Renesas RX Family

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

Introduction

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

The document covers the following items:

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

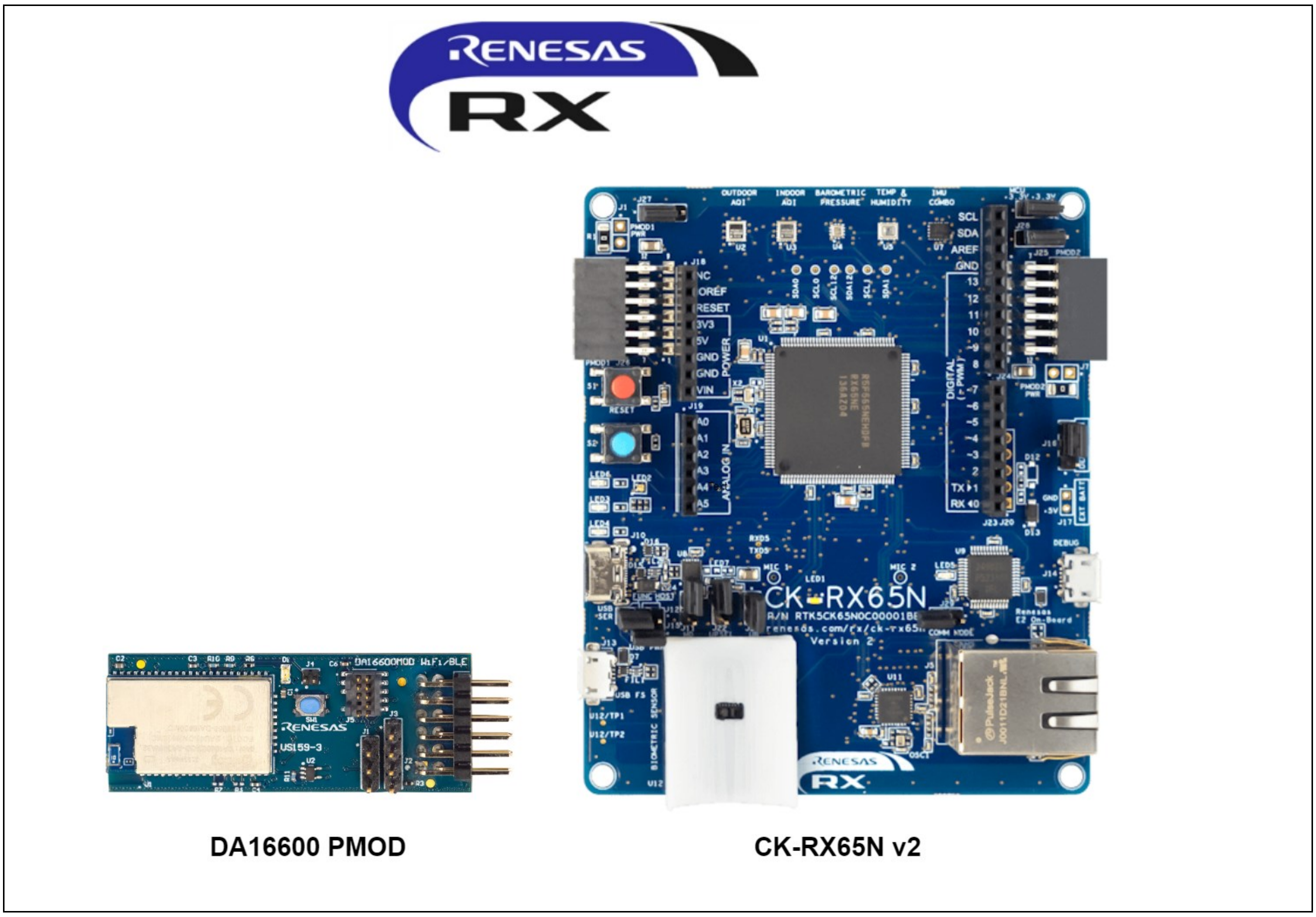


Figure 1. CK-RX65N v2 with Wi-Fi DA16600 Pmod

Contents

1. Terms

The terms used in this document are explained below.

Table 1. Terms

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

2. Overview

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

2.1 Hardware Configuration

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

Table 2. Hardware Configuration

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

2.2 Software Configuration

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

Table 3. Software Configuration

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

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

2.3 Tera term Setting

Table 4. Tera term Setting

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

3. System Diagram

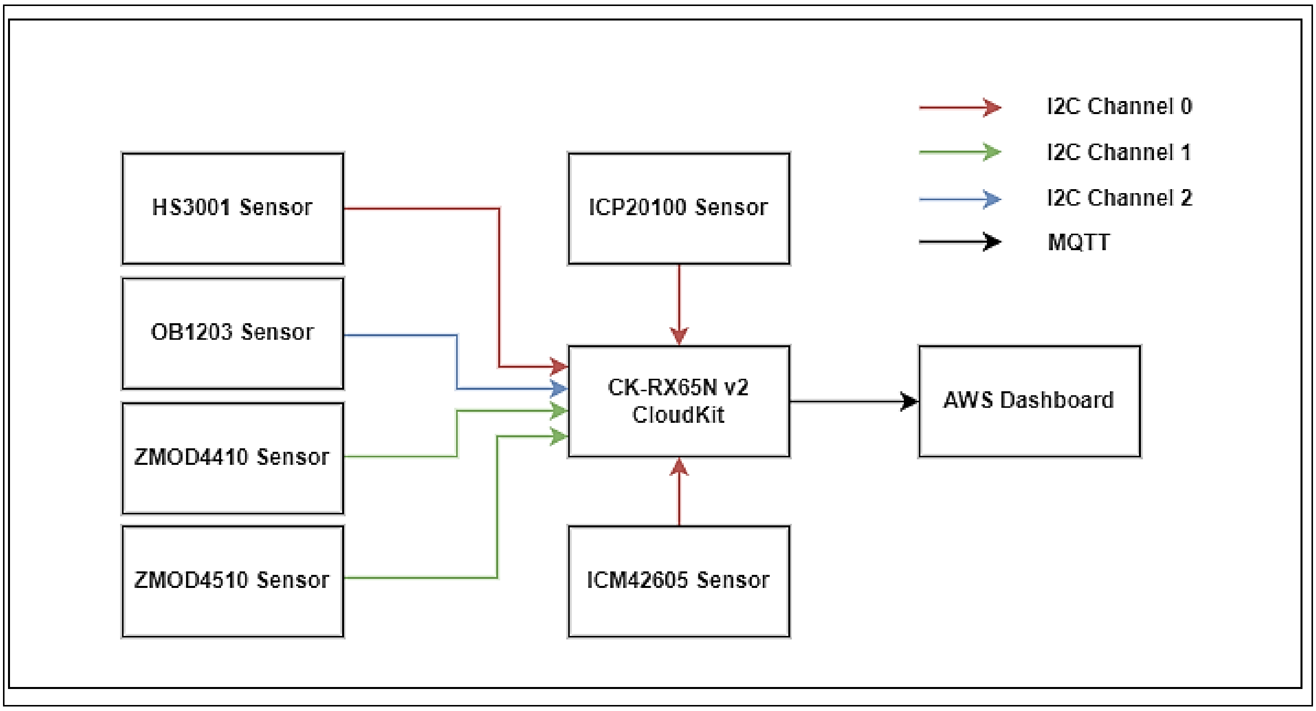


Figure 2. System Diagram

4. Cloud Connectivity Application Example

4.1 Overview

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

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

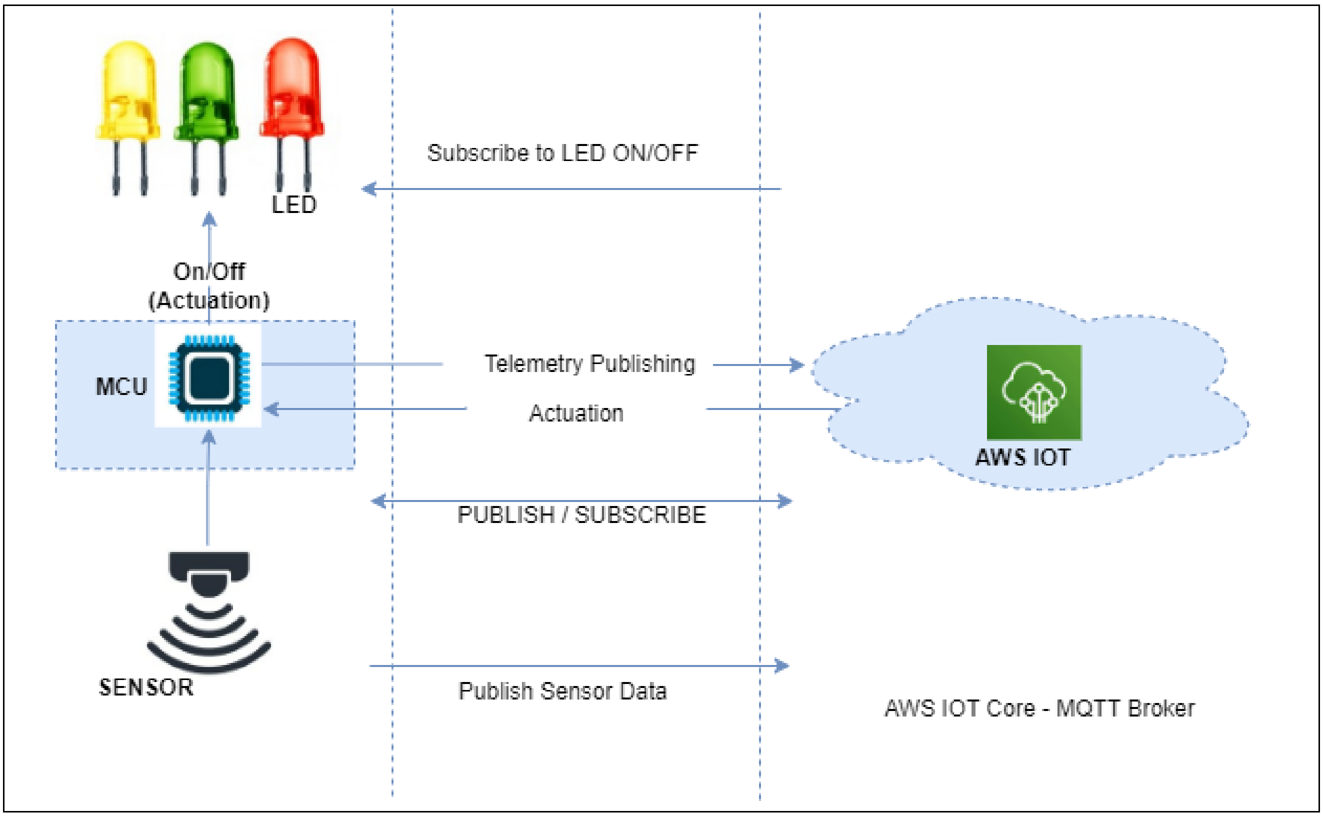
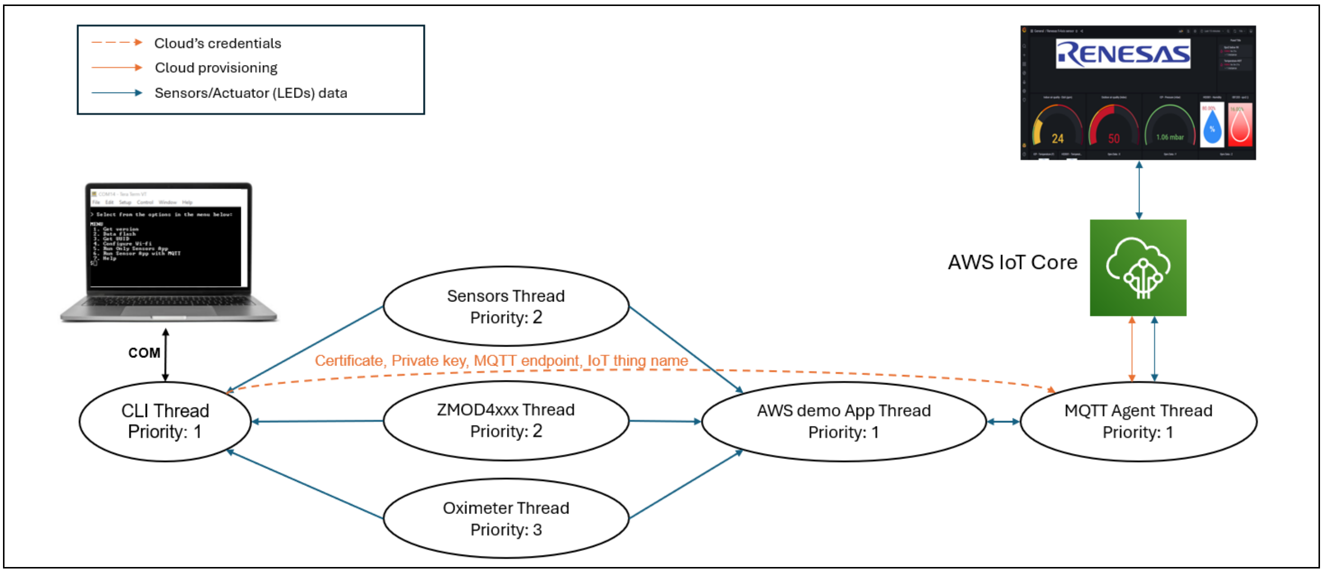


Figure 3. MQTT Publish/Subscribe to/from AWS IoT Core

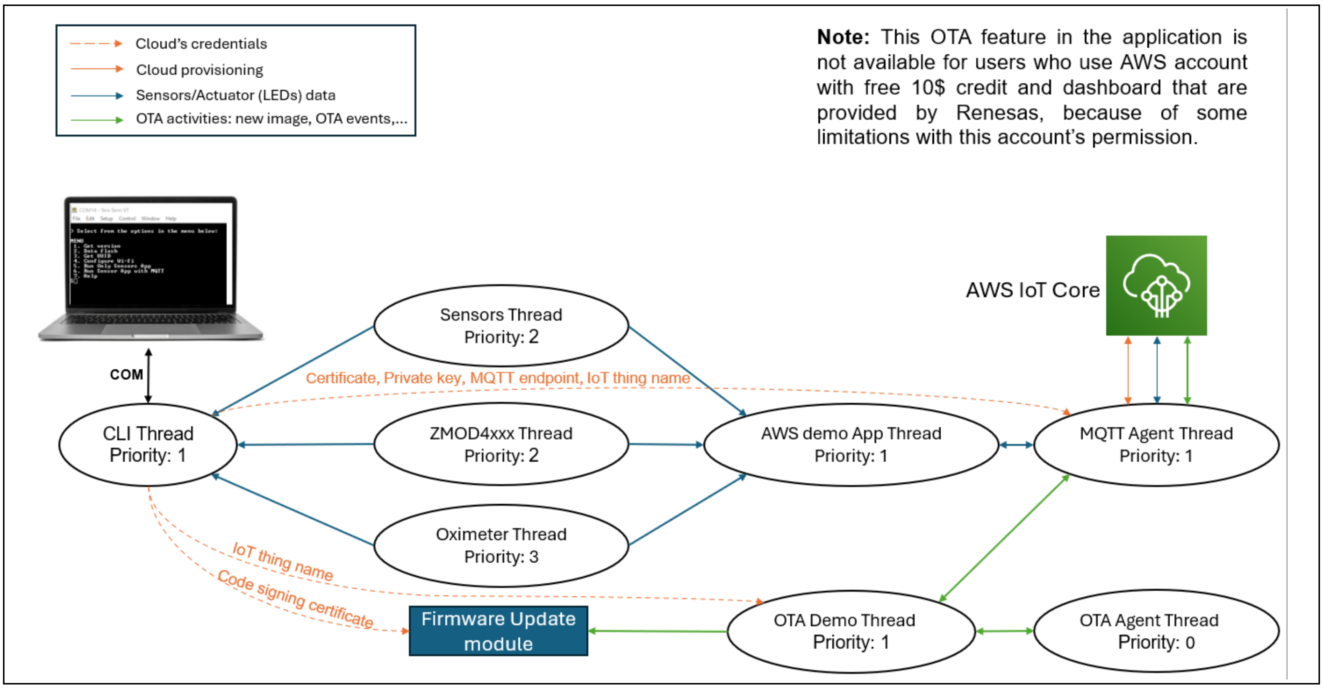


```
> Select from the options in the menu below:

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

Figure 4. Thread Diagram

The application also supports:

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

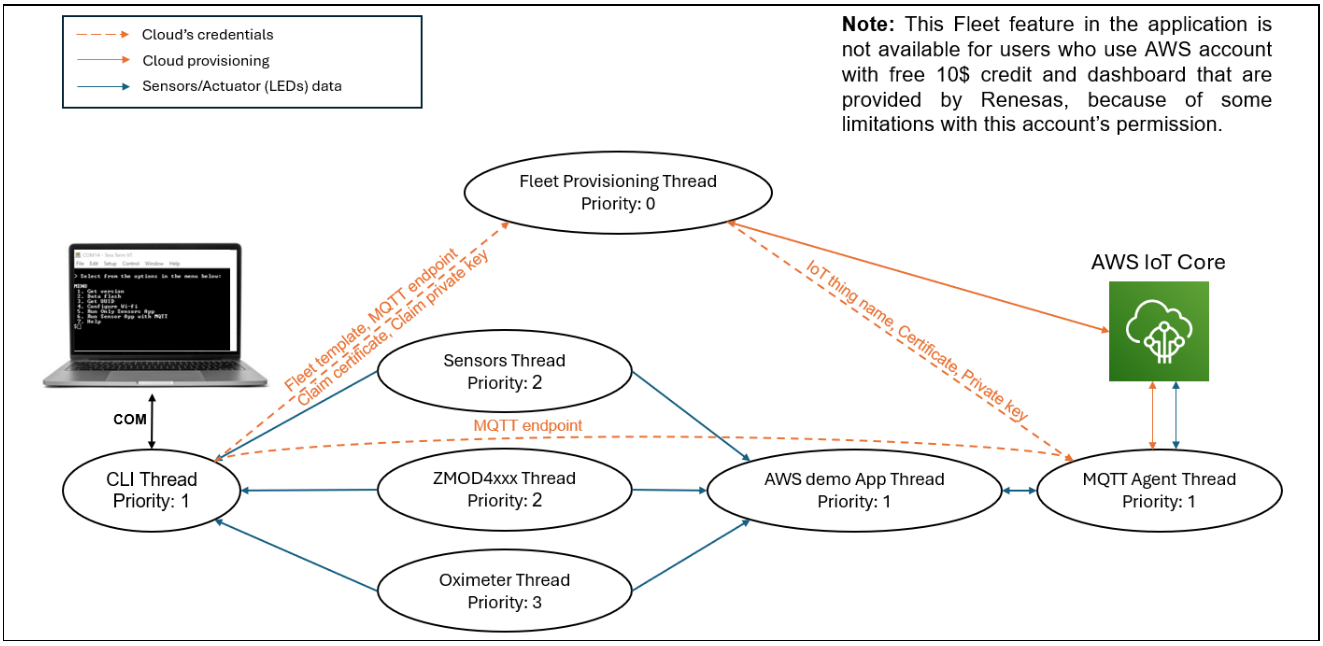


Note: This OTA feature in the application is not available for users who use AWS account with free 10$ credit and dashboard that are provided by Renesas, because of some limitations with this account's permission.

```
COM14 - Tera Term VT
File  Edit  Setup  Control  Window  Help
> Select from the options in the menu below:
MENU
 1. Get version
 2. Data Flash
 3. Get UUID
 4. Configure Wi-Fi
 5. Run Only Sensors App
 6. Run Sensor App with MQTT
 7. Help
```

Figure 5. Thread Diagram when enabling OTA feature

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



Note: This Fleet 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.

