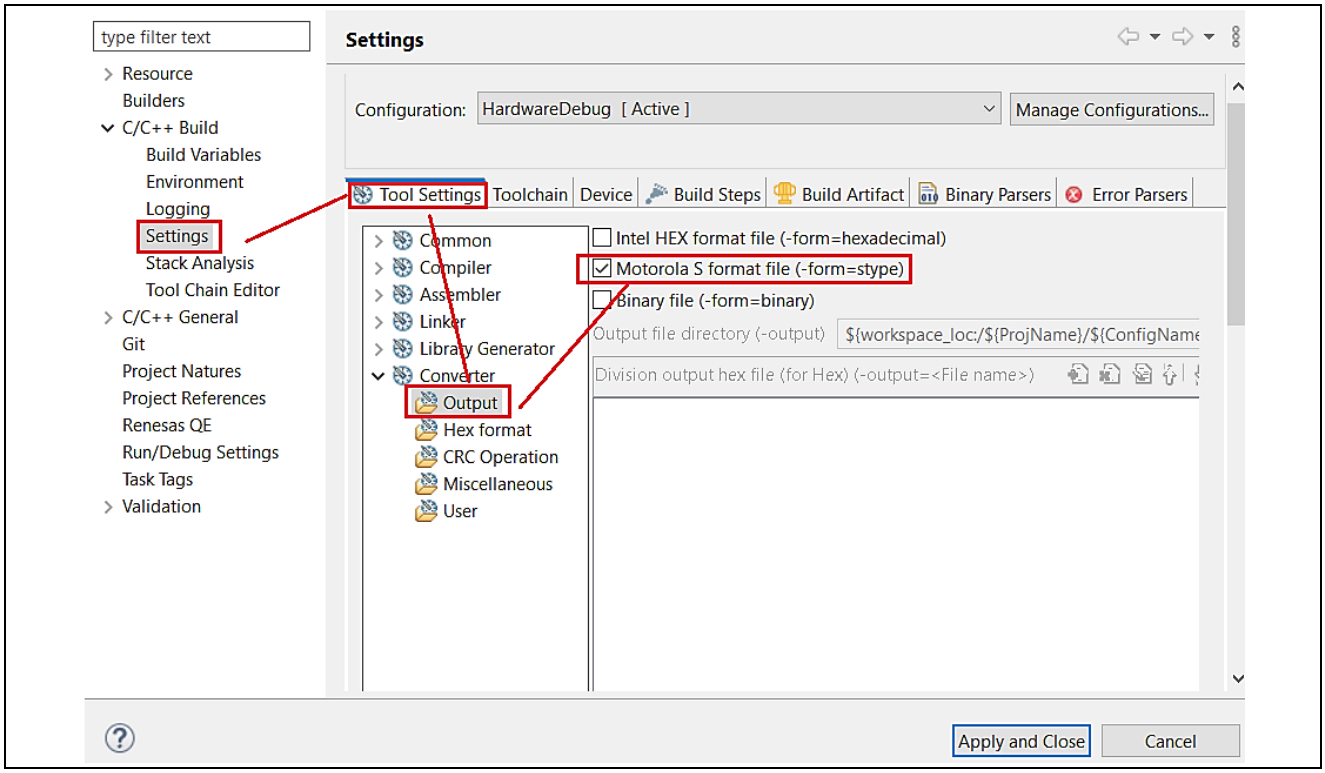

Figure 116. Project setting

4. Device selection setting

   Open the file aws_ether_ck_rx65n.scfg and click the **Board** tab. Click the ellipsis (**…**) beside the **Board** field in the **Device selection** area.

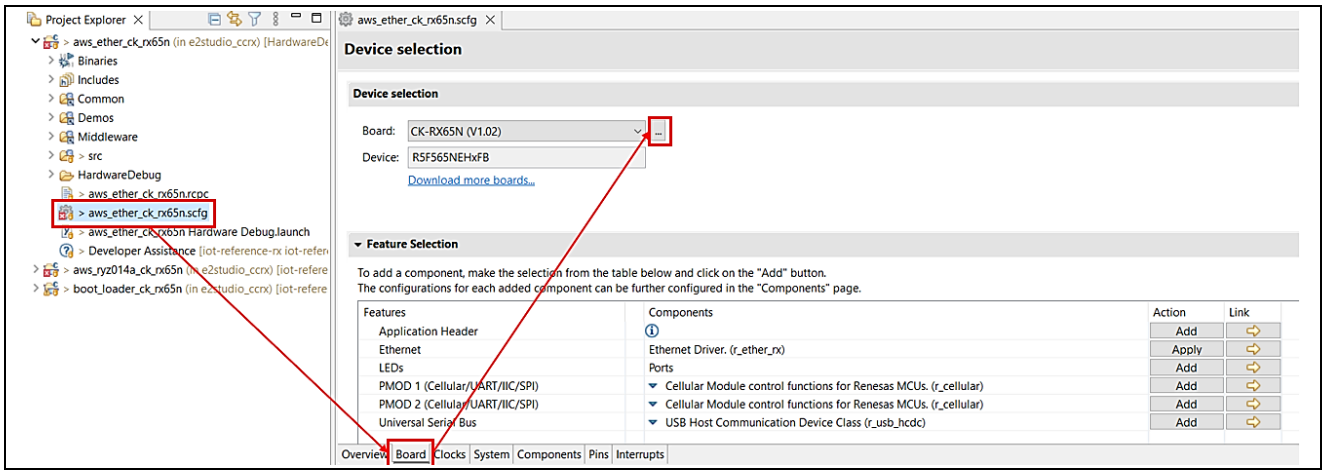

Figure 117. Device selection (1/3)

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

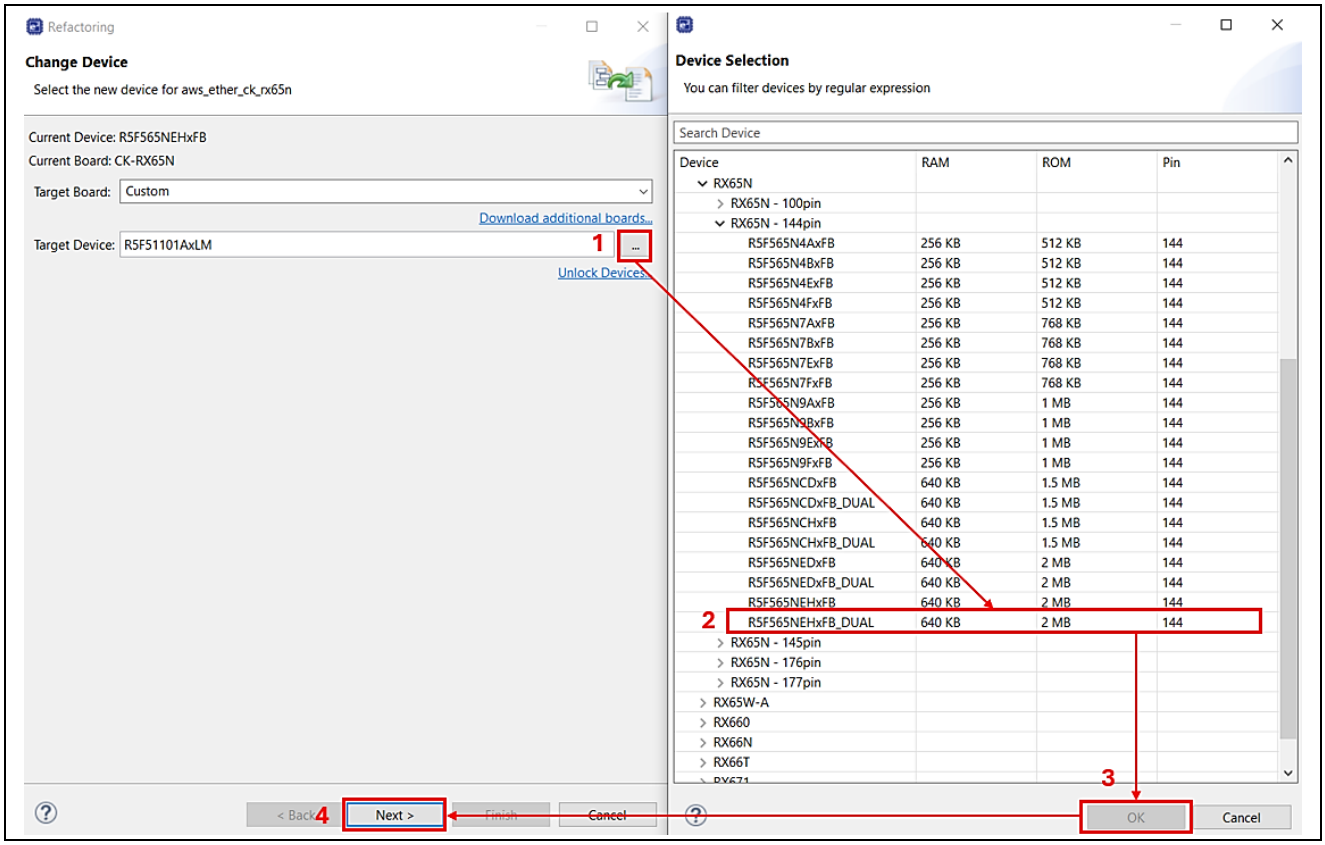

Figure 118. Device selection (2/3)

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

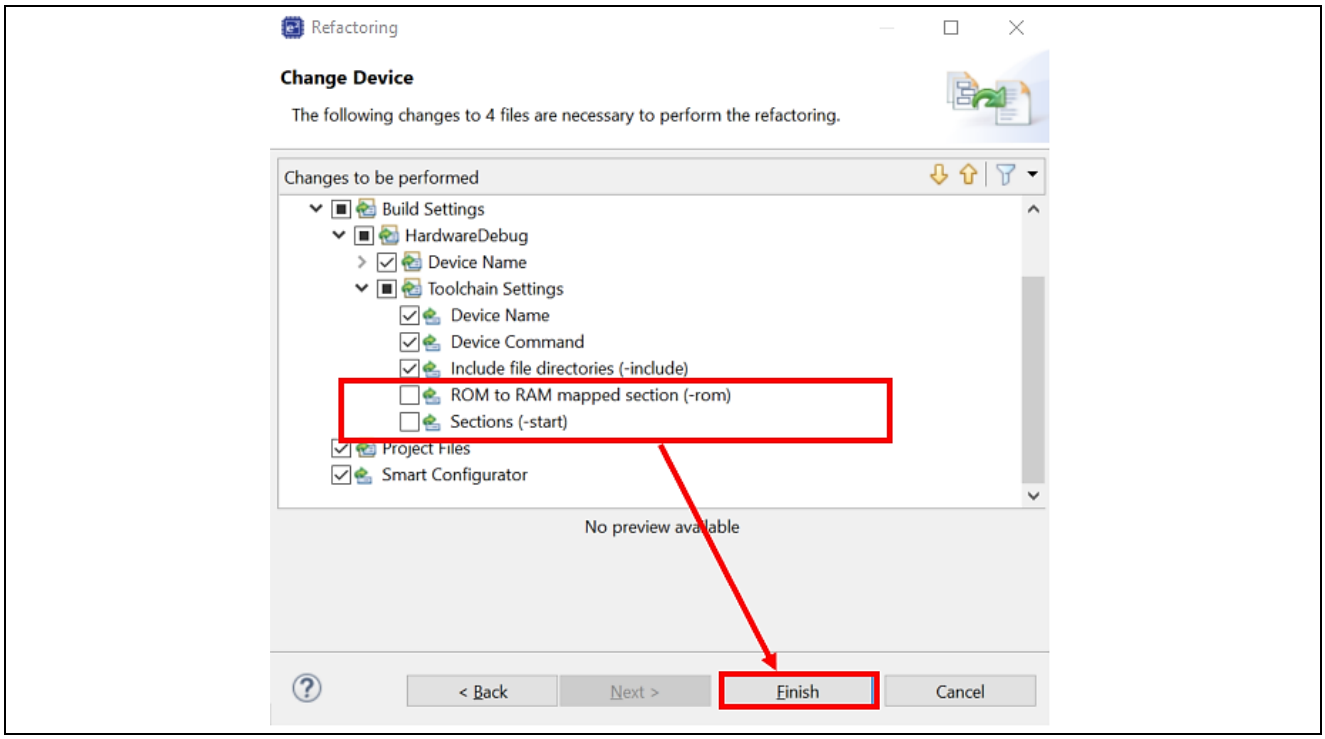

Figure 119. Device selection (3/3)

5. Change the firmware (**aws_ether_ck_rx65n** (or **aws_ryz014a_ck_rx65n**)) vector.

Open the **aws_ether_ck_rx65n** (or **aws_ryz014a_ck_rx65n)** project, 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.**

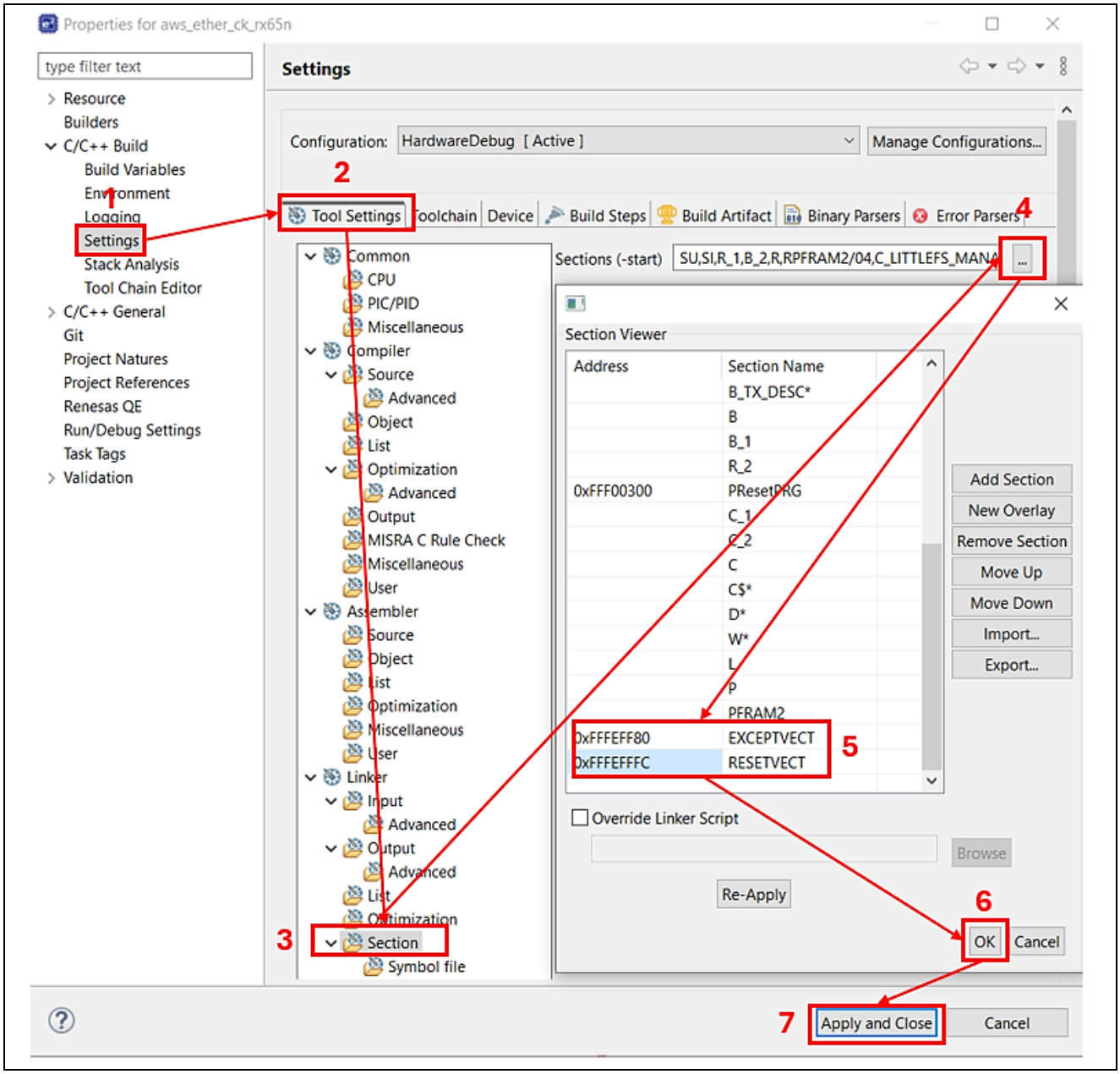

Figure 120. Vector setting

Build the project as described in Topic: **5.1.3 Importing the Project**

6. Generate the initial firmware.

Place the following files in the Renesas Image Generator folder:

The results of building the firmware: **aws_ether_ck_rx65n.mot (or aws_ryz014a_ck_rx65n.mot)**
The results of building the boot loader: **boot_loader_ck_rx65n.mot**
The private key created in **"Extract the private key from the ECDSA key pair."**: **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_ether_ck_rx65n.mot* -ip *RX65N_DualBank_ImageGenerator_PRM.csv* -o *userprog* -ibp *boot_loader_ck_rx65n.mot* -key *secp256r1.privatekey* -vt ecdsa -ff RTOS

6.4.2.2 Running OTA project

1. Start Renesas Flash Programmer and create new project:

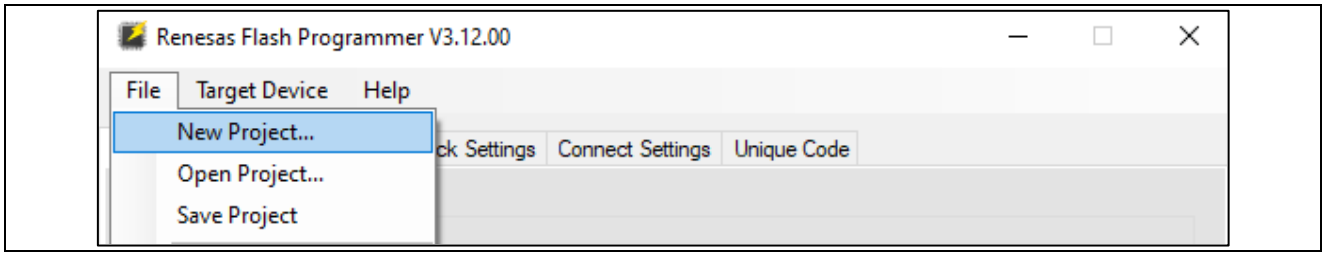

Figure 121. Create New Flash Project (1/4)

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.

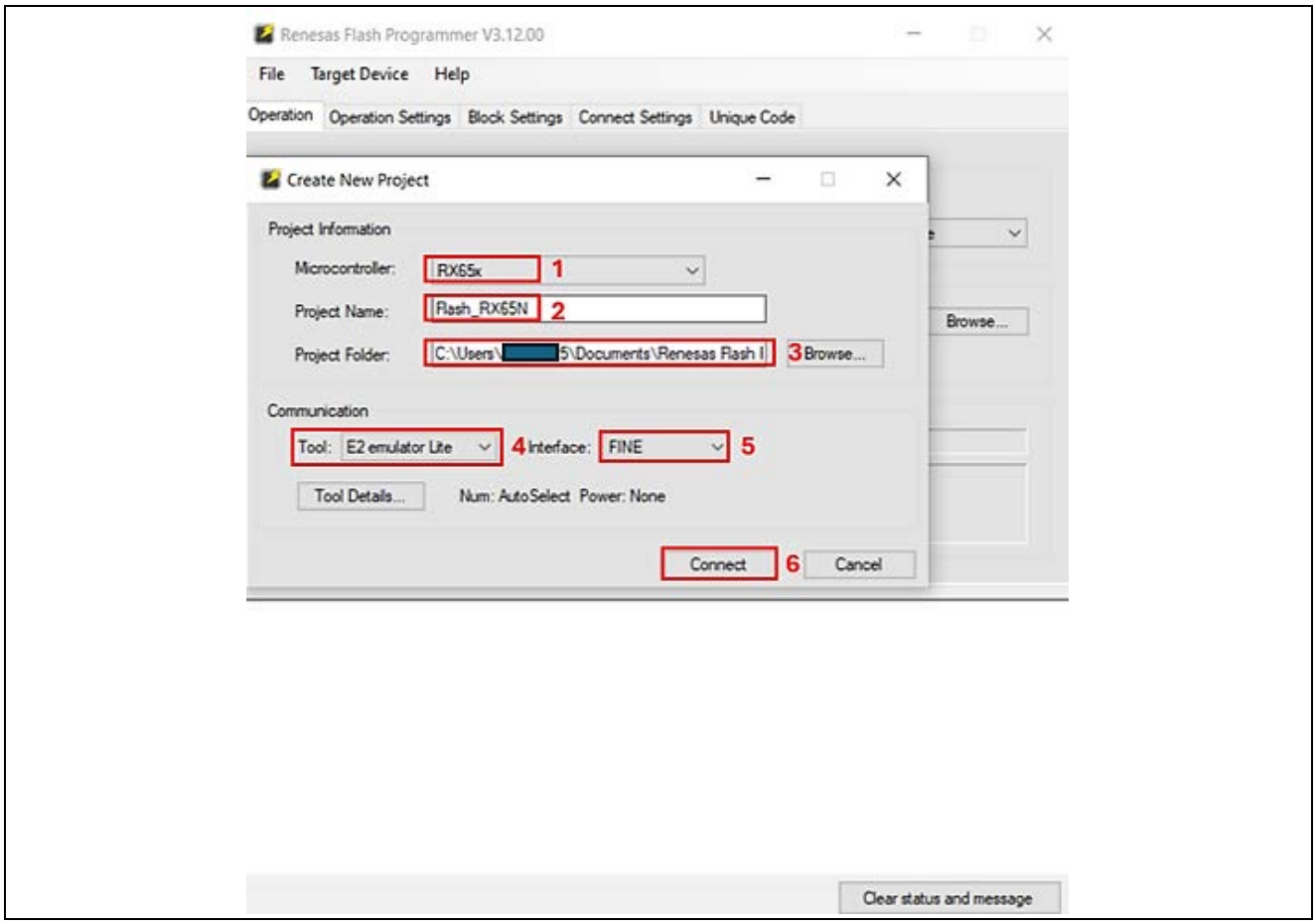

Figure 122. Create New Flash Project (2/4)

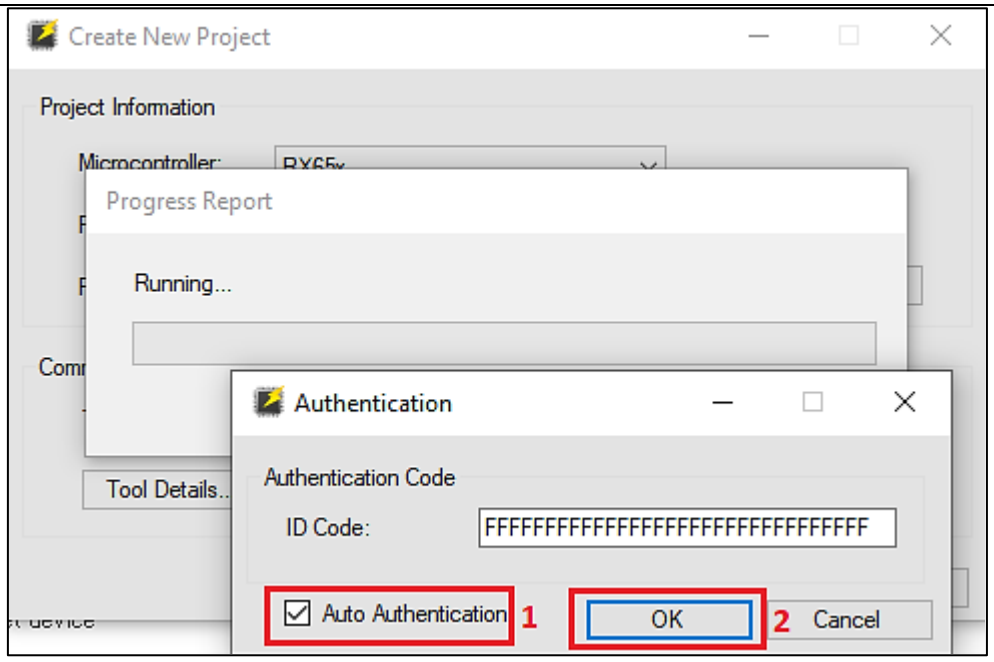

Figure 123. Create New Flash Project (3/4)

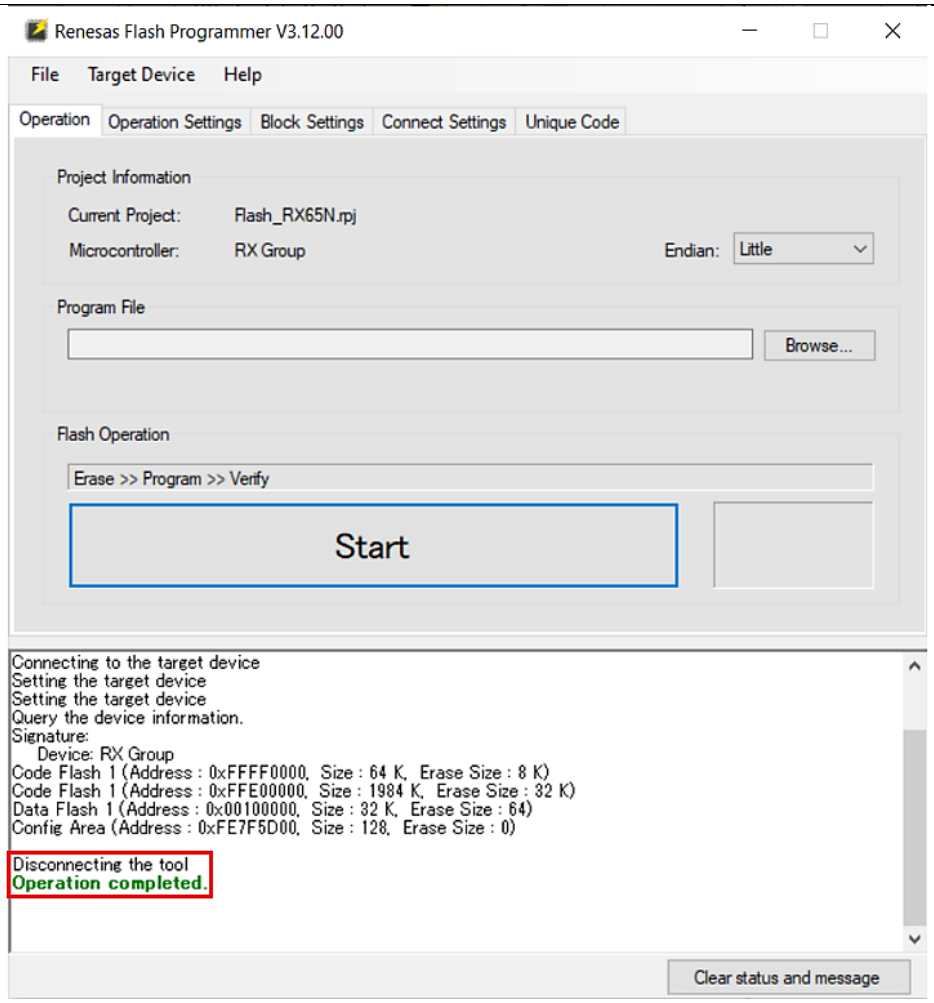

Figure 124. Create New Flash Project (4/4)

2. Choose code flash area in configuration:

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

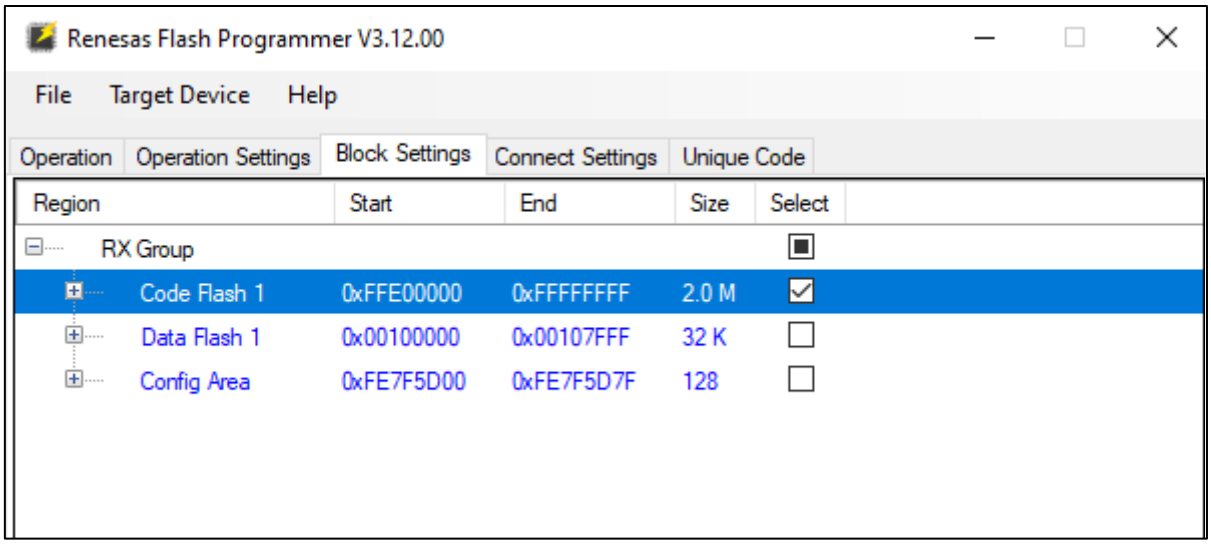

Figure 125. Choose Code Flash Area

3. Erase "Code Flash" before loading new image:

   Choose "**Operation Settings**" > "**Erase**".