```
COM14 - Tera Term VT
File  Edit  Setup  Control  Window  Help
> Select from the options in the menu below:
MENU
 1. Get version
 2. Data Flash
 3. Get UUID
 4. Configure Wi-Fi
 5. Run Only Sensors App
 6. Run Sensor App with MQTT
 7. Help
```

Figure 6. Thread Diagram when enabling Fleet feature

4.2 MQTT/TLS Application Software Overview

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

Table 5. Application Project File

No.	Filename	Purpose
1.	`src/application_code/main.c`	Contains the initialization code of the Wi-Fi connection, provisioning cloud credentials used in Cloud Connectivity, the main function of the application.
2.	`src/application_code/usr_wifi.c`	Contains Wi-Fi initialization functions and data structures.
3.	`src/application_code/usr_wifi.h`	Contains macros, data structures, and functions prototypes used to initialize Wi-Fi across the project.
4.	`src/application_code/CommandLine/common_init.h`	Contains macros, data structures, and functions prototypes used to initialize common in the project.
5.	`src/application_code/CommandLine/console.c`	Contains data structures and functions used to print data on the console using UART
6.	`src/application_code/CommandLine/console.h`	Contains the function prototypes used to print data on the console using UART
7.	`src/application_code/CommandLine/menu_flash.c`	Contains data structures and functions used to provide CLI flash memory-related menu
8.	`src/application_code/CommandLine/menu_flash.h`	Contains the function prototypes and macros used to provide CLI flash memory-related menu
9.	`src/application_code/CommandLine/menu_kis.c`	Contains functions to get the application's version, get UUID and help option for the main menu on the CLI
10.	`src/application_code/CommandLine/menu_kis.h`	Contains the function prototypes and macros used to get the application's version, get UUID, and help option for the main menu on the CLI
11.	`src/application_code/CommandLine/menu_main.c`	Contains data structures and functions used to provide CLI main menu options
12.	`src/application_code/CommandLine/menu_main.h`	Contains the function prototypes and macros used to provide CLI main menu options
13.	`src/application_code/CommandLine/common_utils.h`	Contains macros, data structures, and functions prototypes commonly used across the project.
14.	`src/application_code/CommandLine/r_typedefs.h`	Contains typedef used in the application
15.	`src/application_code/sensor_thread_entry.c`	Contains the code for the sensor thread (HS3001, ICP20100, and ICM42605)
16.	`src/application_code/ICM42605/ICM_42605.c`	Contains the code for the 6-Axis MEMS Motion Tracking™ Sensor
17.	`src/application_code/ICM42605/ICM_42605.h`	Contains the Data structure function prototypes for the 6-Axis MEMS Motion Tracking™ Sensor
18.	`src/application_code/ICM42605/apex_feature.c`	Contains the code for the apex feature of the 6-Axis MEMS Motion Tracking™ Sensor
19.	`src/application_code/ICM42605/icm_i2c.c`	Contains the I2C code to communicate with 6-Axis MEMS Motion Tracking™ Sensor

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

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

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

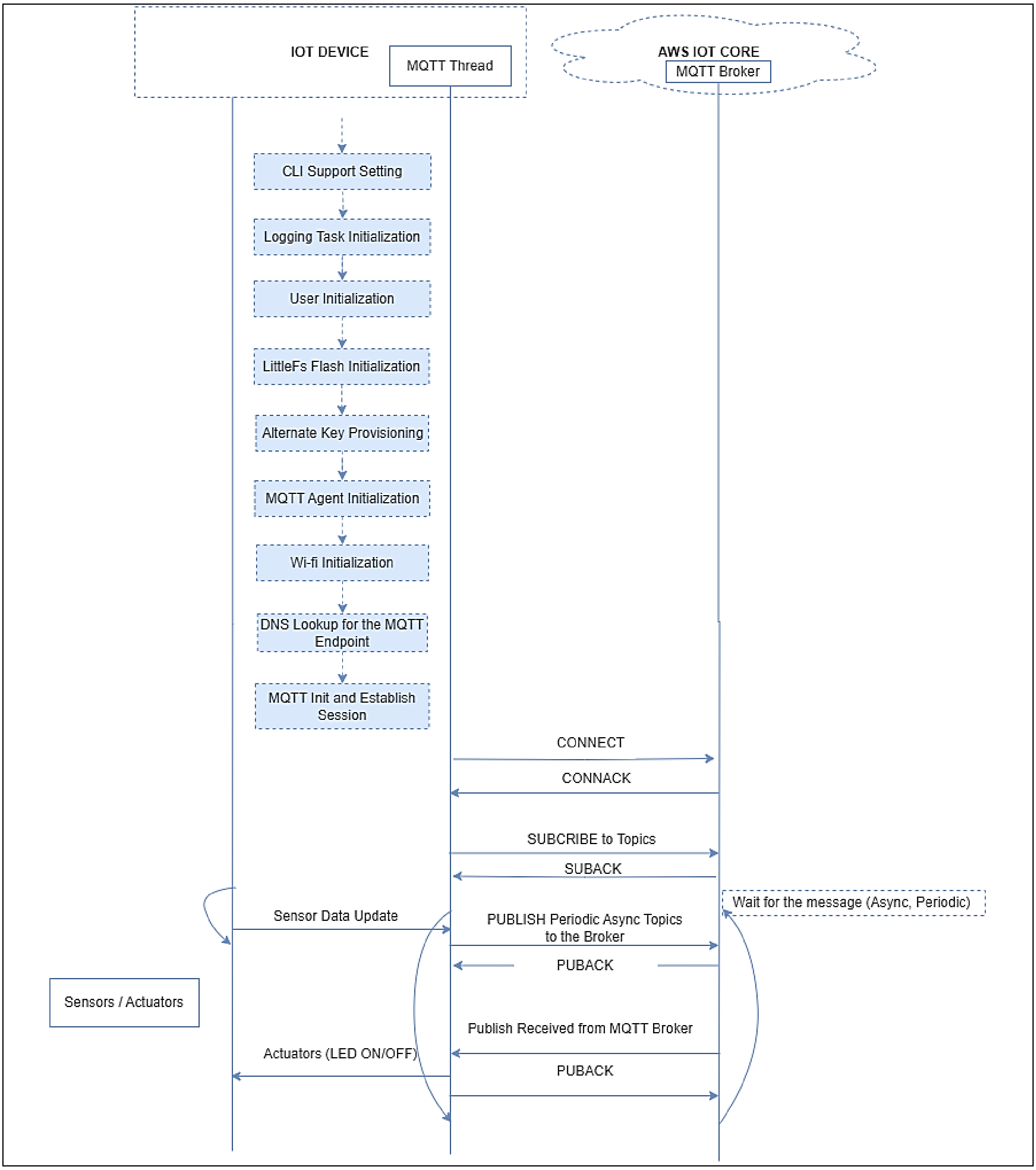

IOT DEVICE
MQTT Thread
AWS IOT CORE
MQTT Broker
CLI Support Setting
Logging Task Initialization
User Initialization
LittleFs Flash Initialization
Alternate Key Provisioning
MQTT Agent Initialization
Wi-fi Initialization
DNS Lookup for the MQTT Endpoint
MQTT Init and Establish Session
CONNECT
CONNACK
SUBCRIBE to Topics
SUBACK
Wait for the message (Async, Periodic)
Sensor Data Update
PUBLISH Periodic Async Topics to the Broker
PUBACK
Sensors / Actuators
Publish Received from MQTT Broker
Actuators (LED ON/OFF)
PUBACK

Figure 7. Application Example Implementation Details

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

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

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

- At most once (0)

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

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

5. Connection to AWS

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

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

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

5.1 Hardware Setup and Import of the Project

5.1.1 Hardware Preparation

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

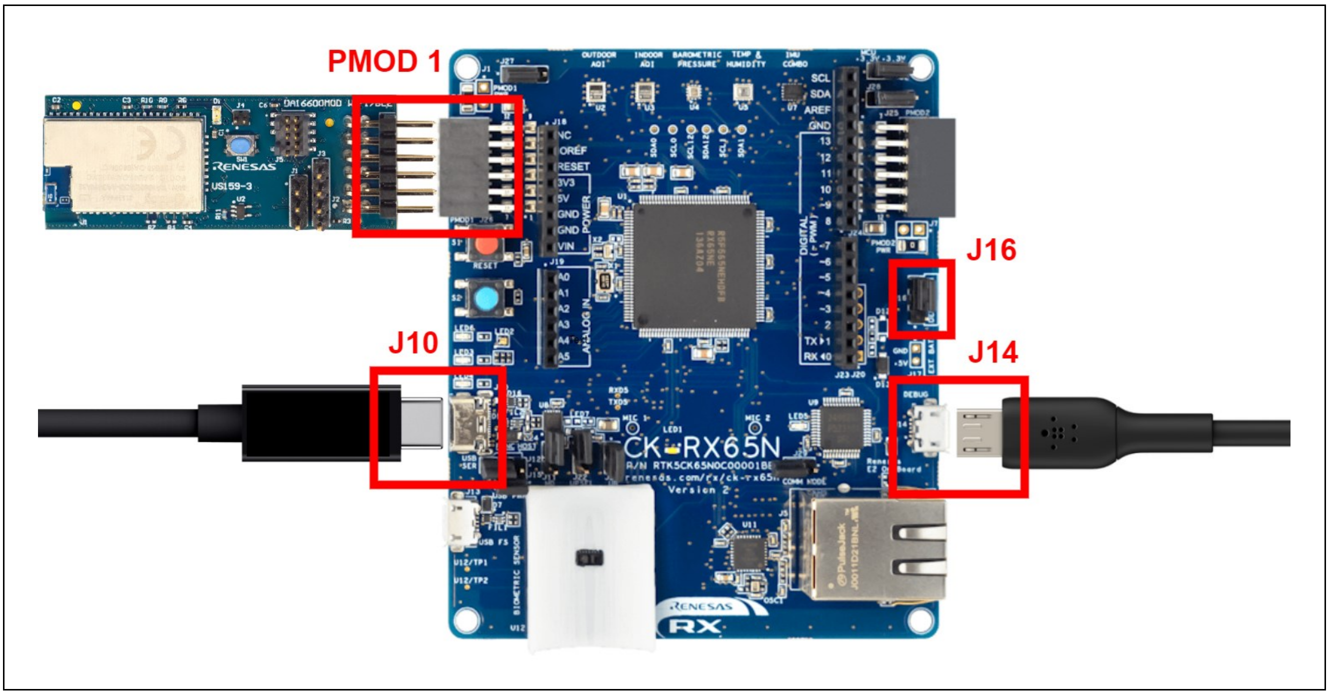
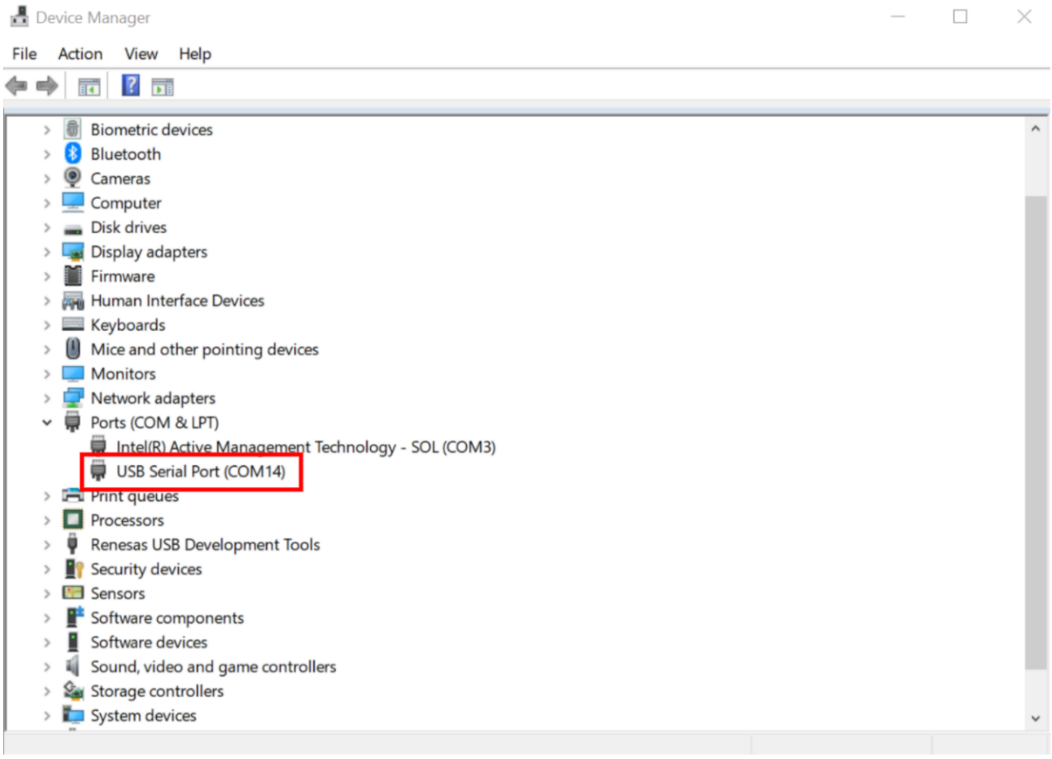


Figure 8. Connecting the USB and DA16600 Pmod

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

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

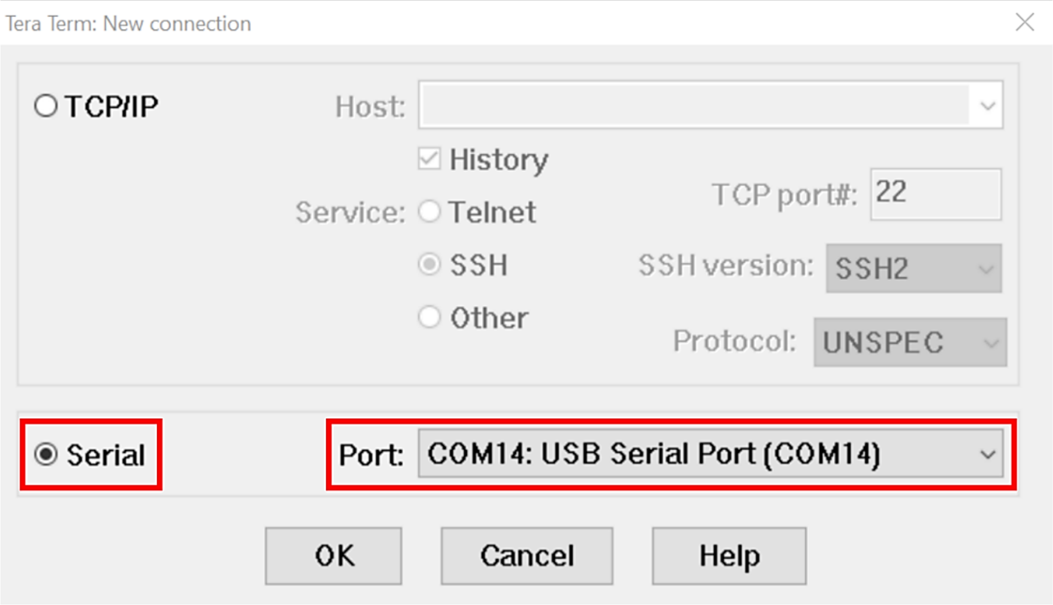
   Note: USB Serial Device drivers are required to communicate between the CK-RX65N v2 board and the PC.



- Biometric devices
- Bluetooth
- Cameras
- Computer
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Intel(R) Active Management Technology - SOL (COM3)
  - USB Serial Port (COM14)
- Print queues
- Processors
- Renesas USB Development Tools
- Security devices
- Sensors
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices

Figure 9. USB Serial Device in Windows Device Manager

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

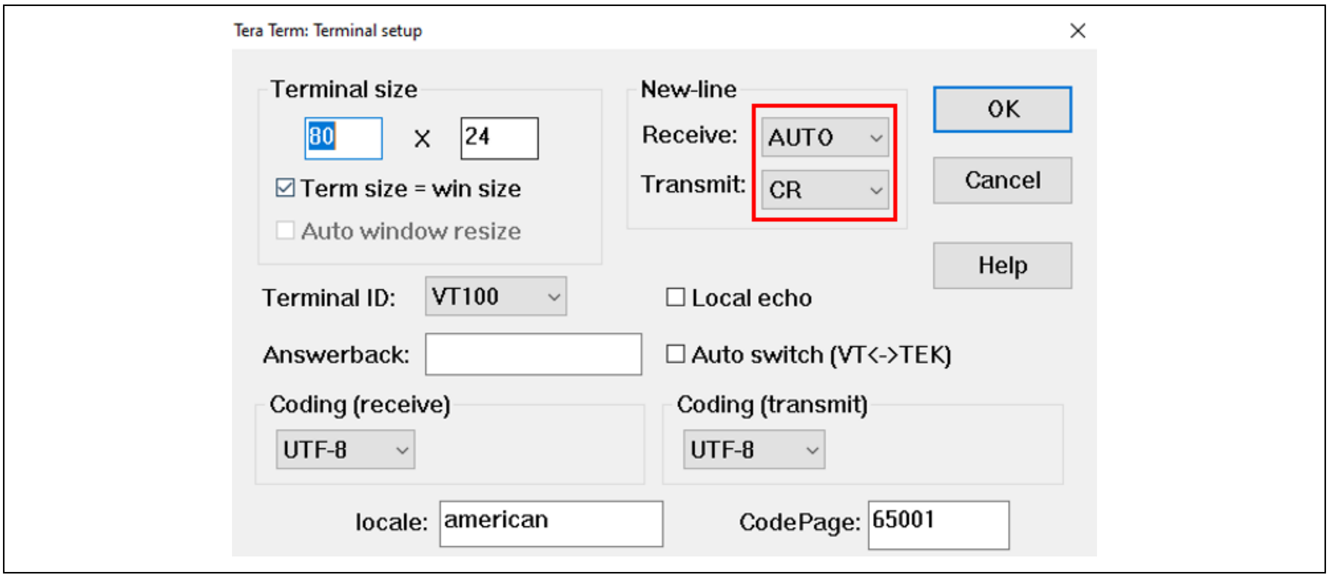


○ TCP/IP Host:
☑ History
Service: ○ Telnet TCP port#: 22
● SSH SSH version: SSH2
○ Other Protocol: UNSPEC

● Serial Port: COM14: USB Serial Port (COM14)

Figure 10. Selecting the Serial Port on Tera Term

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



Tera Term: Terminal setup

Terminal size

80 x 24

☑ Term size = win size

☐ Auto window resize

New-line

Receive: AUTO

Transmit: CR

Terminal ID: VT100

Answerback:

☐ Local echo

☐ Auto switch (VT<->TEK)

Coding (receive): UTF-8

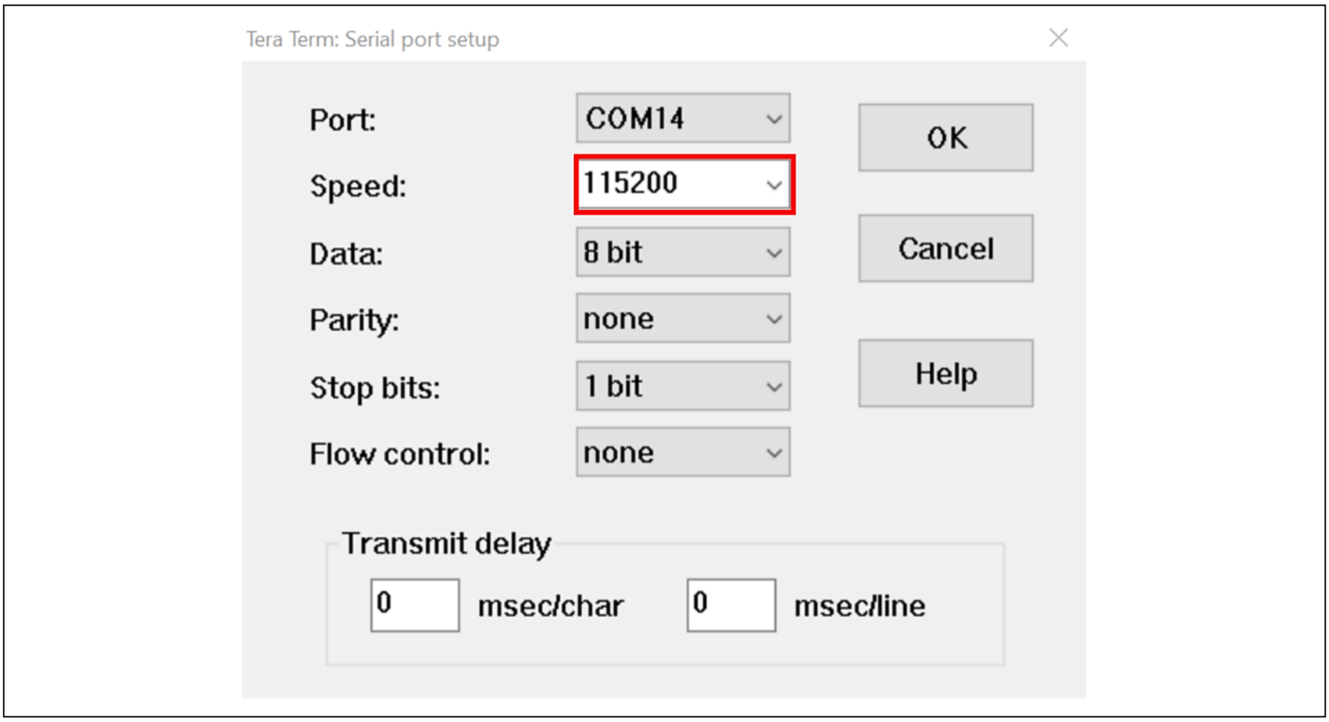
Coding (transmit): UTF-8

locale: american

CodePage: 65001

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

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



Tera Term: Serial port setup

Port: COM14

Speed: 115200

Data: 8 bit

Parity: none

Stop bits: 1 bit

Flow control: none

Transmit delay

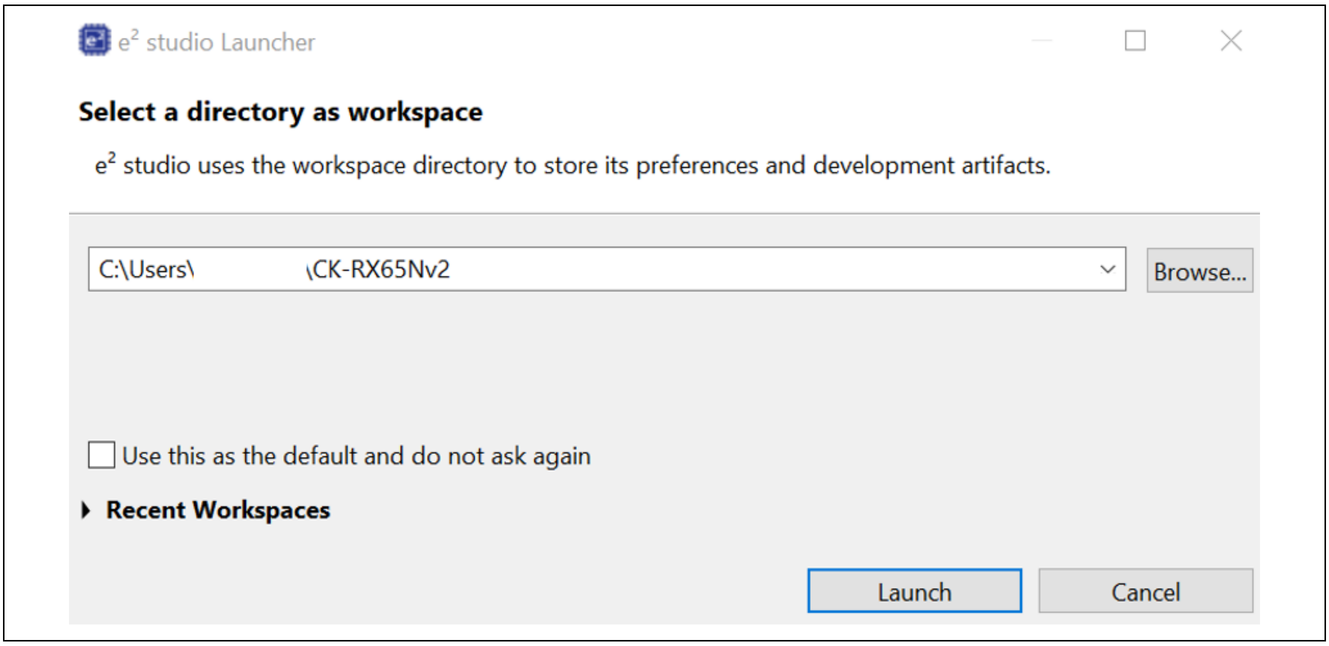
0 msec/char 0 msec/line

Figure 12. Select 115200 on the Speed Pulldown

5.1.3 Import the Project

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

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



e^2 studio Launcher

Select a directory as workspace

e^2 studio uses the workspace directory to store its preferences and development artifacts.

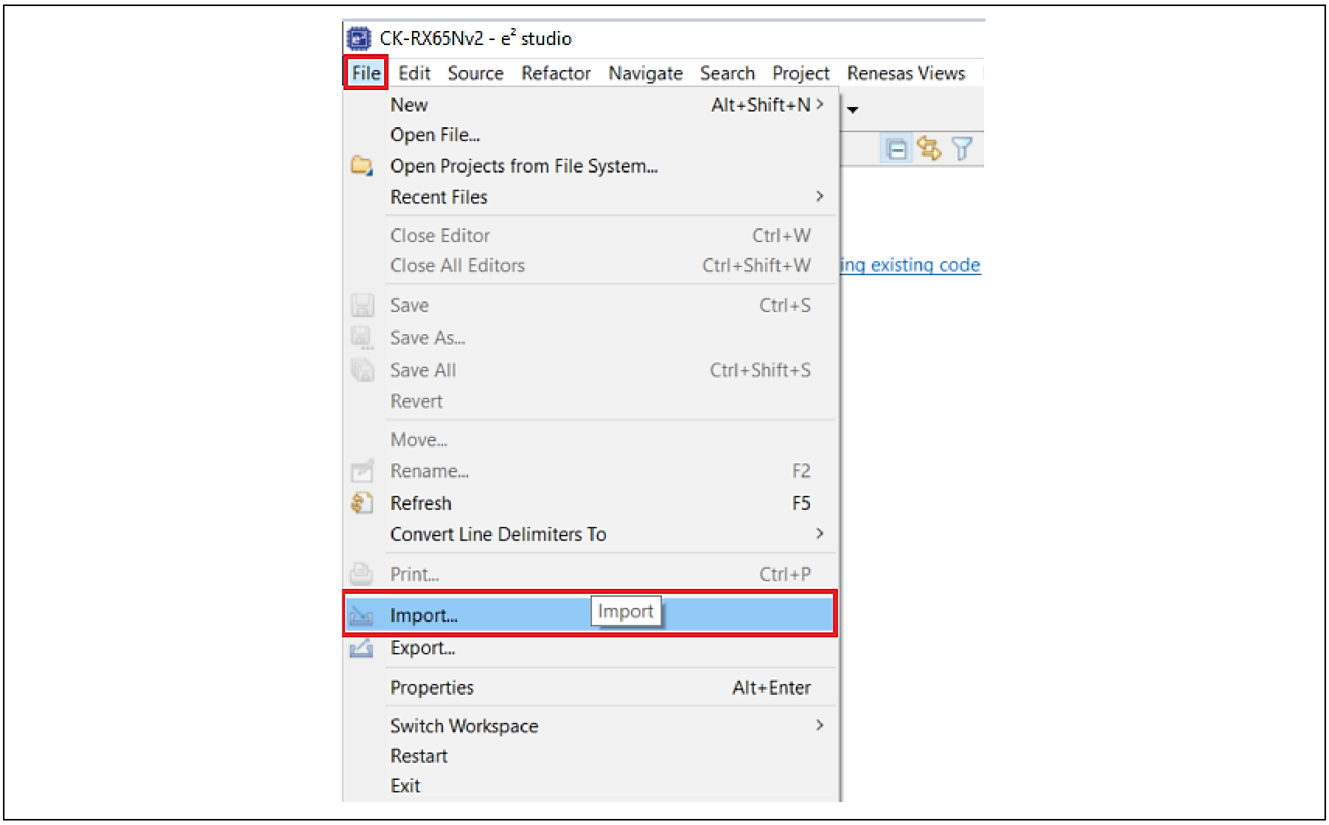
C:\Users\ \CK-RX65Nv2

□ Use this as the default and do not ask again

Recent Workspaces

Figure 13. Launch e^2 studio

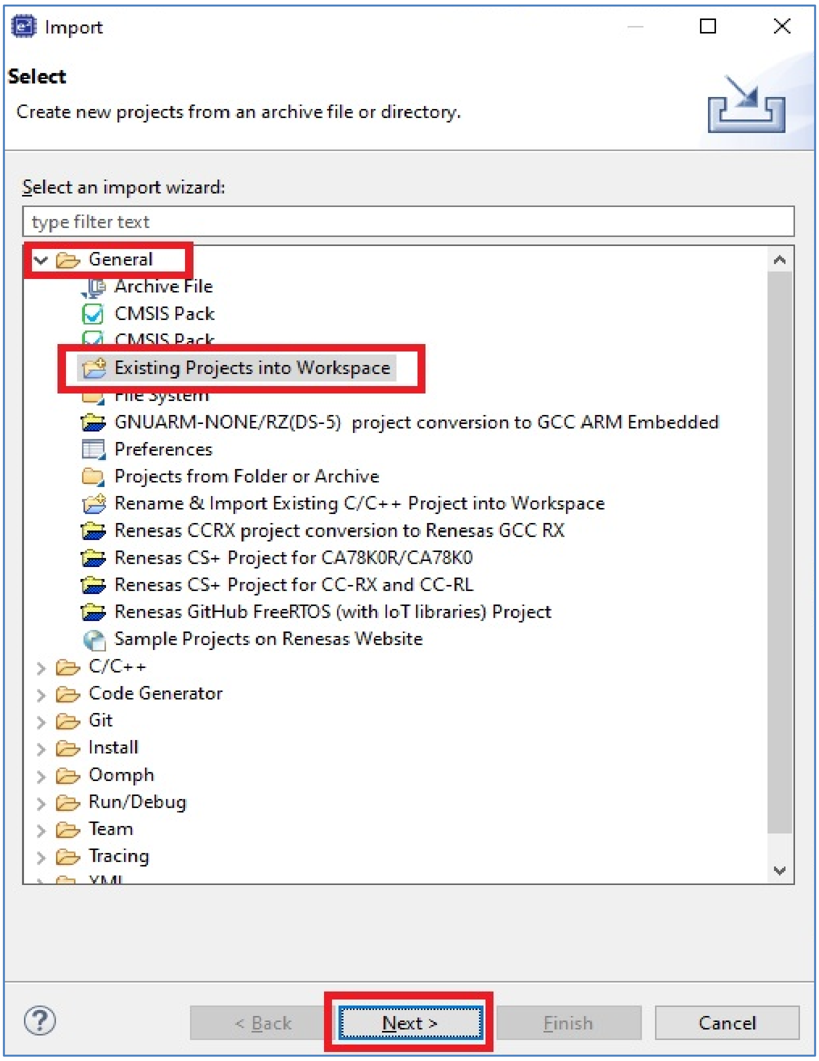
3. Select **File** > **Import…**



CK-RX65Nv2 - e^2 studio

Figure 14. Select Import

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



Import

Select

Create new projects from an archive file or directory.

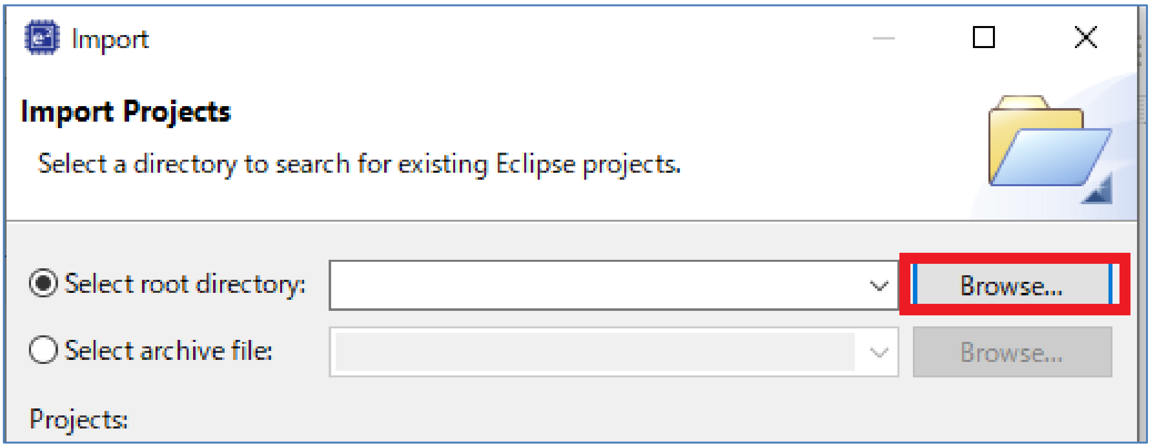
Select an import wizard:

type filter text

- General
  - Archive File
  - CMSIS Pack
  - CMSIS Pack
  - Existing Projects into Workspace
  - File System
  - GNUARM-NONE/RZ(DS-5) project conversion to GCC ARM Embedded
  - Preferences
  - Projects from Folder or Archive
  - Rename & Import Existing C/C++ Project into Workspace
  - Renesas CCRX project conversion to Renesas GCC RX
  - Renesas CS+ Project for CA78K0R/CA78K0
  - Renesas CS+ Project for CC-RX and CC-RL
  - Renesas GitHub FreeRTOS (with IoT libraries) Project
  - Sample Projects on Renesas Website
- C/C++
- Code Generator
- Git
- Install
- Oomph
- Run/Debug
- Team
- Tracing
- XML

Figure 15. Select Existing Projects into Workspace

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

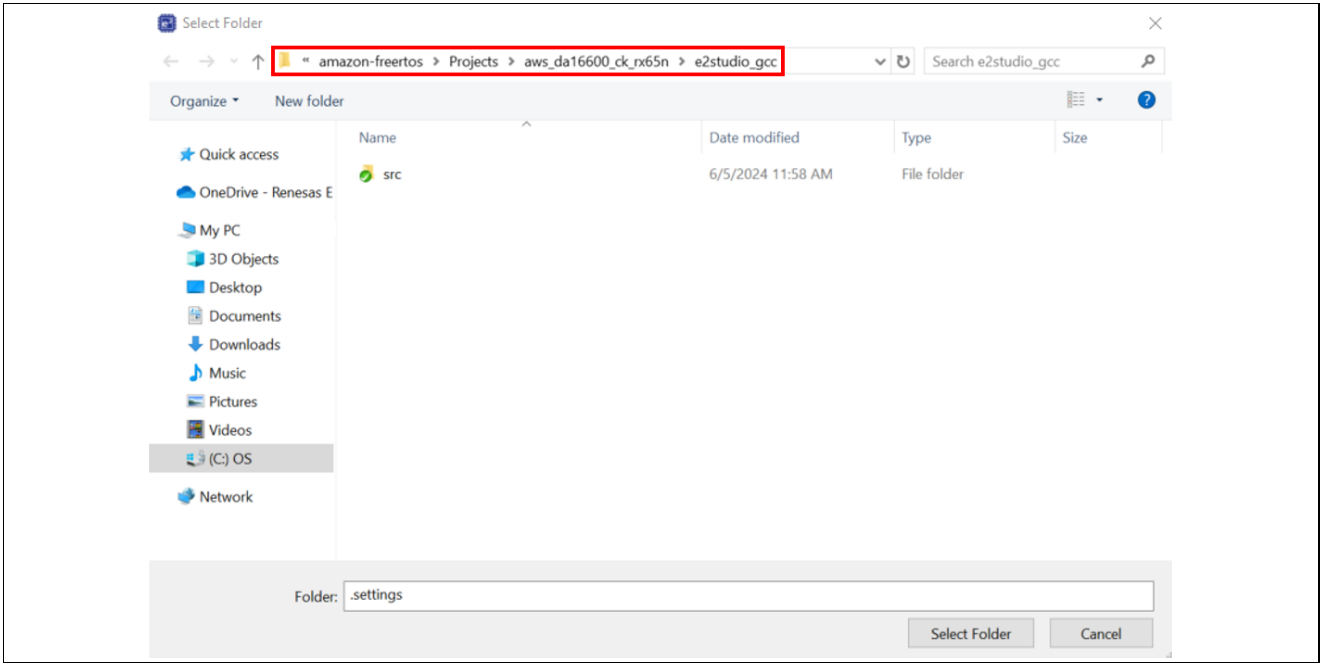
Figure 16. Find the Project

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

Project Details

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

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



Select Folder

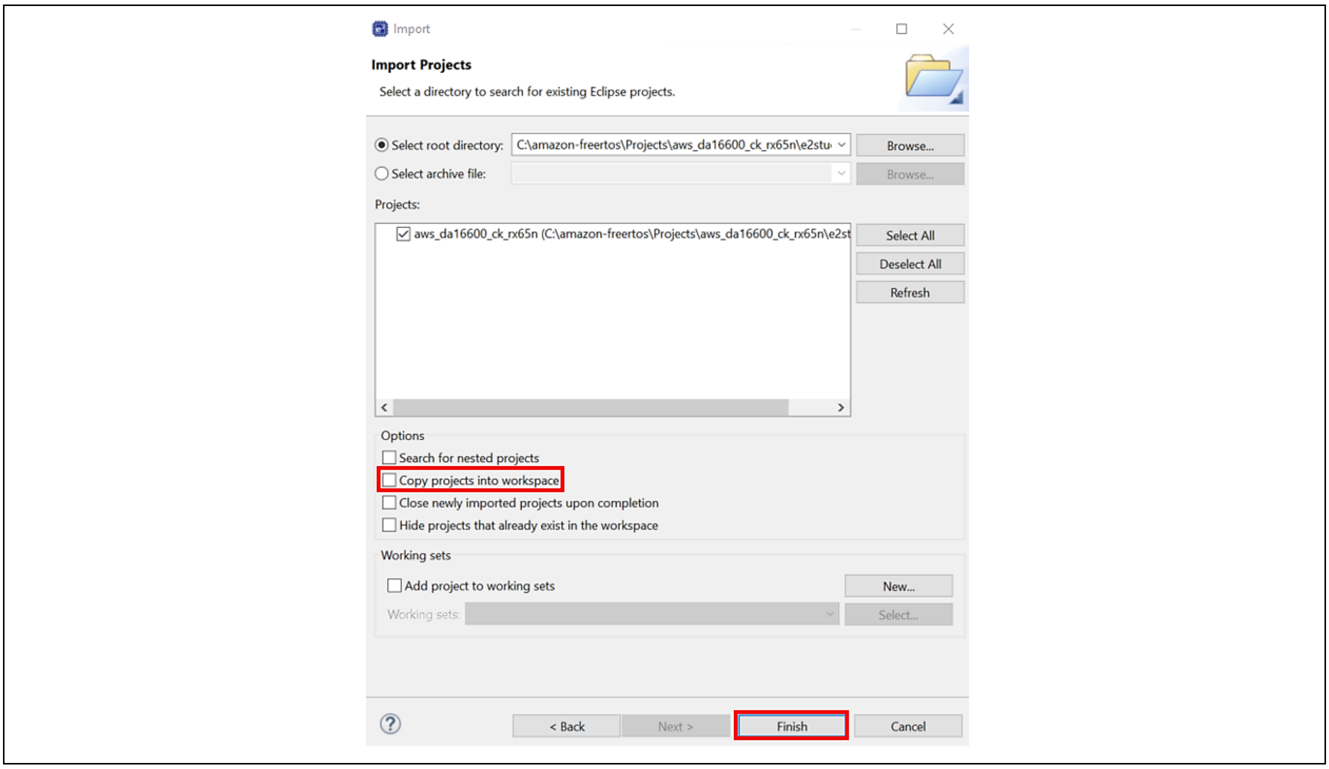
« amazon-freertos › Projects › aws_da16600_ck_rx65n › e2studio_gcc

Name	Date modified	Type	Size
src	6/5/2024 11:58 AM	File folder	

Folder: .settings

Figure 17. Select the Project Folder

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\amazon-freertos\Projects\aws_da16600_ck_rx65n\e2stu

○ Select archive file:

Projects:

☑ aws_da16600_ck_rx65n (C:\amazon-freertos\Projects\aws_da16600_ck_rx65n\e2st

Options

□ Search for nested projects

□ Copy projects into workspace

□ Close newly imported projects upon completion

□ Hide projects that already exist in the workspace

Working sets

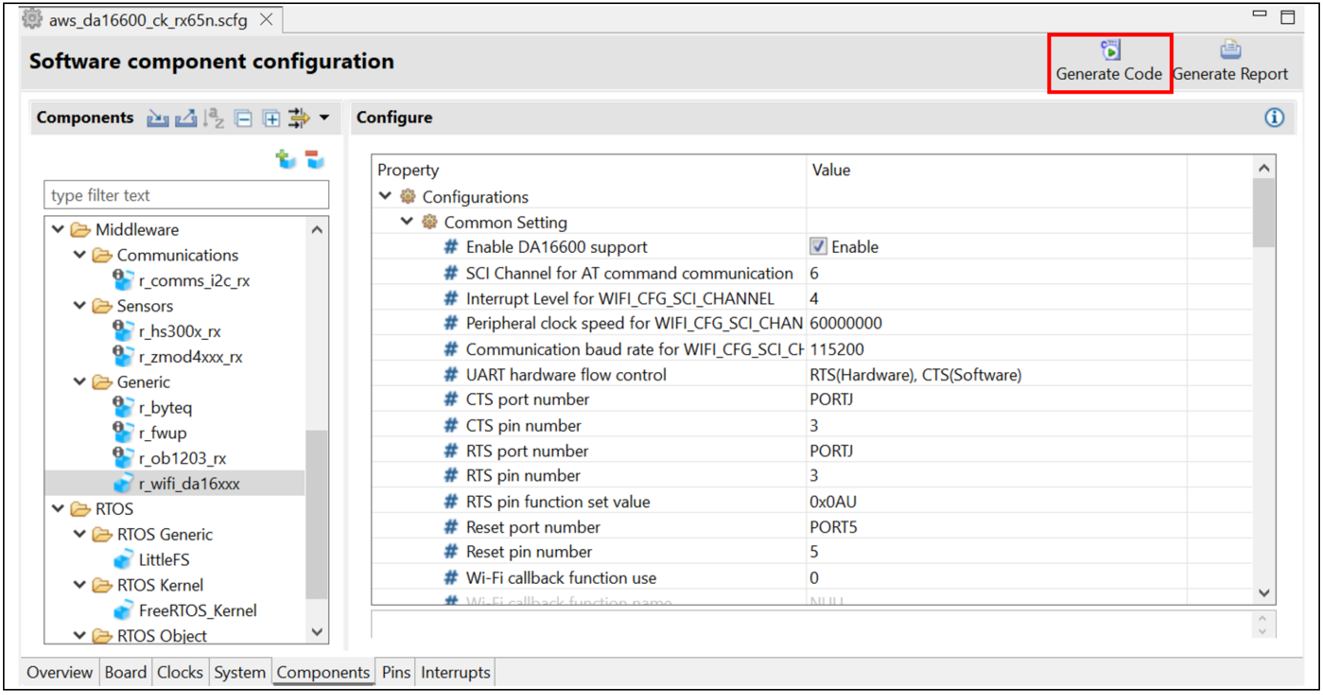
□ Add project to working sets

Working sets:

Figure 18. Import the Project

6. Execute code generation.

This application uses FIT; the user can configure them in the **“aws_da16600_ck_rx65n.scfg”** file**.**
If you have changed the Smart Configurator settings, click **Generate Code**.

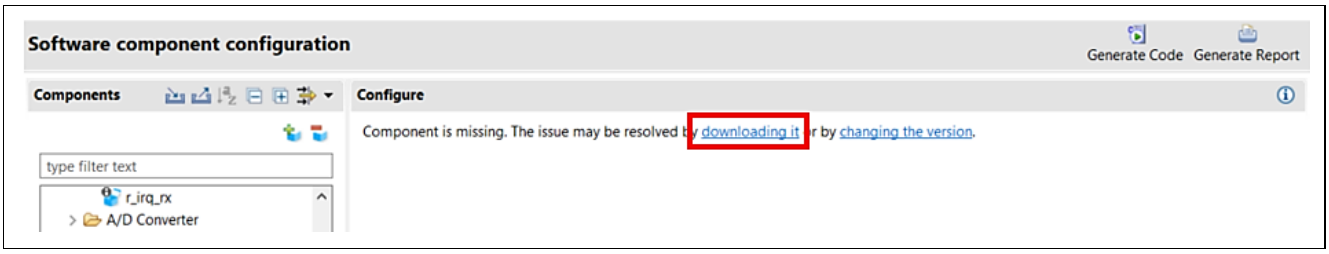


Property	Value
Configurations	
Common Setting	
Enable DA16600 support	☑ Enable
SCI Channel for AT command communication	6
Interrupt Level for WIFI_CFG_SCI_CHANNEL	4
Peripheral clock speed for WIFI_CFG_SCI_CHAN	60000000
Communication baud rate for WIFI_CFG_SCI_CH	115200
UART hardware flow control	RTS(Hardware), CTS(Software)
CTS port number	PORTJ
CTS pin number	3
RTS port number	PORTJ
RTS pin number	3
RTS pin function set value	0x0AU
Reset port number	PORT5
Reset pin number	5
Wi-Fi callback function use	0
Wi-Fi callback function name	NULL

Figure 19. Generate Code

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

1. Choosing **aws_da16600_ck_rx65n.scfg** > **Components** > **downloading it**



Component is missing. The issue may be resolved by downloading it or by changing the version.

Figure 20. Downloading missing components (1/4)

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

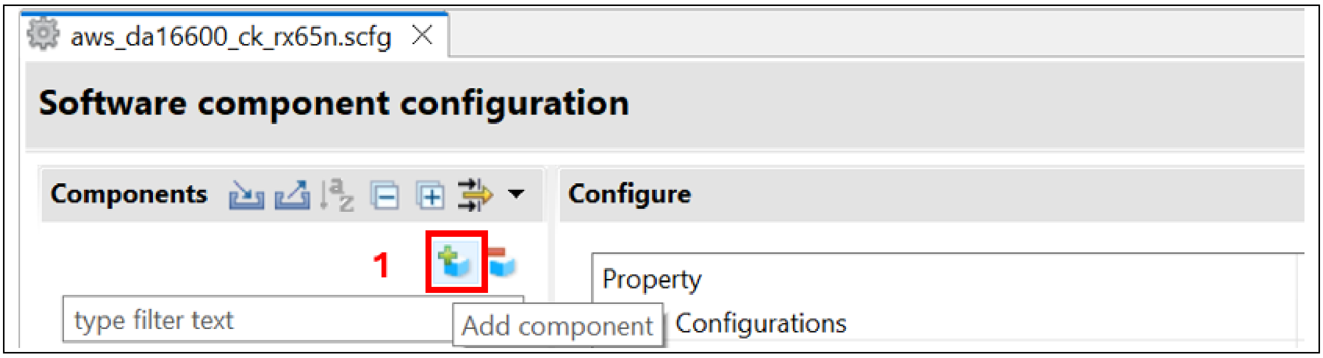
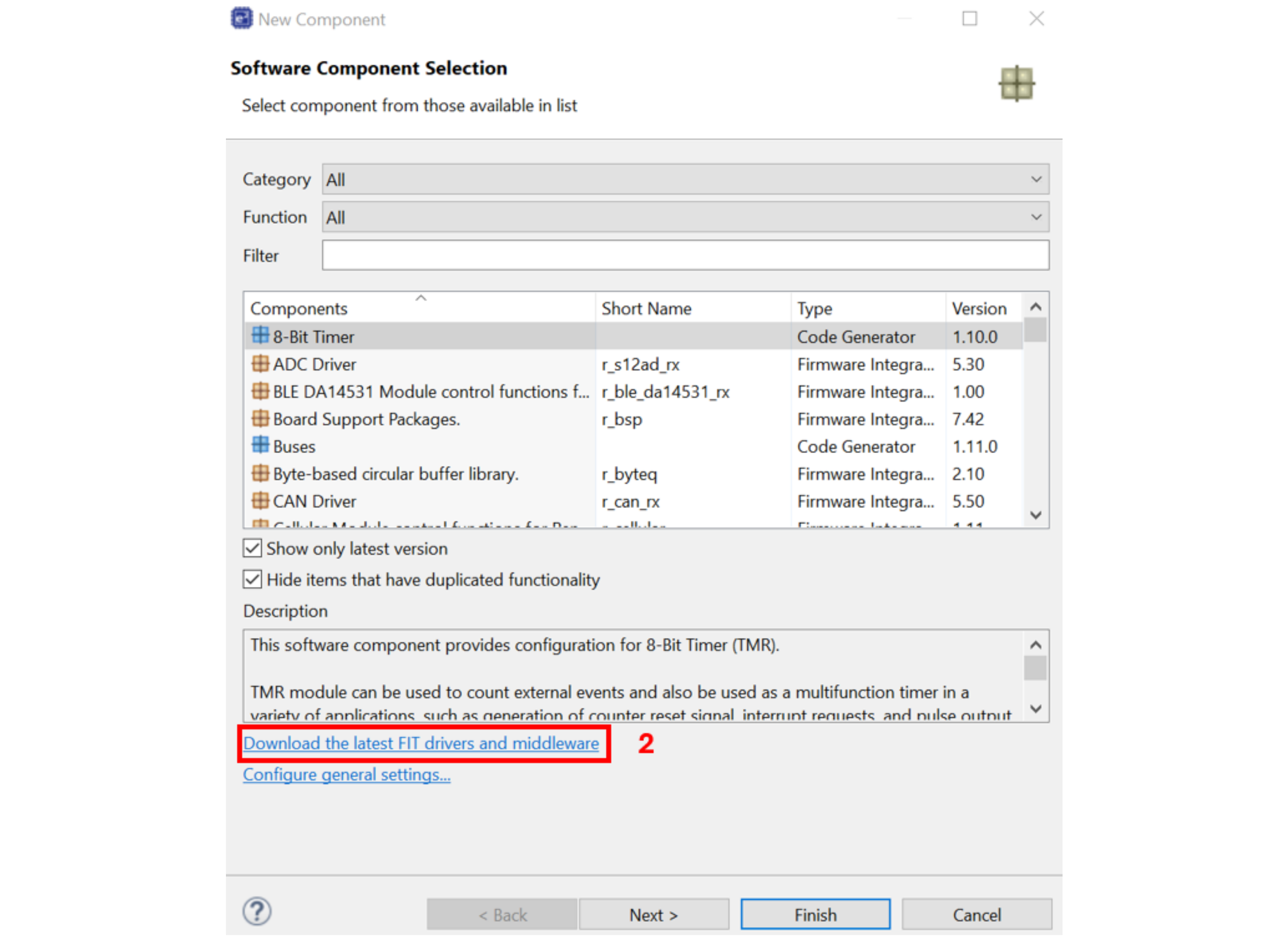


Figure 21. Download missing components (2/4)

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



New Component

Software Component Selection

Select component from those available in list

Category: All

Function: All

Filter:

Components	Short Name	Type	Version
8-Bit Timer		Code Generator	1.10.0
ADC Driver	r_s12ad_rx	Firmware Integra...	5.30