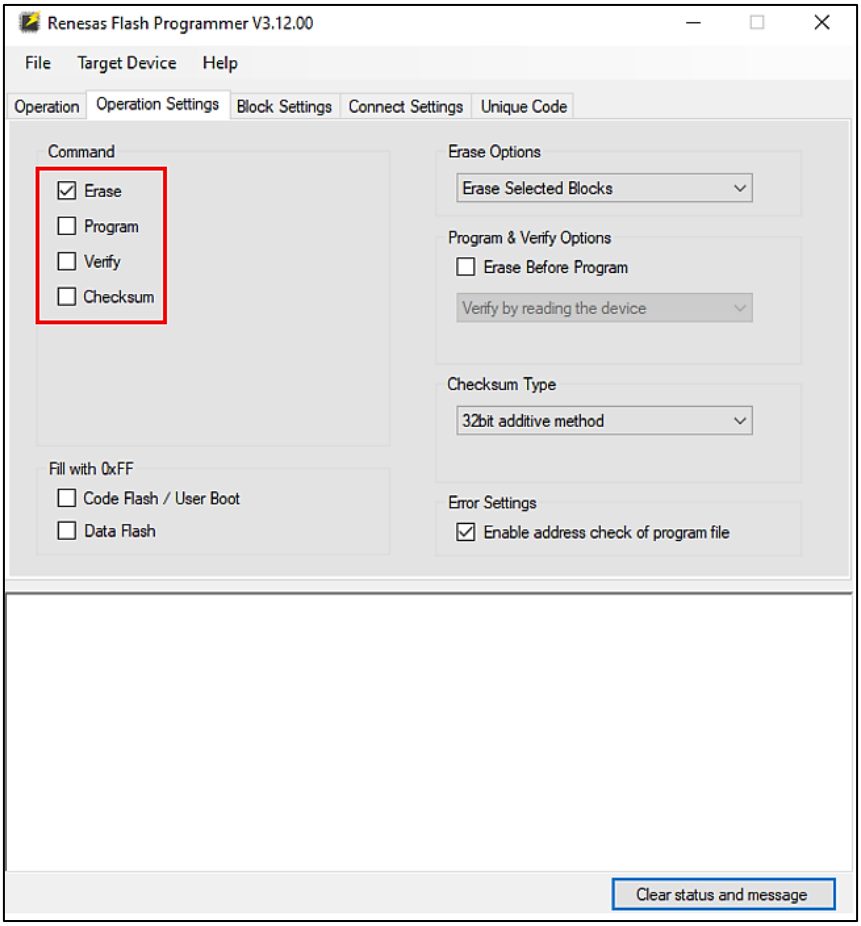

Figure 126. Erase Code Flash (1/2)

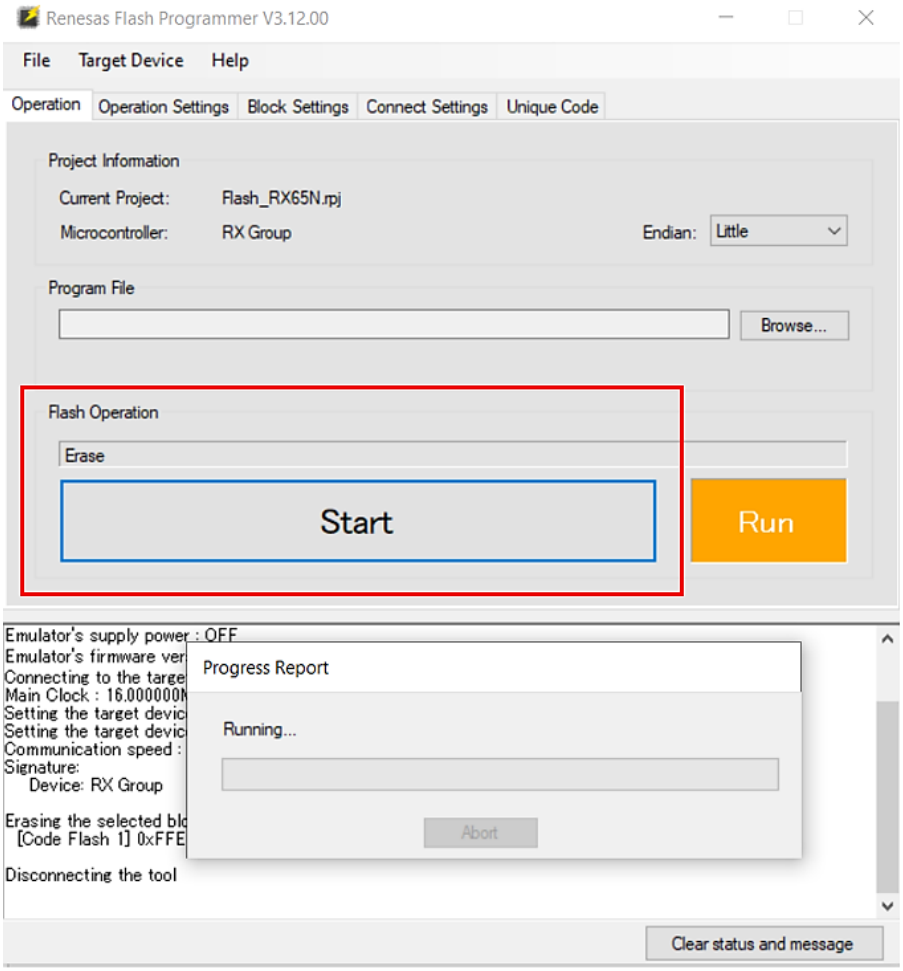

Figure 127. Erase Code Flash (2/2)

4. Write the initial firmware (userprog.mot)

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

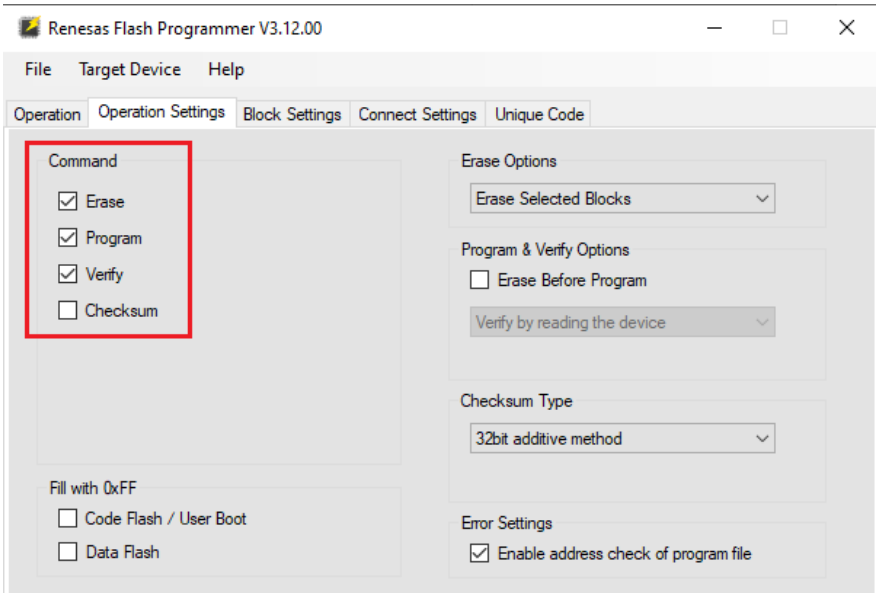

Figure 128. Flash the Firmware (1/3)

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

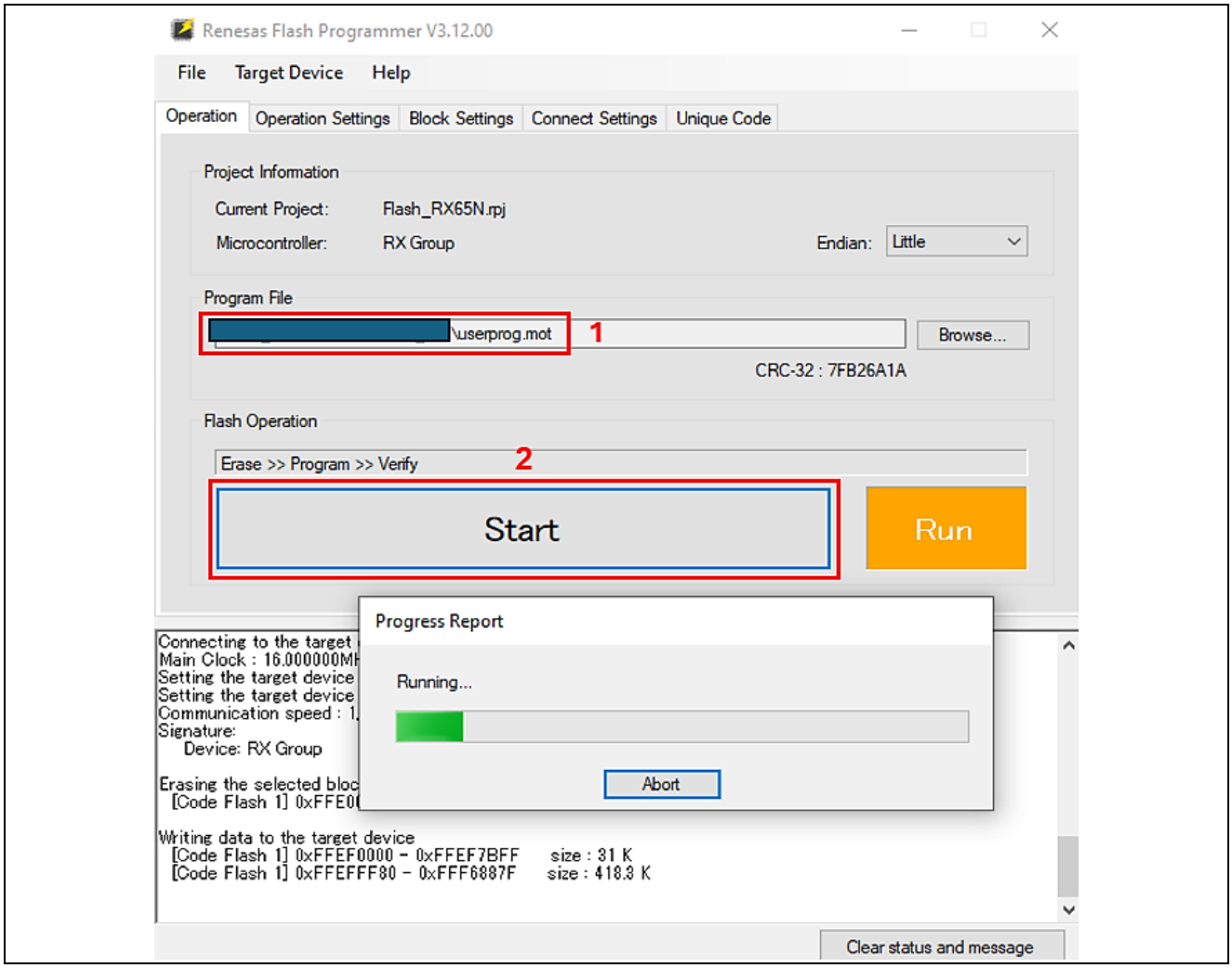

Figure 129. Flash the Firmware (2/3)

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

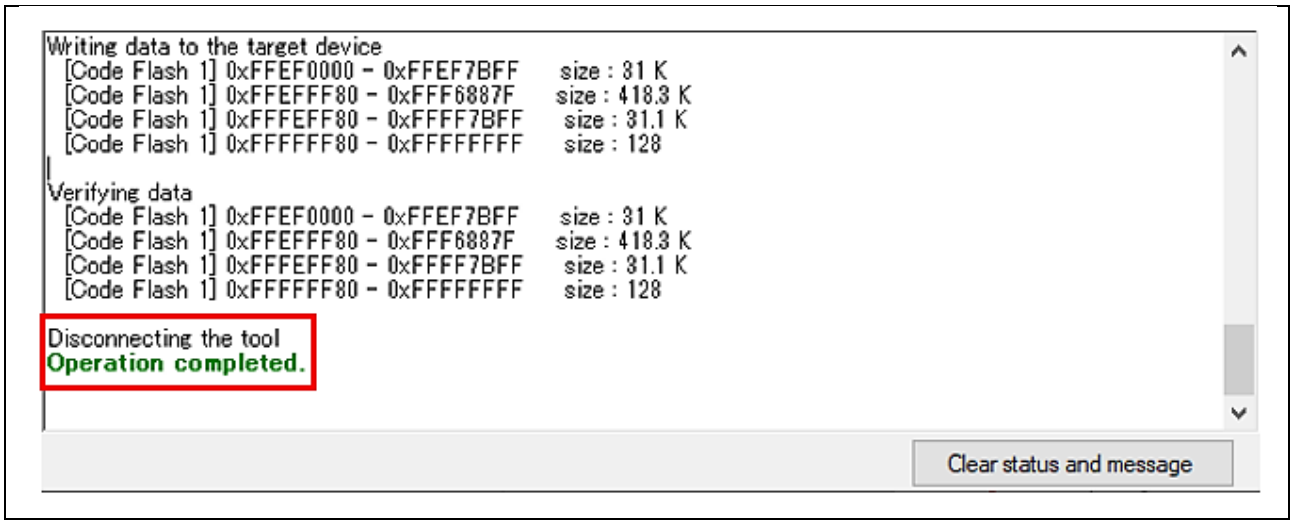

Figure 130. Flash the Firmware (3/3)

5. Running the application:

Set jumper J16 to “Run” mode. The application will run as below:

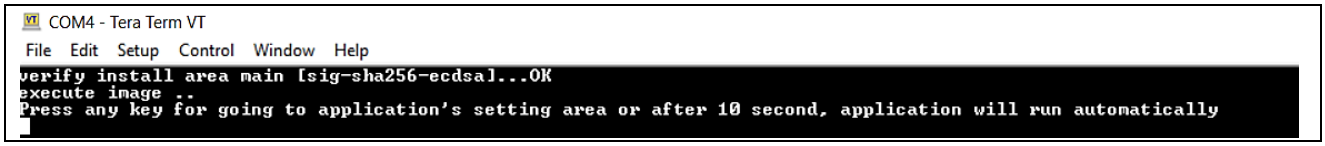

Figure 131. Start Application with OTA

Press any key to configure the application. For setting Cloud’s credentials for IoT Device, please refer to the **“0 ”** section.

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.

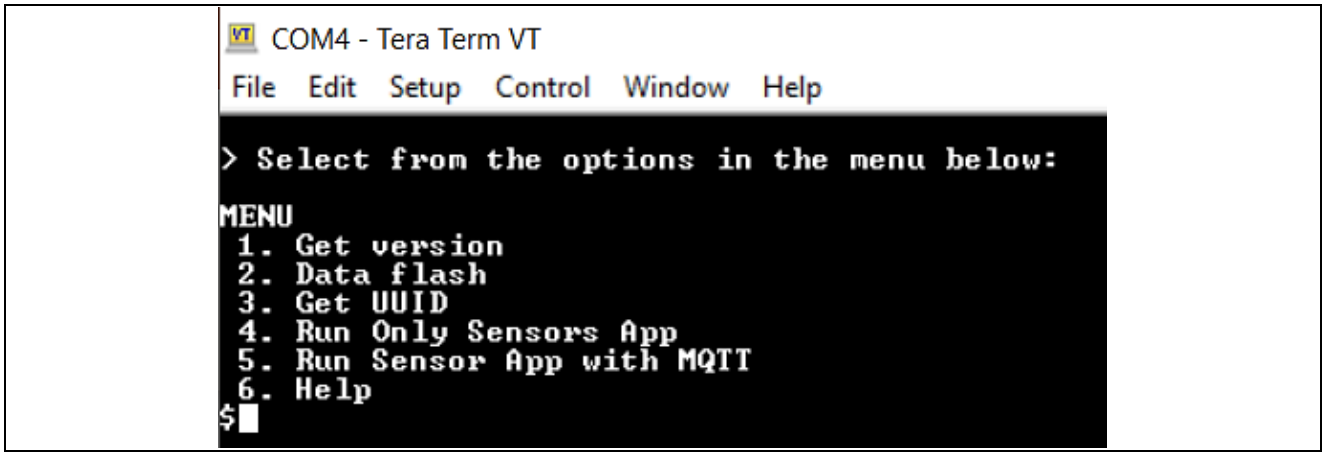

Figure 132. Main Menu

Press **‘f’** for storing code signing certificate:

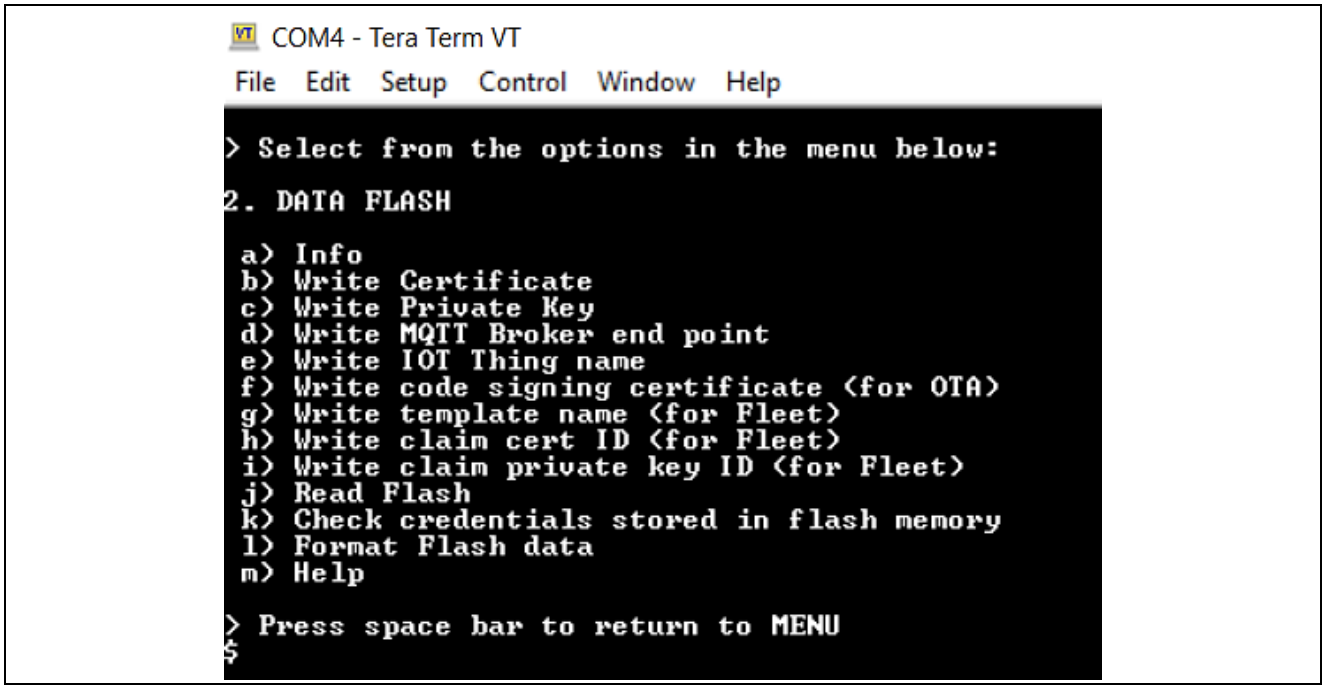

Figure 133. Data Flash Menu

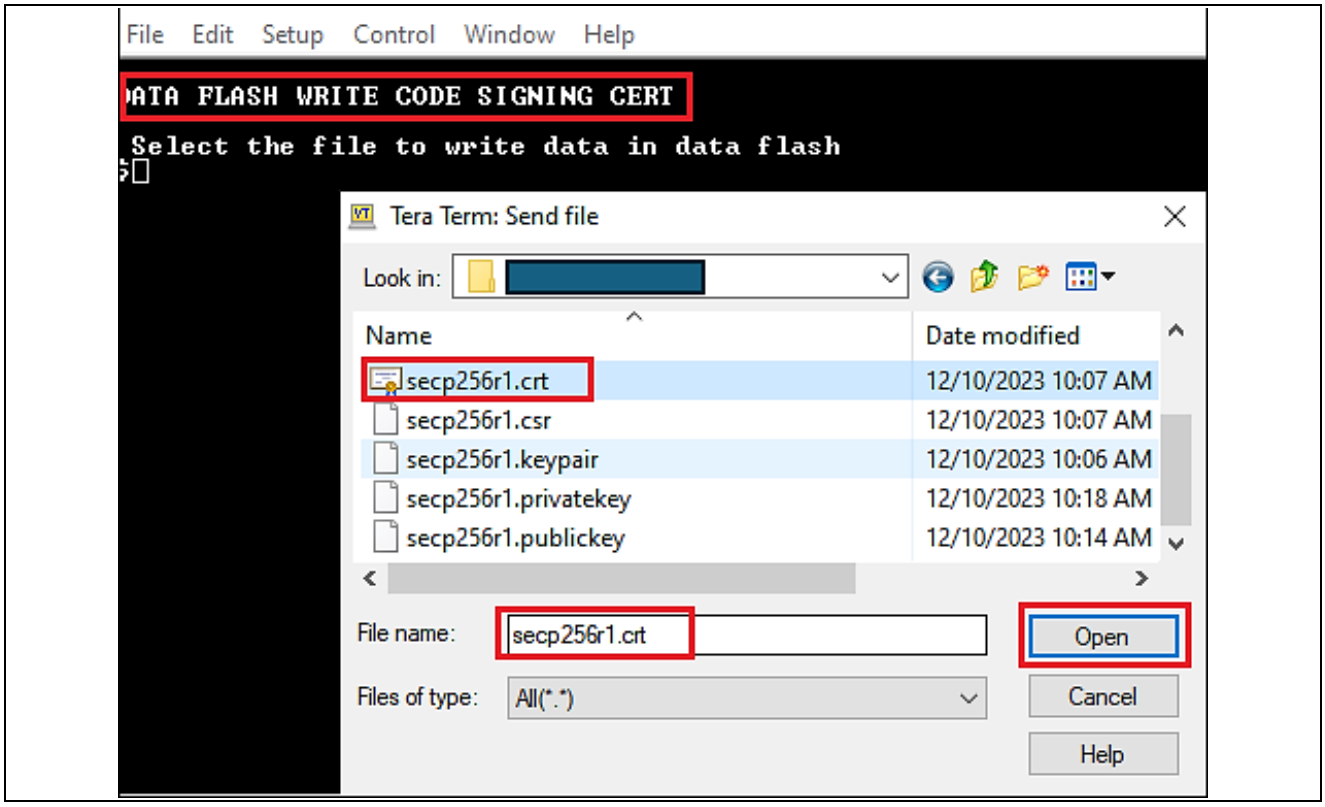

Figure 134. Store code signing certificate into flash (1/2)

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

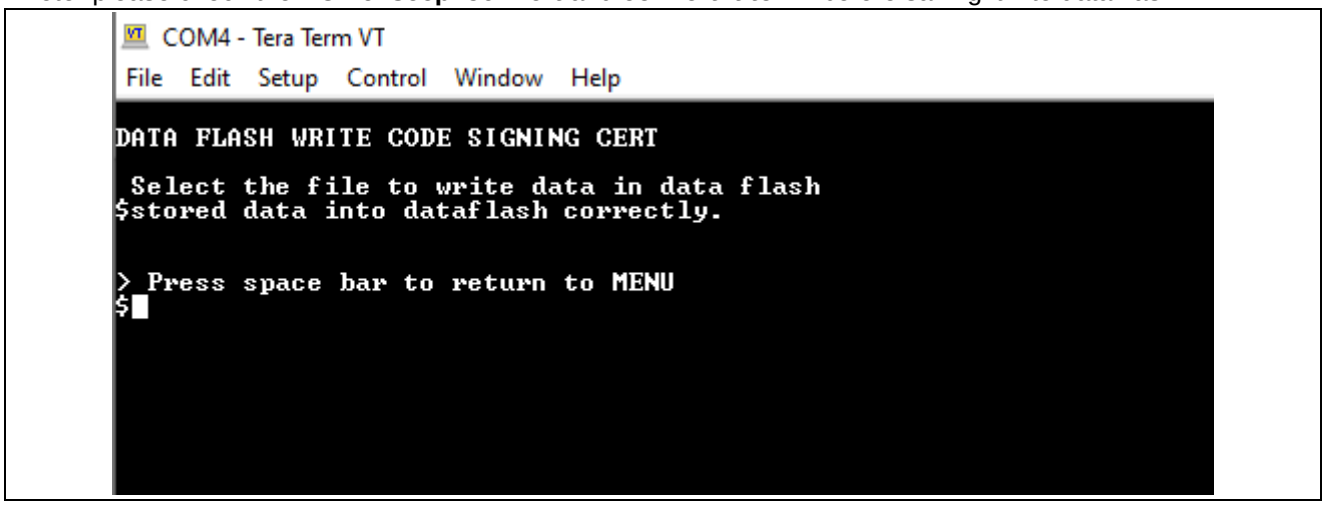

Figure 135. Store code signing certificate into flash (2/2)

After saving all Cloud credentials, user starts the application by choosing option **"Run Sensor App with MQTT"** on Main Menu:

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help

CHECK CREDENTIALS STORED IN DATA FLASH
 Fleet is disabled, do not need Claim private key ID
 Fleet is disabled, do not need Claim cert ID
 Fleet is disabled, do not need template name
 Code signing certificate saved in data flash is verified and successful
 IOT thing name saved in data flash is verified and successful
 MQTT Endpoint saved in data flash is verified and successful
 Private Key saved in data flash is verified and successful
 Certificate saved in data flash is verified and successful
 All credentials in data flash is verified and successful
0 37498 [CLI] Write certificate...