BLE DA14531 Module control functions f...	r_ble_da14531_rx	Firmware Integra...	1.00
Board Support Packages.	r_bsp	Firmware Integra...	7.42
Buses		Code Generator	1.11.0
Byte-based circular buffer library.	r_byteq	Firmware Integra...	2.10
CAN Driver	r_can_rx	Firmware Integra...	5.50

☑ Show only latest version

☑ Hide items that have duplicated functionality

Description

This software component provides configuration for 8-Bit Timer (TMR).

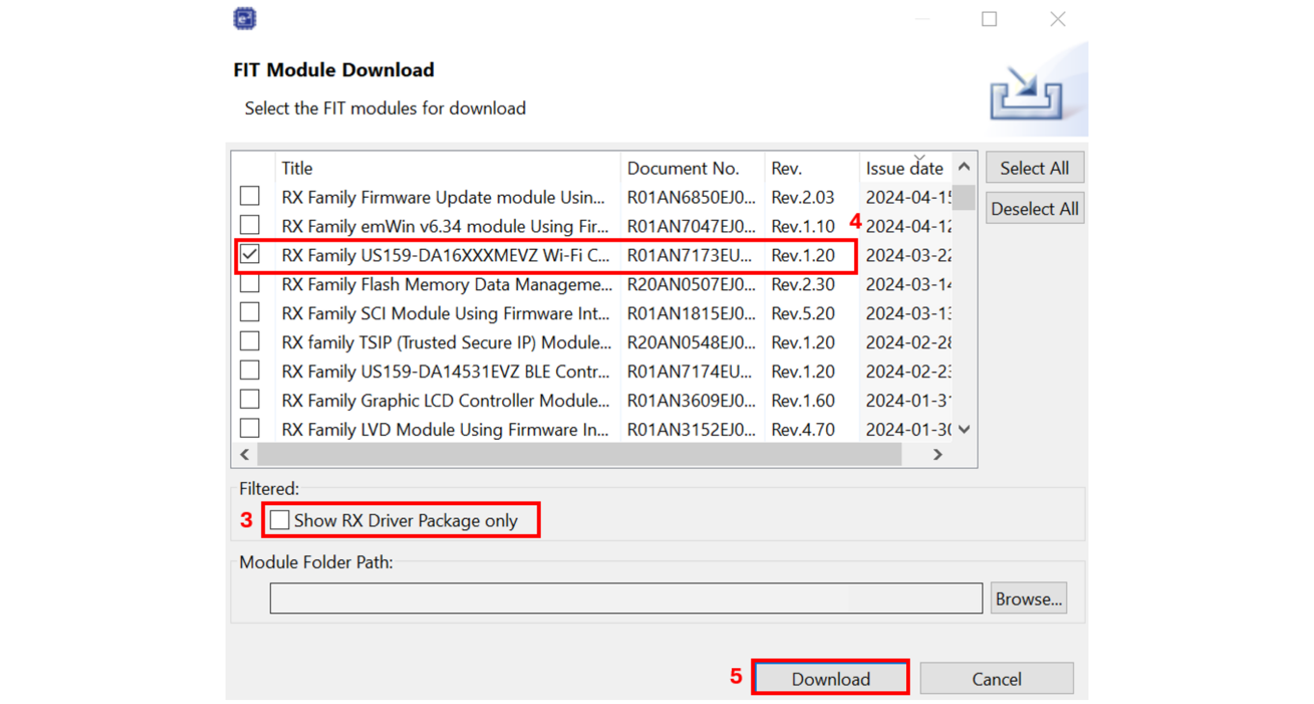
TMR module can be used to count external events and also be used as a multifunction timer in a variety of applications, such as generation of counter reset signal, interrupt requests, and pulse output

Download the latest FIT drivers and middleware 2

Configure general settings...

Figure 22. Download missing components (3/4)

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



FIT Module Download

Select the FIT modules for download

	Title	Document No.	Rev.	Issue date
□	RX Family Firmware Update module Usin...	R01AN6850EJ0...	Rev.2.03	2024-04-1!
□	RX Family emWin v6.34 module Using Fir...	R01AN7047EJ0...	Rev.1.10	2024-04-1:
☑	RX Family US159-DA16XXXMEVZ Wi-Fi C...	R01AN7173EU...	Rev.1.20	2024-03-2:
□	RX Family Flash Memory Data Manageme...	R20AN0507EJ0...	Rev.2.30	2024-03-1
□	RX Family SCI Module Using Firmware Int...	R01AN1815EJ0...	Rev.5.20	2024-03-1:
□	RX family TSIP (Trusted Secure IP) Module...	R20AN0548EJ0...	Rev.1.20	2024-02-2!
□	RX Family US159-DA14531EVZ BLE Contr...	R01AN7174EU...	Rev.1.20	2024-02-2:
□	RX Family Graphic LCD Controller Module...	R01AN3609EJ0...	Rev.1.60	2024-01-3
□	RX Family LVD Module Using Firmware In...	R01AN3152EJ0...	Rev.4.70	2024-01-3(

4

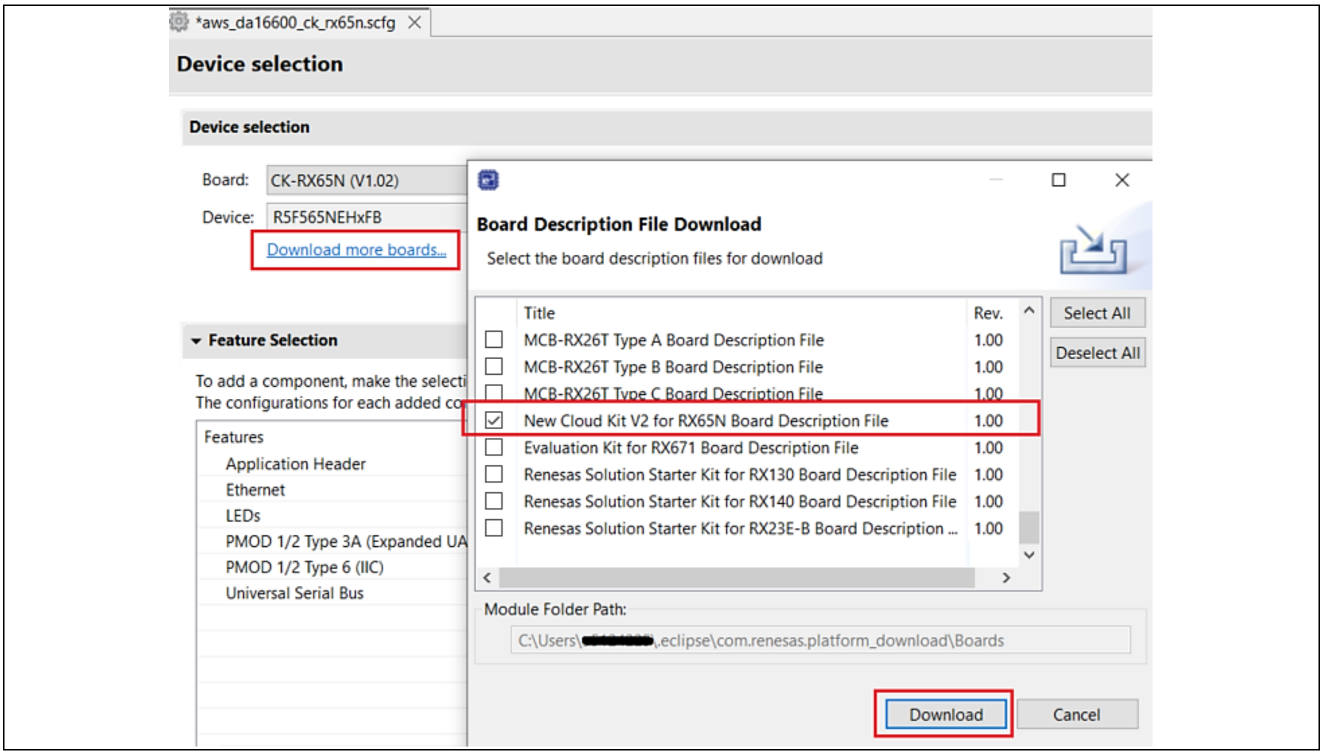
Filtered:

3 □ Show RX Driver Package only

Module Folder Path:

Figure 23. Download missing components (4/4)

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



*aws_da16600_ck_rx65n.scfg

Device selection

Device selection

Board: CK-RX65N (V1.02)

Device: R5F565NEHxFB

Download more boards...

Feature Selection

To add a component, make the selecti
The configurations for each added co

Features
Application Header
Ethernet
LEDs
PMOD 1/2 Type 3A (Expanded UA
PMOD 1/2 Type 6 (IIC)
Universal Serial Bus

Board Description File Download

Select the board description files for download

	Title	Rev.
□	MCB-RX26T Type A Board Description File	1.00
□	MCB-RX26T Type B Board Description File	1.00
□	MCB-RX26T Type C Board Description File	1.00
☑	New Cloud Kit V2 for RX65N Board Description File	1.00
□	Evaluation Kit for RX671 Board Description File	1.00
□	Renesas Solution Starter Kit for RX130 Board Description File	1.00
□	Renesas Solution Starter Kit for RX140 Board Description File	1.00
□	Renesas Solution Starter Kit for RX23E-B Board Description ...	1.00

Module Folder Path: C:\Users\[redacted].eclipse\com.renesas.platform_download\Boards

Figure 24. Download CK-RX65N v2 Board Description File on e^2 studio

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

Table 6. Components Configuration

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

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

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

No	Component	Configuration
		I2C Shared Bus No. for I2C Communication Device6: **I2C Shared Bus0** Slave address for I2C Communication Device6: **0x68** Callback function for I2C Communication Device6: **comms_i2c_callback_icm**
11	**Middleware→Sensors→r_hs300x_rx (v1.23)**	Number of HS300x Sensors: **1**
		Data types from HS300x Sensor: **Humidity and Temperature**
		Programming mode for HS300x sensor: **ON**
		I2C Communication device No. for HS300x sensor device0: **I2C Communication Device0**
		Callback function for HS300x sensor device0: **hs300x_callback**
12	**Middleware→Sensors→r_zmod4xxx_rx (v1.31)**	Number of ZMOD4xxx Sensors: **2**
		Operation mode of ZMOD4XXX Sensor0: **IAQ 1st Gen. (Continuous)**
		I2C Communication device No. for ZMOD4XXX sensor device0: **I2C Communication Device1**
		I2C callback function for ZMOD4XXX sensor device0: **zmod4xxx_user_i2c_callback0**
		Enable IRQ from ZMOD4XXX sensor device0: **Enabled**
		IRQ Callback function for ZMOD4XXX sensor device0: **zmod4xxx_user_irq_callback0**
		IRQ number for ZMOD4XXX sensor device0: **IRQ14**
		IRQ trigger for ZMOD4XXX sensor device0: **Falling**
		IRQ interrupt priority for ZMOD4XXX sensor device0: **Priority 10**
		Operation mode of ZMOD4XXX Sensor1: **OAQ 1st Gen.**
		I2C Communication device No. for ZMOD4XXX sensor device1: **I2C Communication Device2**
		I2C callback function for ZMOD4XXX sensor device1: **zmod4xxx_user_i2c_callback1**
		Enable IRQ from ZMOD4XXX sensor device 1: **Enabled**
		IRQ Callback function for ZMOD4XXX sensor device1: **zmod4xxx_user_irq_callback1**
		IRQ number for ZMOD4XXX sensor device1: **IRQ13**
		IRQ trigger for ZMOD4XXX sensor device1: **Falling**
		IRQ interrupt priority for ZMOD4XXX sensor device1: **Priority 5**
13	**Middleware→Generic→r_byteq (v2.10)**	Memory allocation for queue control blocks: **Static memory allocation**
		Number of static queue control block: **32**
14	**Middleware→Generic→r_fwup (v2.03)**	Select the update mode: **Dual bank**
		Select the function mode: **user for User program**
		Main area start address: **0xFFF00000**
		Buffer area start address: **0xFFE00000**
		Install area size: **0xF0000**
		Select the algorithm of signature verification: **ECDSA + SHA256**
		Enable user disable interrupt function: **Use FWUP r_fwup_wrap_disable_interrupt() function**

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

No	Component	Configuration
		IRQ number for OB1203 sensor device0: **IRQ15**
		IRQ trigger for OB1203 sensor device0: **Falling**
		IRQ interrupt priority for OB1203 sensor device0: **Priority 14**
		Sensor mode of OB1203 Sensor device1: **PPG sensor mode**
		I2C Communication device No. for OB1203 sensor device1: **I2C Communication Device 4**
		I2C callback function for OB1203 sensor device1: **ob1203_comms_i2c_callback**
		Enable IRQ from OB1203 sensor device1: **Enabled**
		IRQ callback function for OB1203 sensor device1: **ob1203_irq_callback**
		IRQ number for OB1203 sensor device1: **IRQ15**
		IRQ trigger for OB1203 sensor device1: **Falling**
		IRQ interrupt priority for OB1203 sensor device1: **Priority 14**
16	**Middleware→Generic→r_wifi_da16xxx (v1.20)**	Enable DA16600 support: ✓ **Enable**
		SCI Channel for AT command communication: **6**
		Interrupt Level for WIFI_CFG_SCI_CHANNEL: **4**
		Communication baud rate for WIFI_CFG_CHANNEL: **115200**
		UART hardware flow control: **RTS(Hardware), CTS(Software)**
		CTS port number: **PORTJ**
		CTS pin number: **3**
		RTS port number: **PORTJ**
		RTS pin number: **3**
		RTS pin function set value: **0x0AU**
		Reset port number: **PORT 5**
		Reset pin number: **5**
		AT command transfer buffer size: **1024**
		AT command receive buffer size: **2048**
		Use SNTP client service: ✓ **Enable**
		Timezone offset in hours (-12 ~ 12): **7** **Note:** Depending on the user's time zone. Please refer to the Settings of Country code and GMT timezone (Only using Wi-Fi)
		Country code: **"VN"** **Note:** Depending on the user's country. Please refer to the Settings of Country code and GMT timezone (Only using Wi-Fi)
		Use FreeRTOS logging functionality: ✓ **Enable**
		Select Wi-Fi protocol → TCP protocol support: ✓ **Enable**
		Creatable sockets number: **4**
		Socket Receive buffer size: **8192**
17	**RTOS→RTOS Generic → LittleFS (V202210.01)**	.block_count: **70**
		Starting block of Data Flash allocated to littleFS: **FLASH_DF_BLOCK_0_MACRO**
18	**RTOS→RTOS Kernel→FreeRTOS_Kernel (V202210.01)**	RTOS scheduler: **Preemptive**
		Maximum number of priorities to the application task: **7**
		The frequency of the RTOS tick interrupt: **(TickType_t) 1000**
		The size of the stack used by the idle task: **768**
		The total amount of RAM available in the FreeRTOS heap: **256**

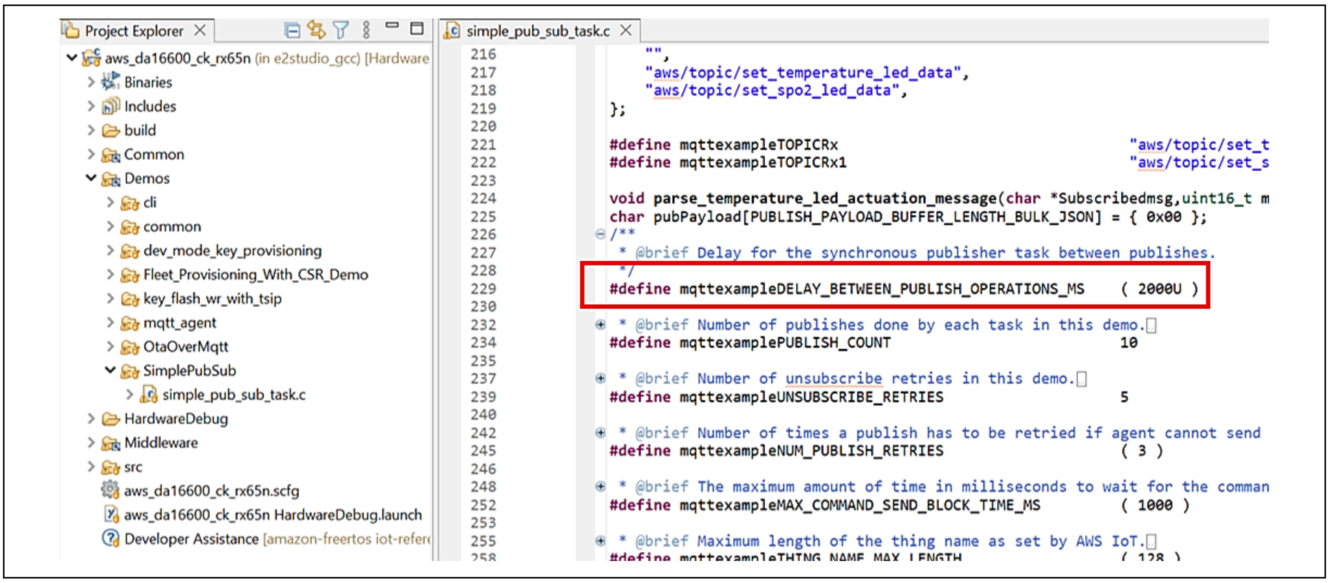
No	Component	Configuration
		The maximum permissible length of name: **12**
		Use trace facility: ✓ **Enable**
		Idle should yield: ✓ **Enable**
		Mutex functionality: ✓ **Enable**
		Recursive mutex functionality: ✓ **Enable**
		Counting semaphore functionality: ✓ **Enable**
		Thread local storage pointers: **3**
		Record stack high address: ✓ **Enable**
		Daemon task startup hook: ✓ **Enable**
		Queue set functionality: ✓ **Enable**
		Tick hook: ✓ **Enable**
		Idle hook: ✓ **Enable**
		The malloc() failed function: ✓ **Enable**
		Check for stack overflow: **Check by tick value and stack pointer**
		Software timer functionality: ✓ **Enable**
		The priority of the software timer task: **6**
		The length of the software timer command queue: **5**
		Kernel interrupt priority: **1**
		Maximum syscall interrupt priority: **4**
		Tick vector: **_CMT0_CMI0**
		Event groups: ✓ **Enable**
		Maximum number of priorities to the application co-routines: **2**
		Dynamic allocation: ✓ **Enable**
		Static allocation: ✓ **Enable**
		vTaskPrioritySet: ✓ **Enable**
		vTaskPriorityGet: ✓ **Enable**
		vTaskDelete: ✓ **Enable**
		vTaskSuspend: ✓ **Enable**
		vTaskDelayUntil: ✓ **Enable**
		vTaskDelay: ✓ **Enable**
		xTaskGetSchedulerState: ✓ **Enable**
		xEventGroupSetBitsFromISR: ✓ **Enable**
		xTimerPendFunctionCall: ✓ **Enable**
		xTaskGetCurrentTaskHandle: ✓ **Enable**
		xTaskAbortDelay: ✓ **Enable**
		Max message length: **192**
		Include time and task name: ✓ **Enable**
		Include demo debug stats: ✓ **Enable**
		Max output size: **850**
		Platform name: **“RenesasRX65N”**
		Include platform.h instead of iodefine.h: ✓ **Enable**
		Enable time-slicing for equal priority tasks: ✓ **Enable**

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

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

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

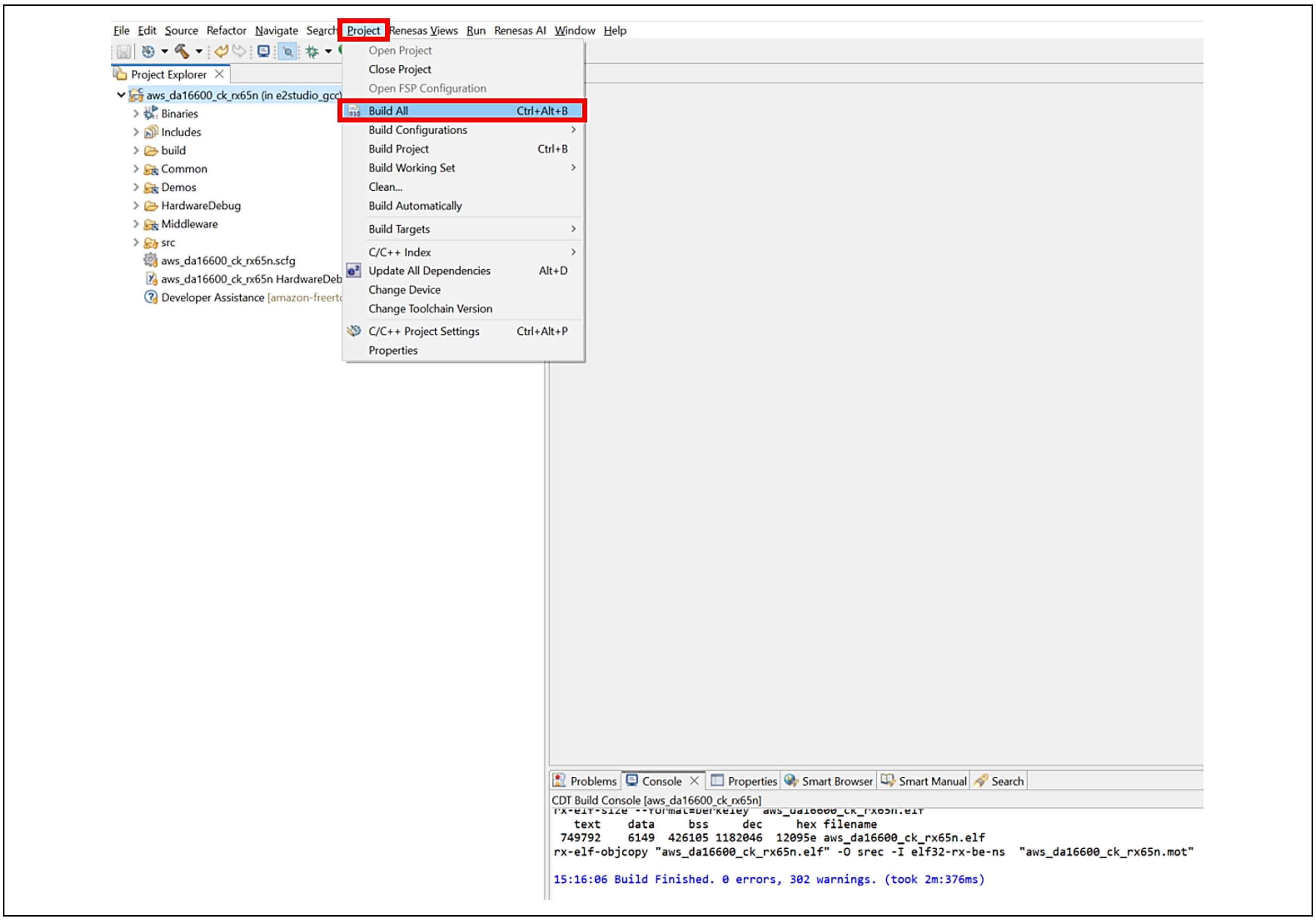
```
#define mqttexampleDELAY_BETWEEN_PUBLISH_OPERATIONS_MS (2000U)
```


Project Explorer
aws_da16600_ck_rx65n (in e2studio_gcc) [Hardware
Binaries
Includes
build
Common
Demos
cli
common
dev_mode_key_provisioning
Fleet_Provisioning_With_CSR_Demo
key_flash_wr_with_tsip
mqtt_agent
OtaOverMqtt
SimplePubSub
simple_pub_sub_task.c
HardwareDebug
Middleware
src
aws_da16600_ck_rx65n.scfg
aws_da16600_ck_rx65n HardwareDebug.launch
Developer Assistance [amazon-freertos iot-refere
simple_pub_sub_task.c