** Alternate Key Provisioning successfully **
1 37696 [IP-Task] prvIPTask started
2 37698 [ETHER_RECEI] Deferred Interrupt Handler Task started
3 37698 [ETHER_RECEI] Queue space: lowest 8
4 37698 [IP-Task] InitializeNetwork returns OK
5 37698 [IP-Task] xNetworkInterfaceInitialise returns 0
6 37798 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
7 37898 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
8 37998 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
9 38098 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
10 38198 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
11 38298 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
12 38398 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
13 38498 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
14 38598 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
15 38698 [ETHER_RECEI] R_ETHER_Read_ZC2: rc = -5
```

Figure 136. Application with OTA (1/2)

While subscribing/publishing messages to MQTT (sensor's data, control LEDs and so forth), application will also check the event of OTA from Cloud.

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help
49 57222 [MQTT] [INFO] ---------Start MQTT Agent Task---------

50 57222 [MQTT] [INFO] Creating a TLS connection to [redacted].us-east-1.amazonaws.com:8883.
51 57253 [ETHER_RECEI] Heap: current 187784 lowest 183808

52 57253 [MQTT] FreeRTOS_ProcessDNSCache: add: '[redacted].us-east-1.amazonaws.com' @ 36a66e33ip (TTL 60)

53 57253 [MQTT] DNS[0x5487]: The answer to '[redacted].us-east-1.amazonaws.com' (54.166.110.51) will be stored

54 57255 [MQTT] FreeRTOS_connect: 26808 to 36a66e33ip:8883

55 57255 [IP-Task] prvSocketSetMSS: 1400 bytes for 36a66e33ip:8883

56 57255 [IP-Task] prvWinScaleFactor: uxRxWinSize 2 MSS 1400 Factor 0

57 57255 [IP-Task] Connect[36a66e33ip:8883]: next timeout 1: 3000 ms

58 57490 [IP-Task] MSS change 1400 -> 1452

59 57490 [IP-Task] TCP: active 26808 => 36a66e33ip:8883 set ESTAB (scaling 1)

60 57491 [MQTT] [INFO] Established TCP connection with [redacted].us-east-1.amazonaws.com.
61 57590 [ETHER_RECEI] Heap: current 127808 lowest 127336

62 58314 [ETHER_RECEI] Heap: current 113712 lowest 111560

63 60304 [MQTT] [INFO] (Network connection 805234) TLS handshake successful.
64 60304 [MQTT] [INFO] (Network connection 805234) Connection to [redacted].us-east-1.amazonaws.com established.
65 60304 [MQTT] [INFO] Creating an MQTT connection to the broker.
66 60581 [MQTT] [INFO] MQTT connection established with the broker.
67 60581 [MQTT] [INFO] Successfully connected to MQTT broker.
68 60587 [OTA Demo Ta] [INFO] ---------Start OTA Task---------

70 60593 [sensor_thre] I2C bus 0 setup success

71 60593 [sensor_thre] HS3001 open sensor instance successful: 0

72 60593 [sensor_thre] ICP10101 open sensor instance successful: 0

73 60593 [ob1203_thre]
OB1203 Device open success
74 60602 [sensor_thre] ICM20948 open sensor instance successful: 0

69 60592 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.2
75 60746 [MainAWSDemo] [INFO] ---------Start AWS - MQTT Demo Task ---------

76 60757 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
77 60781 [ETHER_RECEI] Heap: current 83096 lowest 83096

78 60781 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
79 62209 [MainAWSDemo] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
80 62843 [zmod_thread] ZMOD4410 open sensor instance successful: 0

81 63049 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
82 63075 [zmod_thread] ZMOD4510 open sensor instance successful: 0

83 63076 [zmod_thread] Task zmod4410 measurement Success:0

84 63208 [OTA Agent T] [INFO] Subscribed to topic $aws/things/[redacted]/jobs/notify-next.


85 63215 [OTA Agent T] [INFO] Subscribed to MQTT topic: $aws/things/[redacted]/jobs/notify-next
86 64223 [MainAWSDemo] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
87 64230 [MainAWSDemo] [Send Data] ZMOD4410-IAQ TVOC: 000.000
82 64230 [MainAWSDemo] [Send Data] ZMOD4410-IAQ TVOC: 000.000
```

Figure 137. 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: The previous version is 0.9.2, so the new version we can choose is 0.9.3)

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

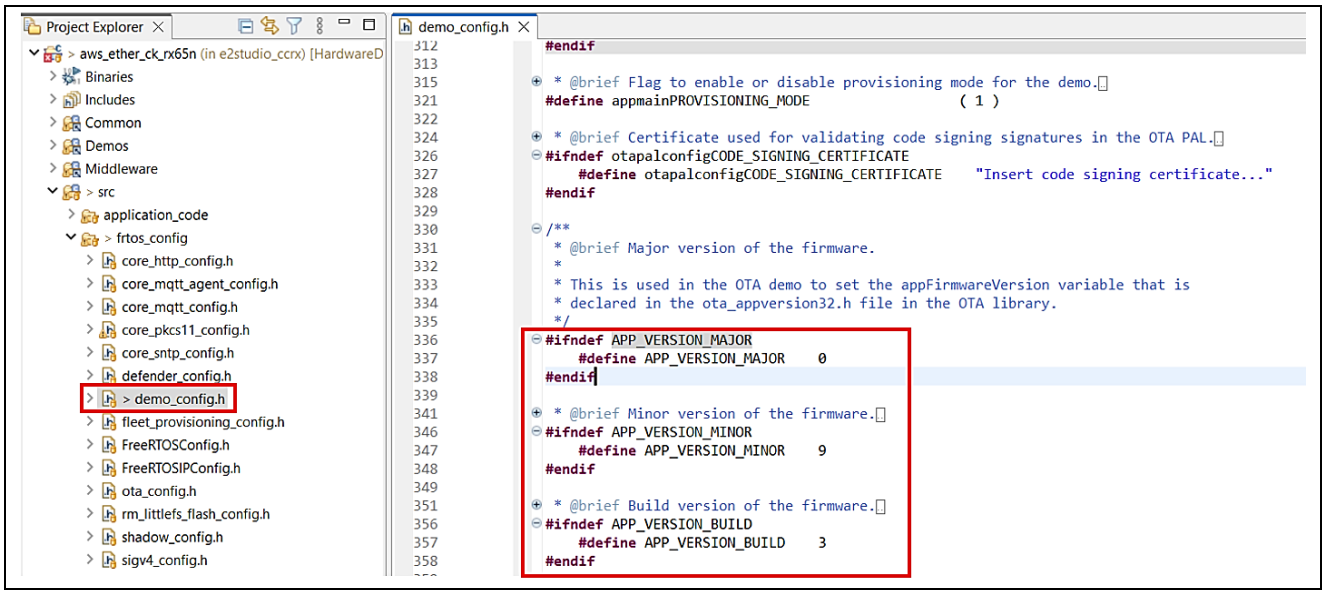

Figure 138. Setting New Version for Firmware

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_ether_ck_rx65n.mot), and then execute the following command at the command prompt:

$ python image-gen.py -iup *aws_ether_ck_rx65n.mot* -ip *RX65N_DualBank_ImageGenerator_PRM.csv* -o *user_093* -key *secp256r1.privatekey* -vt ecdsa -ff RTOS

This command generates a file named user_093.rsu.

6.5.2 Updating the firmware

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

6.5.2.1 Creating New Job

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

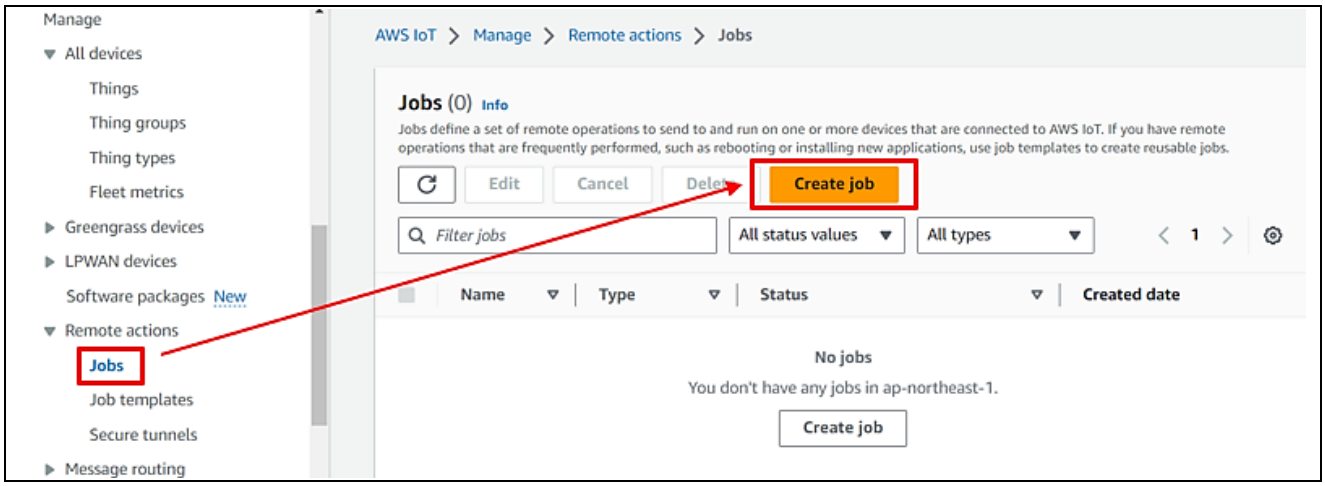

Figure 139. Create New Job for OTA (1/2)

6.5.2.2 Creating FreeRTOS OTA Job Update

Select **Create FreeRTOS OTA update job** and then click **Next**.

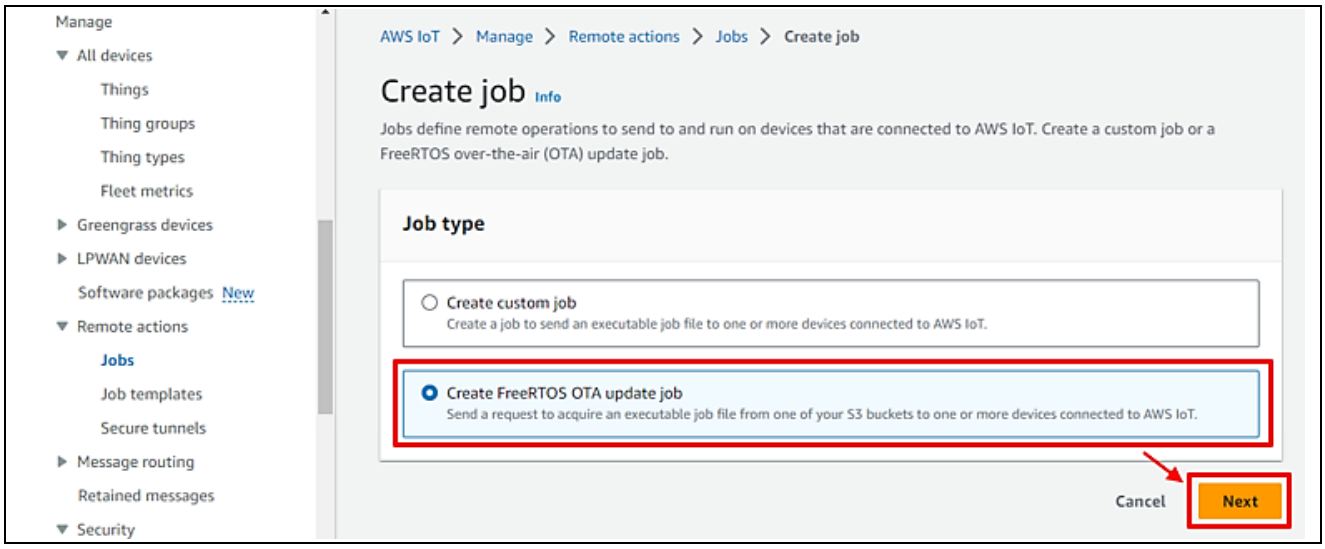

Figure 140. Create New Job for OTA (2/2)

6.5.2.3 Entering a Job Name

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

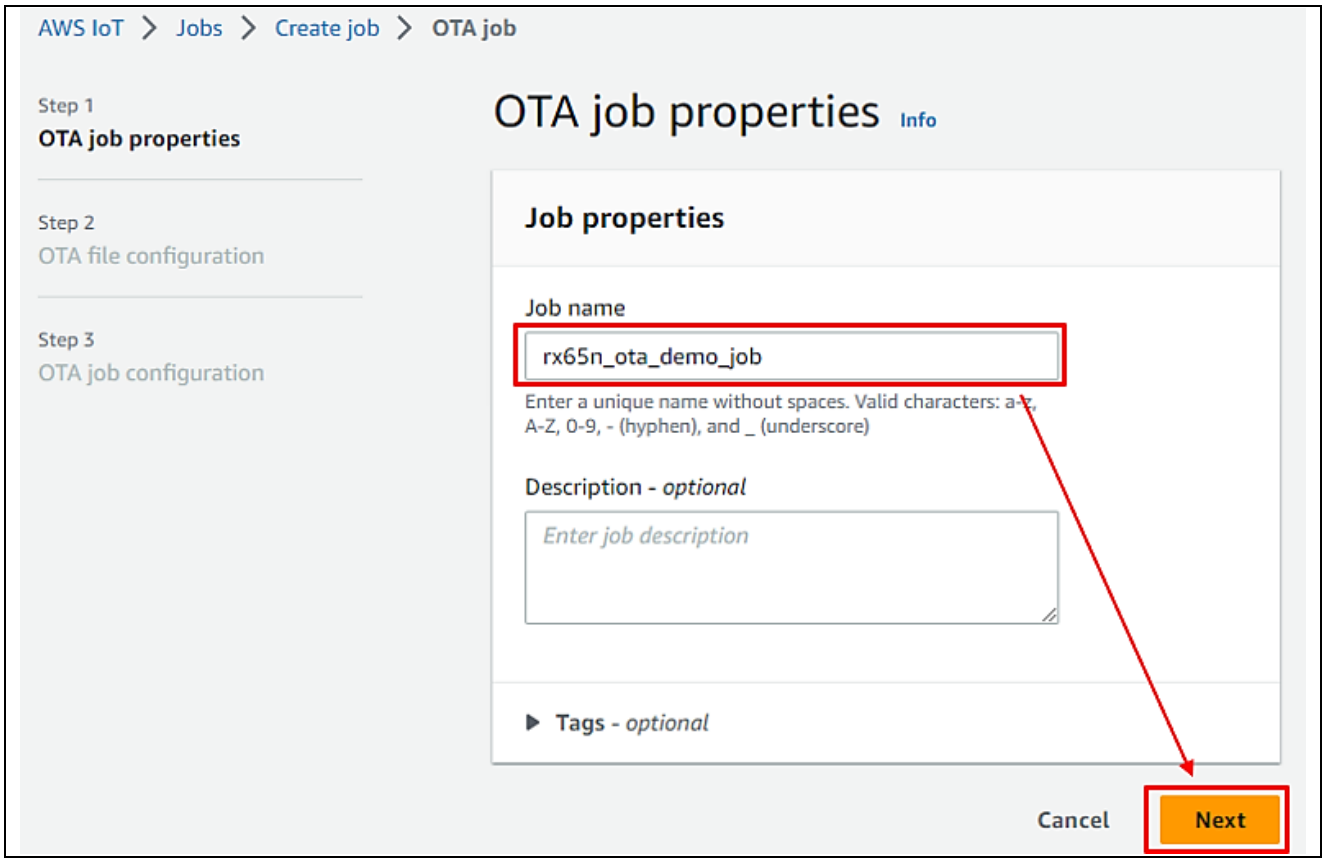

Figure 141. Enter OTA Job Name

6.5.2.4 Updating Devices

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

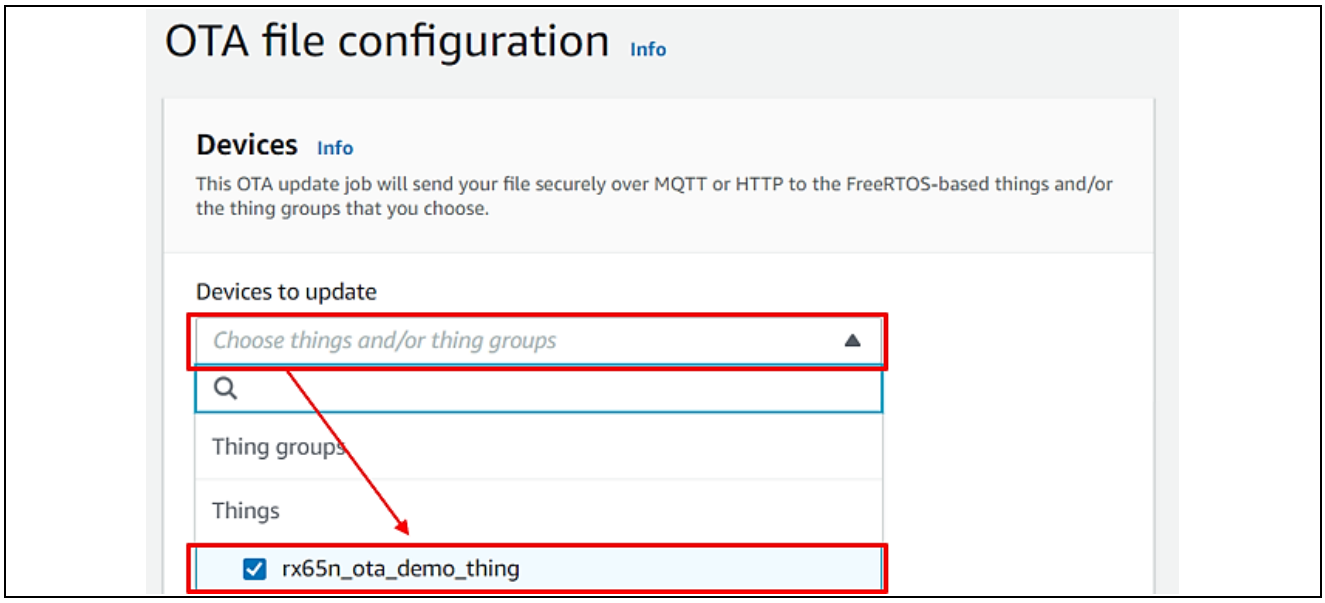

Figure 142. Choose Device to Update

6.5.2.5 Creating New Profile

Click **Create new profile**.

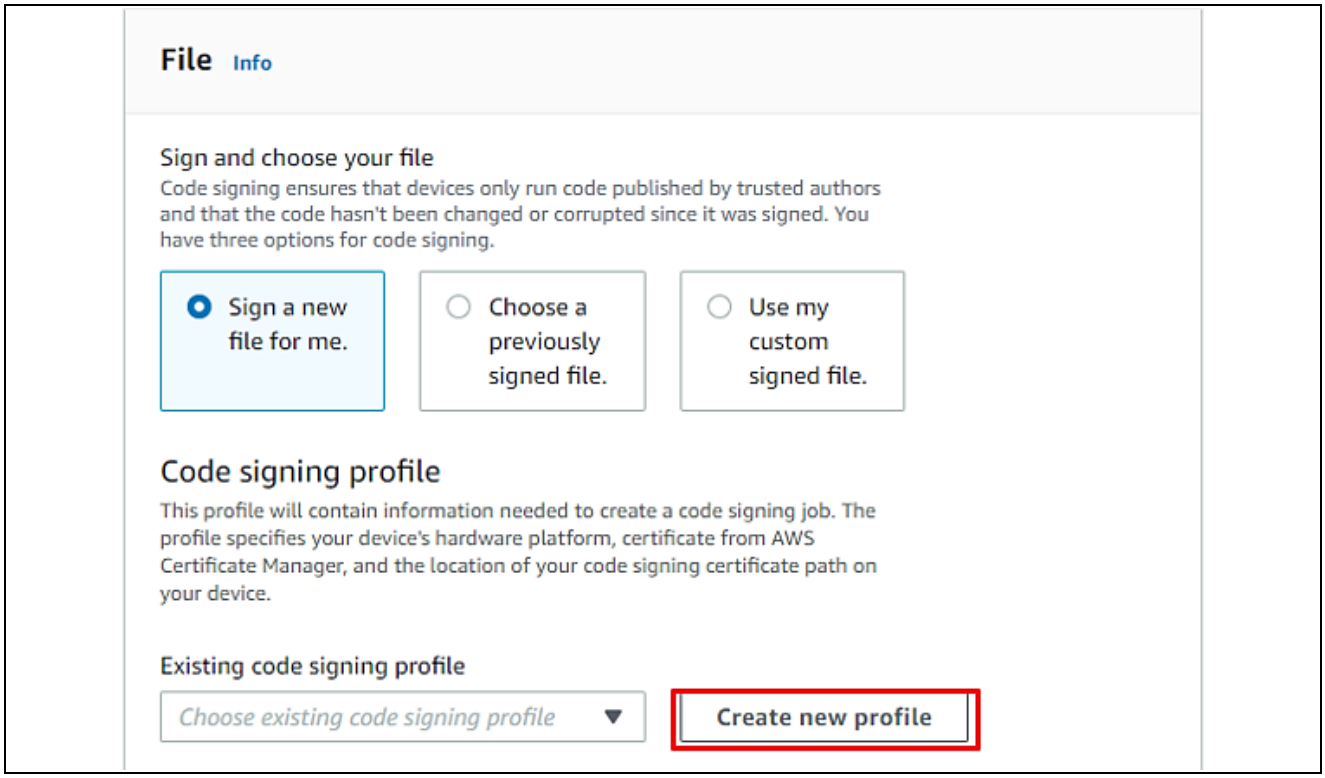

Figure 143. Create New Code Signing Profile

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.

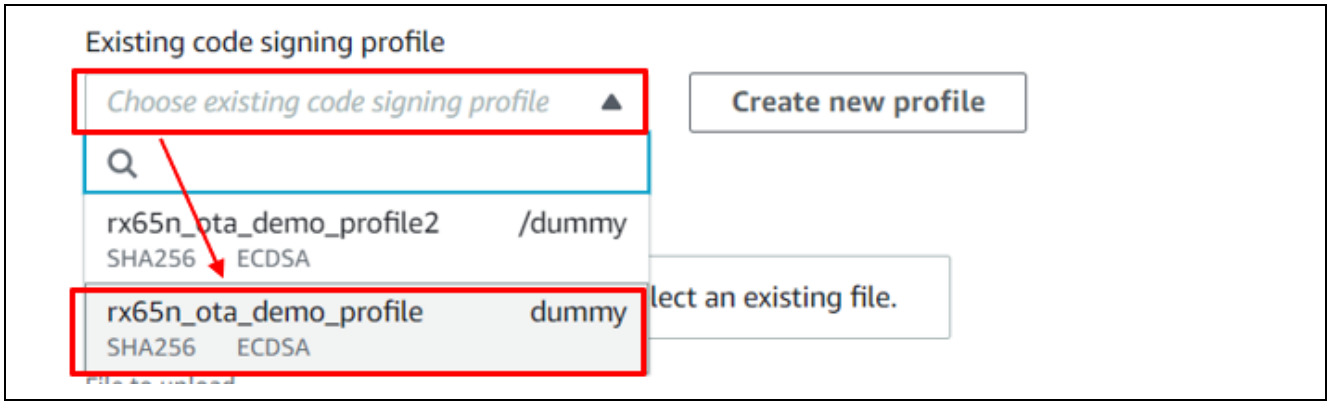

Figure 144. Choose Existing Code Signing Profile

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.

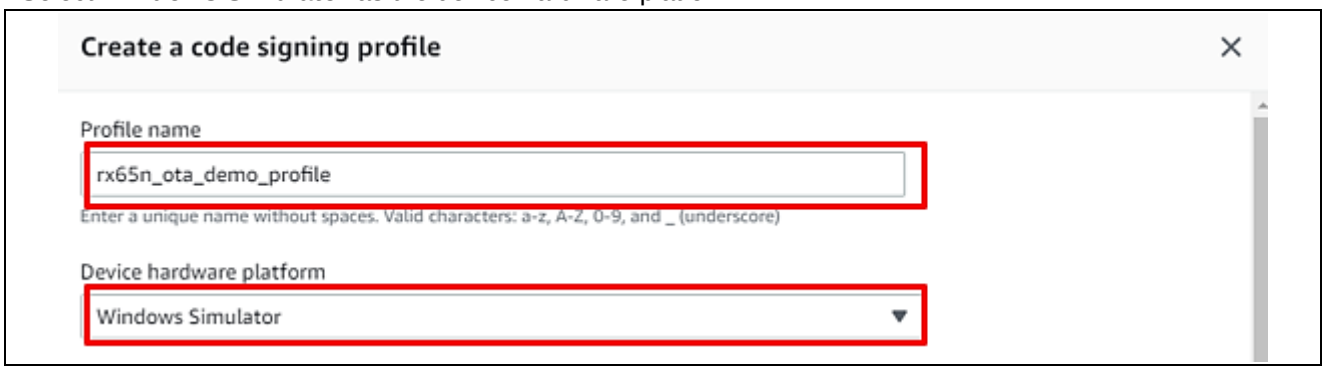

Figure 145. Create New Code Signing (1/2)

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 above files in **6.4**.
> Click **Import**

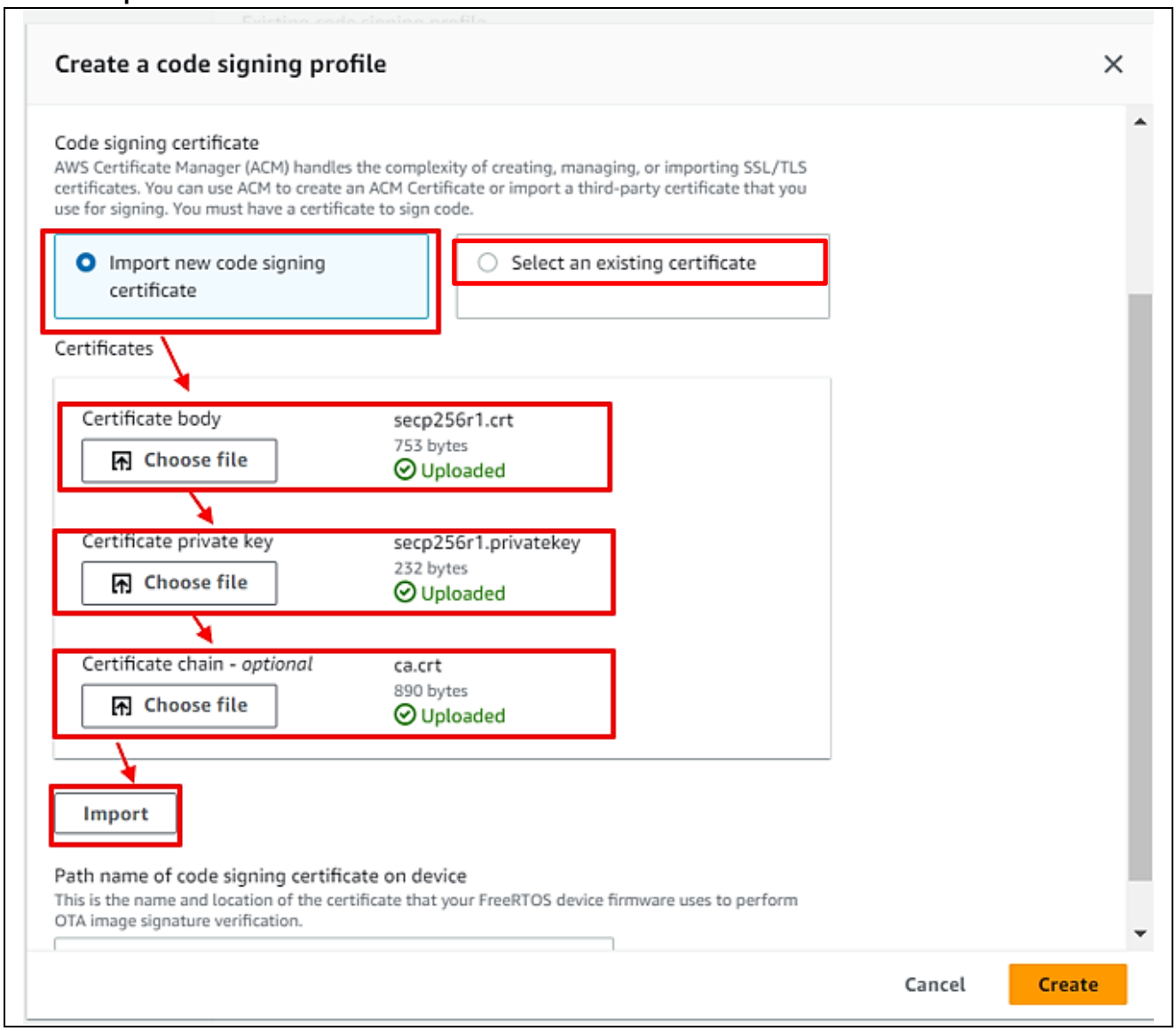

Figure 146. Create New Code Signing (2/2)

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 147. 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.

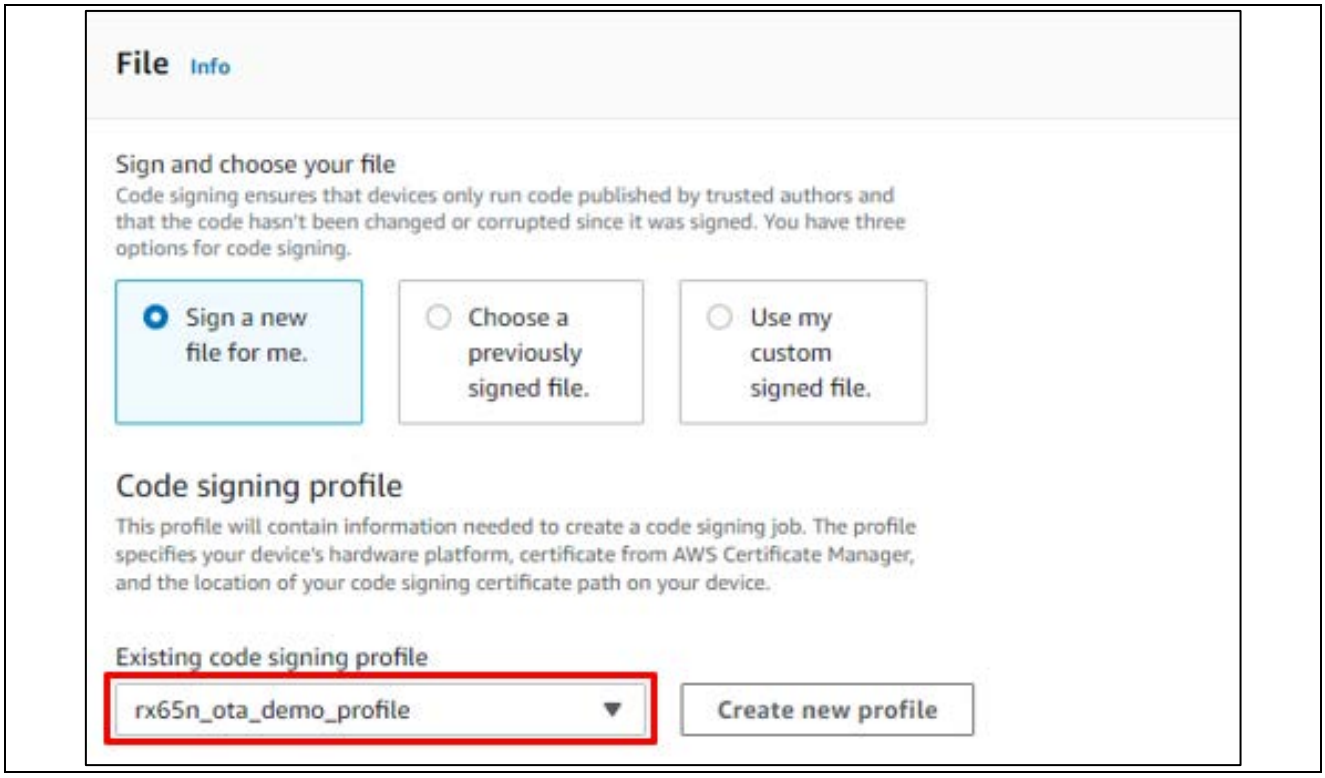

Figure 148. Choose Existing Code Signing Profile

6.5.2.10 Updating the Firmware

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

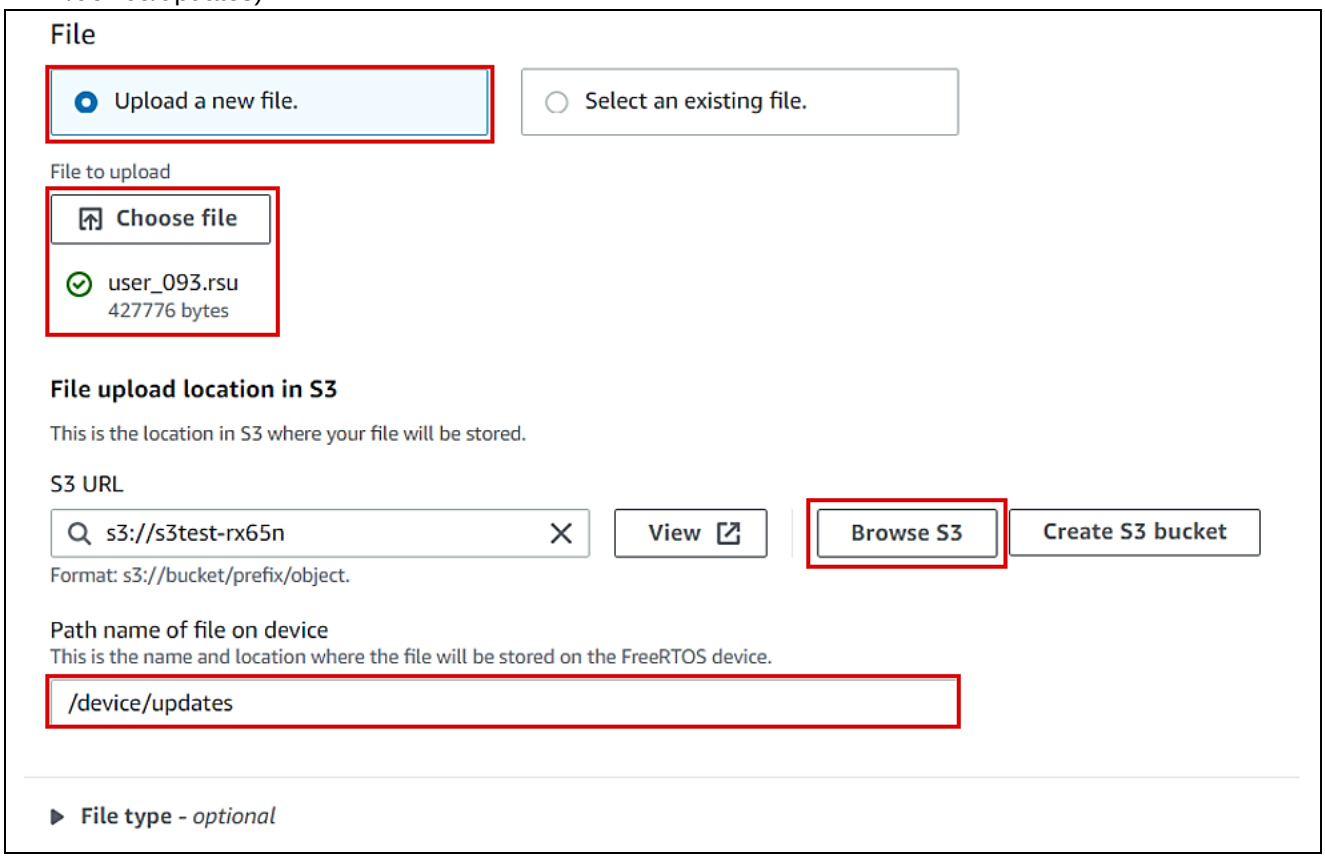

Figure 149. Choose Firmware to Update

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**.

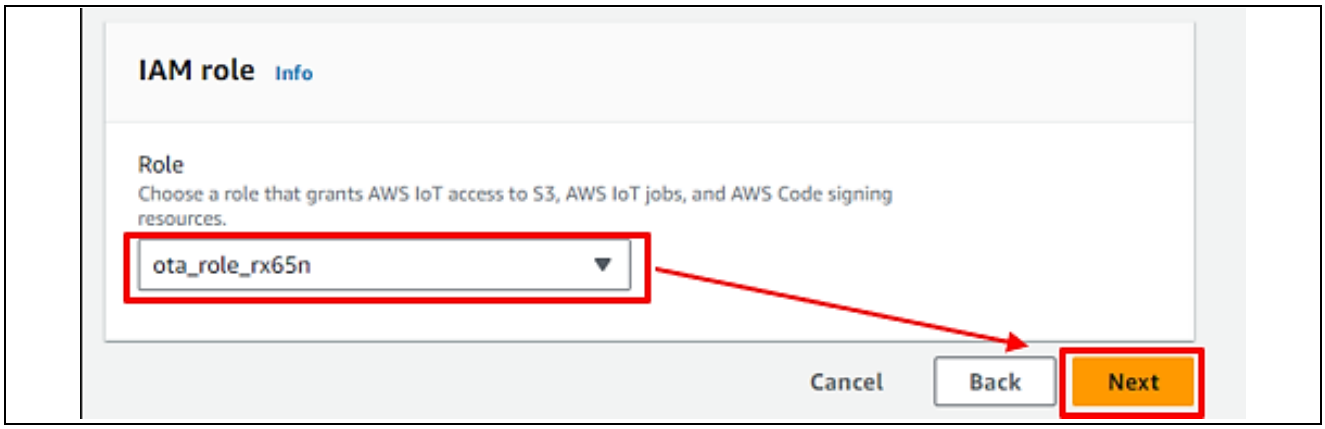

Figure 150. Choose Role for Creating Job

Click **Create job**.

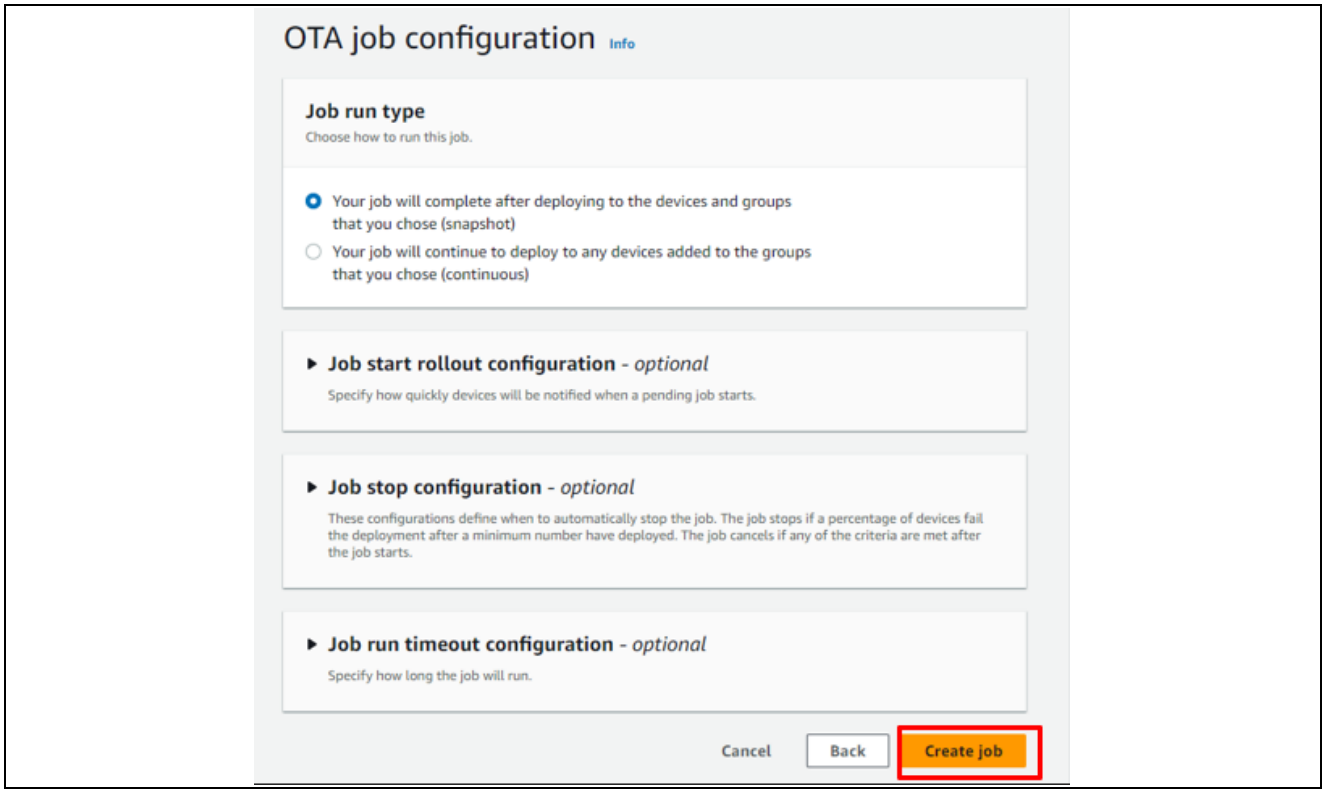

Figure 151. Create Job

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.

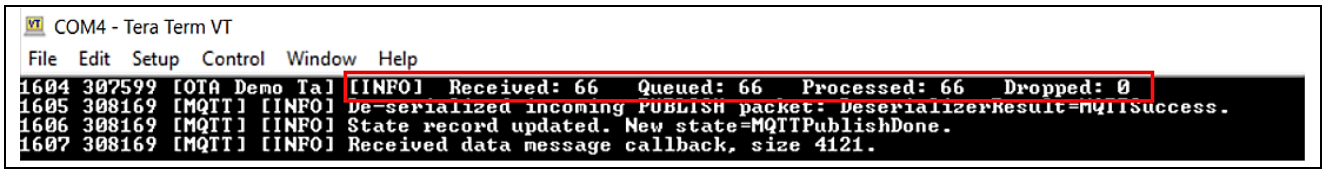

Figure 152. OTA Job is Processed

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

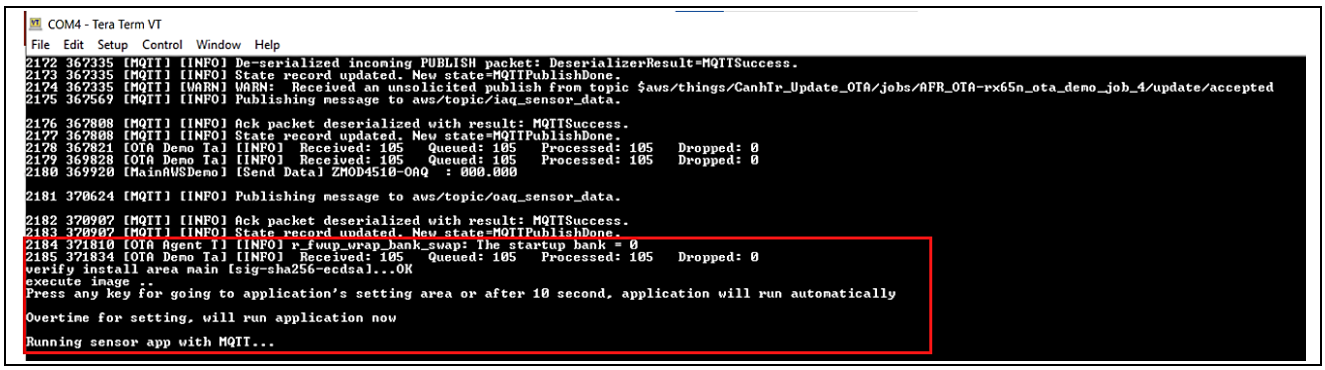

Figure 153. Device Reset and Update New Firmware

6.5.2.13 Confirming that the Firmware Version is a New Version

Example: 0.9.3 (updated at **6.5.1.1**)

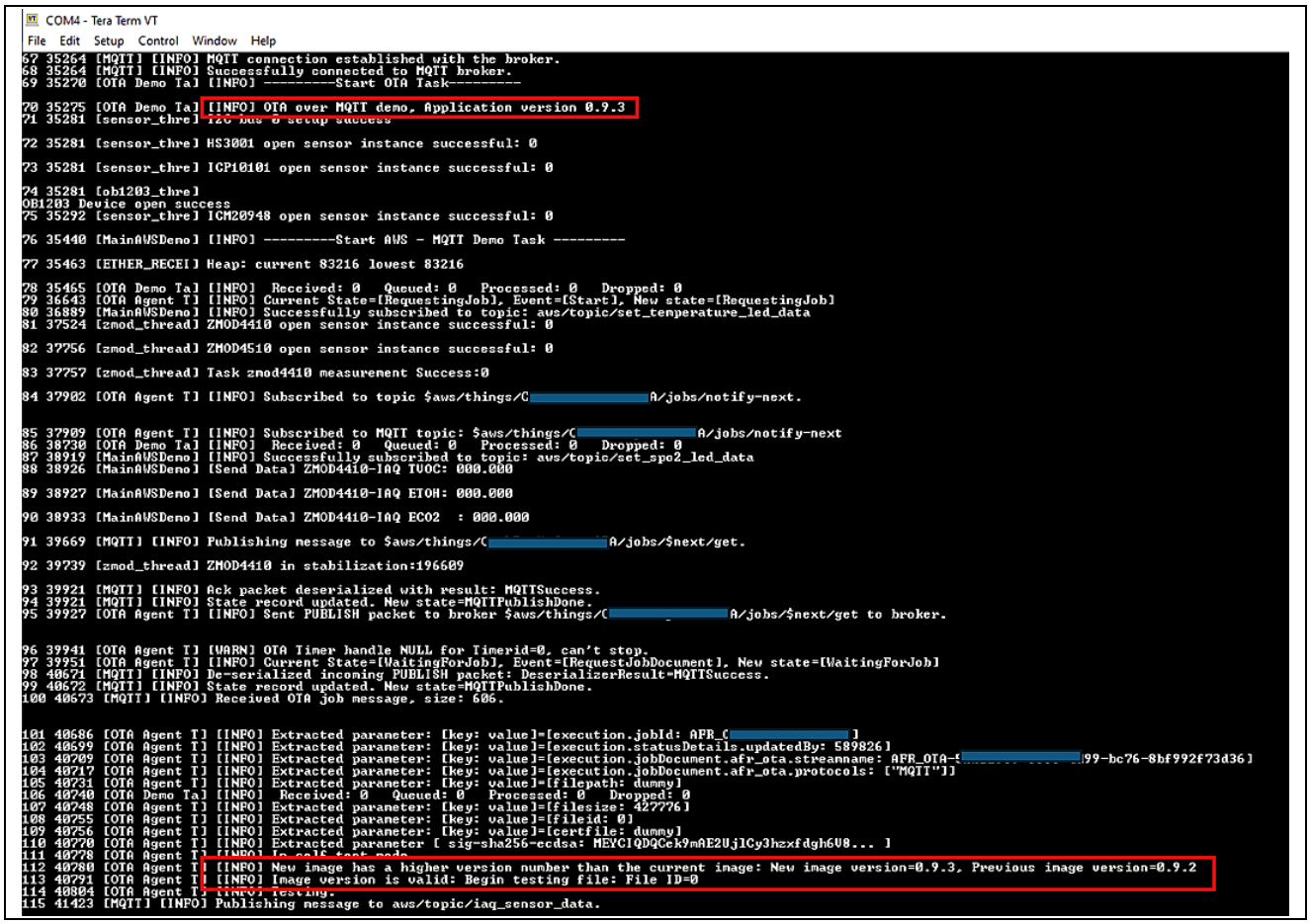

Figure 154. Firmware with New Version is Updated

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

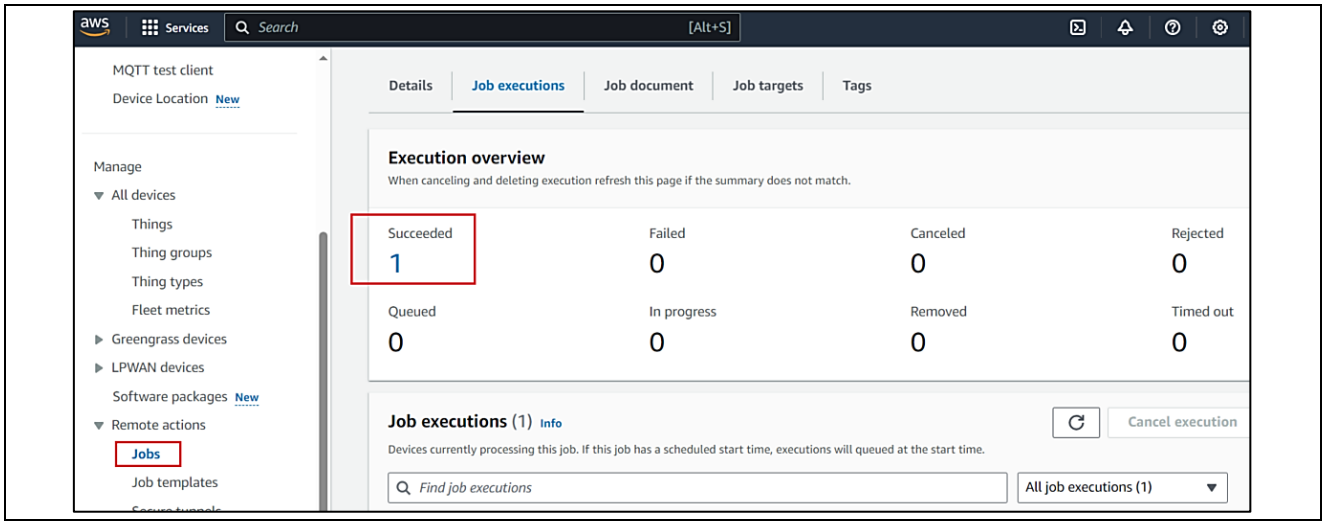

Figure 155. Job's Status in AWS Portal

7. Fleet Provisioning

7.1 Overview Fleet Provisioning

This section describes the steps on using Fleet Provisioning in these applications.