```
        "",
        "aws/topic/set_temperature_led_data",
        "aws/topic/set_spo2_led_data",
};

#define mqttexampleTOPICRx                                        "aws/topic/set_t
#define mqttexampleTOPICRx1                                       "aws/topic/set_s

void parse_temperature_led_actuation_message(char *Subscribedmsg,uint16_t m
char pubPayload[PUBLISH_PAYLOAD_BUFFER_LENGTH_BULK_JSON] = { 0x00 };
/**
 * @brief Delay for the synchronous publisher task between publishes.
 */
#define mqttexampleDELAY_BETWEEN_PUBLISH_OPERATIONS_MS    ( 2000U )

 * @brief Number of publishes done by each task in this demo.
#define mqttexamplePUBLISH_COUNT                          10

 * @brief Number of unsubscribe retries in this demo.
#define mqttexampleUNSUBSCRIBE_RETRIES                    5

 * @brief Number of times a publish has to be retried if agent cannot send
#define mqttexampleNUM_PUBLISH_RETRIES                    ( 3 )

 * @brief The maximum amount of time in milliseconds to wait for the comman
#define mqttexampleMAX_COMMAND_SEND_BLOCK_TIME_MS         ( 1000 )

 * @brief Maximum length of the thing name as set by AWS IoT.
#define mqttexampleTHING_NAME_MAX_LENGTH                  ( 128 )
```

Figure 25. Data Publishing Interval Settings (Optional)

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

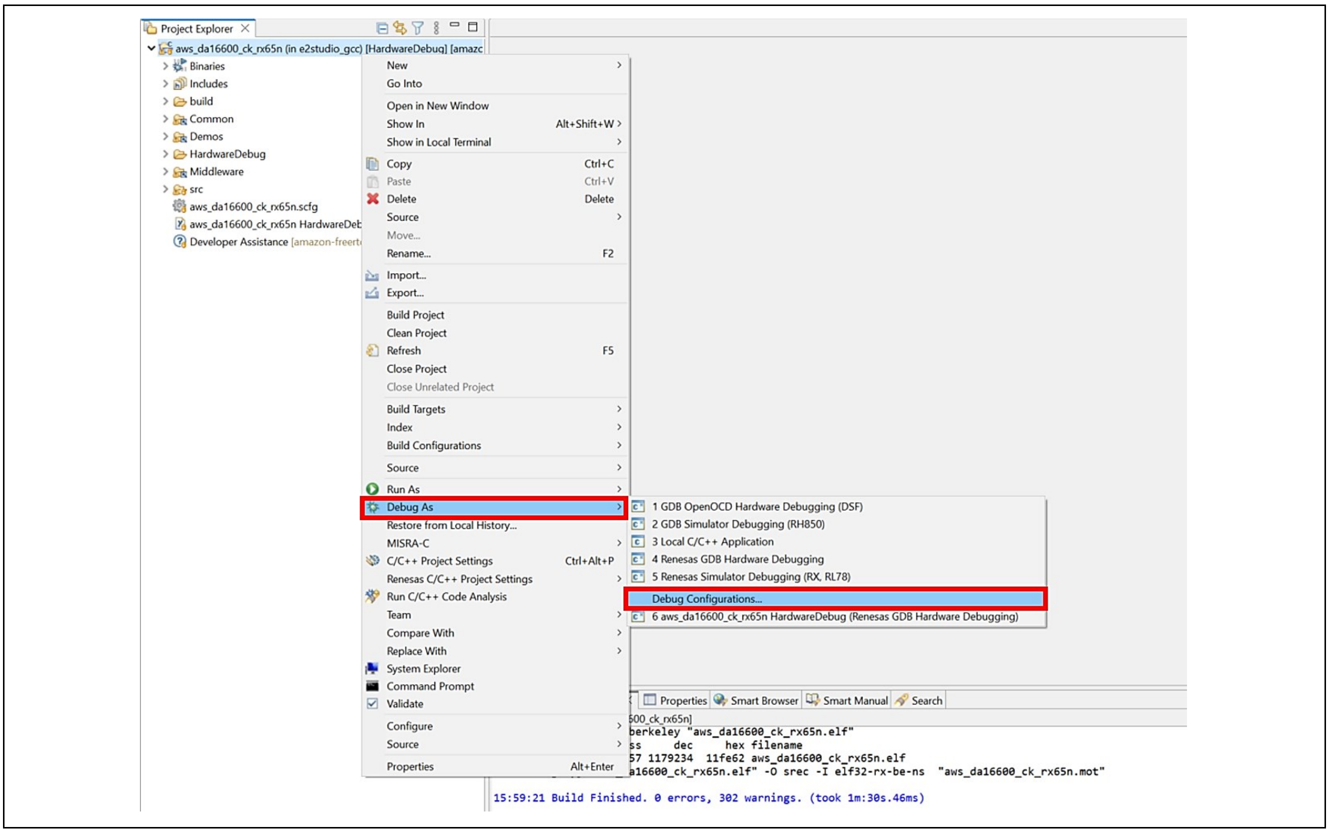


CDT Build Console [aws_da16600_ck_rx65n]

```
rx-elf-size --format=berkeley  aws_da16600_ck_rx65n.elf
   text    data     bss     dec     hex filename
 749792    6149  426105 1182046  12095e aws_da16600_ck_rx65n.elf
rx-elf-objcopy "aws_da16600_ck_rx65n.elf" -O srec -I elf32-rx-be-ns  "aws_da16600_ck_rx65n.mot"

15:16:06 Build Finished. 0 errors, 302 warnings. (took 2m:376ms)
```

Figure 26. Build the Project

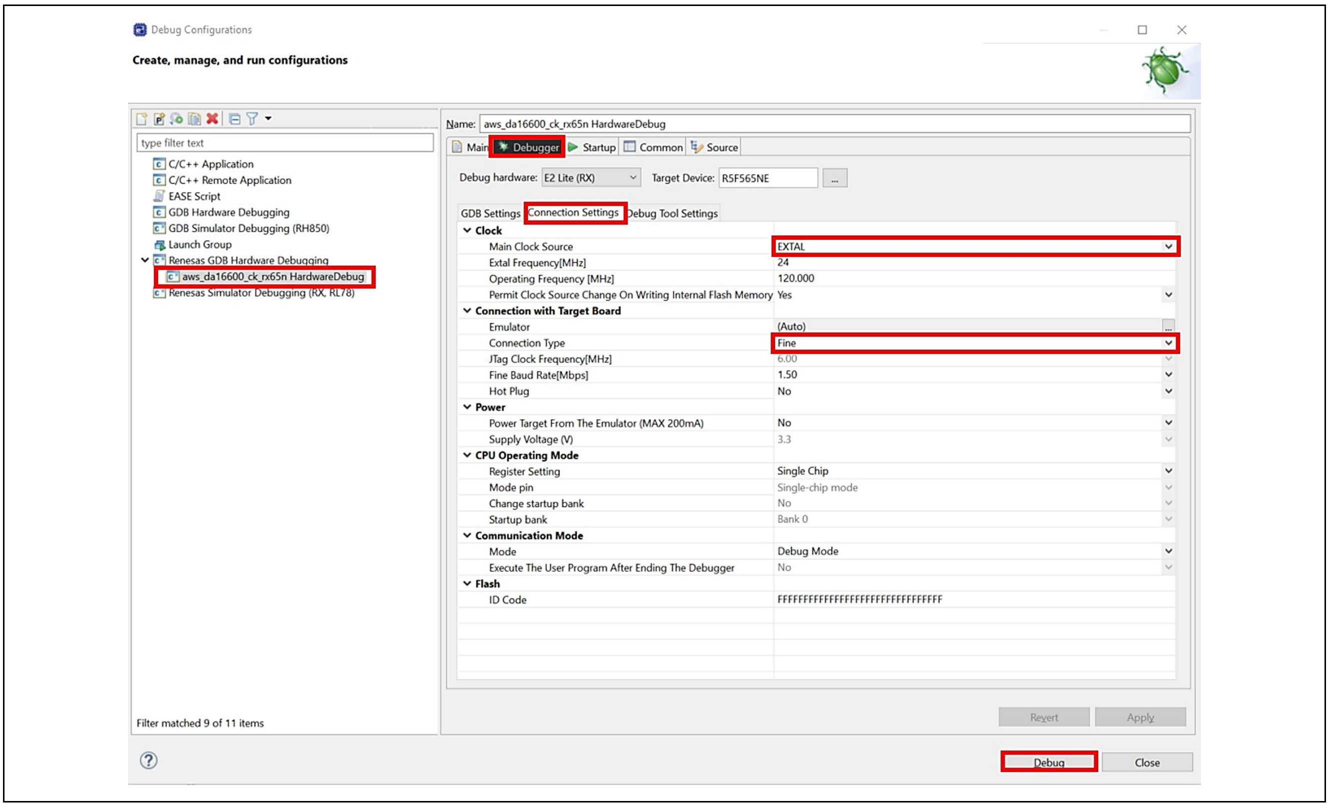
9. Debug configuration and load image to the board:
   Open **Debug Configurations**: Right-click on project **> Debug As > Debug Configurations**…



```
500_ck_rx65n]
berkeley "aws_da16600_ck_rx65n.elf"
ss     dec     hex filename
57 1179234  11fe62 aws_da16600_ck_rx65n.elf
a16600_ck_rx65n.elf" -O srec -I elf32-rx-be-ns  "aws_da16600_ck_rx65n.mot"

15:59:21 Build Finished. 0 errors, 302 warnings. (took 1m:30s.46ms)
```

Figure 27. Configuration Debugger (1/2)

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



Name: aws_da16600_ck_rx65n HardwareDebug

Debug hardware: E2 Lite (RX) Target Device: R5F565NE

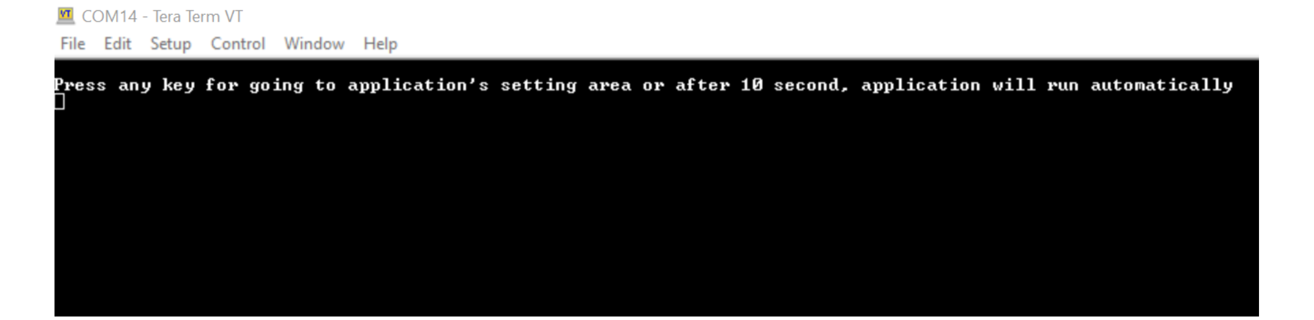
Clock	
Main Clock Source	EXTAL
Extal Frequency[MHz]	24
Operating Frequency [MHz]	120.000
Permit Clock Source Change On Writing Internal Flash Memory	Yes
Connection with Target Board	
Emulator	(Auto)
Connection Type	Fine
JTag Clock Frequency[MHz]	6.00
Fine Baud Rate[Mbps]	1.50
Hot Plug	No
Power	
Power Target From The Emulator (MAX 200mA)	No
Supply Voltage (V)	3.3
CPU Operating Mode	
Register Setting	Single Chip
Mode pin	Single-chip mode
Change startup bank	No
Startup bank	Bank 0
Communication Mode	
Mode	Debug Mode
Execute The User Program After Ending The Debugger	No
Flash	
ID Code	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Figure 28. Configuration Debugger (2/2)

5.1.4 Running the Application Project

To run the Application project, use the following instructions.

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

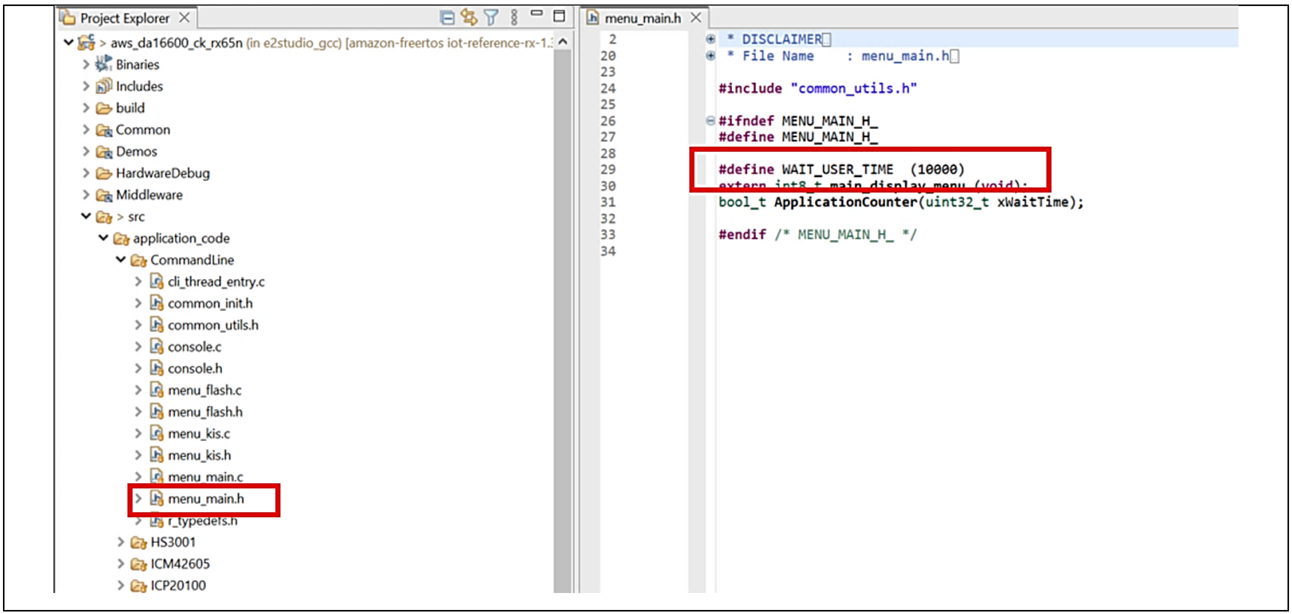


```
Press any key for going to application's setting area or after 10 second, application will run automatically
```

Figure 29. Start the Application

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

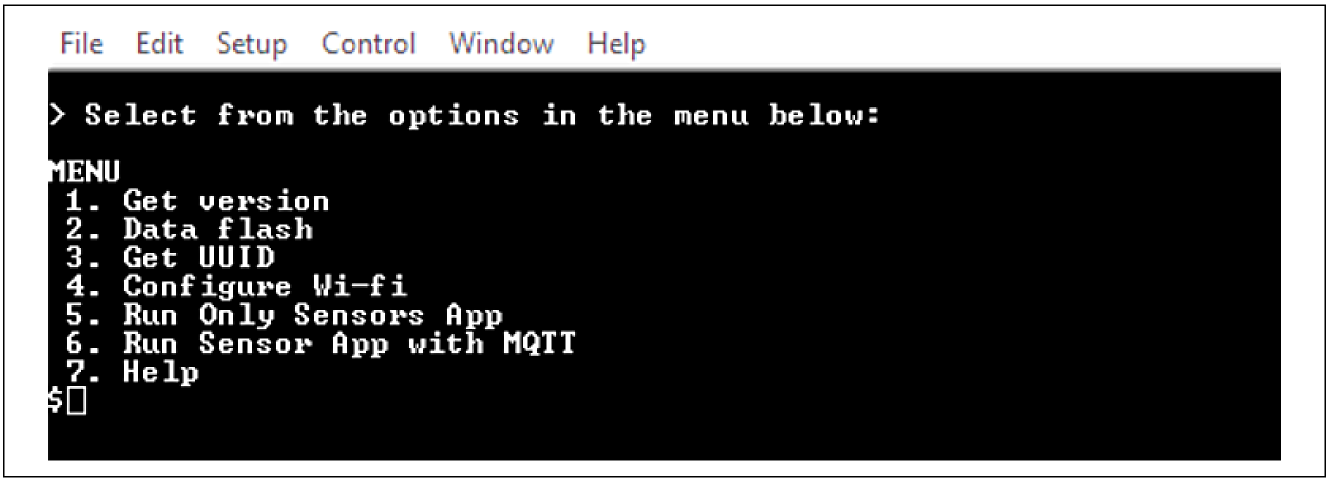
Note: After 10s, the application will run automatically with the option **"6. Run Sensor App with MQTT"** in the application's menu. The user can modify this time's value by changing the value of **WAIT_USER_TIME** macro in the file: **Projects\aws_da16600_ck_rx65n\e2studio_gcc\src\application_code\CommandLine\menu_main.h** before building the project:



```
 * DISCLAIMER
 * File Name    : menu_main.h

#include "common_utils.h"

#ifndef MENU_MAIN_H_
#define MENU_MAIN_H_

#define WAIT_USER_TIME  (10000)
extern int8_t main_display_menu (void);
bool_t ApplicationCounter(uint32_t xWaitTime);

#endif /* MENU_MAIN_H_ */
```

Figure 30. Modify macro to Wait for User Input

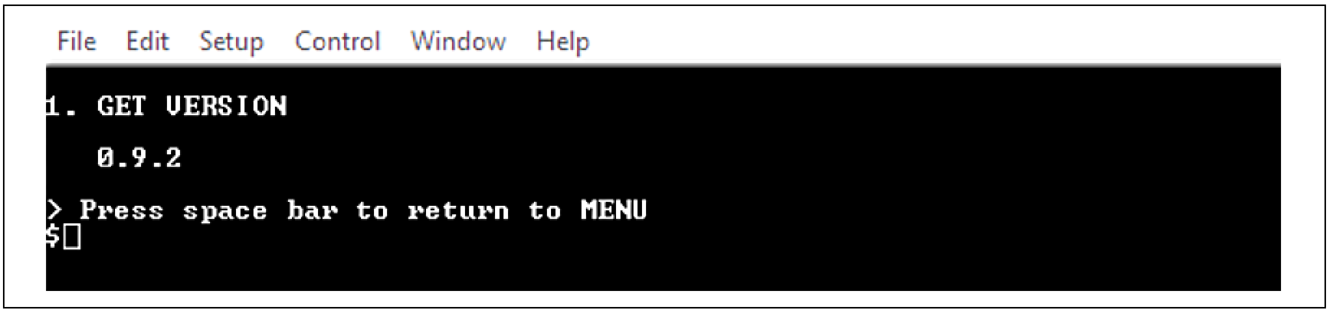
After pressing any key, the settings are as shown below.