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

7.2 Setting up AWS for Fleet Provisioning

It is necessary to configure AWS settings 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.

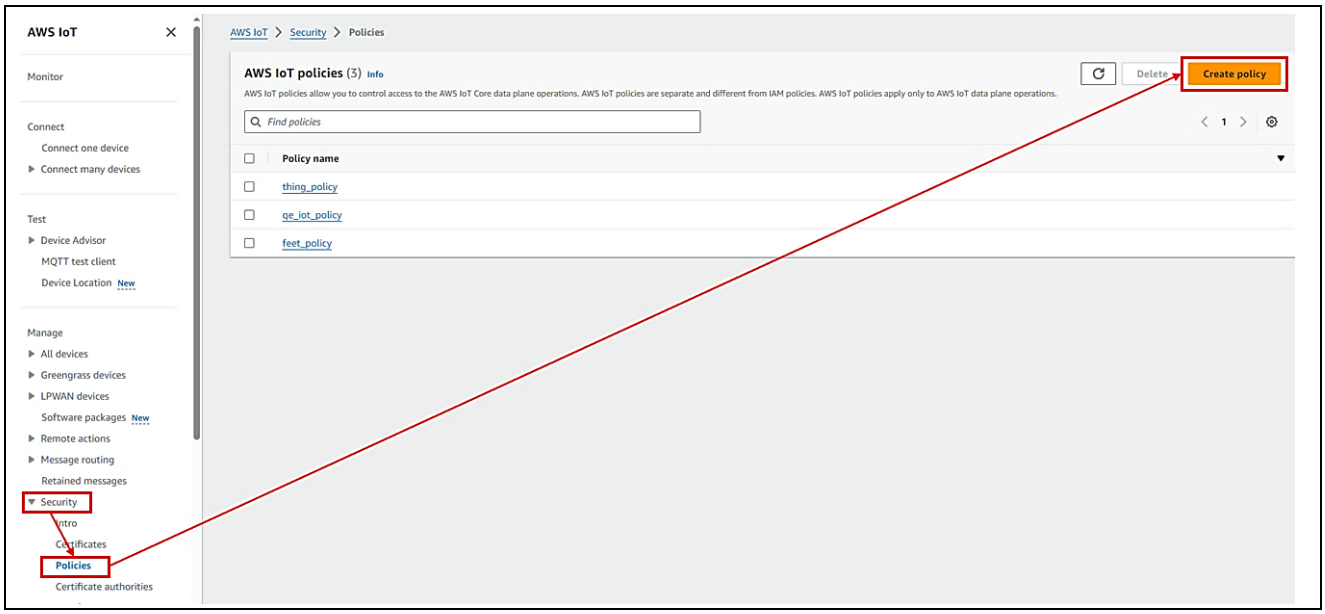

Figure 156. Creating an AWS IoT Policy (1/2)

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

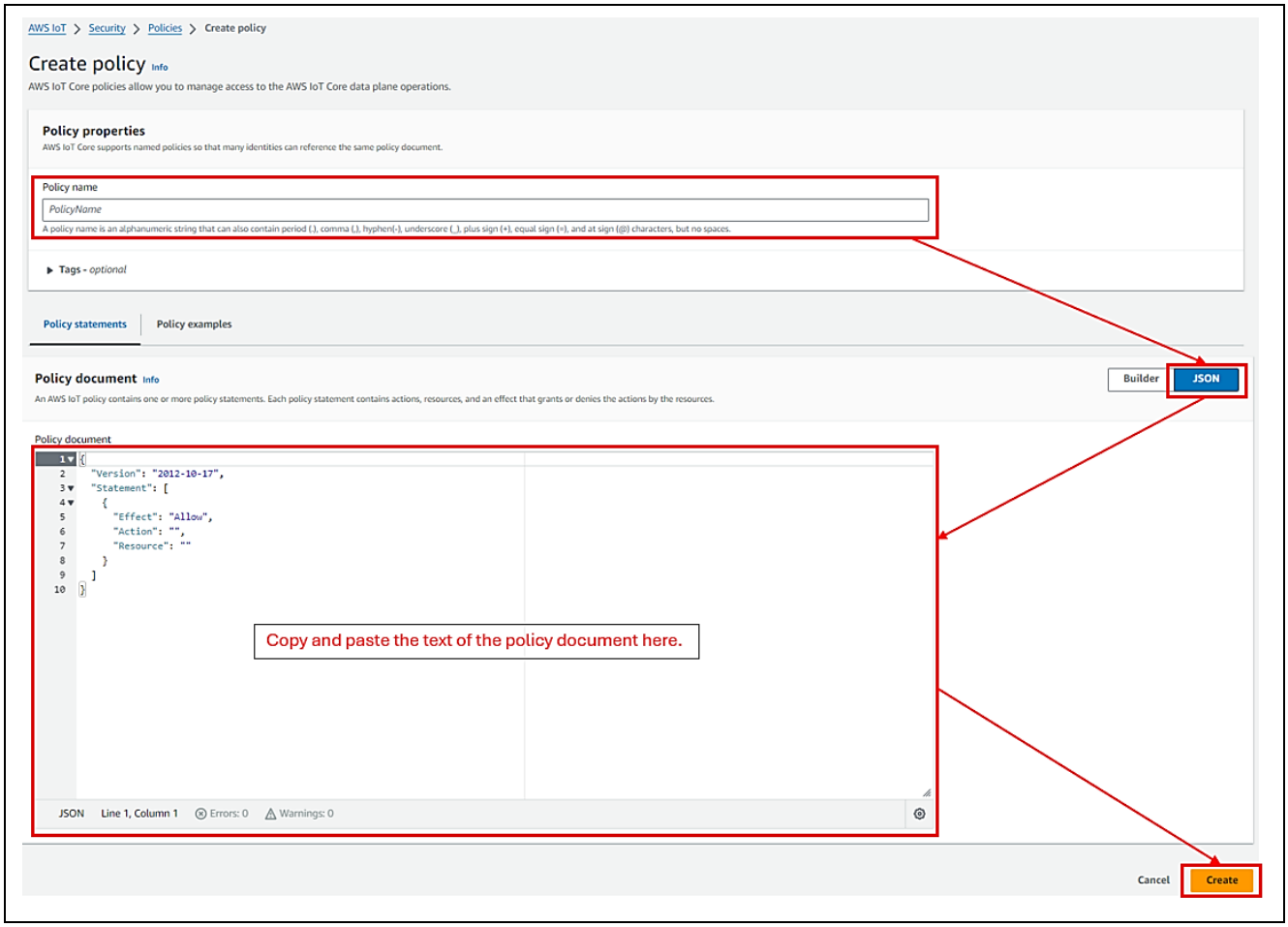

Figure 157. Creating an AWS IoT Policy (2/2)

Click the JSON button to display the policy document input field, then copy and paste the policy document shown in **Table 11. Policy Document** into the input field. When copying and pasting the policy document in **Table 11. Policy Document**, 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 11. 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**.

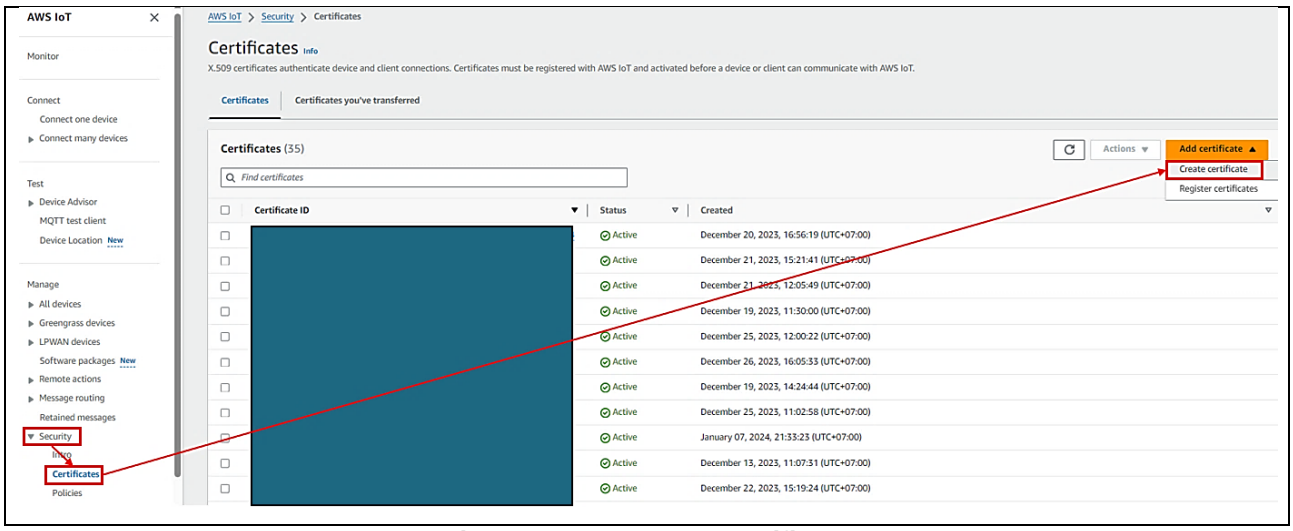

Figure 158. Create a Certificate

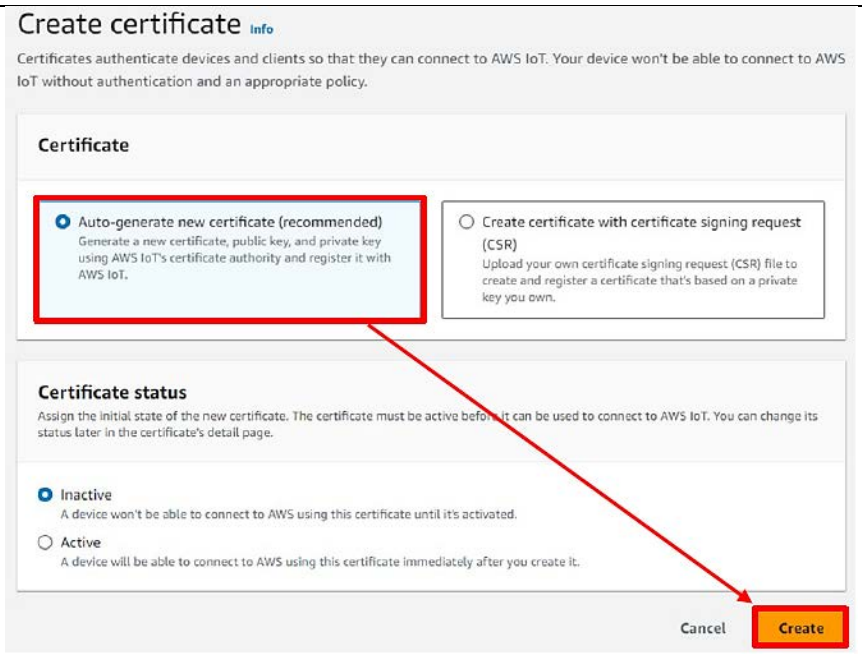

Figure 159. Creating a Certificate Automatically

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

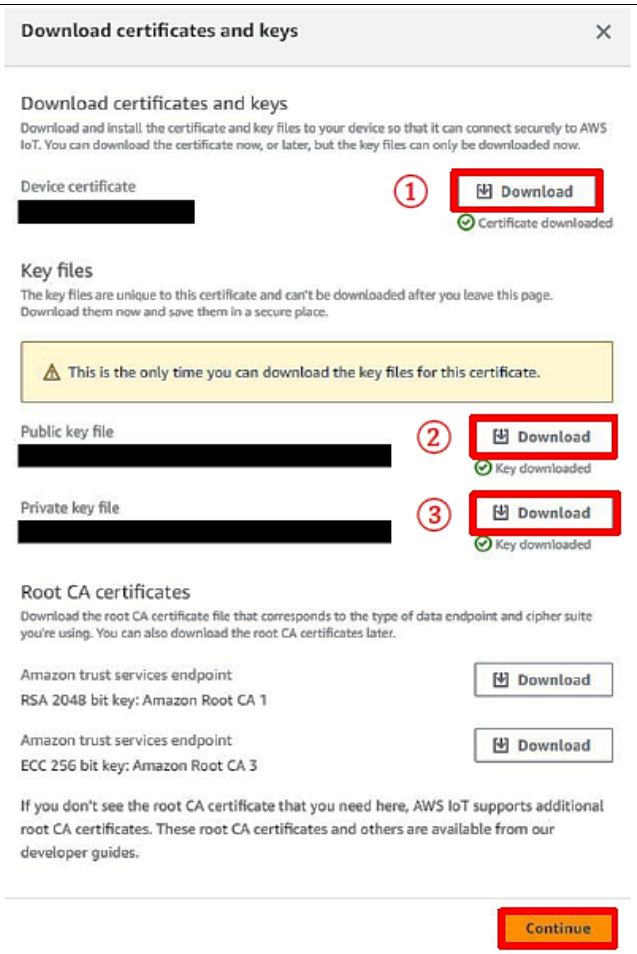

Figure 160. Downloading the Certificate and Key Pair

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

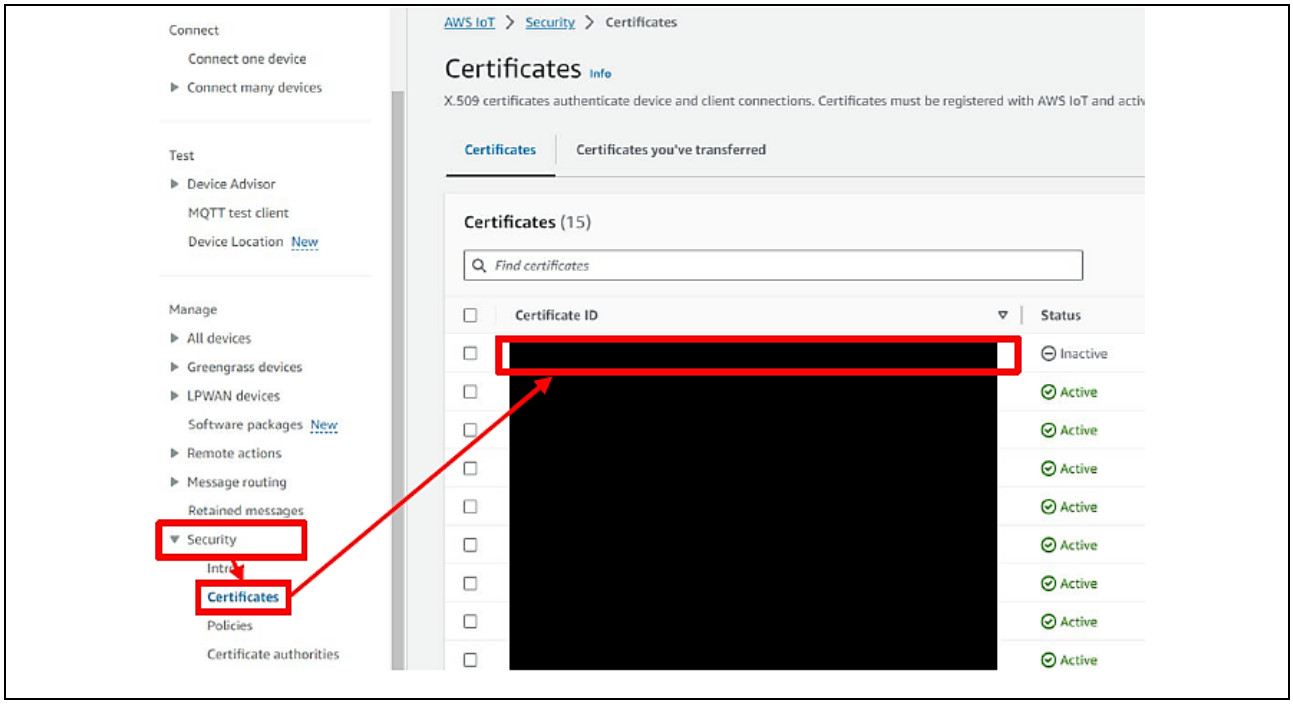

Figure 161. Certificate Settings

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

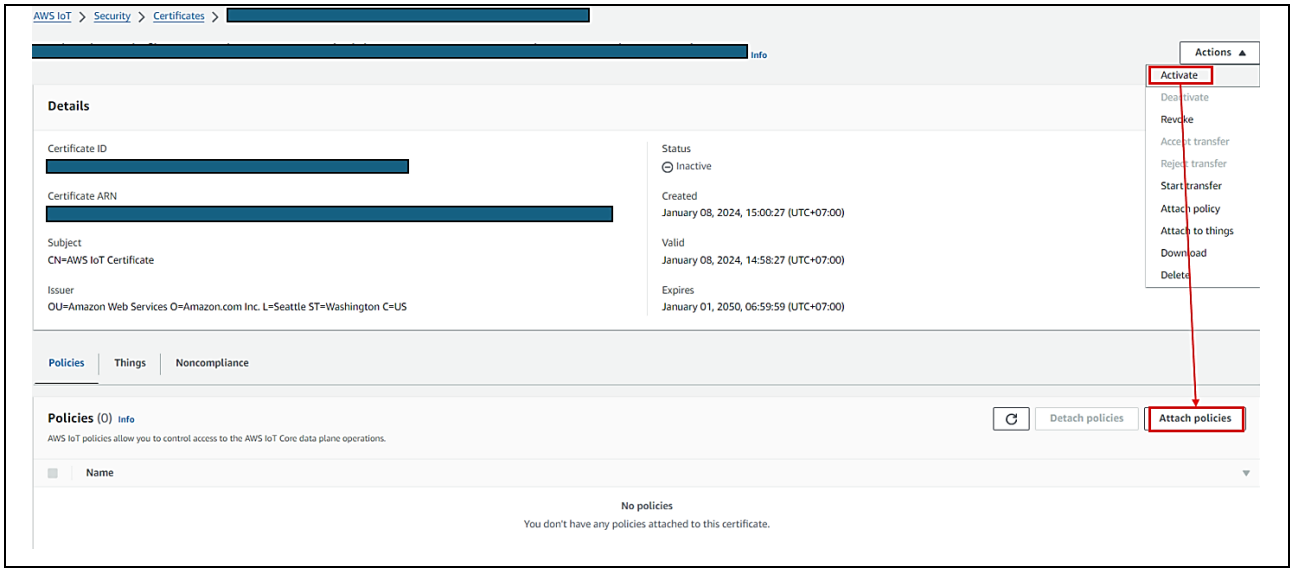

Figure 162. Certificate Settings: Attach Policies (1/2)

Clicking the Attach policies button opens the dialog box shown in **Figure 163**.

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.

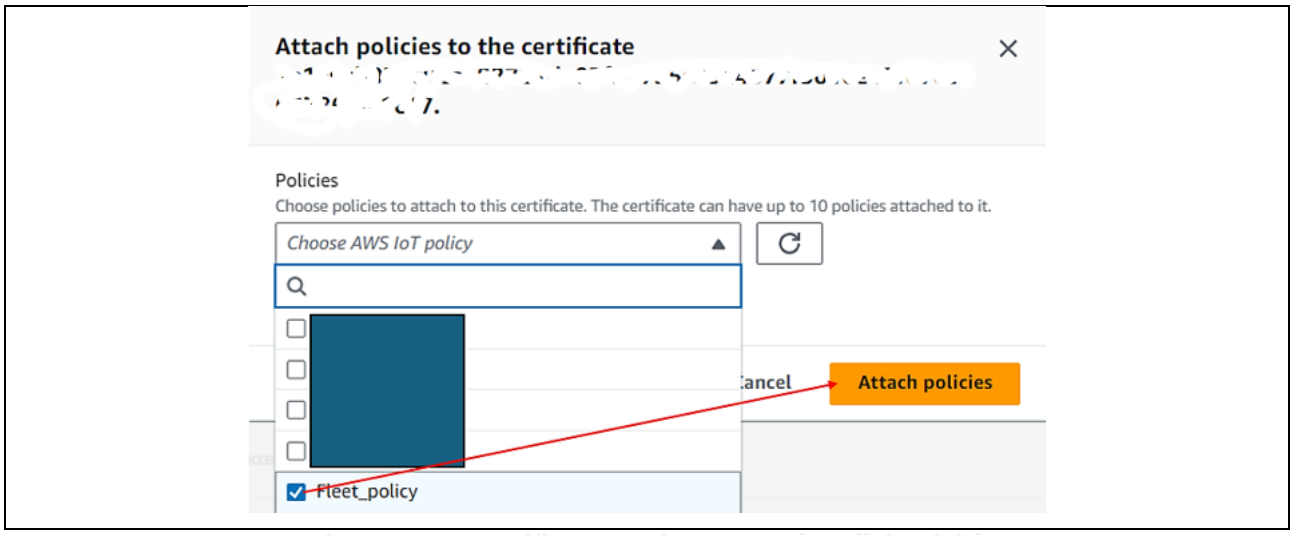

Figure 163. Certificate Settings: Attach Policies (2/2)

7.2.3 Creating a Fleet Provisioning Template

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

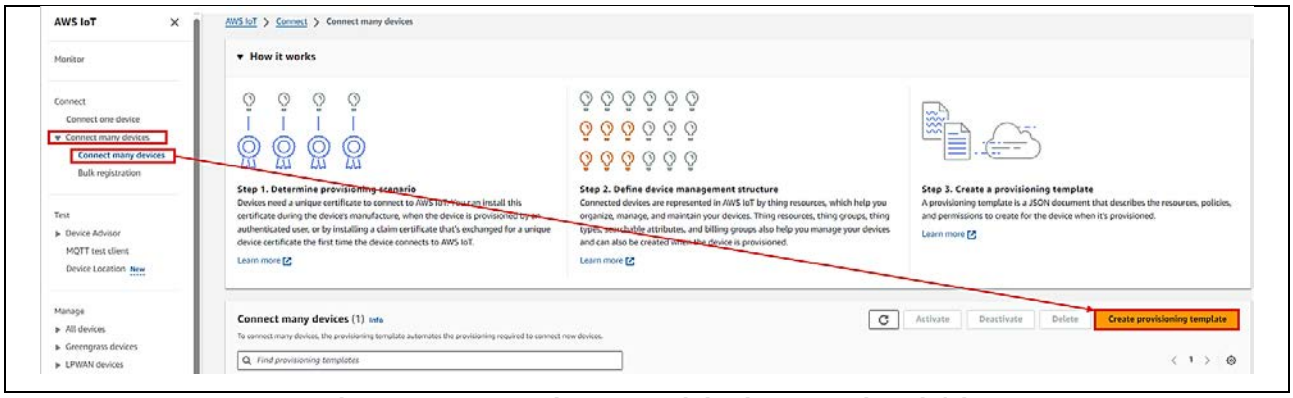

Figure 164. Creating a Provisioning Template (1/7)

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

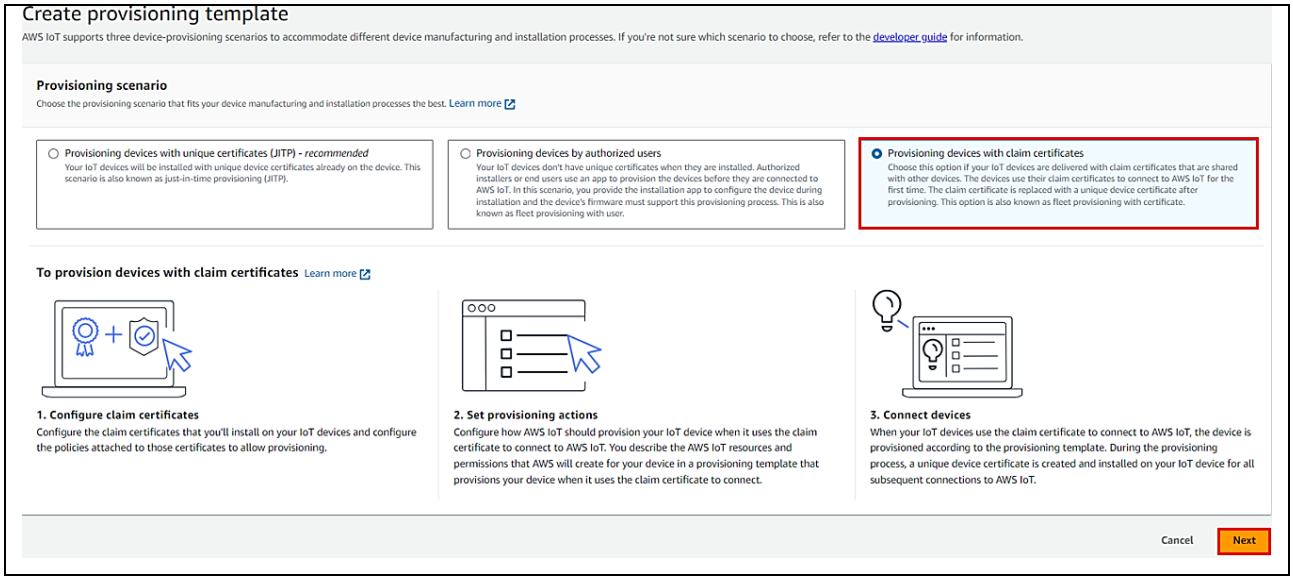

Figure 165. Creating a Provisioning Template (2/7)

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.

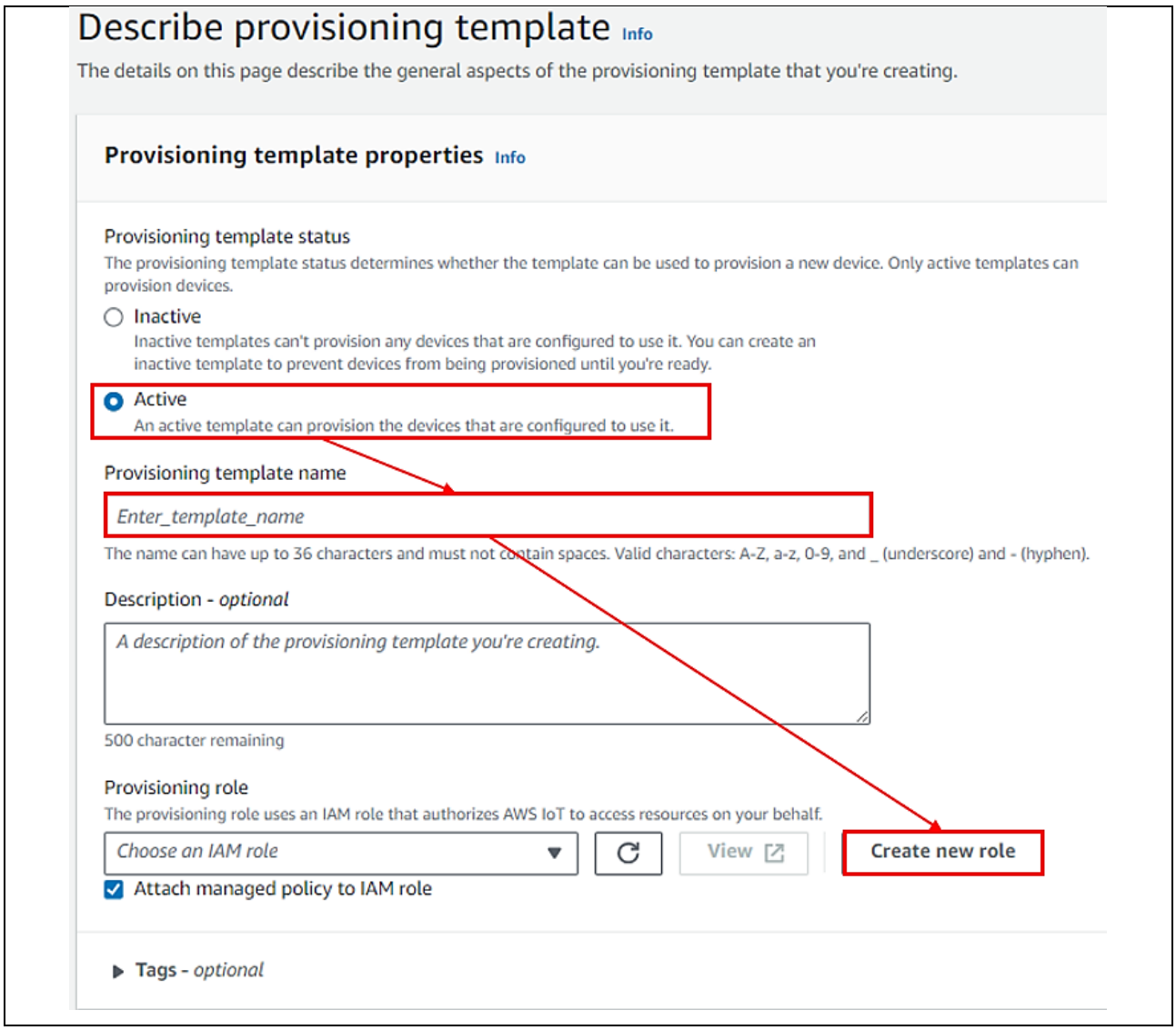

Figure 166. Creating a Provisioning Template (3/7)

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.

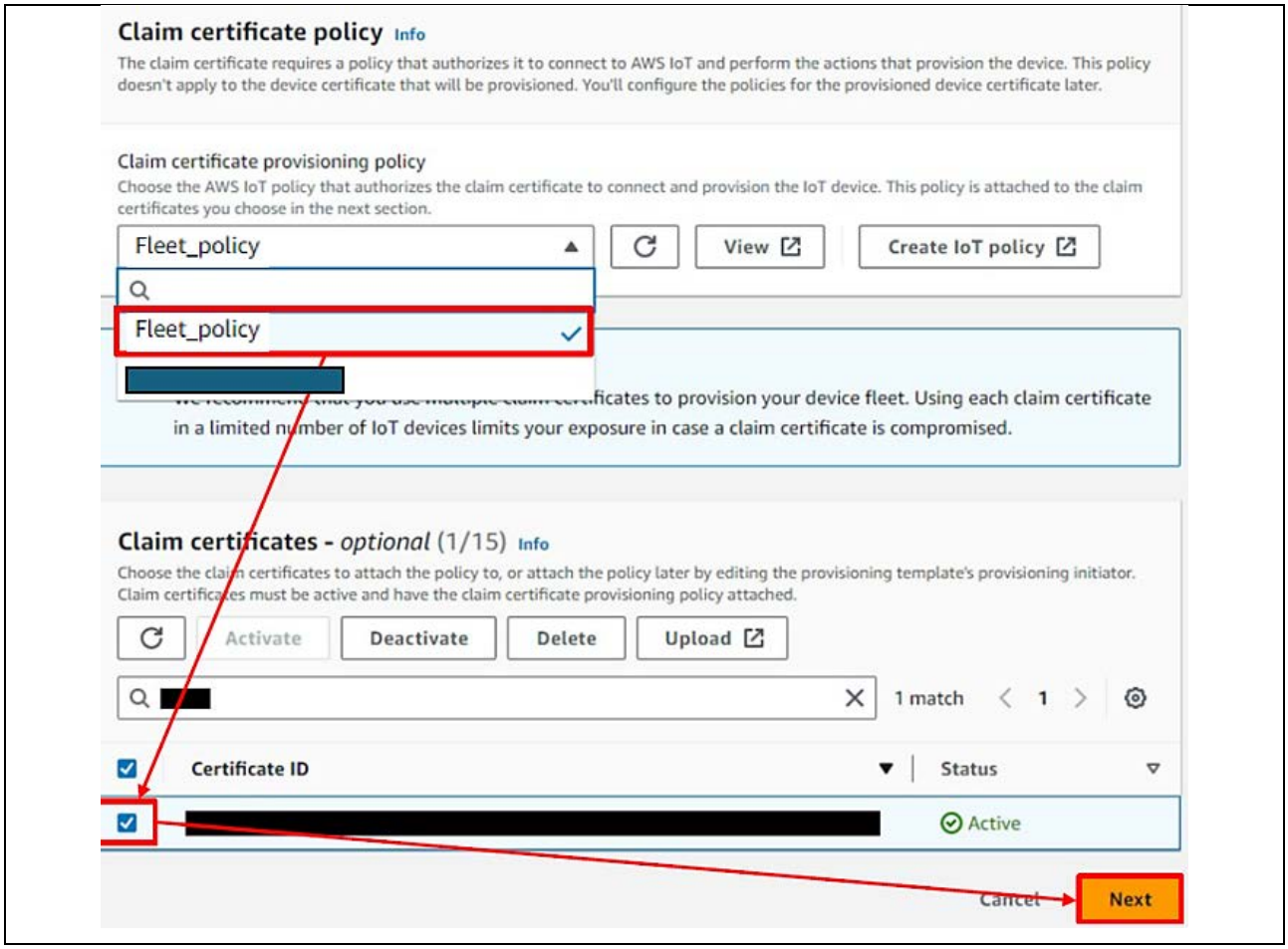

Figure 167. Creating a Provisioning Template (4/7)

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

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

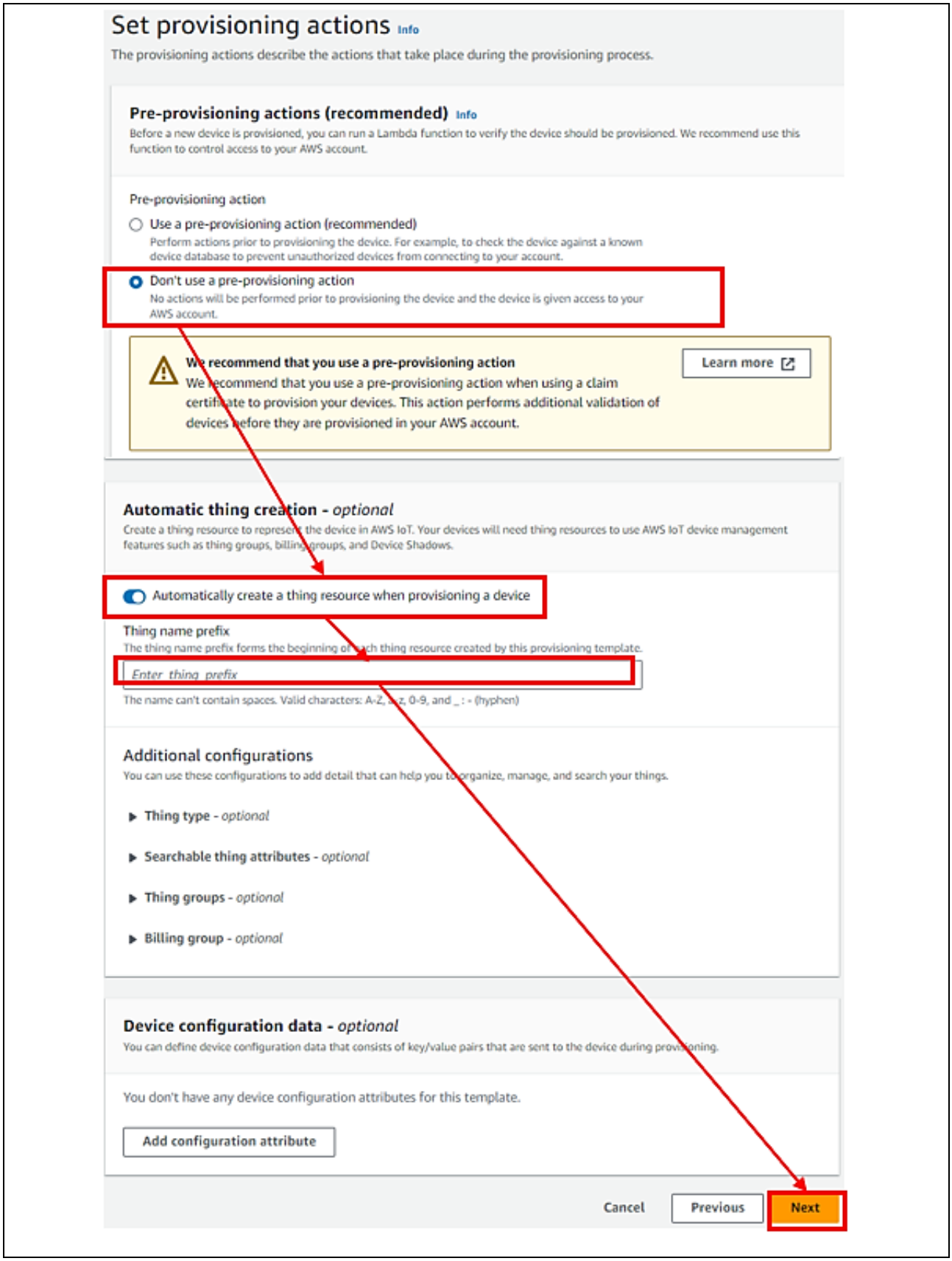

Figure 168. Creating a Provisioning Template (5/7)

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.

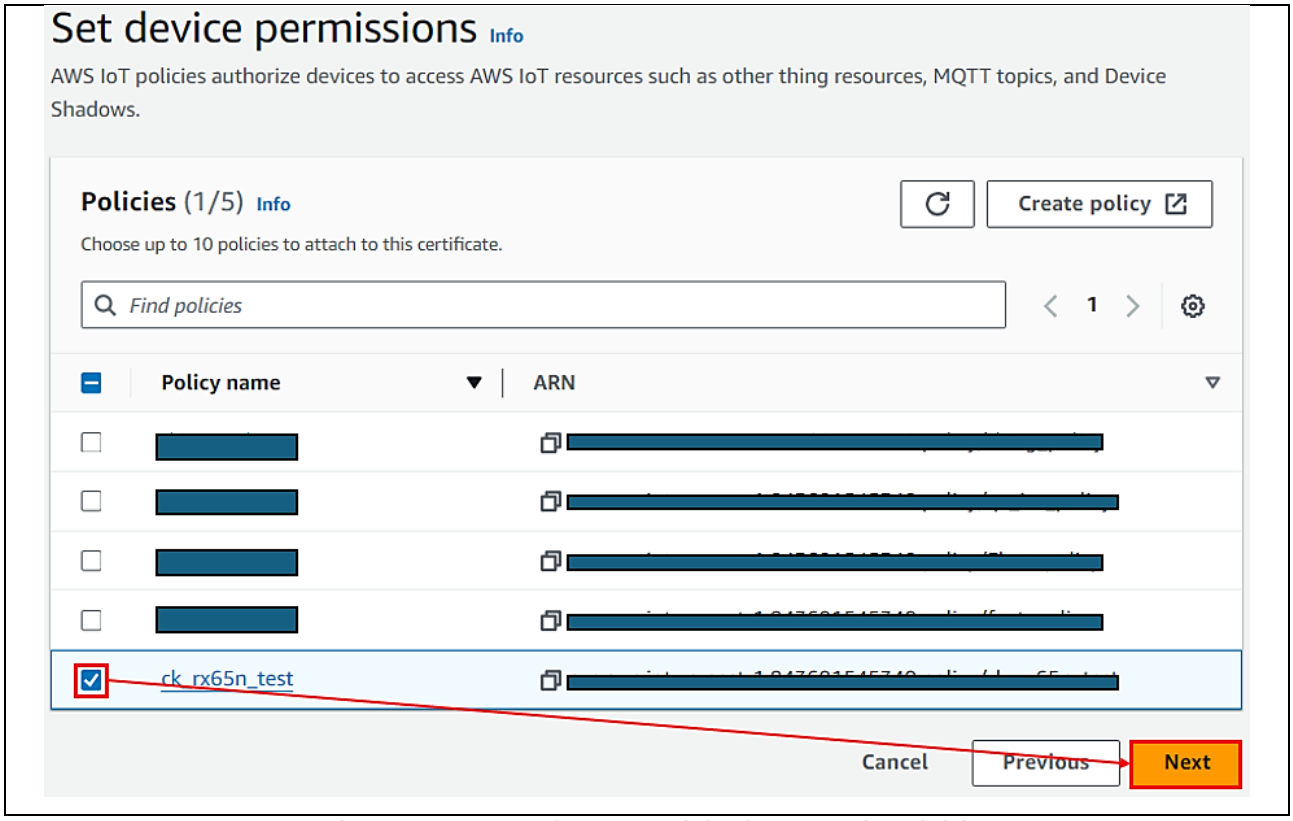

Figure 169. Creating a Provisioning Template (6/7)

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

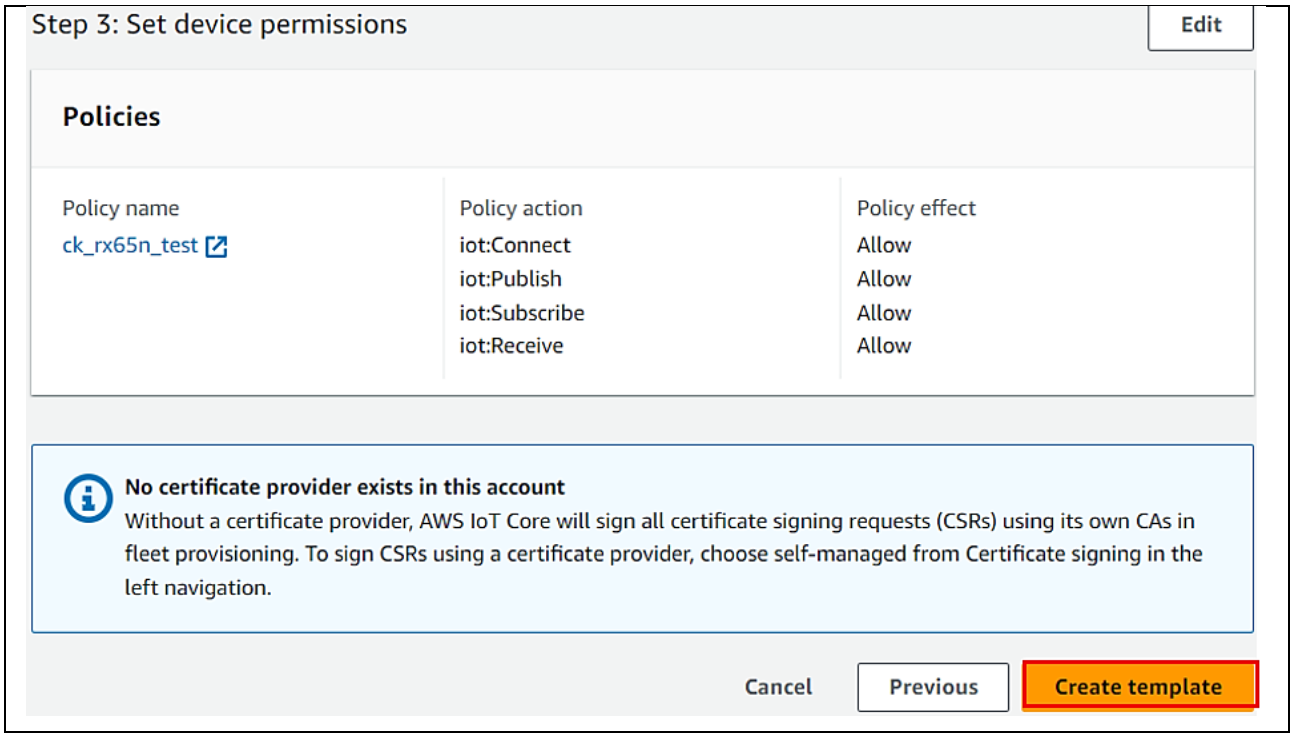

Figure 170. Creating a Provisioning Template (7/7)

7.3 Setting up the Project

Change the value of **ENABLE_FLEET_PROVISIONING_DEMO** to **1** in **aws_ether_ck_rx65n/src/frtos_config/demo_config.h:**

```
demo_config.h
76
77  /* Please select a provisioning method
78   * (0) : Pre-provisioning
79   * (1) : Fleet provisioning
80   */
81  #define ENABLE_FLEET_PROVISIONING_DEMO      (1)
```

Figure 171. Enable Fleet Provisioning macro

Build and Debug the project as described in Topic: **5.1.3 Importing the Project** (number **9, 10**)

7.4 Running Fleet Provisioning

Note**:** User can run both Fleet Provisioning and 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)

```
File  Edit  Setup  Control  Window  Help

Press any key for going to application's setting area or after 10 second, applic
ation will run automatically

Going to setting area!!!!
```

Figure 172. Save credentials for Fleet Provisioning (1/8)

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help

> Select from the options in the menu below:

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

Figure 173. 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)**

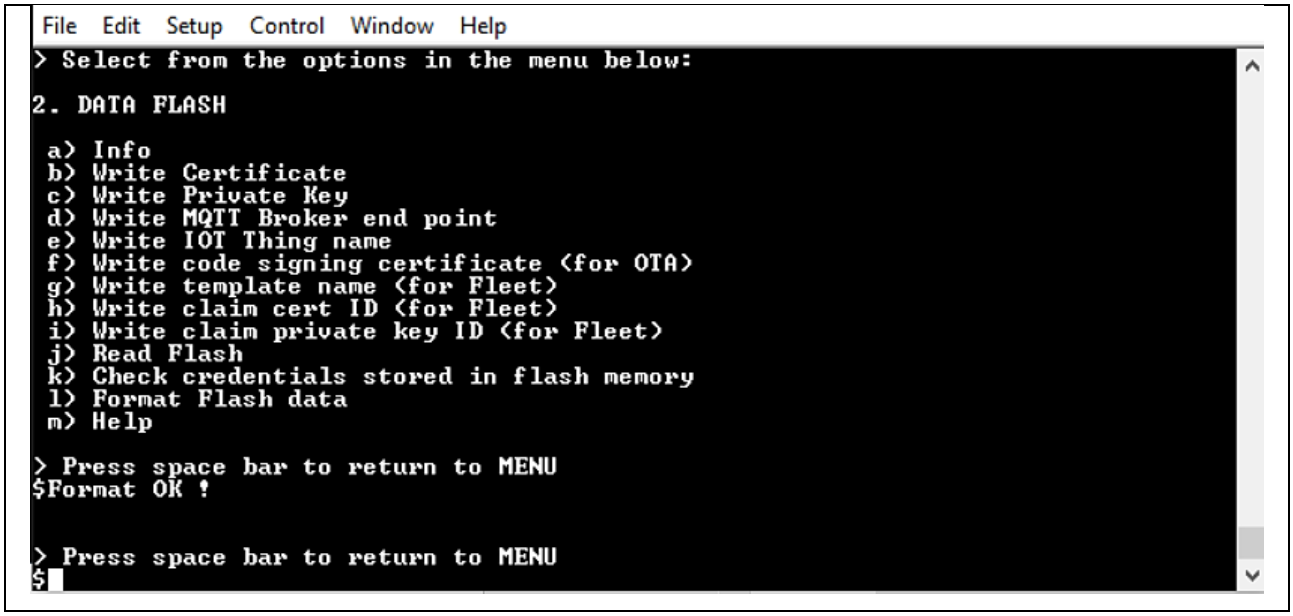

Figure 174. Save credentials for Fleet Provisioning (3/8)

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**:

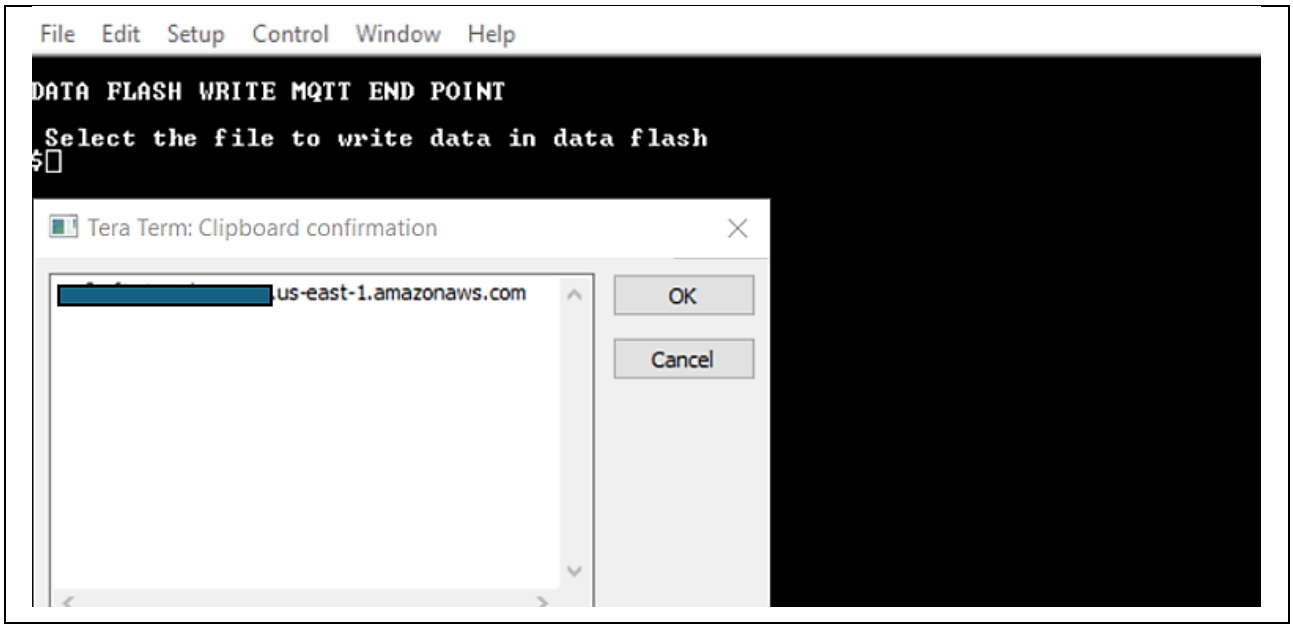

Figure 175. Save credentials for Fleet Provisioning (4/8)

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, choose "**Paste<CR>**", verify and confirm the valid string then press **OK**.

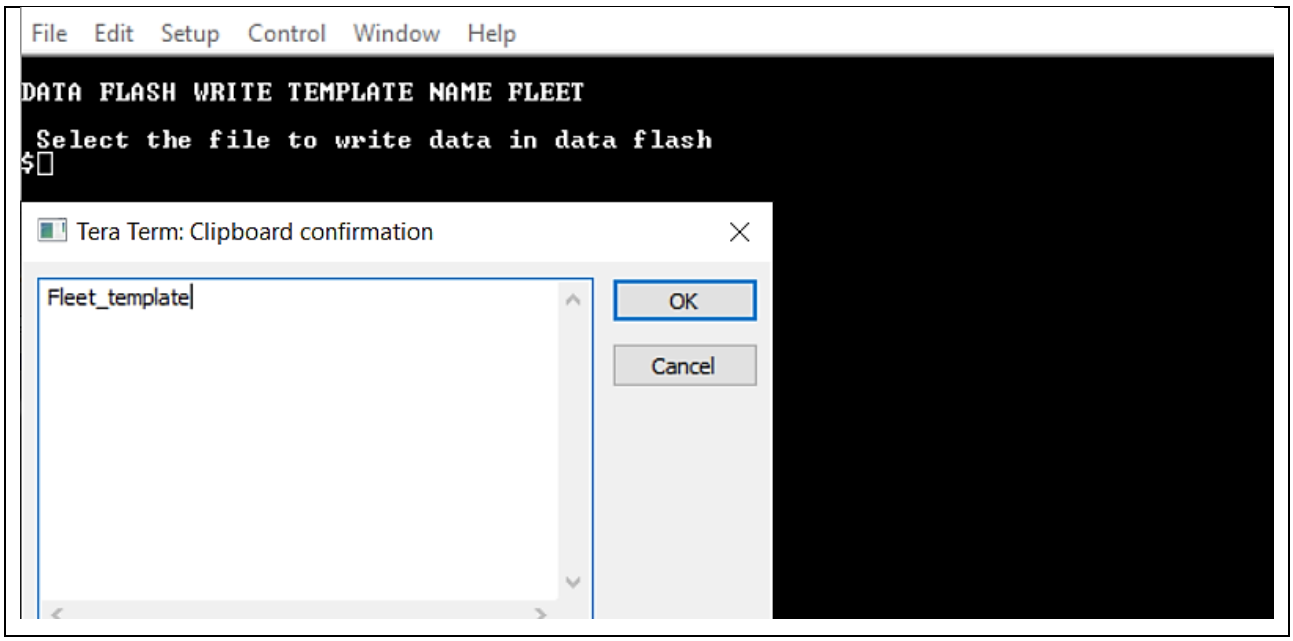

Figure 176. Save credentials for Fleet Provisioning (5/8)

5. Next, storing fleet claim cert ID, claim private key ID that users created at **0,** users downloaded (**Figure 160. Downloading the Certificate and Key Pair**):

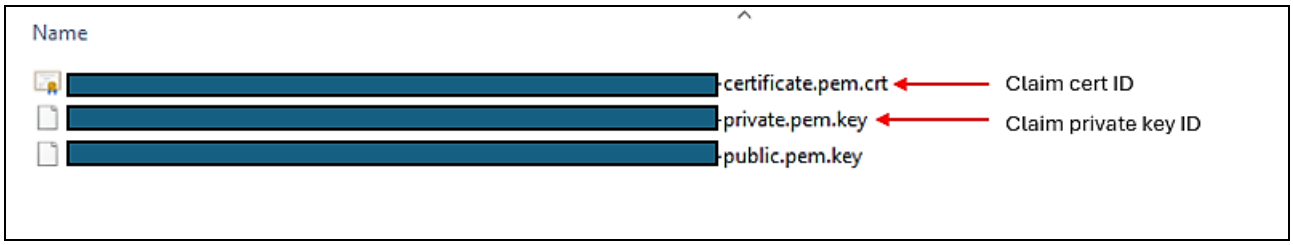

Figure 177. Save credentials for Fleet Provisioning (6/8)

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**".