```
> Select from the options in the menu below:

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

Figure 31. Main Menu

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



```
1. GET VERSION

   0.9.2

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

Figure 32. Get Version Information

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

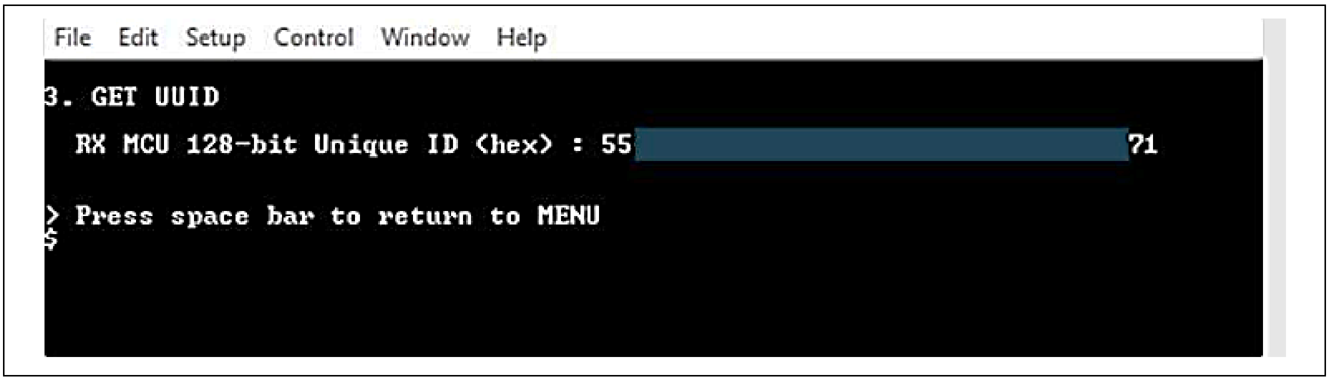
5.2 For Users Using the Provided Dashboard and AWS Account of Kit

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

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

5.2.1 Getting the UUID Information of the Board

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



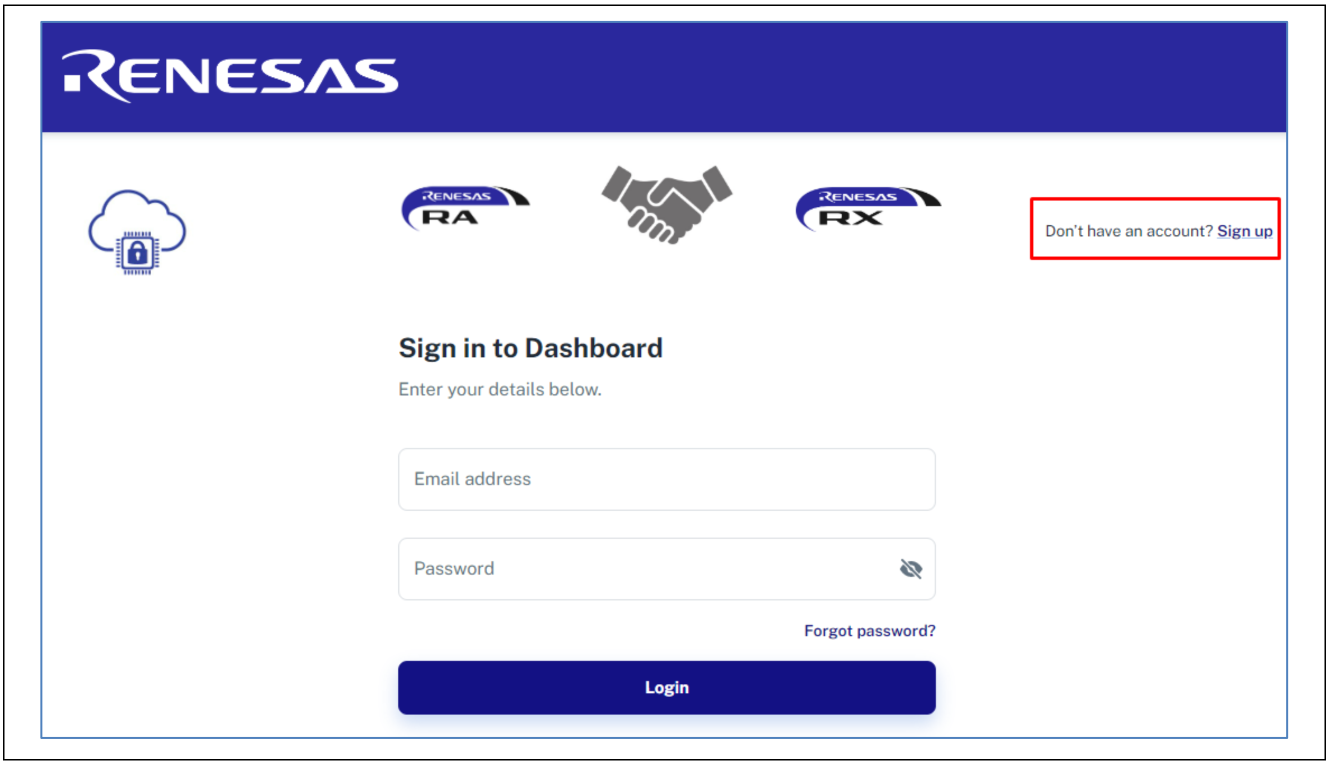
```
3. GET UUID

  RX MCU 128-bit Unique ID (hex) : 55[redacted]71

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

Figure 33. Getting Board UUID Information

5.2.2 To Get the Account 10 USD of Trial of AWS

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

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



Don't have an account? Sign up

Sign in to Dashboard

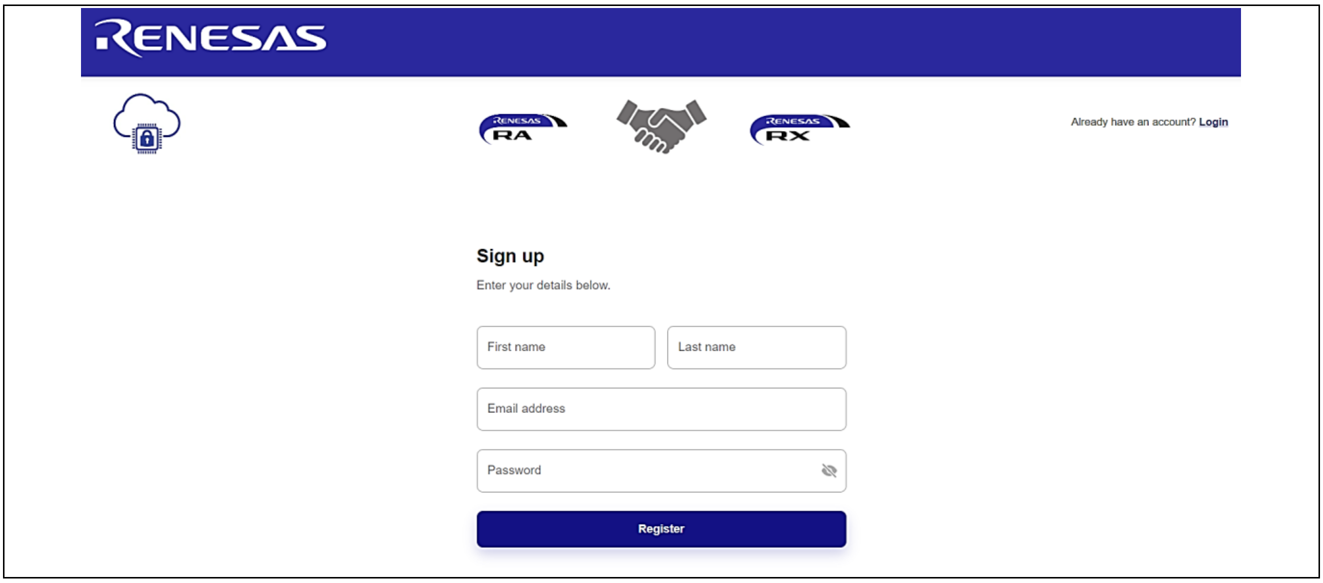
Enter your details below.

Email address

Password

Forgot password?

Figure 34. Creating Account



Already have an account? Login

Sign up

Enter your details below.

First name

Last name

Email address

Password

Figure 35. Registering Information

The rules for a valid first name and last name:

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

The rules for a valid email address:

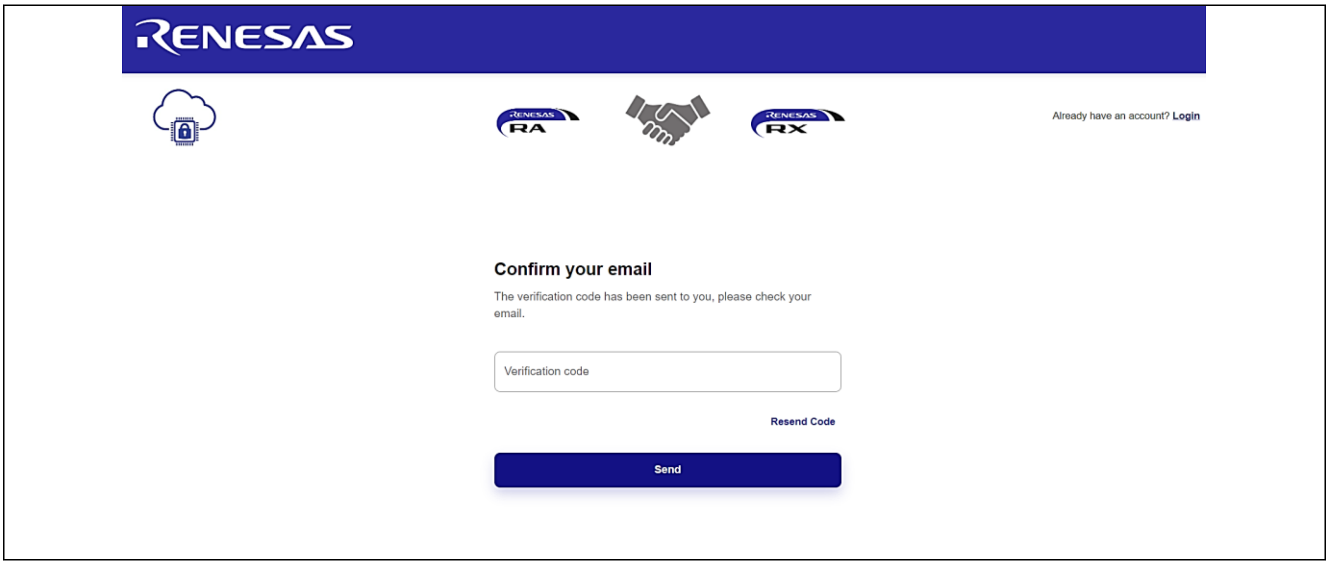
- The address must be a minimum of 6 and a maximum of 64 characters long.
- All characters must be 7-bit ASCII characters.
- There must be one and only one @ symbol, which separates the local name from the domain name.
- The local name cannot contain any of the following characters: whitespace, " ' ()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

The rules for a valid password:

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

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

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



Confirm your email

The verification code has been sent to you, please check your email.

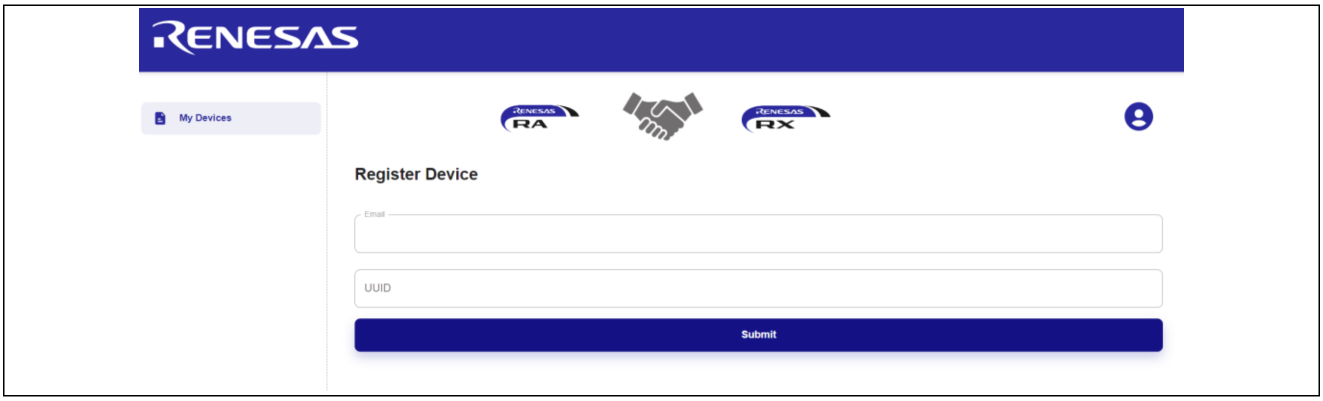
Verification code

Figure 36 Confirming Email

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

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

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



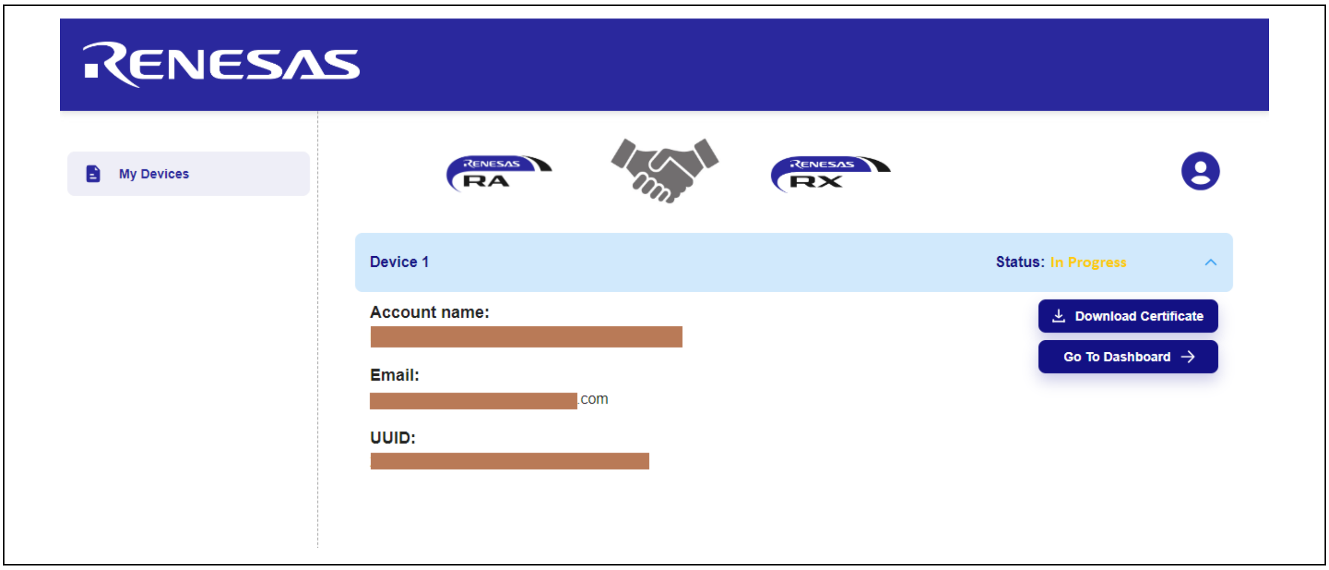
Register Device

Email

UUID

Figure 37. Register Device

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



Device 1 Status: In Progress

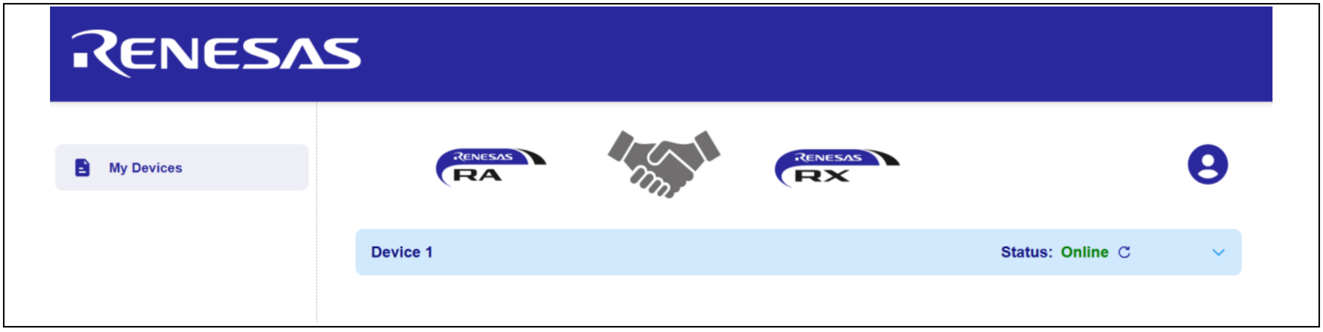
Account name: [redacted]

Email: [redacted].com

UUID: [redacted]

Figure 38. Device Registration In Progress

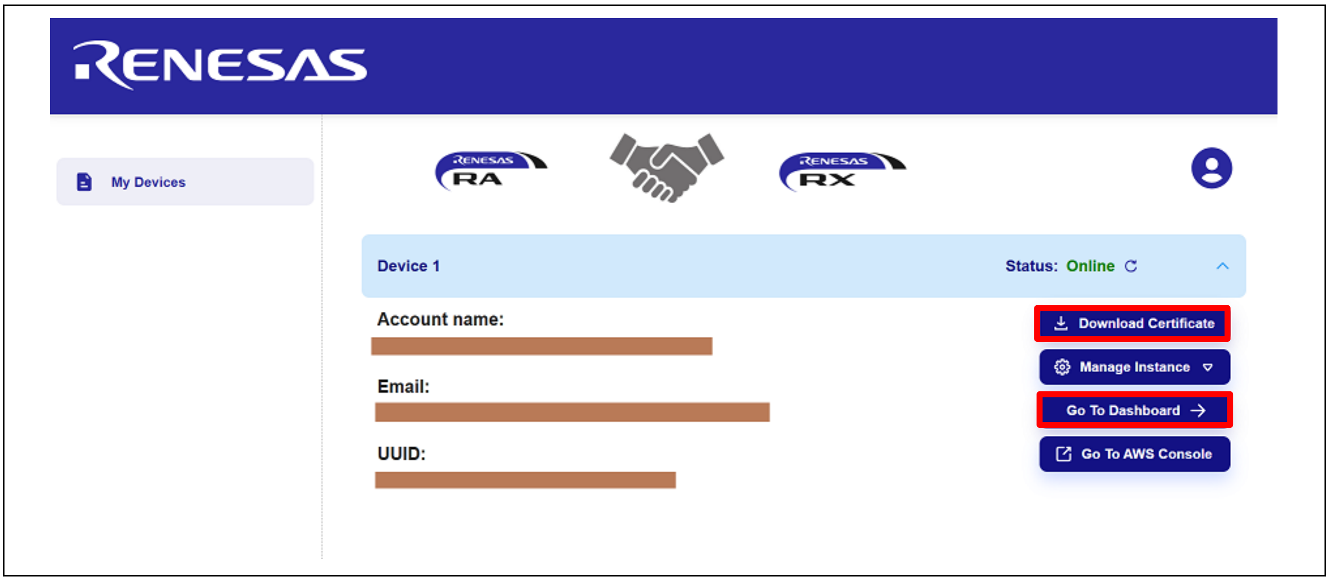
5. Refresh the page and the status of the registration changes to 'Online'.



Device 1 Status: Online

Figure 39. Active Device

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



Device 1 Status: Online

Account name: [redacted]

Email: [redacted]

UUID: [redacted]

Figure 40. Completing Device Provisioning

7. Click “**Go to Dashboard**” to access the dashboard. First time users will access the dashboard with default ‘username’ is **admin** and default ‘password’ is **admin1234**. Once completed, users can access the dashboard.

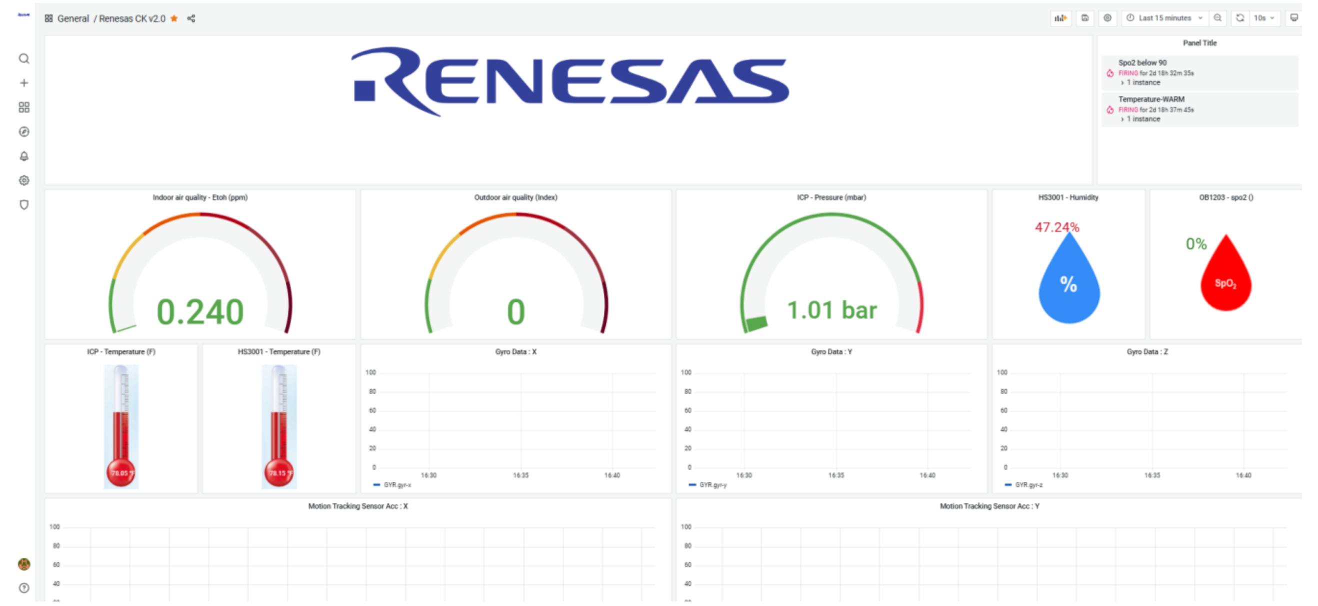


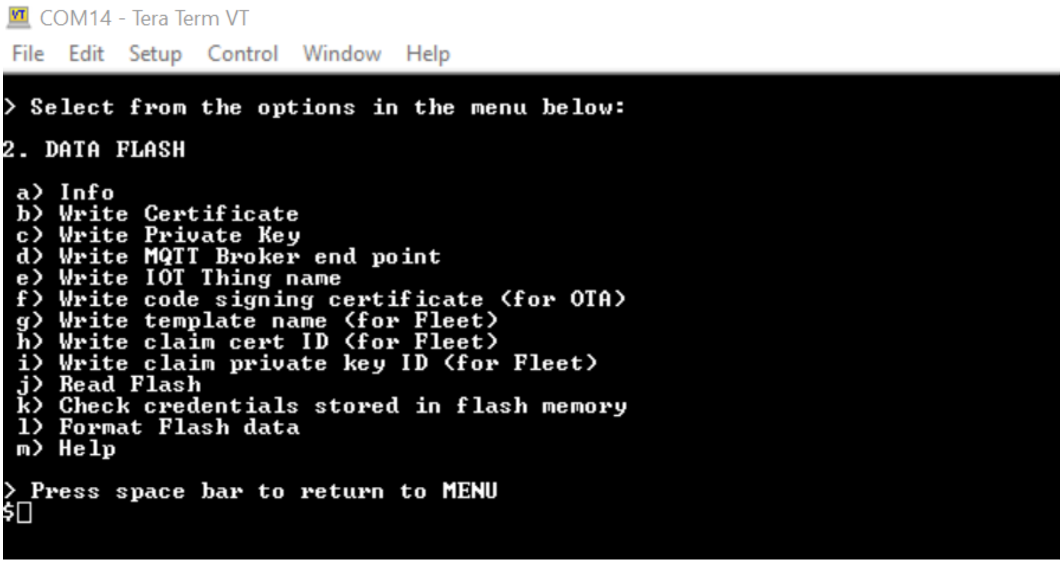
Figure 41. Dashboard for this Application

5.3 Software Preparation-Run Project from IDE

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

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

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

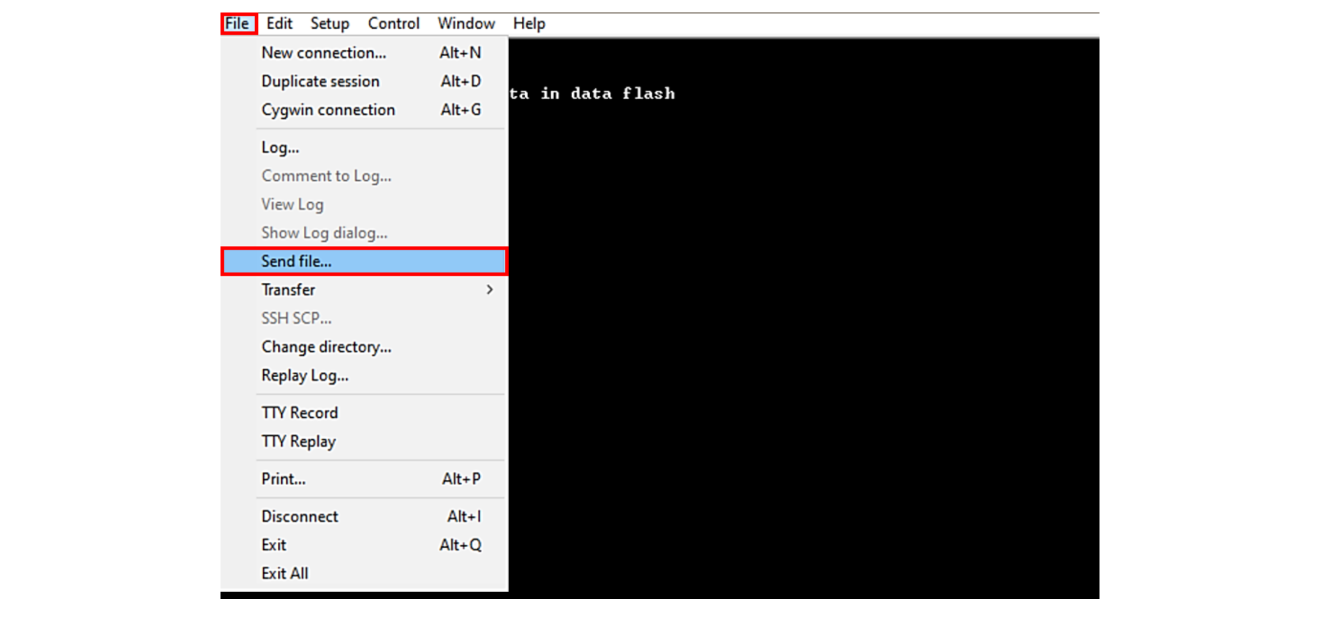


```
> Select from the options in the menu below:

2. DATA FLASH

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

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

Figure 42. Data Flash Related Menu and Commands

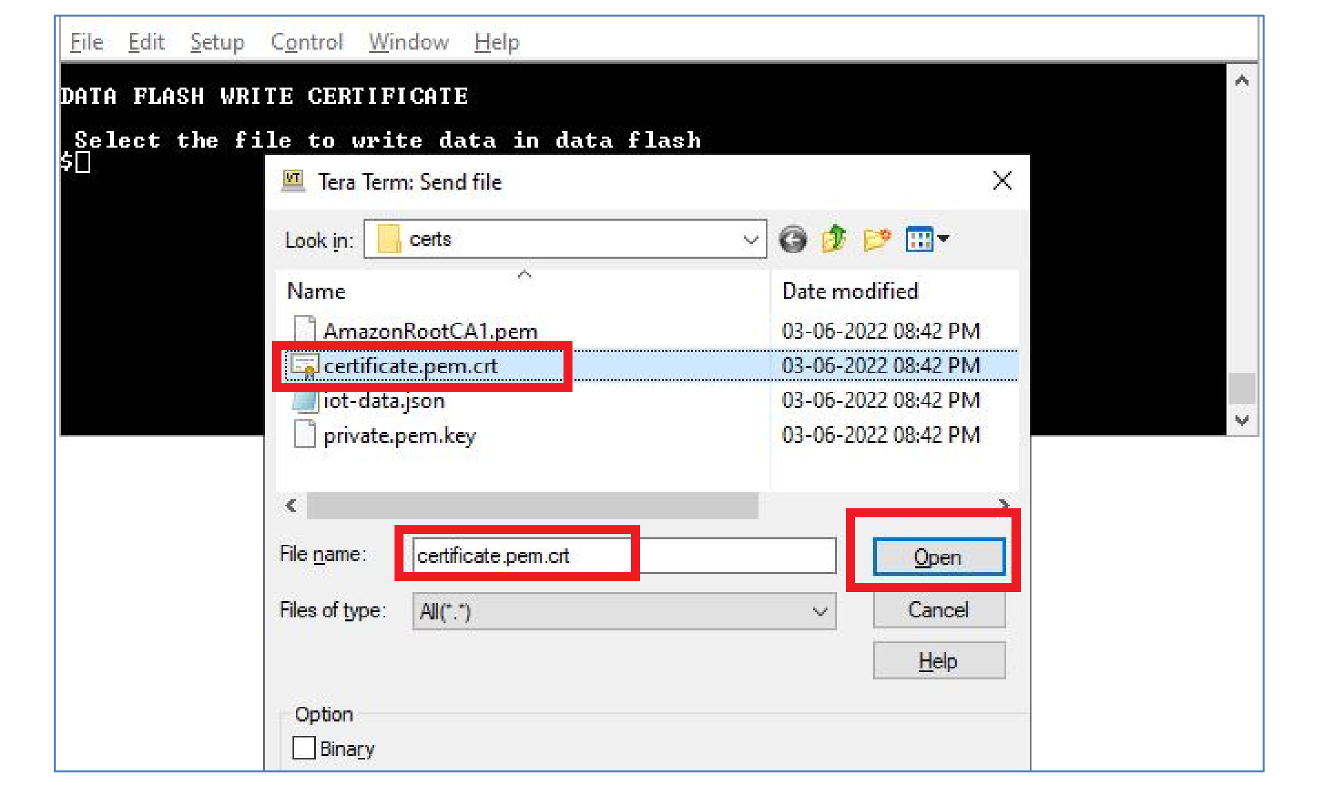
2. Please unzip the `cert.zip` from the dashboard (downloaded at **Figure 40. Completing Device Provisioning**)

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