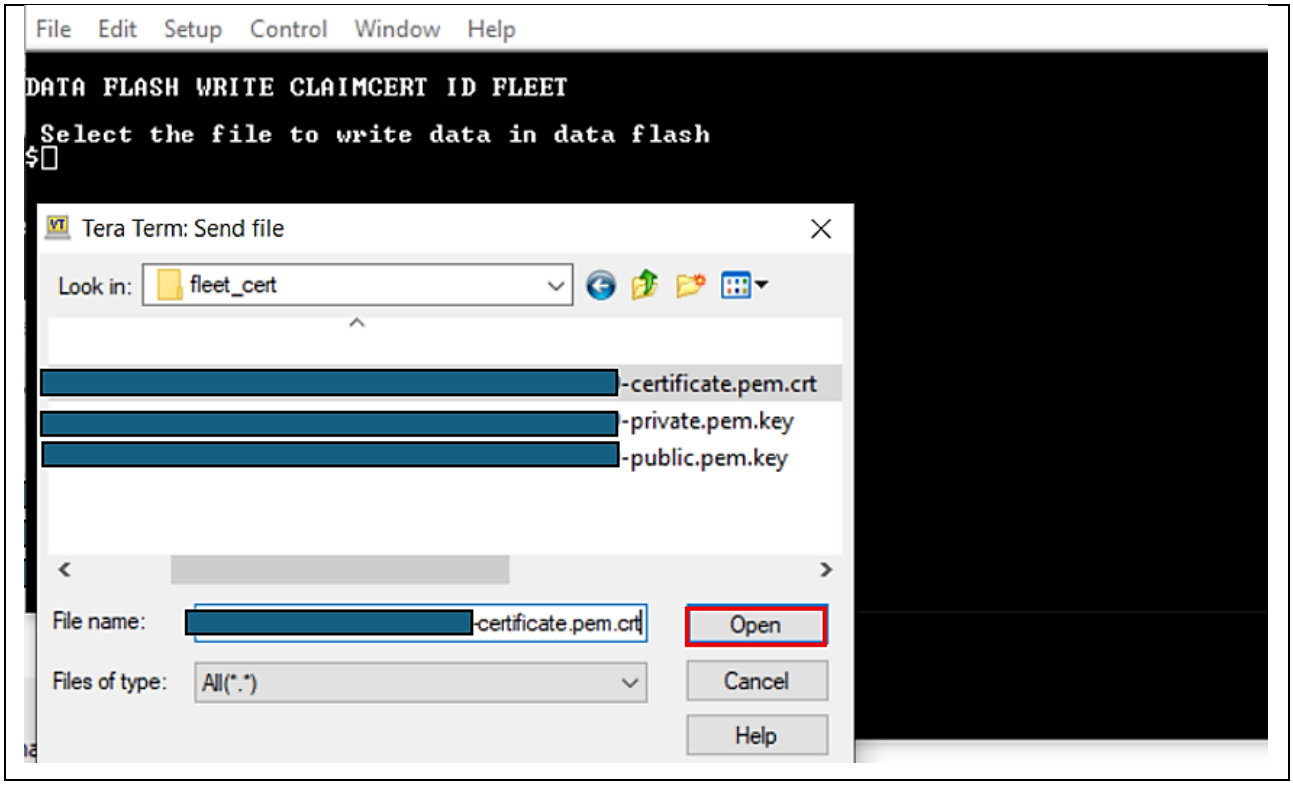

Figure 178. Save Credentials for Fleet Provisioning (7/8)

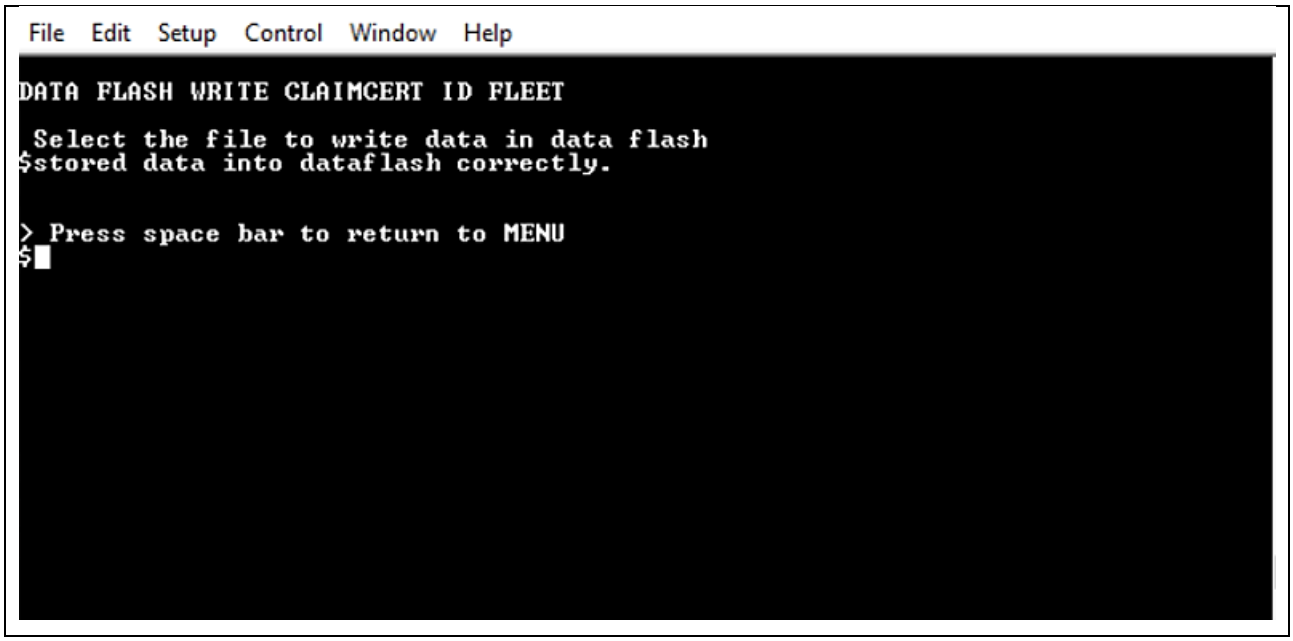

Figure 179. Save Credentials for Fleet Provisioning (8/8)

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. Start application with Fleet Provisioning:

Back to the Application Menu and press **'Run Sensor App with MQTT'** to run application:

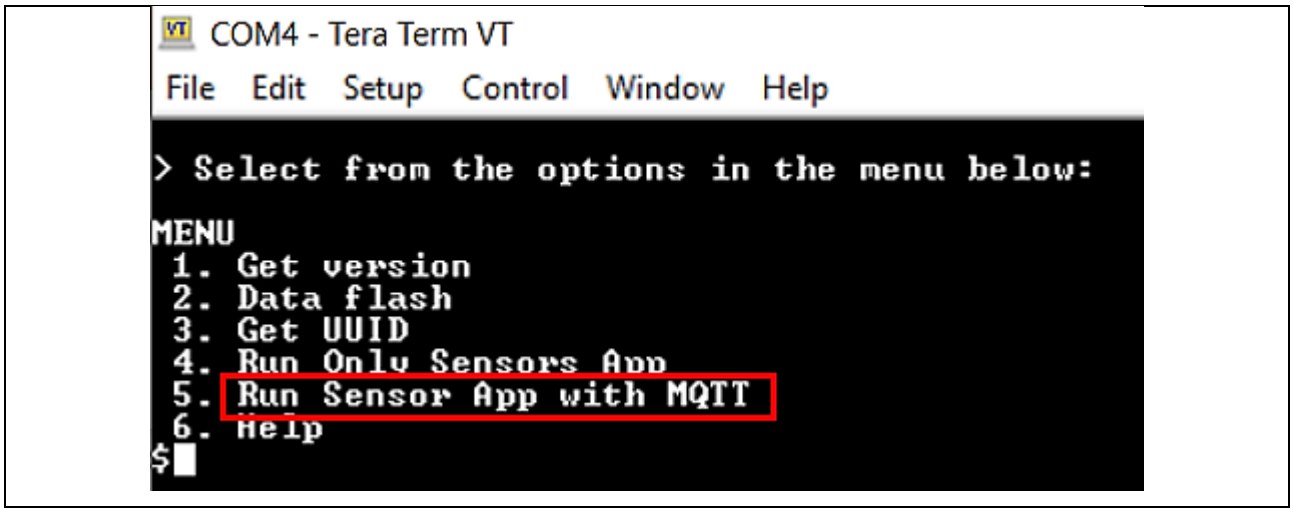

Figure 180. Running Application with Fleet (1/8)

Application will check stored data flash. When it has enough necessary credentials, 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.

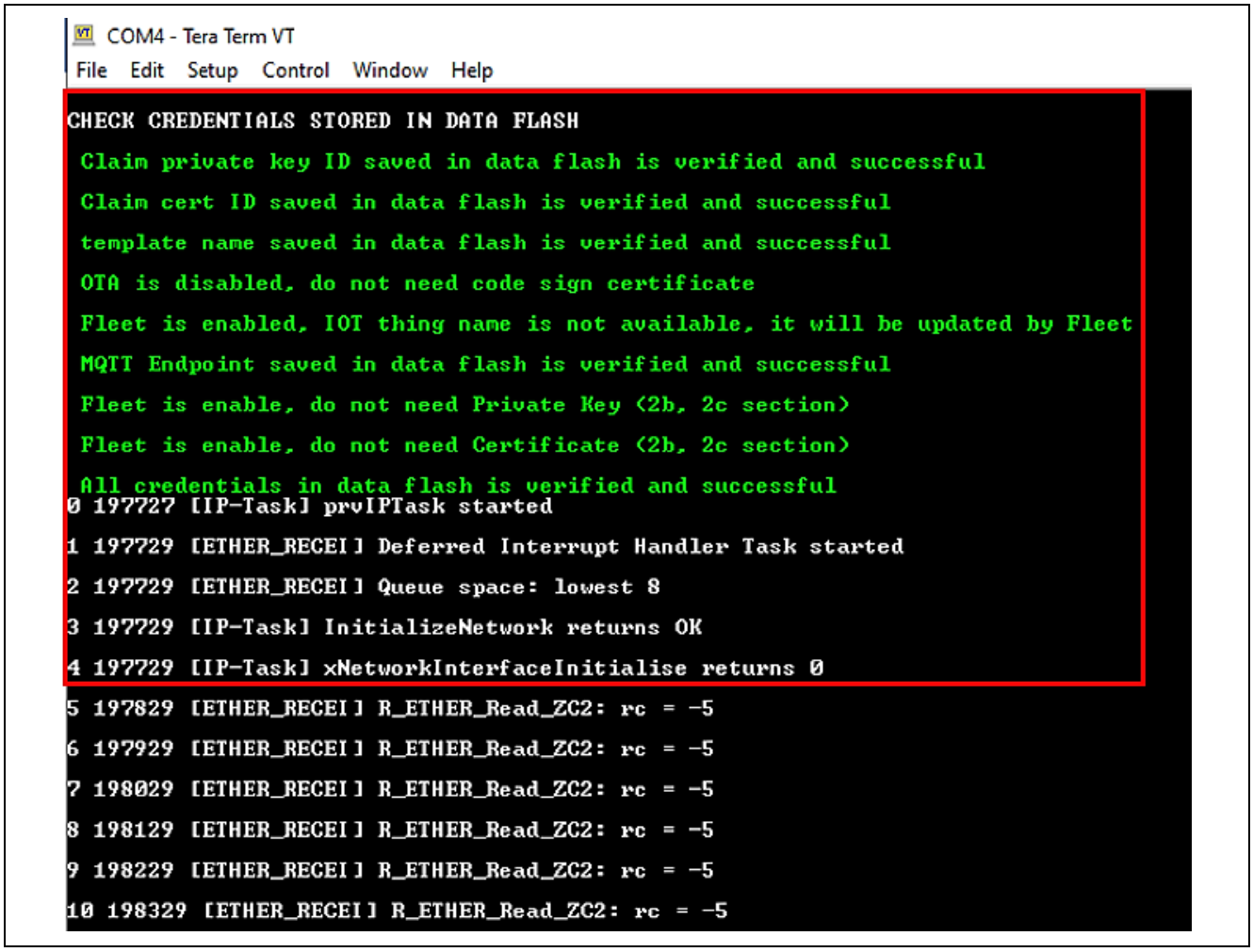

Figure 181. Running Application with Fleet (2/8)

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help

41 216763 [IP-Task] Subnet Mask: 255.255.255.0

42 216763 [IP-Task] Gateway Address: 192.168.3.1

43 216763 [IP-Task] DNS Server Address: 8.8.8.8

44 216769 [ETHER_RECEI] Network buffers: 1 lowest 1

45 217005 [CLI] Initialise the RTOS's TCP/IP stack

46 217005 [CLI] ---------STARTING DEMO---------

47 217009 [DemoTask] [INFO] ---------Start Fleet Provisioning Task---------

48 217016 [ETHER_RECEI] Heap: current 186344 lowest 182896

49 217195 [DemoTask] [INFO] Establishing MQTT session with claim certificate...
50 217202 [DemoTask] [INFO] Using default rootCA cert.
51 217202 [DemoTask] [INFO] Create a TCP connection to a[redacted]-ats.iot.us-east-1.amazonaws.com:8883.
52 217244 [DemoTask] FreeRTOS_ProcessDNSCache: add: '[redacted]-ats.iot.us-east-1.amazonaws.com' @ 34378843ip (TTL 60)

53 217251 [DemoTask] DNS[0xDDDC]: The answer to '[redacted]-ats.iot.us-east-1.amazonaws.com' (52.55.136.67) will be stored

54 217262 [DemoTask] FreeRTOS_connect: 54649 to 34378843ip:8883

55 217266 [IP-Task] prvSocketSetMSS: 1400 bytes for 34378843ip:8883

56 217267 [IP-Task] prvWinScaleFactor: uxRxWinSize 2 MSS 1400 Factor 0

57 217267 [IP-Task] Connect[34378843ip:8883]: next timeout 1: 3000 ms

58 217502 [IP-Task] MSS change 1400 -> 1452

59 217502 [IP-Task] TCP: active 54649 => 34378843ip:8883 set ESTAB (scaling 1)

60 217513 [DemoTask] [INFO] Established TCP connection with a[redacted]-ats.iot.us-east-1.amazonaws.com.
61 217558 [ETHER_RECEI] Heap: current 126208 lowest 126064

62 218120 [ETHER_RECEI] Heap: current 112216 lowest 110064

63 220068 [DemoTask] [INFO] (Network connection 804d14) TLS handshake successful.
64 220075 [DemoTask] [INFO] (Network connection 804d14) Connection to a[redacted]-ats.iot.us-east-1.amazonaws.com established.
65 220362 [DemoTask] [INFO] MQTT connection established with the broker.
66 220362 [DemoTask] [INFO] MQTT connection successfully established with broker.


67 220375 [DemoTask] [INFO] Established connection with claim credentials.
68 220376 [DemoTask] [INFO] SUBSCRIBE topic $aws/certificates/create-from-csr/cbor/accepted to broker.


69 220653 [DemoTask] [INFO] MQTT_PACKET_TYPE_SUBACK.


70 225575 [DemoTask] [INFO] SUBSCRIBE topic $aws/certificates/create-from-csr/cbor/rejected to broker.


71 225842 [DemoTask] [INFO] MQTT_PACKET_TYPE_SUBACK.


72 230761 [DemoTask] [INFO] the published payload:íxcertificateSigningRequesty-----BEGIN CERTIFICATE REQUEST-----
MI[redacted]JEMzMz
MF8wMDZDM73 230776 [DemoTask] [INFO] PUBLISH sent for topic $aws/certificates/create-from-csr/cbor to broker with packet ID 3.
```

Figure 182. 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.

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help

127 274576 [DemoTask] [INFO] Sucessfully established connection with provisioned credentials.
128 274583 [DemoTask] [INFO] (Network connection 804d14) TLS close-notify sent.
129 274818 [IP-Task] TCP: No active socket on port 5706 (34378843ip:8883)

130 274818 [DemoTask] [INFO] Demo iteration 1 is successful.
131 274825 [DemoTask] [INFO] Demo completed successfully.
132 274830 [DemoTask] [INFO] -------Fleet Provisioning Task Finished-------
```

Figure 183. 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:

```
152 275479 [MQTT] [INFO] Established TCP connection with a[redacted]ats.iot.us-east-1.amazonaws.com.
153 276944 [MQTT] [INFO] (Network connection 805888) TLS handshake successful.
154 276944 [MQTT] [INFO] (Network connection 805888) Connection to a[redacted]ats.iot.us-east-1.amazonaws.com established.
155 276944 [MQTT] [INFO] Creating an MQTT connection to the broker.
156 277229 [MQTT] [INFO] MQTT connection established with the broker.
157 277229 [MQTT] [INFO] Successfully connected to MQTT broker.
158 277235 [sensor_thre] I2C bus 0 setup success

159 277236 [sensor_thre] HS3001 open sensor instance successful: 0

160 277236 [sensor_thre] ICP10101 open sensor instance successful: 0

161 277236 [obi203_thre]
OB1203 Device open success
162 277245 [sensor_thre] ICM20948 open sensor instance successful: 0

163 277288 [MainAWSDemo] [INFO] --------Start AWS - MQTT Demo Task --------

164 278738 [MainAWSDemo] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
165 279492 [zmod_thread] ZMOD4410 open sensor instance successful: 0

166 279722 [zmod_thread] ZMOD4510 open sensor instance successful: 0

167 279723 [zmod_thread] Task zmod4410 measurement Success:0

168 279741 [MainAWSDemo] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
169 279748 [MainAWSDemo] [Send Data] ZMOD4410-IAQ TVOC: 000.000

170 279750 [MainAWSDemo] [Send Data] ZMOD4410-IAQ ETOH: 000.000

171 279756 [MainAWSDemo] [Send Data] ZMOD4410-IAQ ECO2  : 000.000
```

Figure 184. 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: FleetFPDemoID_xxxxxxxxxxx**"

```
COM4 - Tera Term VT
File  Edit  Setup  Control  Window  Help

98 257203 [DemoTask] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
99 257210 [DemoTask] [INFO] State record updated. New state=MQTTPubAckSend.
100 257212 [DemoTask] [INFO] Received accepted response from Fleet Provisioning RegisterThing API.
101 261913 [DemoTask] [INFO] Received AWS IoT Thing name: FleetFPDemoID_[redacted]
102 261927 [DemoTask] [INFO] AWS IoT Thing name is saved to Data Flash
103 261927 [DemoTask] [INFO] UNSUBSCRIBE sent topic $aws/provisioning-templates/Fleet/provision/cbor/accepted to broker.
```

Figure 185. Running Application with Fleet (5/8)

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

Under **All devices**, select **Things**. The registered thing is shown below (It will have a name that matches the "Received AWS IoT Thing name" shown in **Figure 185**):

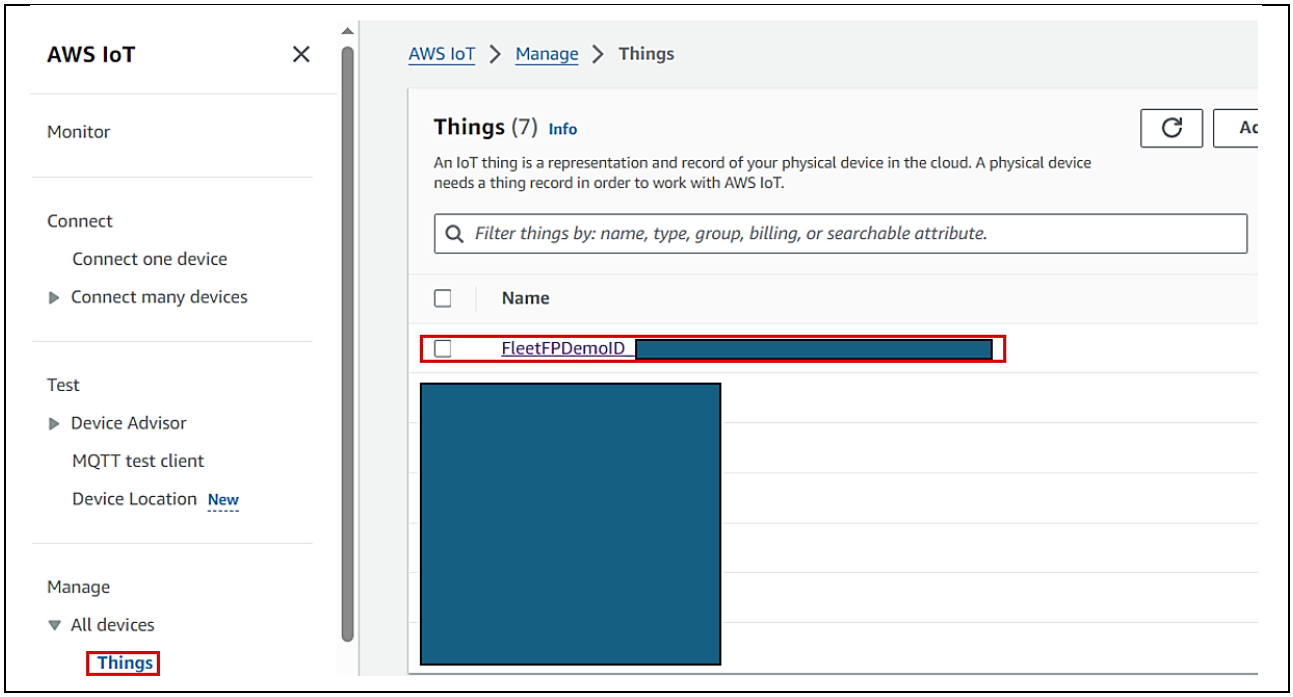

Figure 186. Running Application with Fleet (6/8)

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

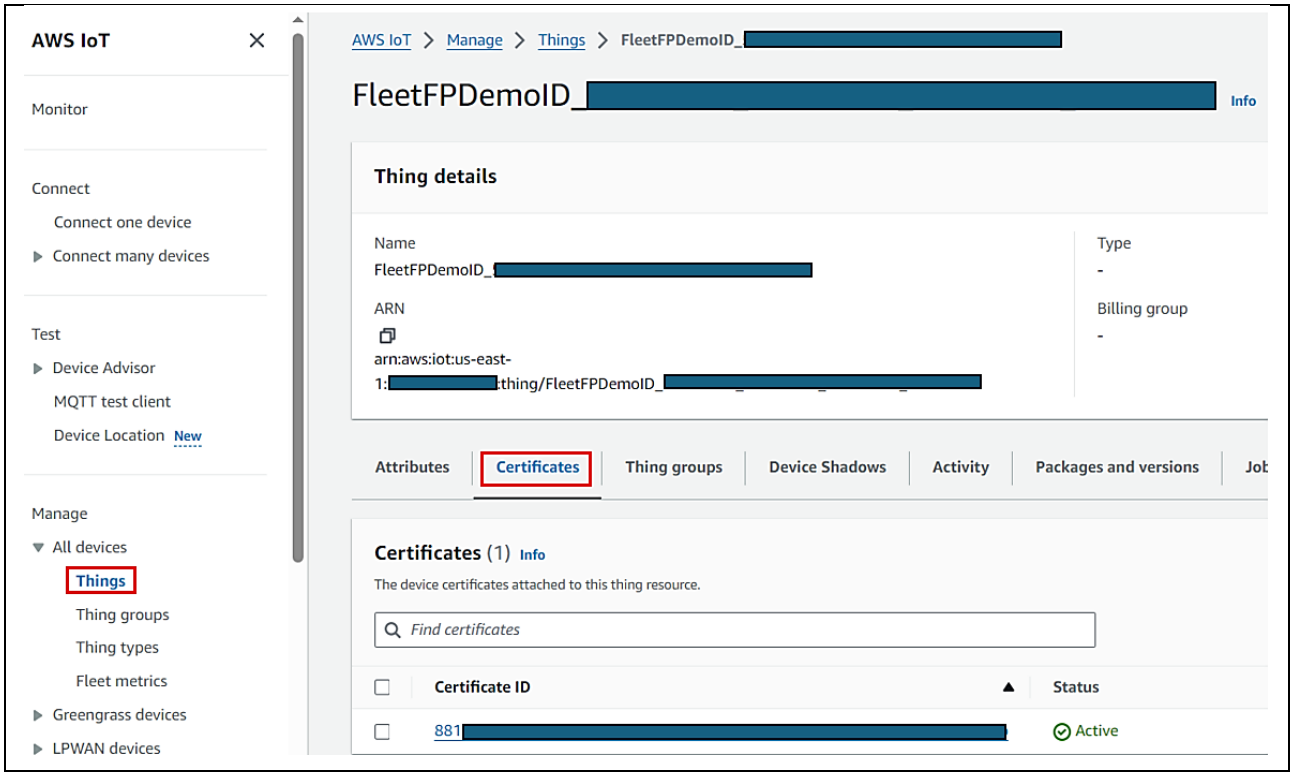

Figure 187. Running Application with Fleet (7/8)

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

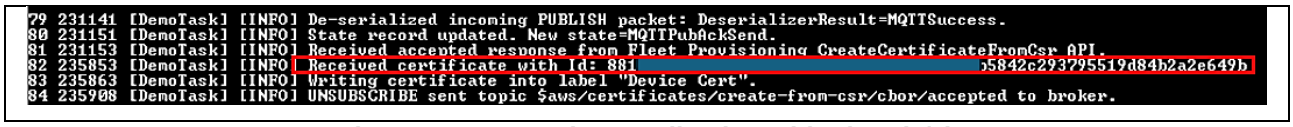

Figure 188. Running Application with Fleet (8/8)

8. Note and Troubleshooting

8.1 Sensor Stabilization Time

Each sensor has a stabilization time during which correct values cannot be read.