```
ta in data flash
```

Figure 43. Accessing the Device Certificate



```
DATA FLASH WRITE CERTIFICATE

 Select the file to write data in data flash
$
```

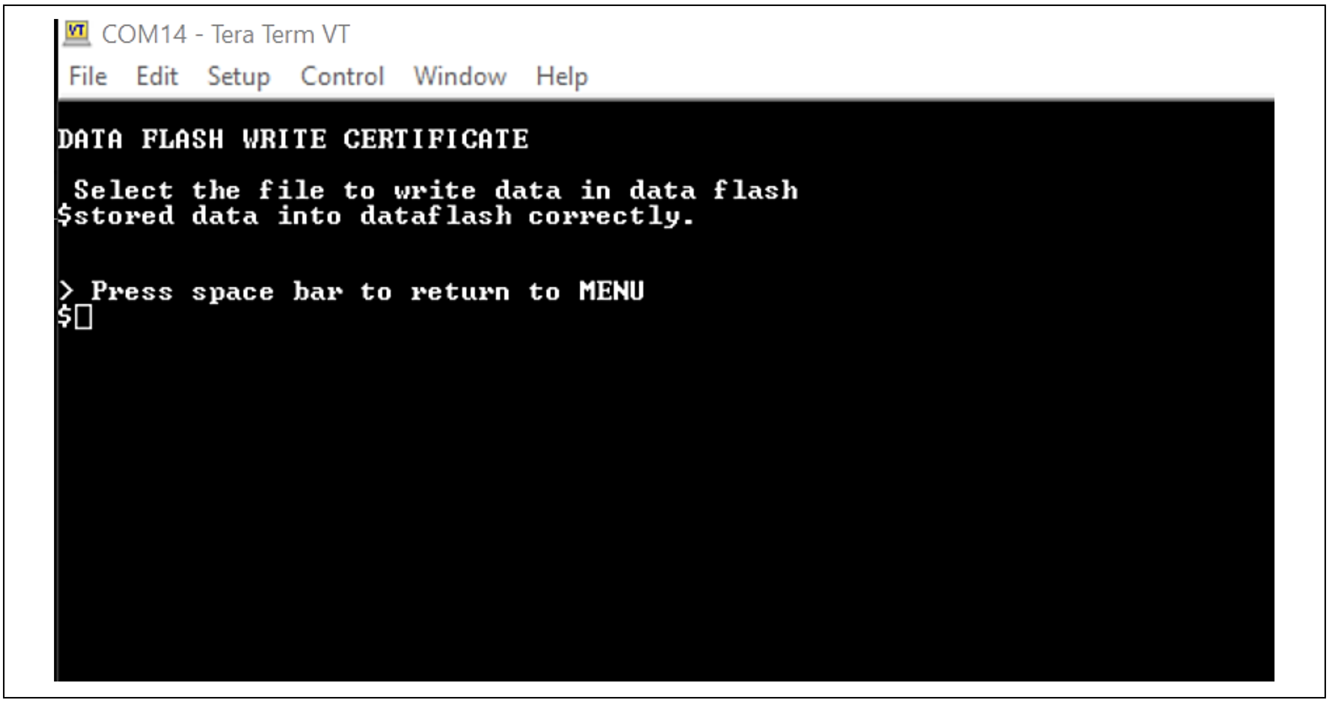
Look in: certs

Name	Date modified
AmazonRootCA1.pem	03-06-2022 08:42 PM
certificate.pem.crt	03-06-2022 08:42 PM
iot-data.json	03-06-2022 08:42 PM
private.pem.key	03-06-2022 08:42 PM

File name: certificate.pem.crt

Files of type: All(*.*)

Figure 44. Downloading the Device Certificate into the Data Flash



```
DATA FLASH WRITE CERTIFICATE

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


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

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

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

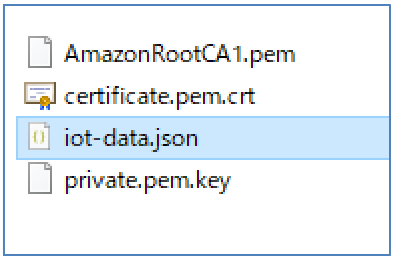
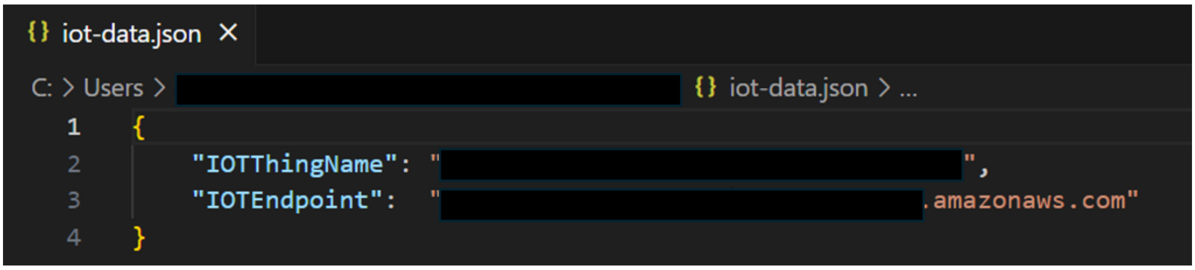


Figure 46. Getting the IoT Things Name and IoT Endpoint Information

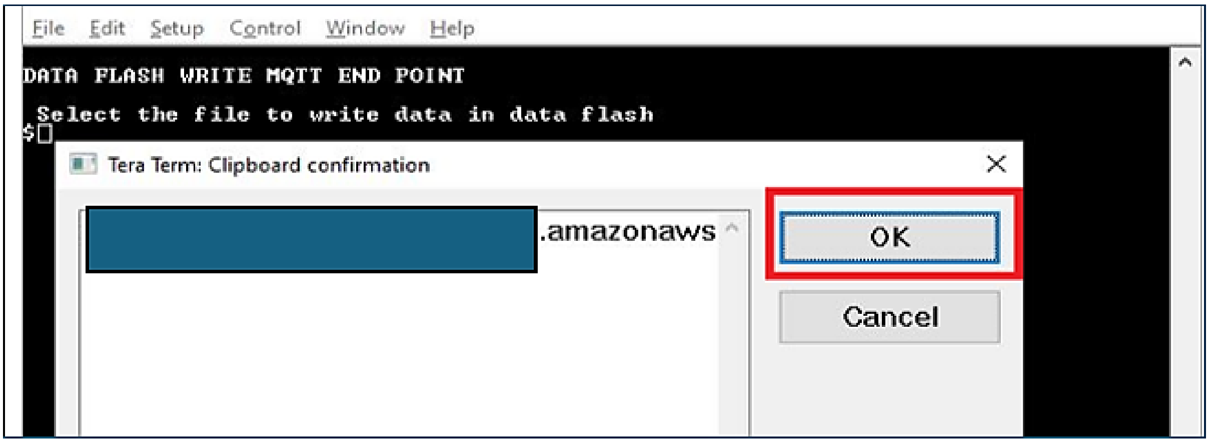
6. To store the **MQTT Broker end point,** copy the end point string between the quotes **xxxxxxxxxx.iot.us-east-1.amazonaws.com** from the downloaded certificate link, press the option '**d**' and click the **Edit** tab of the Tera Term and **Paste<CR>** and verify and confirm the valid string and press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

   Note: Please copy the IOTEndpoint without "".



```
{
    "IOTThingName": "[redacted]",
    "IOTEndpoint":  "[redacted].amazonaws.com"
}
```

Figure 47. Copy IoT Endpoint Information

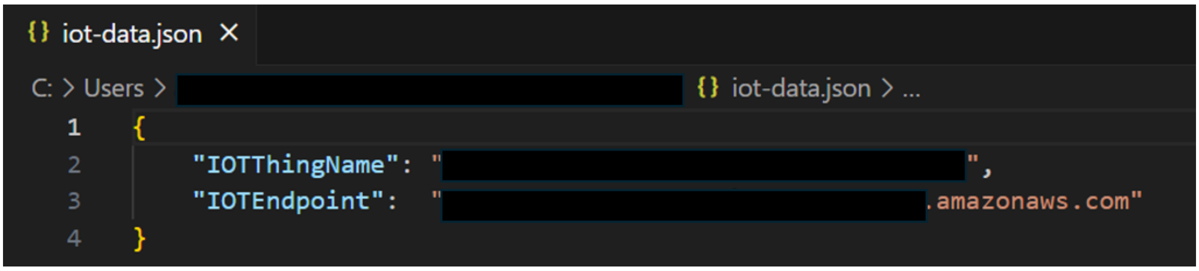

File Edit Setup Control Window Help
Tera Term: Clipboard confirmation
OK
Cancel

```
DATA FLASH WRITE MQTT END POINT
 Select the file to write data in data flash
$
```

[redacted].amazonaws

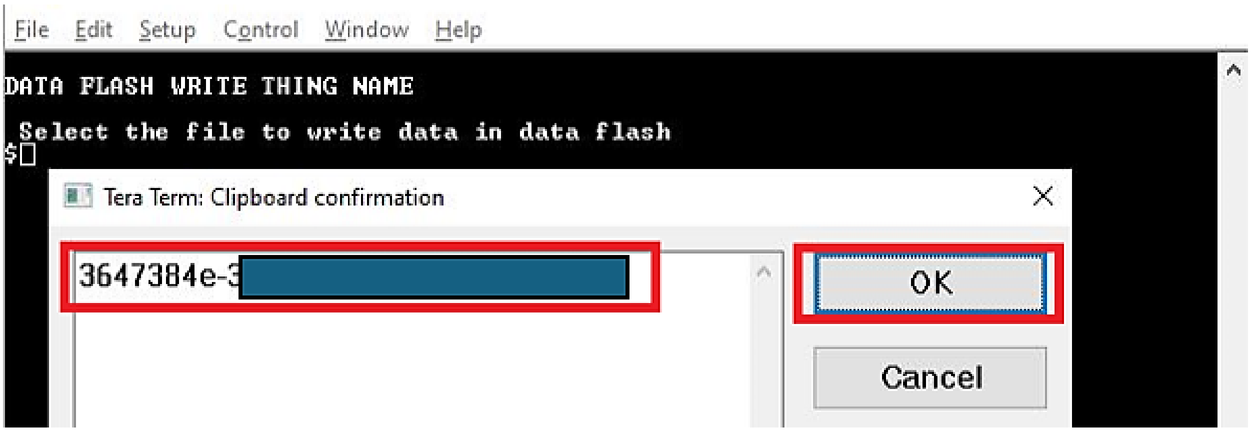
Figure 48. Storing the MQTT IoT Endpoint into the Data Flash

7. To store the IOT Thing Name, copy the Thing Name string between the quotes xxxxxxxx-xxxx-xxxxxxxx-xxxx of IoT thing Name from the downloaded certificate link, press the option ‘**e**’ and click the **Edit** tab of the Tera Term and “**Paste<CR>**” and verify and confirm the valid string and press OK.
   Note: Please copy the IOTThingName without “”.



```
{
    "IOTThingName": "[redacted]",
    "IOTEndpoint":  "[redacted].amazonaws.com"
}
```

Figure 49. Copy the IOTThingName


File Edit Setup Control Window Help
Tera Term: Clipboard confirmation
OK
Cancel

```
DATA FLASH WRITE THING NAME
 Select the file to write data in data flash
$
```

3647384e-3[redacted]

Figure 50. Storing the Thing Name into the Data Flash

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

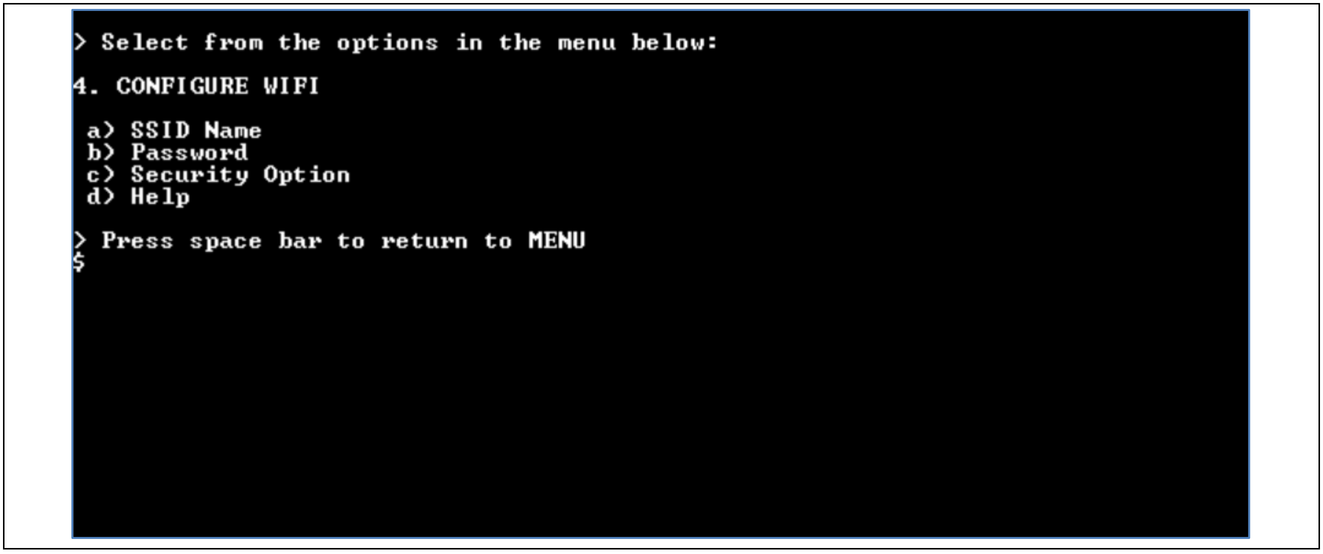
   Note: Validation of the stored data is very limited and validates a minimum set of data points. Users are required to input the valid data to the flash obtained from the Dashboard for the proper working of the Application.

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

5.3.2 Storing the SSID Name, Password, and Security Option

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

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



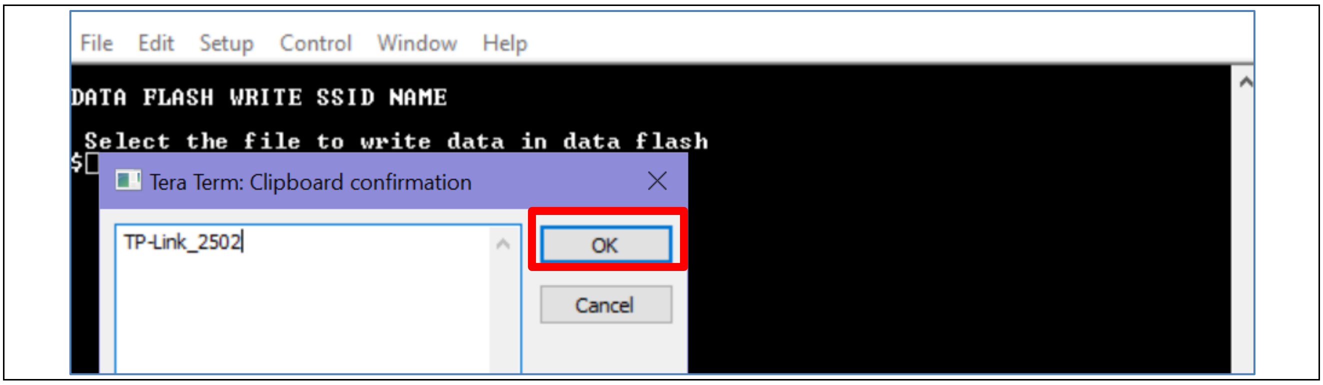
```
> Select from the options in the menu below:

4. CONFIGURE WIFI

 a) SSID Name
 b) Password
 c) Security Option
 d) Help

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

Figure 51. Configure Wi-Fi related Menu and Commands

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

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

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



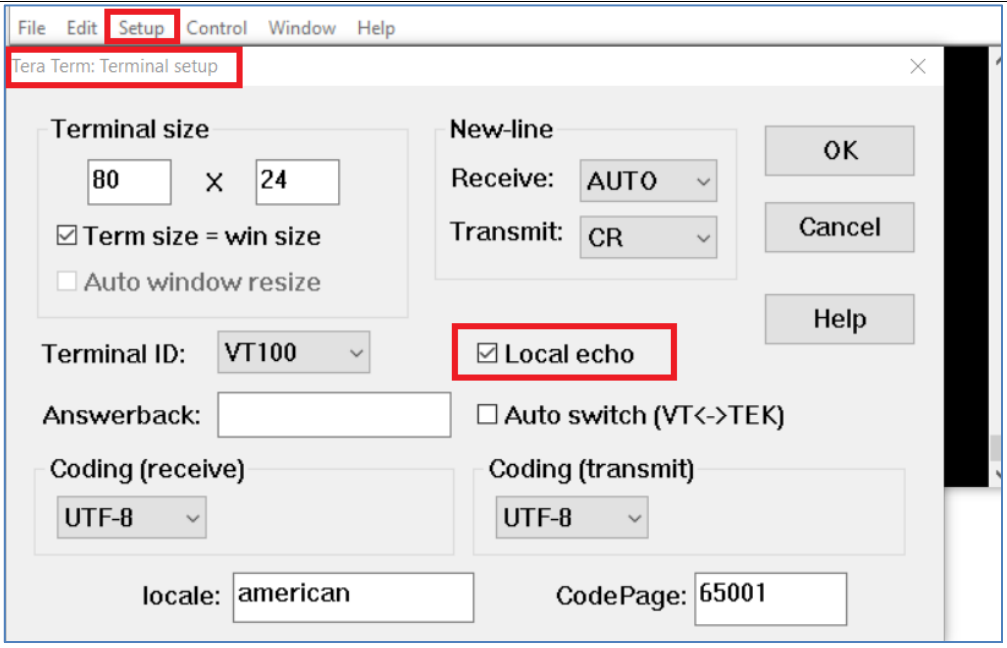
File Edit Setup Control Window Help

```
DATA FLASH WRITE SSID NAME

 Select the file to write data in data flash
$
```

Tera Term: Clipboard confirmation
TP-Link_2502
OK
Cancel

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

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

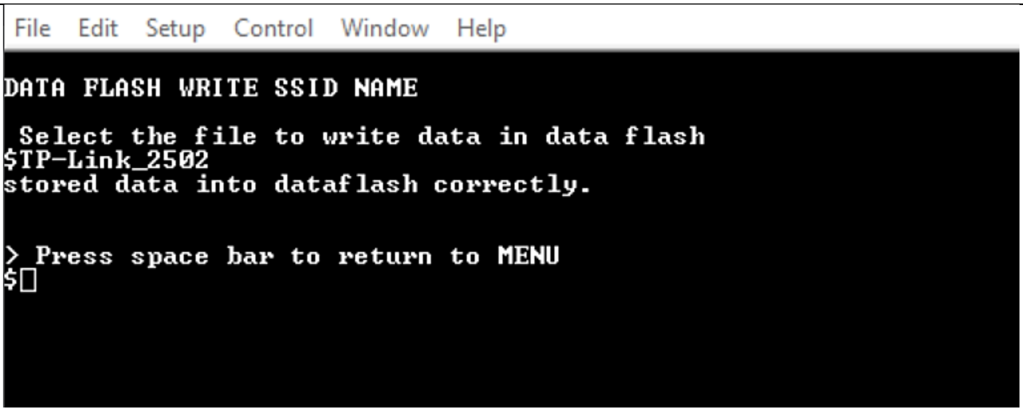


File Edit Setup Control Window Help
Tera Term: Terminal setup
OK
Cancel
Help

Terminal size: 80 × 24
☑ Term size = win size
□ Auto window resize
New-line
Receive: AUTO
Transmit: CR
Terminal ID: VT100
☑ Local echo
Answerback:
□ Auto switch (VT<->TEK)
Coding (receive): UTF-8
Coding (transmit): UTF-8
locale: american
CodePage: 65001

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

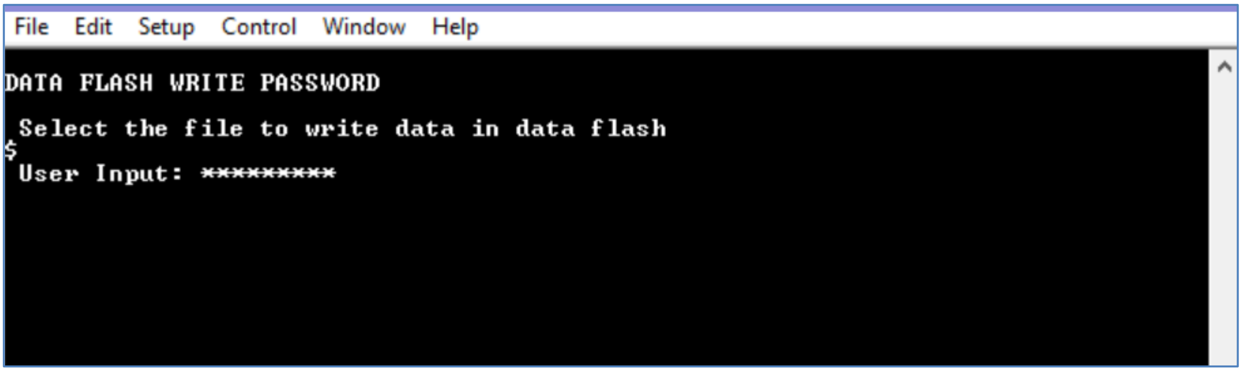
— Input SSID and press **Enter**.



```
DATA FLASH WRITE SSID NAME

 Select the file to write data in data flash
$TP-Link_2502
stored data into dataflash correctly.

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

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

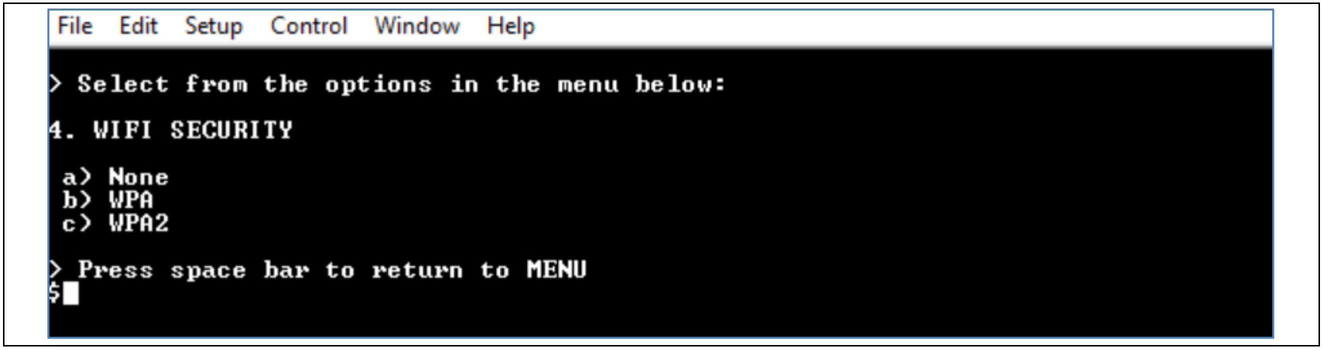
3. To store the password, in the Wi-Fi menu, press the option **'b'**, input the password, confirm the valid string then press Enter. The maximum length of the password is 32 characters, and the minimum length is 1 character.



```
DATA FLASH WRITE PASSWORD

 Select the file to write data in data flash
$
 User Input: *********
```