The following table shows the details of sensor stabilization times.

Table 12. Sensor Stabilization Time

Sensor Name	When power up first time	After soft or hard reset
ZMOD4410 IAQ	Up to 1 minutes	Up to 1 minute
ZMOD4510 OAQ	Up to 1.5 hours	Up to 1 hour
OB1203	Up to 20 minutes (After putting finger on sensor, it may take up to 60 seconds to sense data)	Up to 20 seconds (After putting finger on sensor, it may take up to 60 seconds to sense data)
HS3001	Up to 30 seconds	Up to 10 seconds
ICP	Up to 30 seconds	Up to 10 seconds
ICM	Up to 30 seconds	Up to 10 seconds

8.2 Connection Issue When Using Ethernet (Wired cable)

The Ether PHY only supports full-duplex communication. If your router or Ethernet hub only supports half duplex, it cannot connect to the internet. Please use full duplex devices.

8.3 Current Supply Short Issue When Using RYZ014A

If the CK-RX65N V1 board is not powered through the Debug port (J14), the current available to the board may be limited to 100mA. When using the supplied RYZ014A Pmod module with other code (found here: RYZ014A - LTE Cat-M1 Cellular IoT Module | Renesas) be aware that this Pmod has a maximum operating current of **480 mA** dependent upon the LTE band, Tx/Rx settings, and network coverage. Please ensure that the host board can supply sufficient power or provide supplemental USB power via CN4 on the Pmod to avoid RF instability.

8.4 When Build Errors Occur

If a 'No such file or directory' error occurs, the project path may be too long. When the path is longer than 256 characters, e^2 studio outputs errors at build time.
When this error occurs, move the project to a shorter path location (such as, under C:\).

8.5 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 Getting the Account 10 USD for Trial of AWS**, try the following:

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

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

8.6 Notes on Performing Firmware Update Over-The-Air on AWS FreeRTOS

The following symptoms occur when performing Firmware Update Over-The-Air (FOTA) on AWS FreeRTOS.

1. Device (thing) names and credential information are overwritten.
2. Processing of OTA fails after recovering from low power consumption mode.

For more information, refer to the Tool News at:

[Notes] RX Family Notes on Performing Firmware Update Over-The-Air on AWS FreeRTOS (renesas.com)

8.7 When the Trial 10 USD is Used Up

When the trial 10 USD is used up, status will change to “Quarantined”. At this time, the user cannot download the certificate or go to Dashboard.

Figure 189. Account Used Up Trial 10 USD

Note: Please add payment options before the account is quarantined and enable the option on the dashboard. If the account is quarantined, please contact Renesas Customer Support.

To enable option on the dashboard, go to the user profile.

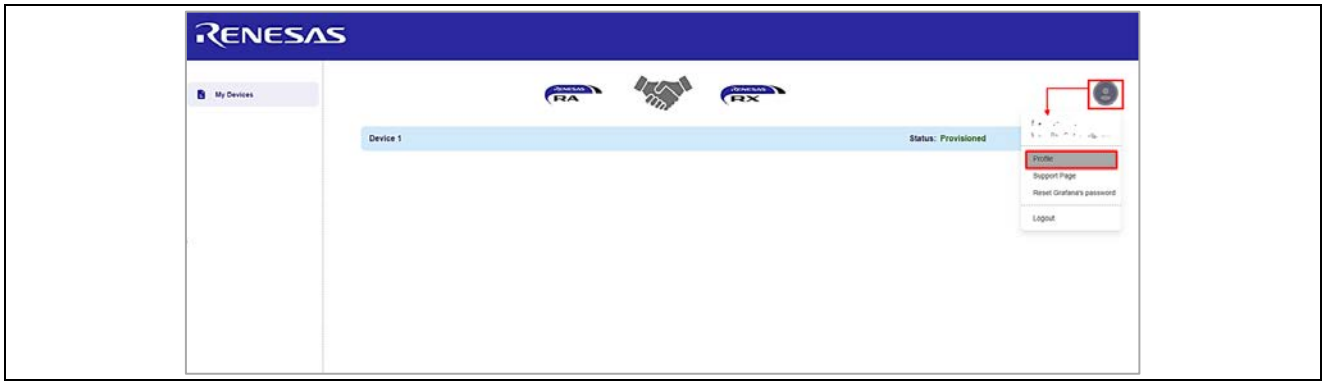

Figure 190. Choose Profile

Choose "**Yes**" for the **Payment Preference Update Confirmation**.

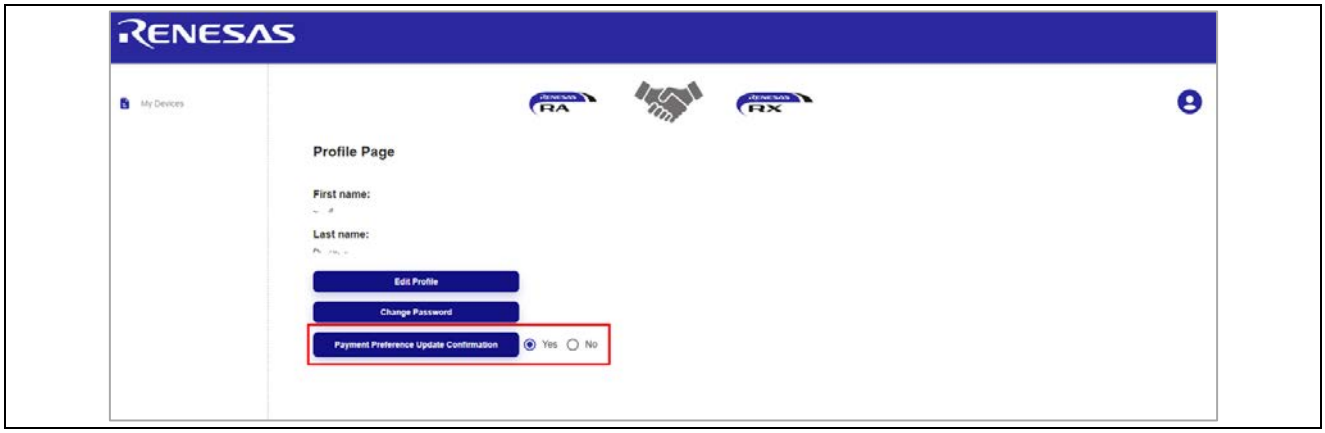

Figure 191. Payment Confirmation

To add a credit card to continue using this account or using another AWS account, go to https://cloud-ra-rx.awsapps.com/start#/, sign up with AWS account.

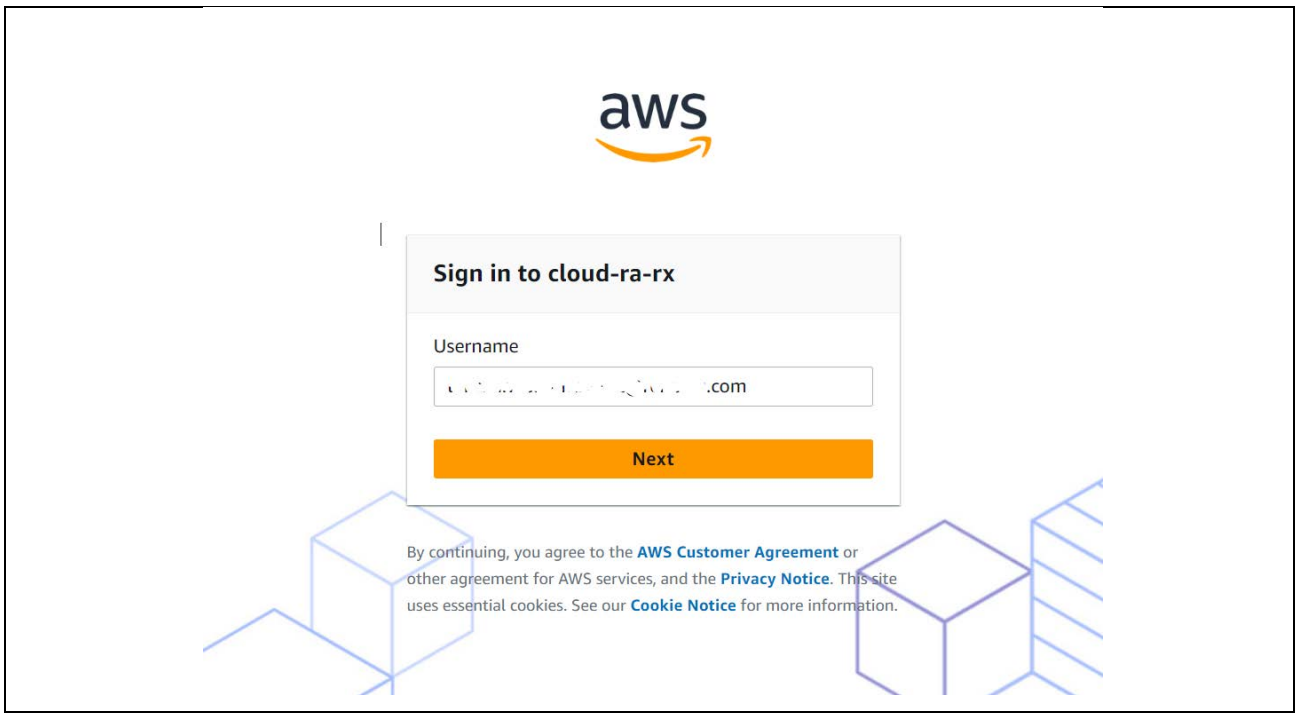

Figure 192. Sign up to AWS Account

Go to **Manage Console > Account > AWS Billing > Payment preferences > Add payment method.**

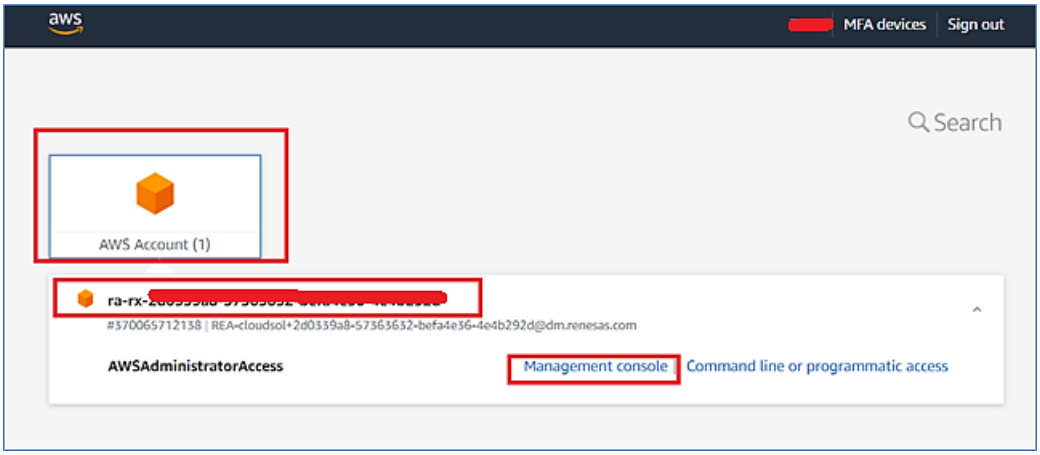

Figure 193. Add Credit Card for Trial Account (1/4)

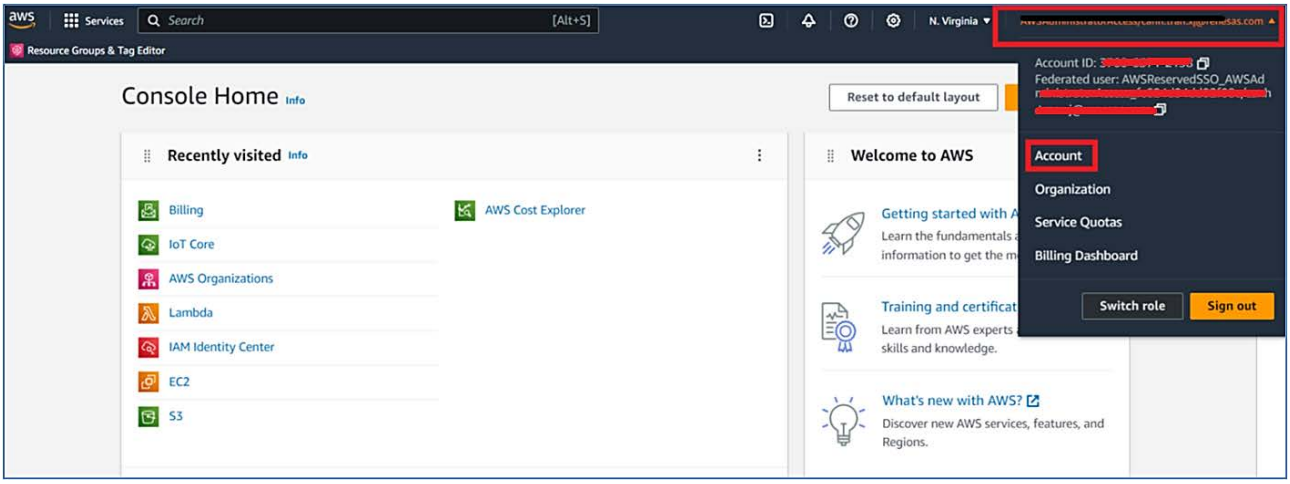

Figure 194. Add Credit Card for Trial Account (2/4)

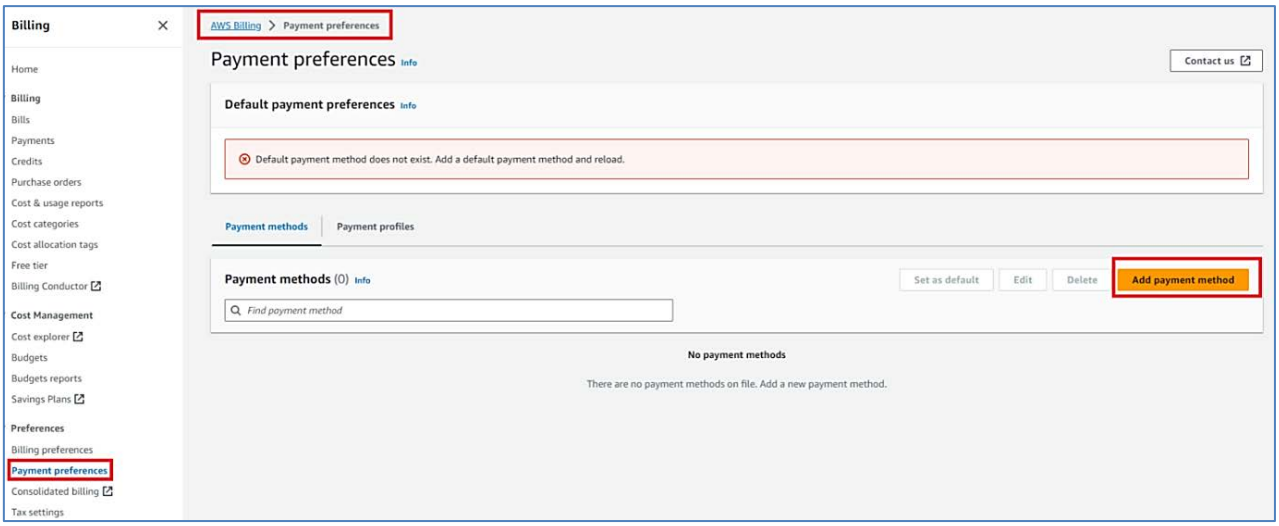

Figure 195. Add Credit Card for Trial Account (3/4)

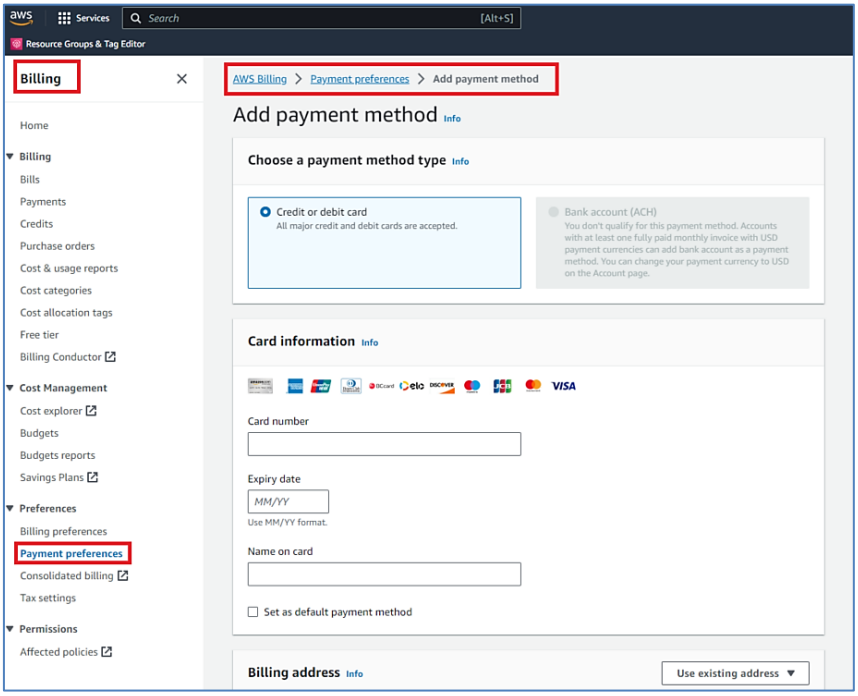

Figure 196. Add Credit Card for Trial Account (4/4)

8.8 How to Enable/Disable EC2 Instance

AWS trial accounts start billing immediately after device registration. The following steps will disable the EC2 instance, saving the user's credits.

Access AWS account from https://cloud-ra-rx.awsapps.com/start#/ using the dashboard credentials. (Refer to **Figure 192** to log in).

To disable the EC2 instance, use the following steps:

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

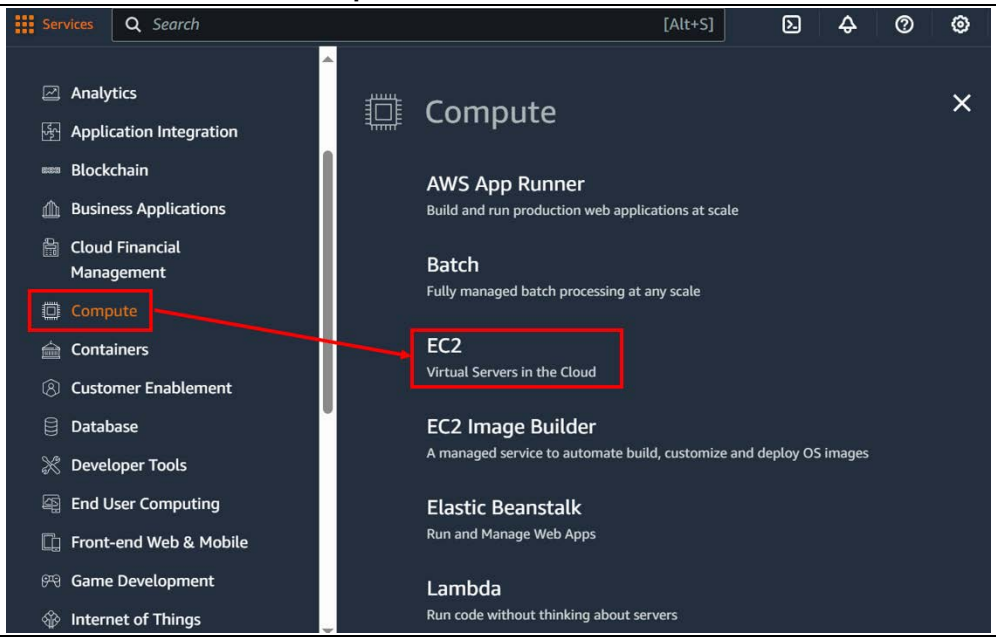

Figure 197. EC2 AWS Service

2. Choose the instance, then change the **Instance state** to **Stop instance**

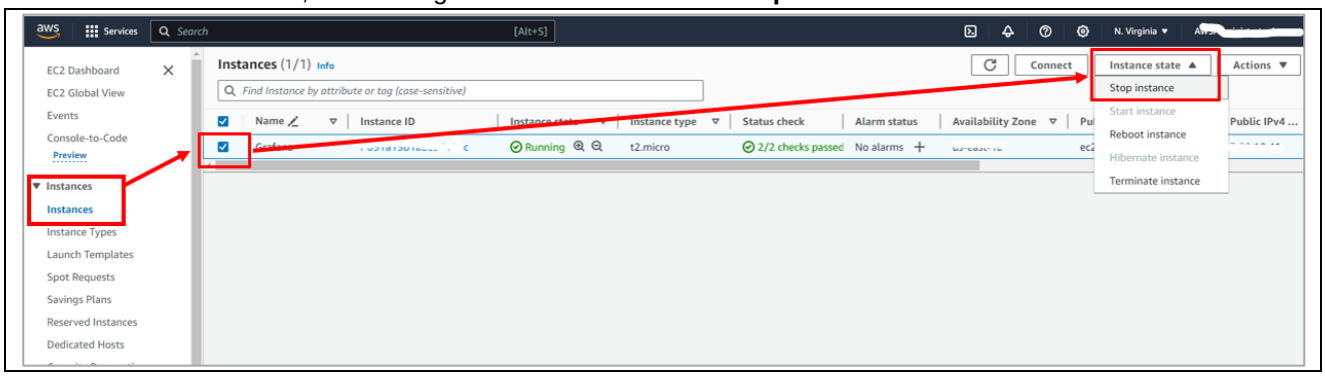

Figure 198. Disable Instance

Note**:** If you wish to use the dashboard again, please enable the EC2 instance before using it.

8.9 How to check the total amount spent in AWS account

Access AWS account from https://cloud-ra-rx.awsapps.com/start#/ using the dashboard credentials. (Refer to the **Figure 192** to login).

Go to **Account > Bills.**

User can choose the **"Billing period"** to see the amount spent during that period:

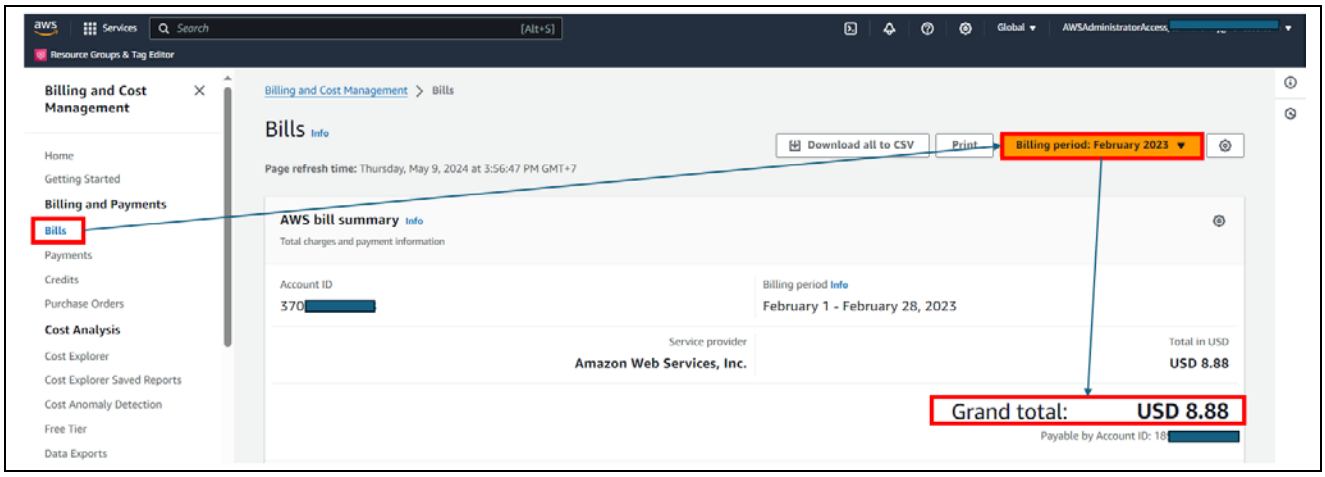

Figure 199. Check the amount spent in AWS Account

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 Installing Python** 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 pins 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.

9. 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 V1 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	Jun.14.22		First version
1.10	Jun.29.22	15	Improved 5.1.4 Activating a SIM card on MicroAI Launchpad of activation way of SIM
		49	Added the restrictions section
		Program	Fixed the program about getting the issue of UUID lacking parts of the number.
1.20	Jul.06.22	10-11	Added 8. Apply Patch files
		50	Removed “SpO2 sensor of OB1203 (Cellular version of Project)” from the restriction section. Updated 5.1 Communication time (Cellular version of Project).
		Program	Updated Cellular version of Project - Supported SpO2 sensor of OB1203. - Added patch file to project.
1.30	Jun.02.23	11	Added the settings for Truphone SIM
		26	Added 4.1.6 about the activation procedure for Truphone SIM
		50	Added 6.4 about when build errors occur.
			Added 6.5 about when unable to log in to the Dashboard (Grafana account)
			Added 6.6 Notes on Performing Firmware Update Over-The-Air on AWS FreeRTOS
1.31	Jan.19.24	—	Fix typo mistake
		5	Updated Figure 2
		5	Updated description for 4. Connection to AWS section
		11, 19	Removed information of MicroAI Sim card, added note about SIM Card’s information
		—	Removed 4.1.5.2 Activate a SIM card on MicroAI Launchpad Section due to the MicroAI SIM card is discontinued to support CK-RX65N.
		44	Correct description for “LED Alerts” Section
		45	Added “Automatic alert from Renesas Dashboard feature” into Restrictions section
		23-26, 33-40, 46-50	Updated information about latest Renesas Dashboard (add payment, enable/disable EC2 instance, …), AWS
1.40	Jun.24.24	49-106	Added OTA feature (section 6), Fleet Provisioning feature (section 7)
		6-8	Add section “4. Cloud Connectivity Application Example”: Add application overview.
		9-48	Re-arrange section “5. Connection to AWS”
		110	Add restrictions (8.9 → 8.14) in “Note and Troubleshooting”

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/.