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

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

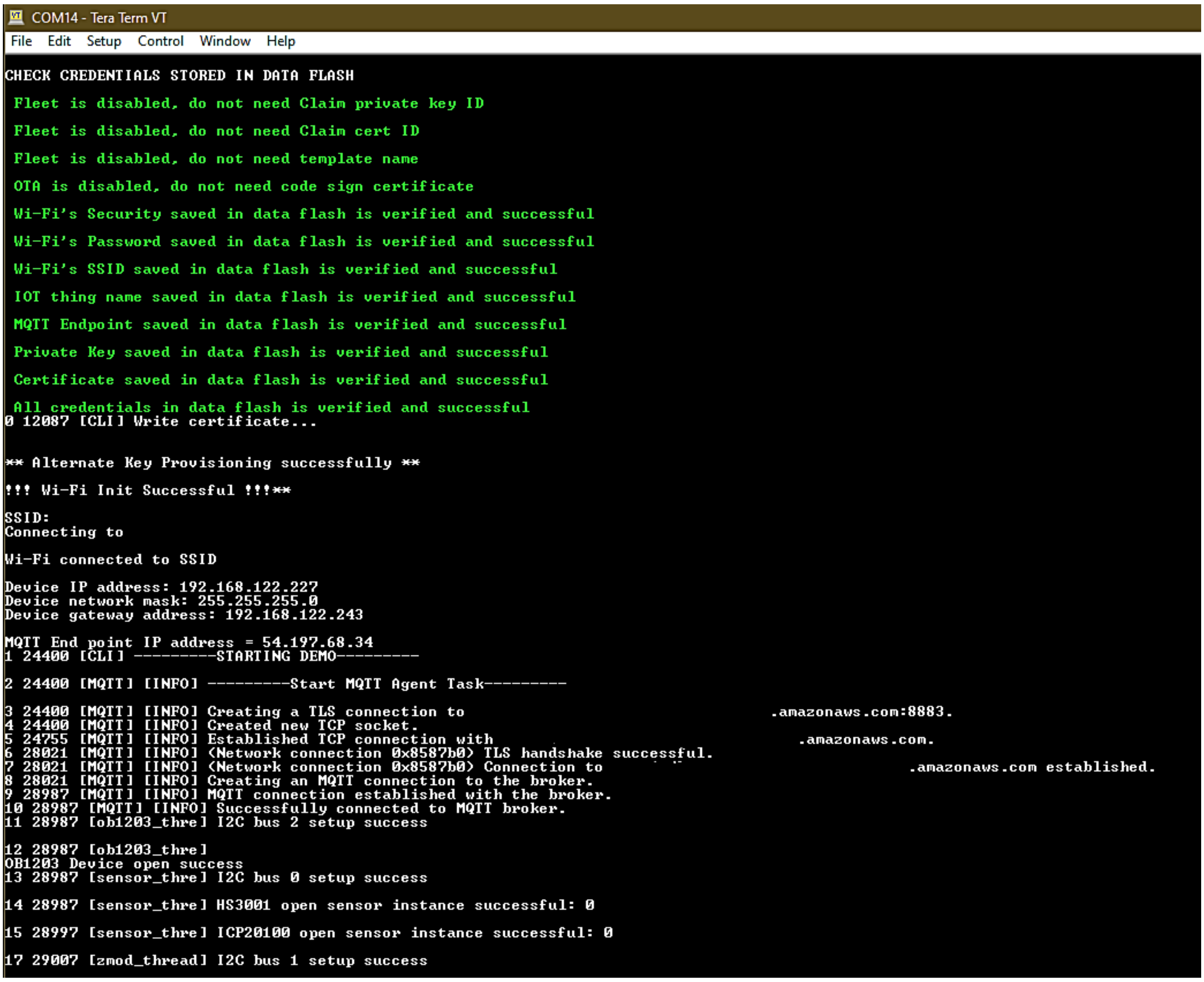


```
> Select from the options in the menu below:

4. WIFI SECURITY

 a> None
 b> WPA
 c> WPA2

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

Figure 56. Storing the Security into the Data Flash

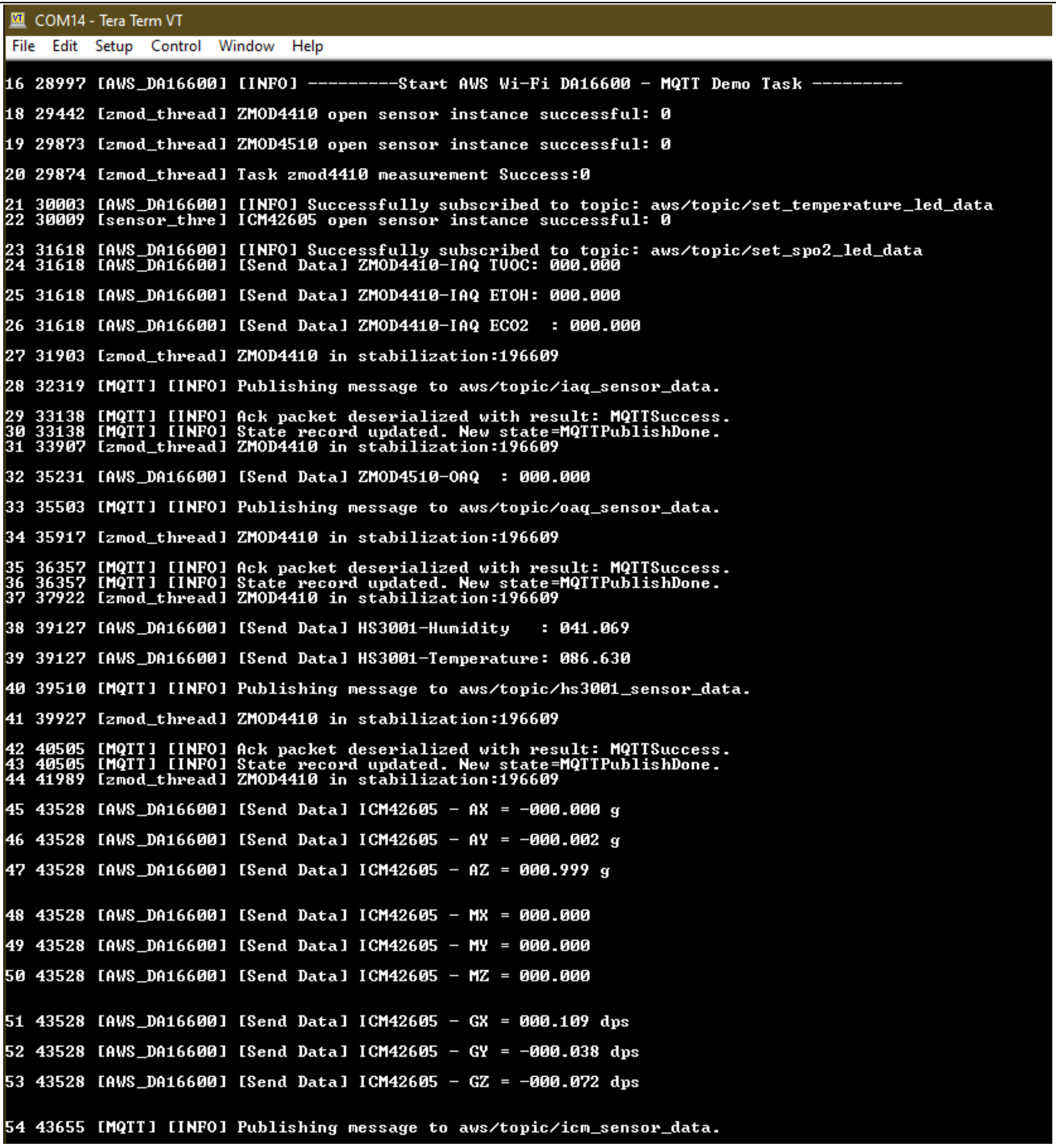
5.3.3 Starting the Application

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



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

** Alternate Key Provisioning successfully **
!!! Wi-Fi Init Successful !!!**
SSID:
Connecting to
Wi-Fi connected to SSID
Device IP address: 192.168.122.227
Device network mask: 255.255.255.0
Device gateway address: 192.168.122.243

MQTT End point IP address = 54.197.68.34
1 24400 [CLI] ---------STARTING DEMO---------

2 24400 [MQTT] [INFO] ---------Start MQTT Agent Task---------

3 24400 [MQTT] [INFO] Creating a TLS connection to                                  .amazonaws.com:8883.
4 24400 [MQTT] [INFO] Created new TCP socket.
5 24755 [MQTT] [INFO] Established TCP connection with                                  .amazonaws.com.
6 28021 [MQTT] [INFO] (Network connection 0x8587b0) TLS handshake successful.
7 28021 [MQTT] [INFO] (Network connection 0x8587b0) Connection to                                  .amazonaws.com established.
8 28021 [MQTT] [INFO] Creating an MQTT connection to the broker.
9 28987 [MQTT] [INFO] MQTT connection established with the broker.
10 28987 [MQTT] [INFO] Successfully connected to MQTT broker.
11 28987 [ob1203_thre] I2C bus 2 setup success

12 28987 [ob1203_thre]
OB1203 Device open success
13 28987 [sensor_thre] I2C bus 0 setup success

14 28987 [sensor_thre] HS3001 open sensor instance successful: 0

15 28997 [sensor_thre] ICP20100 open sensor instance successful: 0

17 29007 [zmod_thread] I2C bus 1 setup success
```

Figure 57. Welcome Screen on the Console



```
16 28997 [AWS_DA16600] [INFO] ---------Start AWS Wi-Fi DA16600 - MQTT Demo Task ---------

18 29442 [zmod_thread] ZMOD4410 open sensor instance successful: 0

19 29873 [zmod_thread] ZMOD4510 open sensor instance successful: 0

20 29874 [zmod_thread] Task zmod4410 measurement Success:0

21 30003 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
22 30009 [sensor_thre] ICM42605 open sensor instance successful: 0

23 31618 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
24 31618 [AWS_DA16600] [Send Data] ZMOD4410-IAQ TVOC: 000.000

25 31618 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ETOH: 000.000

26 31618 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ECO2  : 000.000

27 31903 [zmod_thread] ZMOD4410 in stabilization:196609

28 32319 [MQTT] [INFO] Publishing message to aws/topic/iaq_sensor_data.

29 33138 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
30 33138 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
31 33907 [zmod_thread] ZMOD4410 in stabilization:196609

32 35231 [AWS_DA16600] [Send Data] ZMOD4510-OAQ  : 000.000

33 35503 [MQTT] [INFO] Publishing message to aws/topic/oaq_sensor_data.

34 35917 [zmod_thread] ZMOD4410 in stabilization:196609

35 36357 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
36 36357 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
37 37922 [zmod_thread] ZMOD4410 in stabilization:196609

38 39127 [AWS_DA16600] [Send Data] HS3001-Humidity    : 041.069

39 39127 [AWS_DA16600] [Send Data] HS3001-Temperature: 086.630

40 39510 [MQTT] [INFO] Publishing message to aws/topic/hs3001_sensor_data.

41 39927 [zmod_thread] ZMOD4410 in stabilization:196609

42 40505 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
43 40505 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
44 41989 [zmod_thread] ZMOD4410 in stabilization:196609
45 43528 [AWS_DA16600] [Send Data] ICM42605 - AX = -000.000 g

46 43528 [AWS_DA16600] [Send Data] ICM42605 - AY = -000.002 g

47 43528 [AWS_DA16600] [Send Data] ICM42605 - AZ = 000.999 g


48 43528 [AWS_DA16600] [Send Data] ICM42605 - MX = 000.000

49 43528 [AWS_DA16600] [Send Data] ICM42605 - MY = 000.000

50 43528 [AWS_DA16600] [Send Data] ICM42605 - MZ = 000.000


51 43528 [AWS_DA16600] [Send Data] ICM42605 - GX = 000.109 dps

52 43528 [AWS_DA16600] [Send Data] ICM42605 - GY = -000.038 dps

53 43528 [AWS_DA16600] [Send Data] ICM42605 - GZ = -000.072 dps


54 43655 [MQTT] [INFO] Publishing message to aws/topic/icm_sensor_data.
```

Figure 58. Application with MQTT

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

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

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

5.4 For User Who Use Their Own AWS Account

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

5.4.1 Get an AWS Account

Get an AWS account > Click the **Sign into the Console** button.

5.4.2 Log in to the AWS Management Console

Amazon Web Services > My Account > AWS Management Console

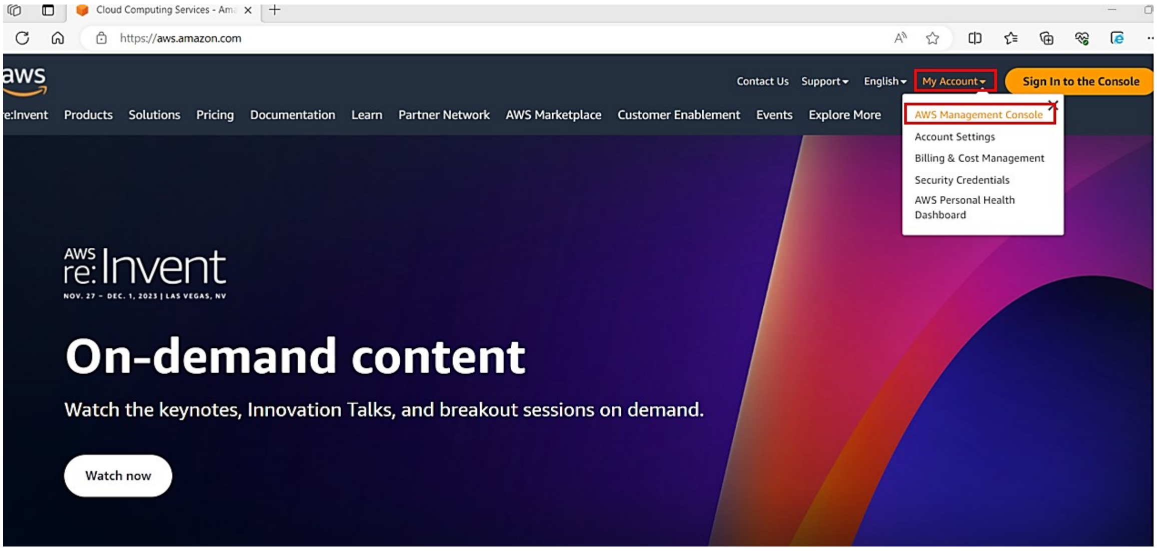

https://aws.amazon.com
Contact Us
Support
English
My Account
Sign In to the Console
AWS Management Console
Account Settings
Billing & Cost Management
Security Credentials
AWS Personal Health Dashboard
re:Invent Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace Customer Enablement Events Explore More
AWS re:Invent
NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV
On-demand content
Watch the keynotes, Innovation Talks, and breakout sessions on demand.
Watch now

Figure 59. Login the AWS

5.4.3 Move to IoT Core Control Panel

AWS services > All services > IoT Core

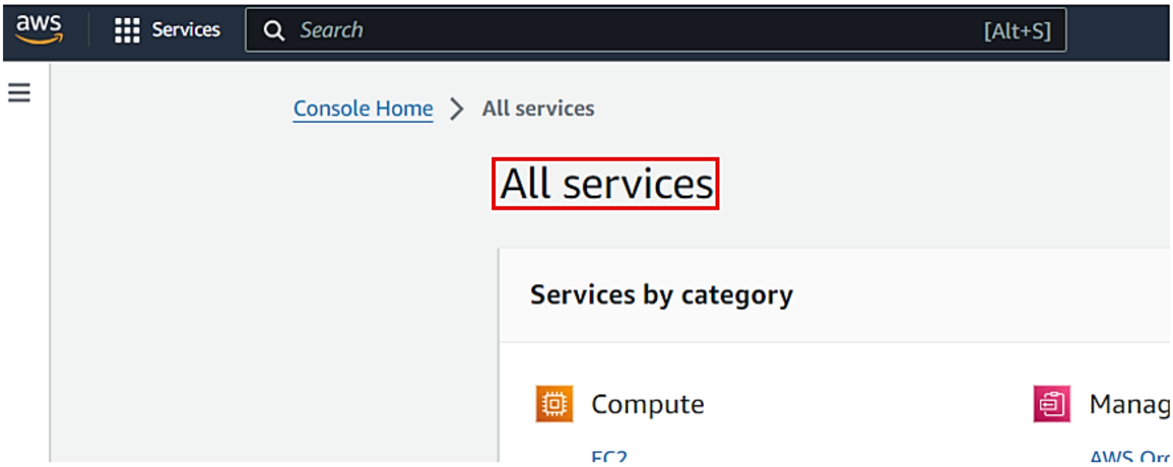
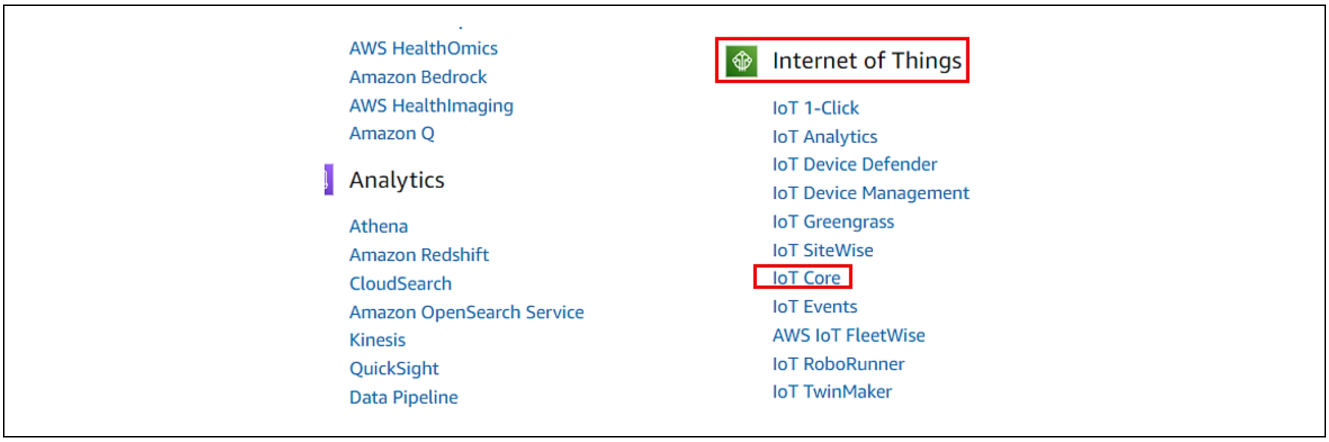


Figure 60. Search the IoT Core (1/2)



AWS HealthOmics
Amazon Bedrock
AWS HealthImaging
Amazon Q

Analytics

Athena
Amazon Redshift
CloudSearch
Amazon OpenSearch Service
Kinesis
QuickSight
Data Pipeline

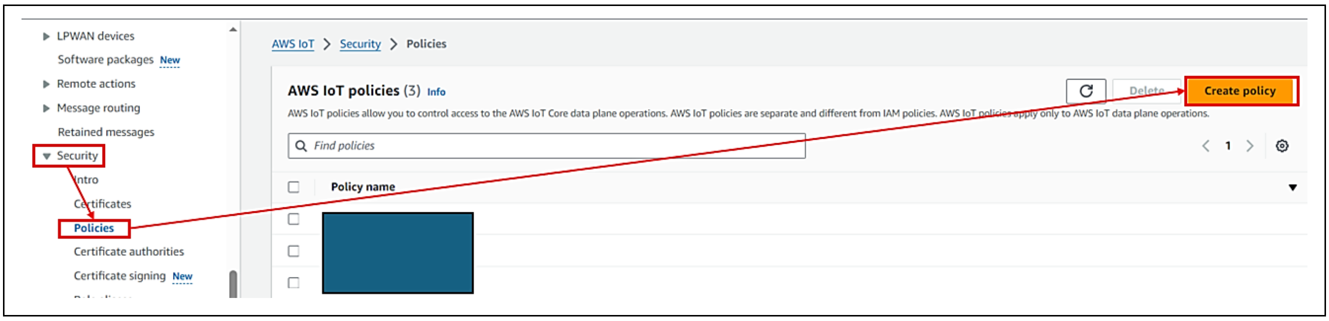
Internet of Things

IoT 1-Click
IoT Analytics
IoT Device Defender
IoT Device Management
IoT Greengrass
IoT SiteWise
IoT Core
IoT Events
AWS IoT FleetWise
IoT RoboRunner
IoT TwinMaker

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

5.4.4 Create a Security Policy

Secure > Policies > Create a policy.



▶ LPWAN devices
Software packages New
▶ Remote actions
▶ Message routing
Retained messages
▼ Security
Intro
Certificates
Policies
Certificate authorities
Certificate signing New

AWS IoT > Security > Policies

AWS IoT policies (3) Info

AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

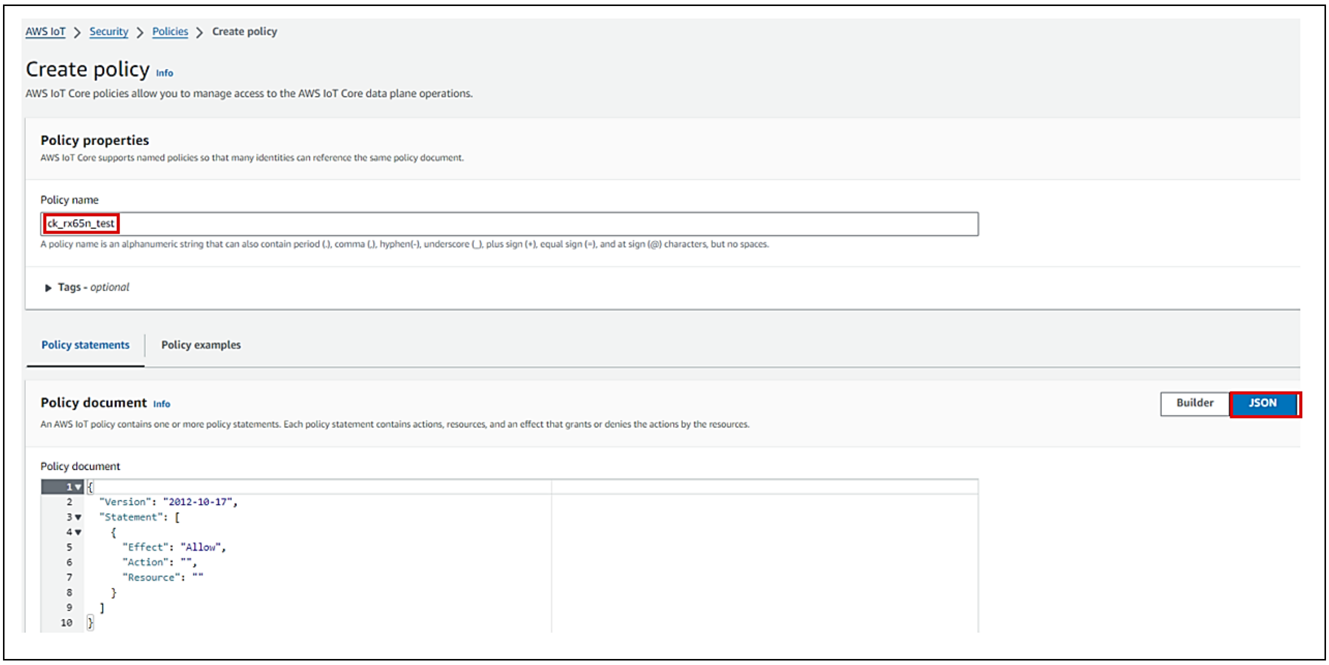
Delete | Create policy

Find policies

< 1 >

□	Policy name
□	[redacted]
□	
□	

Figure 62. Create the Policy (1/3)



AWS IoT > Security > Policies > Create policy

Create policy Info

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

Policy properties

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name: ck_rx65n_test

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

▶ Tags - *optional*

Policy statements | Policy examples

Policy document Info

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

Builder | JSON

Policy document

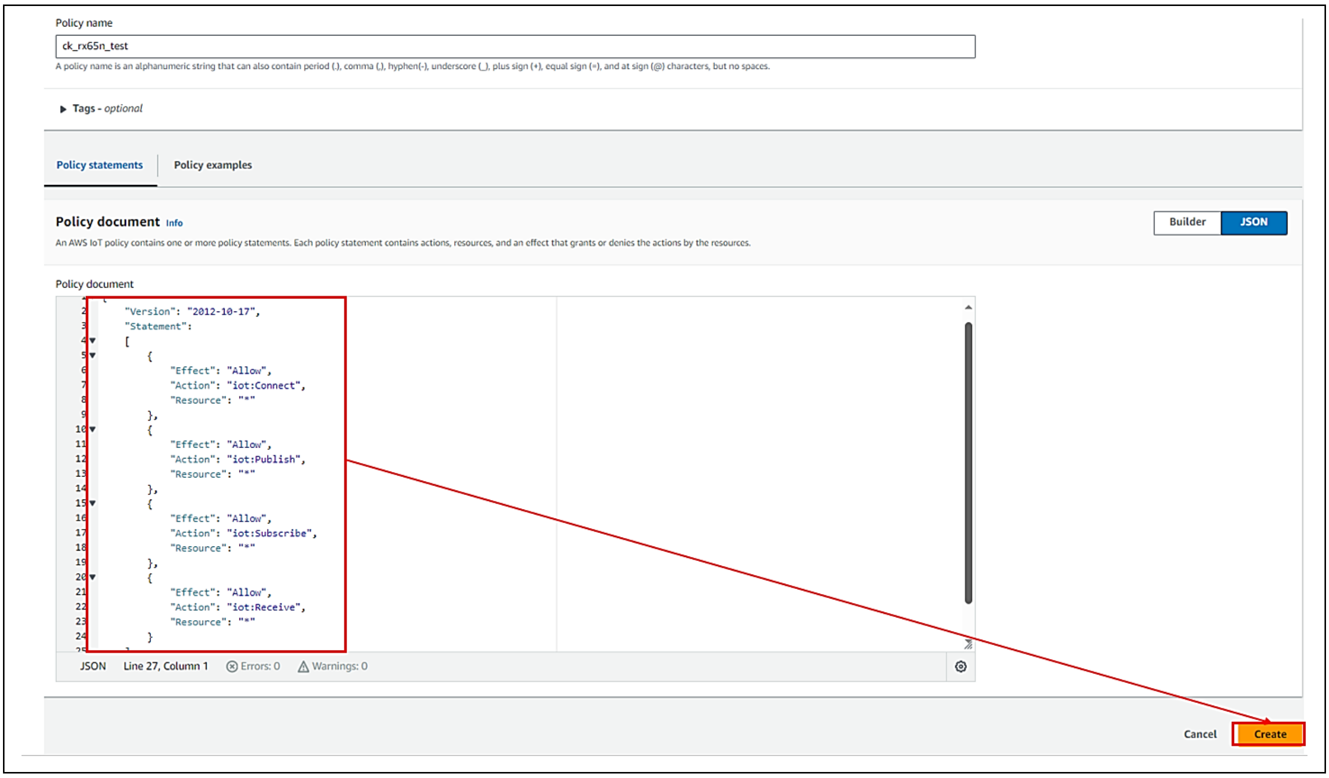
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "",
      "Resource": ""
    }
  ]
}
```

Figure 63. Create the Policy (2/3)

Copy the following code:

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

Paste the copied code into the policy syntax > Create.



Policy name

ck_rx65n_test

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

Policy document Info

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

Policy document

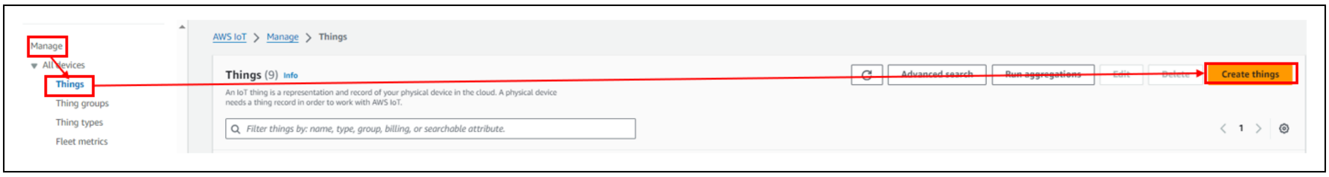
```
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Publish",
            "Resource": "*"
        },
        {
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

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

Figure 64. Create the Policy (3/3)

5.4.5 Register your device (thing) with AWS IoT

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

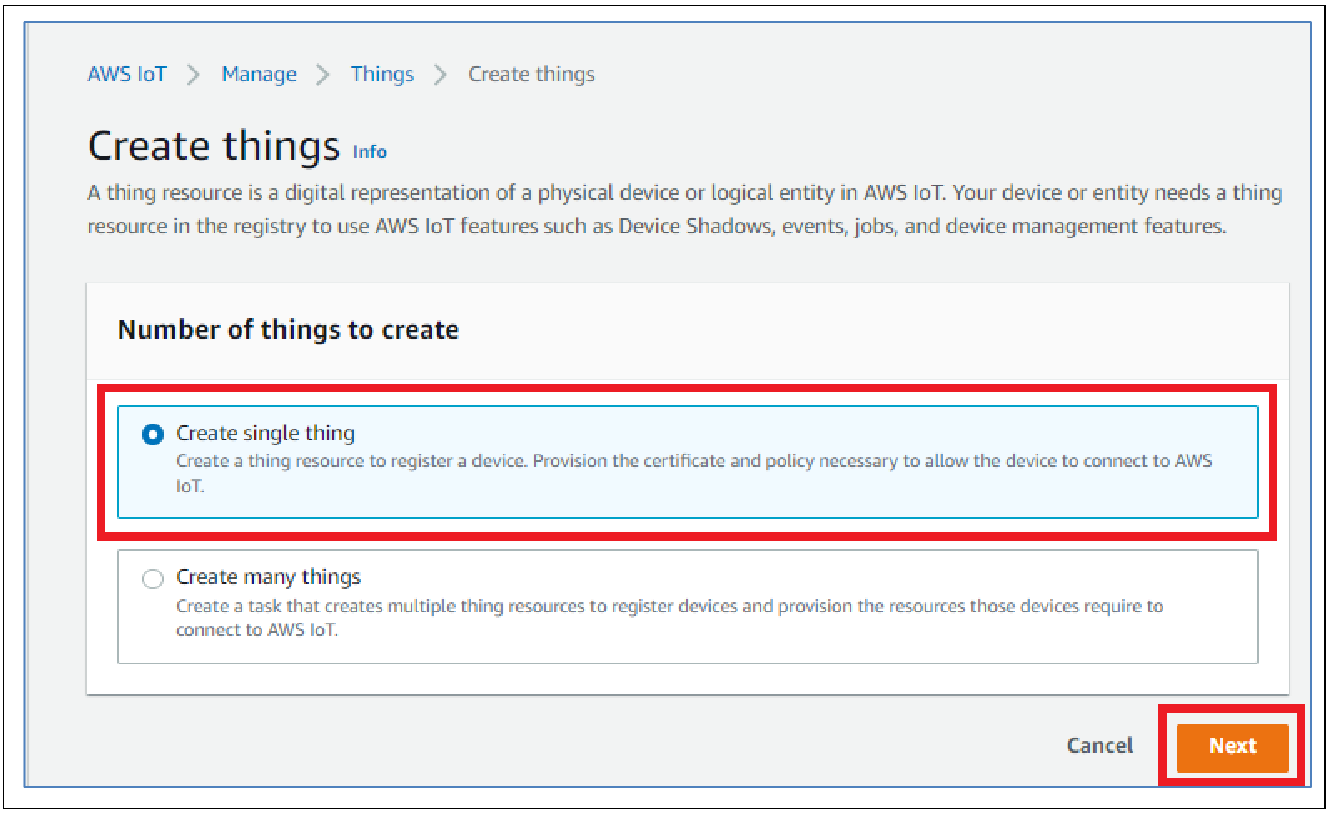


An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Filter things by: name, type, group, billing, or searchable attribute.

Figure 65. Creating the Things (1/5)

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



A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

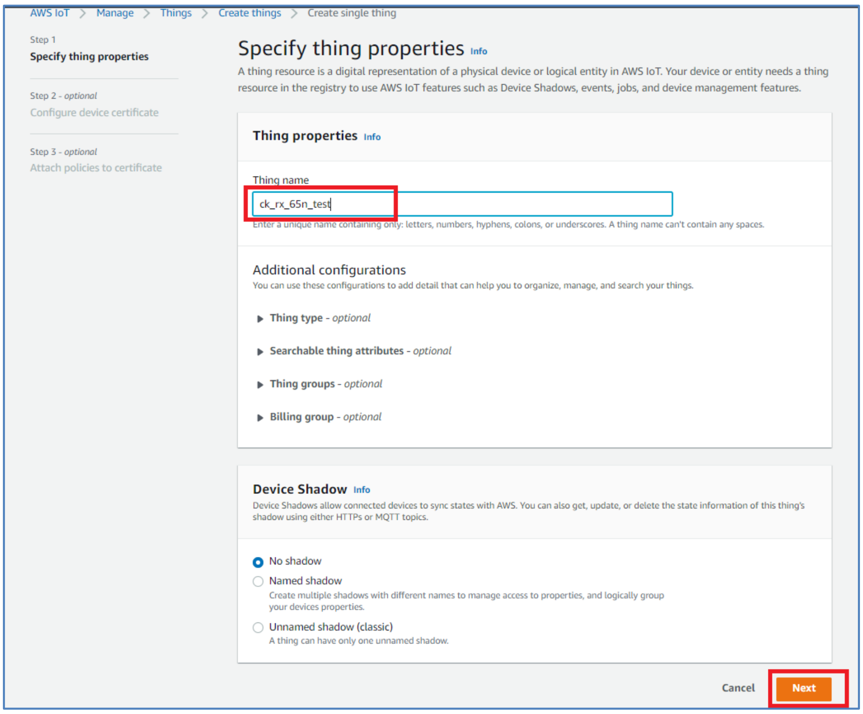
Number of things to create

● Create single thing
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

○ Create many things
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Figure 66. Creating the Things (2/5)

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



AWS IoT > Manage > Things > Create things > Create single thing

Step 1
Specify thing properties

Step 2 - *optional*
Configure device certificate

Step 3 - *optional*
Attach policies to certificate

Specify thing properties Info

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties Info

Thing name
ck_rx_65n_test
Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

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

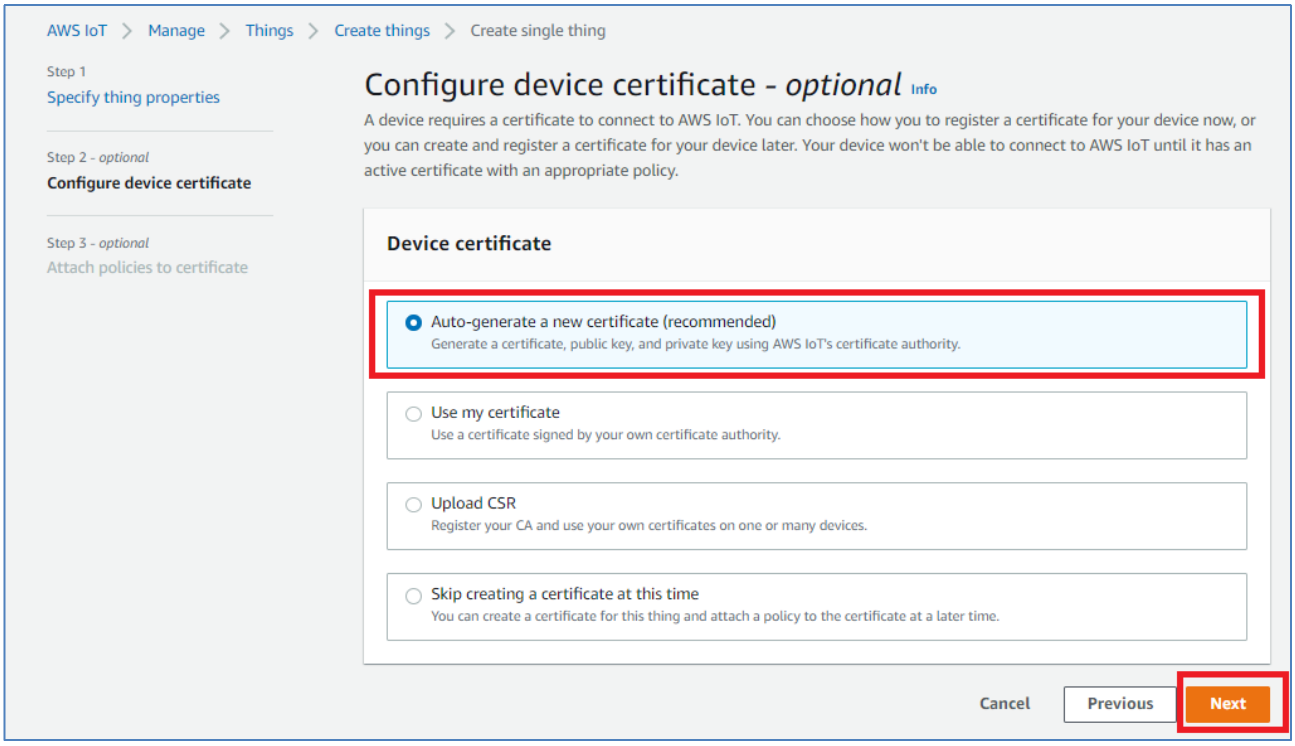
▶ **Thing type** - *optional*
▶ **Searchable thing attributes** - *optional*
▶ **Thing groups** - *optional*
▶ **Billing group** - *optional*

Device Shadow Info
Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

● No shadow
○ Named shadow
Create multiple shadows with different names to manage access to properties, and logically group your devices properties.
○ Unnamed shadow (classic)
A thing can have only one unnamed shadow.

Figure 67. Creating the Things (3/5)

Auto-generate a new certificate.



AWS IoT > Manage > Things > Create things > Create single thing

Step 1
Specify thing properties

Step 2 - *optional*
Configure device certificate

Step 3 - *optional*
Attach policies to certificate

Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how you to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

● Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

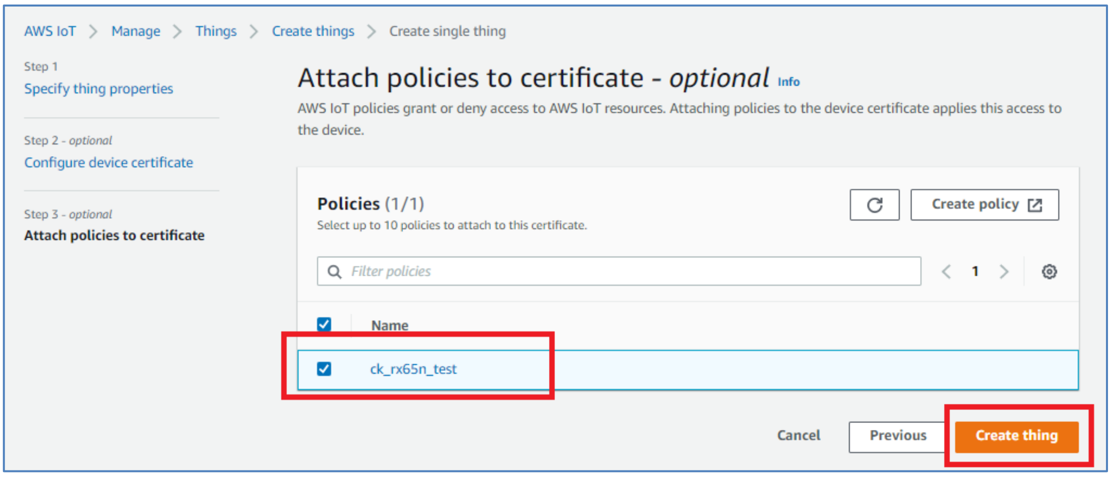
○ Use my certificate
Use a certificate signed by your own certificate authority.

○ Upload CSR
Register your CA and use your own certificates on one or many devices.

○ Skip creating a certificate at this time
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Figure 68. Creating the Things (4/5)

- Add a policy for your thing.



AWS IoT > Manage > Things > Create things > Create single thing

Step 1
Specify thing properties

Step 2 - *optional*
Configure device certificate

Step 3 - *optional*
Attach policies to certificate

Attach policies to certificate - *optional* Info

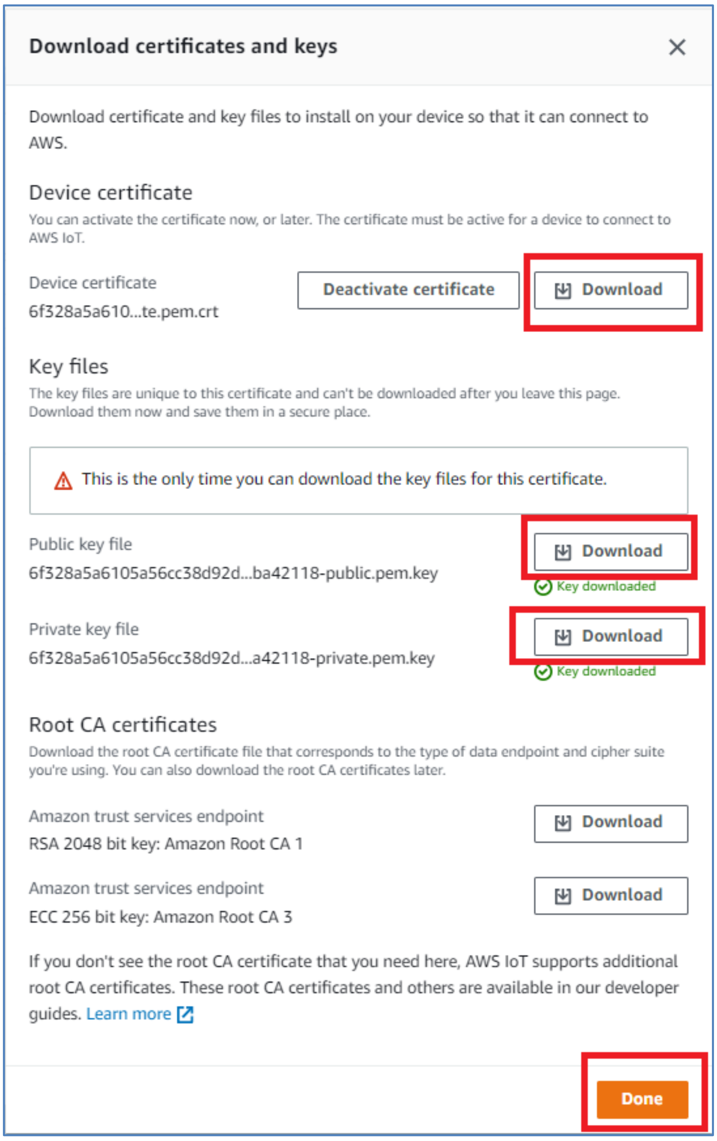
AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

Policies (1/1)
Select up to 10 policies to attach to this certificate.

☑	Name
☑	ck_rx65n_test

Figure 69. Creating the Things (5/5)

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



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
6f328a5a610...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file
6f328a5a6105a56cc38d92d...ba42118-public.pem.key

Key downloaded

Private key file
6f328a5a6105a56cc38d92d...a42118-private.pem.key

Key downloaded

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

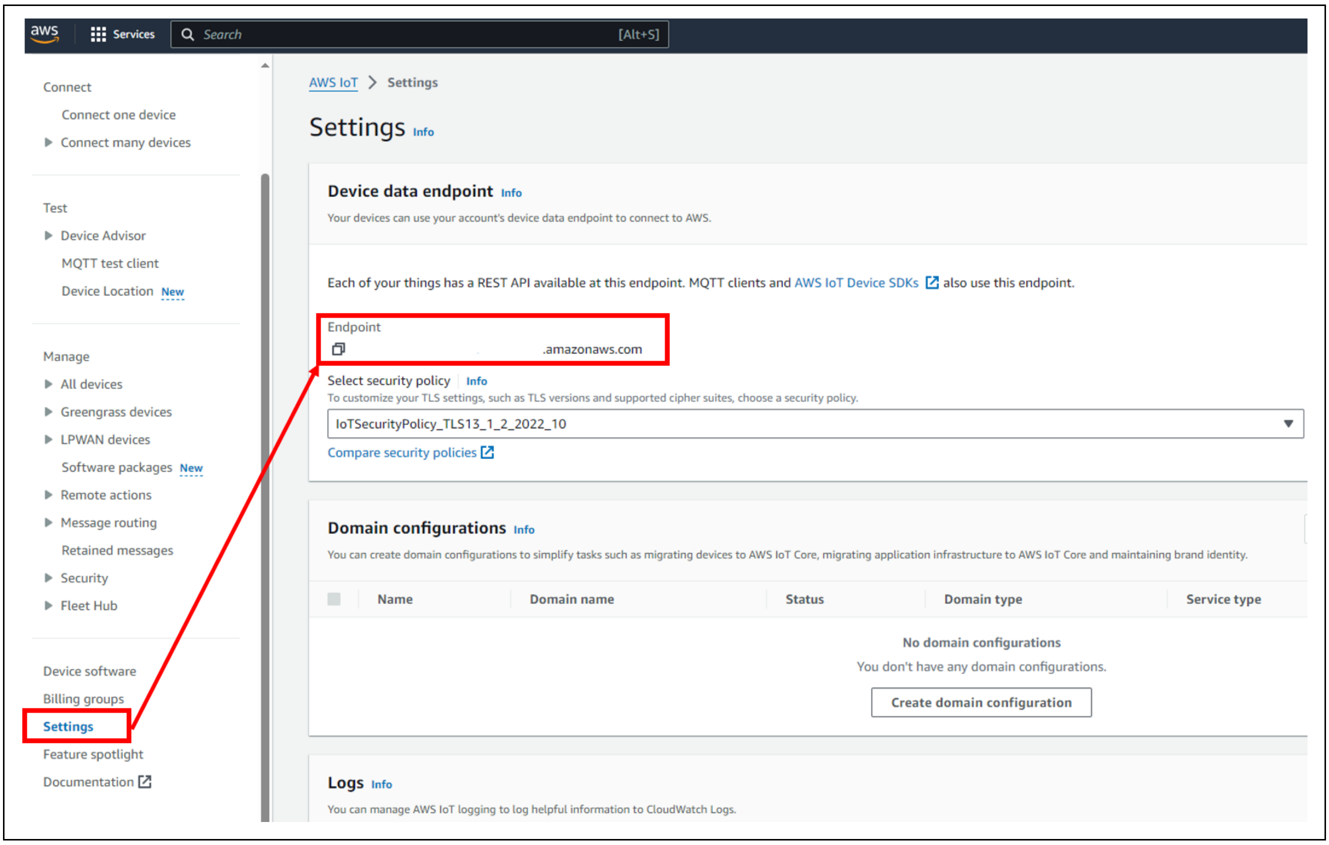
Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more

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

5.4.6 Check AWS IoT Endpoints

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



Settings

Device data endpoint

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint: .amazonaws.com

Select security policy

To customize your TLS settings, such as TLS versions and supported cipher suites, choose a security policy.

IoTSecurityPolicy_TLS13_1_2_2022_10

Compare security policies

Domain configurations

You can create domain configurations to simplify tasks such as migrating devices to AWS IoT Core, migrating application infrastructure to AWS IoT Core and maintaining brand identity.

Name	Domain name	Status	Domain type	Service type

No domain configurations

You don't have any domain configurations.

Logs

You can manage AWS IoT logging to log helpful information to CloudWatch Logs.

Figure 71. Check AWS IoT Endpoints

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

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

5.4.7 Running Application

After collecting Cloud Credentials, it includes:

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

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

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

5.5 Verifying the Application Project using AWS Dashboard and Renesas Dashboard

5.5.1 Subscribe to a Topic Message on the AWS IoT

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

Note: Wait for the board to get the IP address from the service provider upon successful Wi-Fi initialization, and for the board to resolve the DNS lookup for the endpoint. After the successful MQTT connection message on the console ***"An MQTT connection is established with <MQTT_BROKER_ENDPOINT>"***, the device is ready for Publishing and Subscription of messages.

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

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

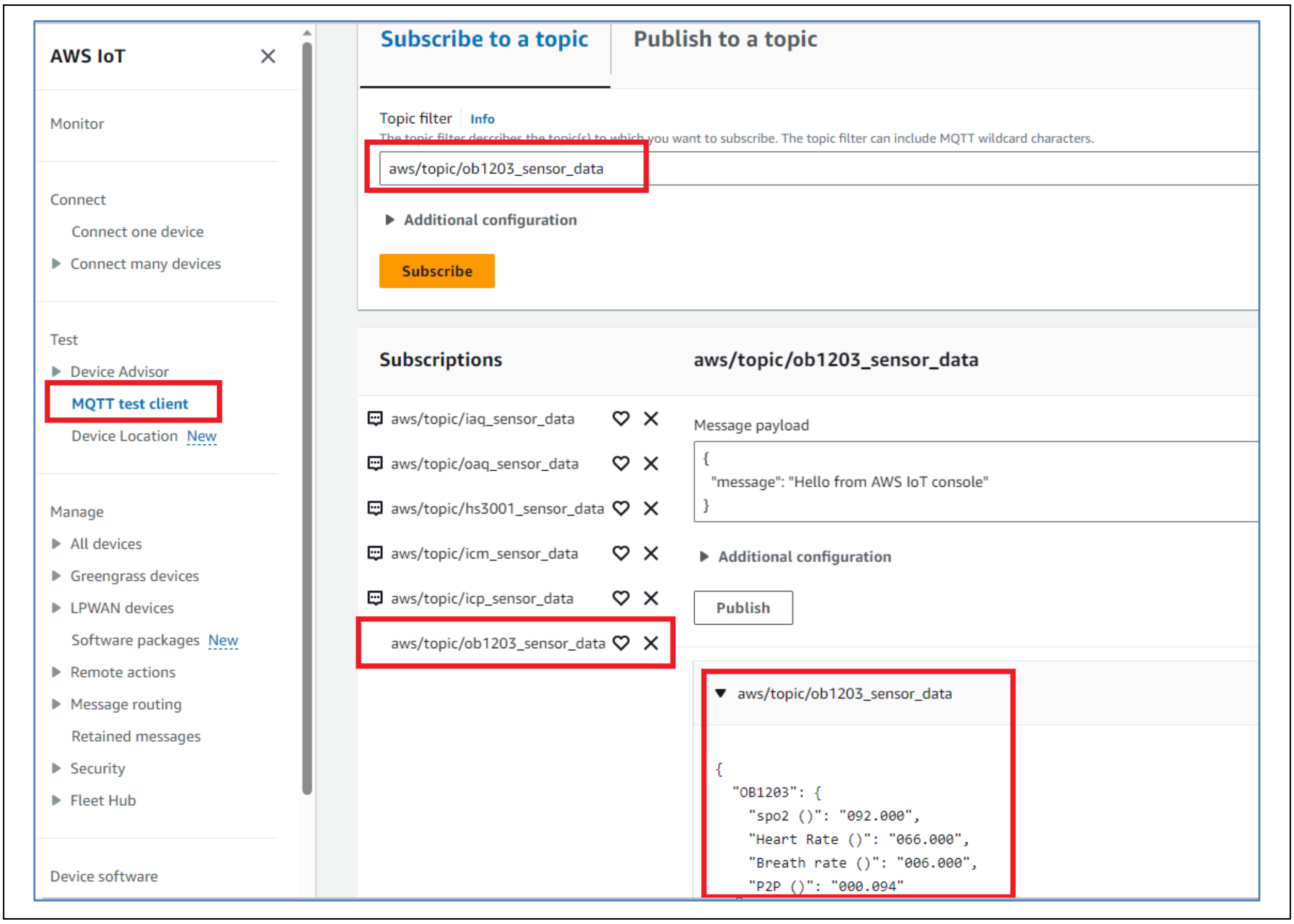
On the AWS cloud Dashboard side, go to IoT Core and select **Test**, then choose **MQTT test client**. Subscribe to a topic listed below one at a time. A sample snapshot of subscribing to the topics is shown below.

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

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

```
"aws/topic/iaq_sensor_data"
"aws/topic/oaq_sensor_data"
"aws/topic/hs3001_sensor_data"
"aws/topic/icm_sensor_data"
"aws/topic/icp_sensor_data"
"aws/topic/ob1203_sensor_data"
```

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



Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

aws/topic/ob1203_sensor_data

Subscriptions: 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

aws/topic/ob1203_sensor_data

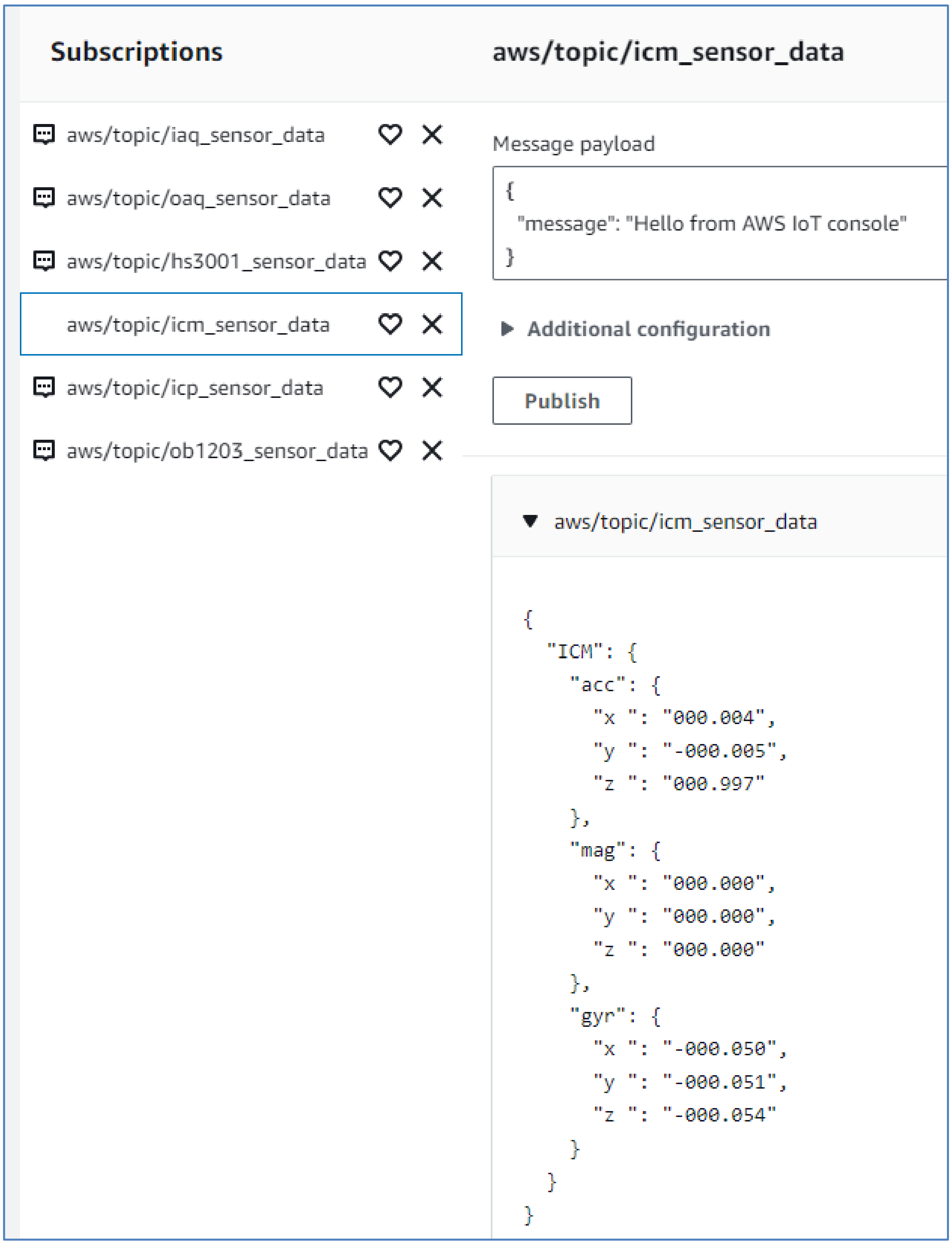
Message payload

```
{
  "message": "Hello from AWS IoT console"
}
```

aws/topic/ob1203_sensor_data

```
{
  "OB1203": {
    "spo2 ()": "092.000",
    "Heart Rate ()": "066.000",
    "Breath rate ()": "006.000",
    "P2P ()": "000.094"
```

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



Subscriptions

- 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

aws/topic/icm_sensor_data

Message payload

```
{
  "message": "Hello from AWS IoT console"
}
```

▶ Additional configuration

Publish

▼ aws/topic/icm_sensor_data

```
{
  "ICM": {
    "acc": {
      "x ": "000.004",
      "y ": "-000.005",
      "z ": "000.997"
    },
    "mag": {
      "x ": "000.000",
      "y ": "000.000",
      "z ": "000.000"
    },
    "gyr": {
      "x ": "-000.050",
      "y ": "-000.051",
      "z ": "-000.054"
    }
  }
}
```

Figure 73. Subscribed Messages on the AWS IoT Screen

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

5.5.2.1 With AWS Dashboard

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

If we publish the below data from AWS console

HS3001 temperature alerts:

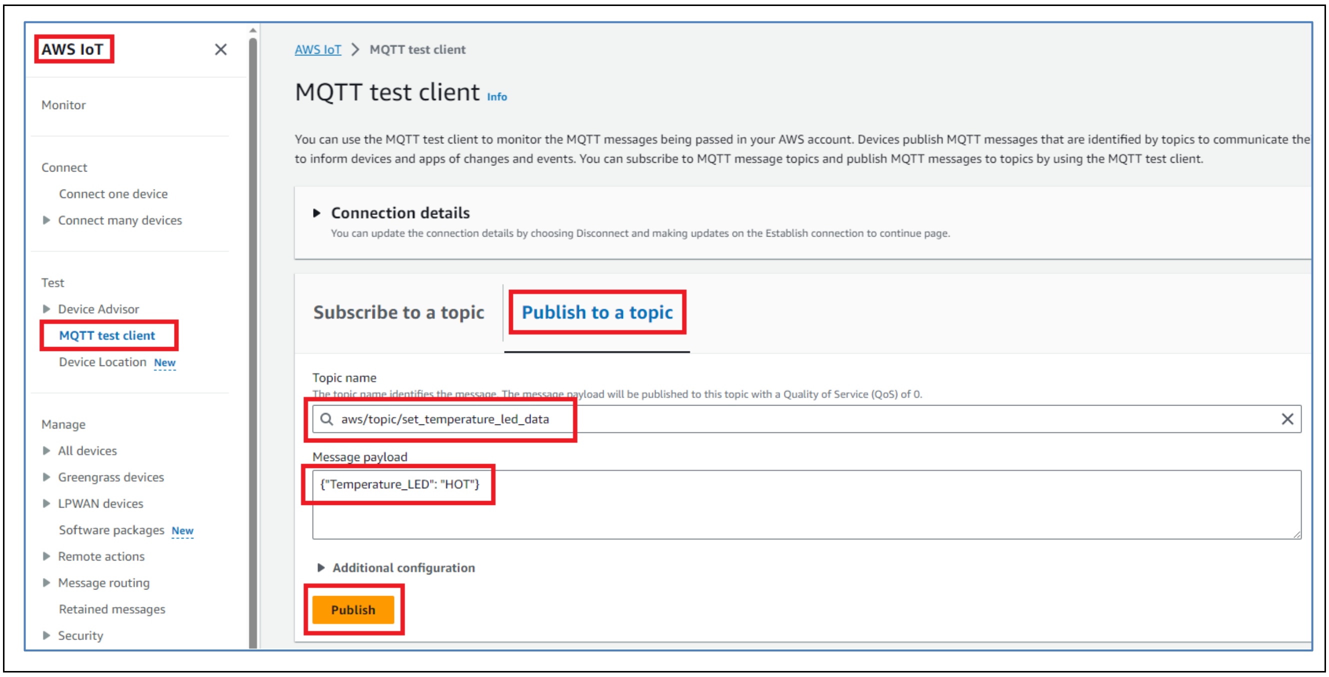
Based on temperature dashboard will send the alert messages to CK-RX65N v2 kit via below topic

Topic: `aws/topic/set_temperature_led_data`

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

Example:

Click **Test** > **MQTT test client**



MQTT test client

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate the to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

aws/topic/set_temperature_led_data

Message payload

{"Temperature_LED": "HOT"}

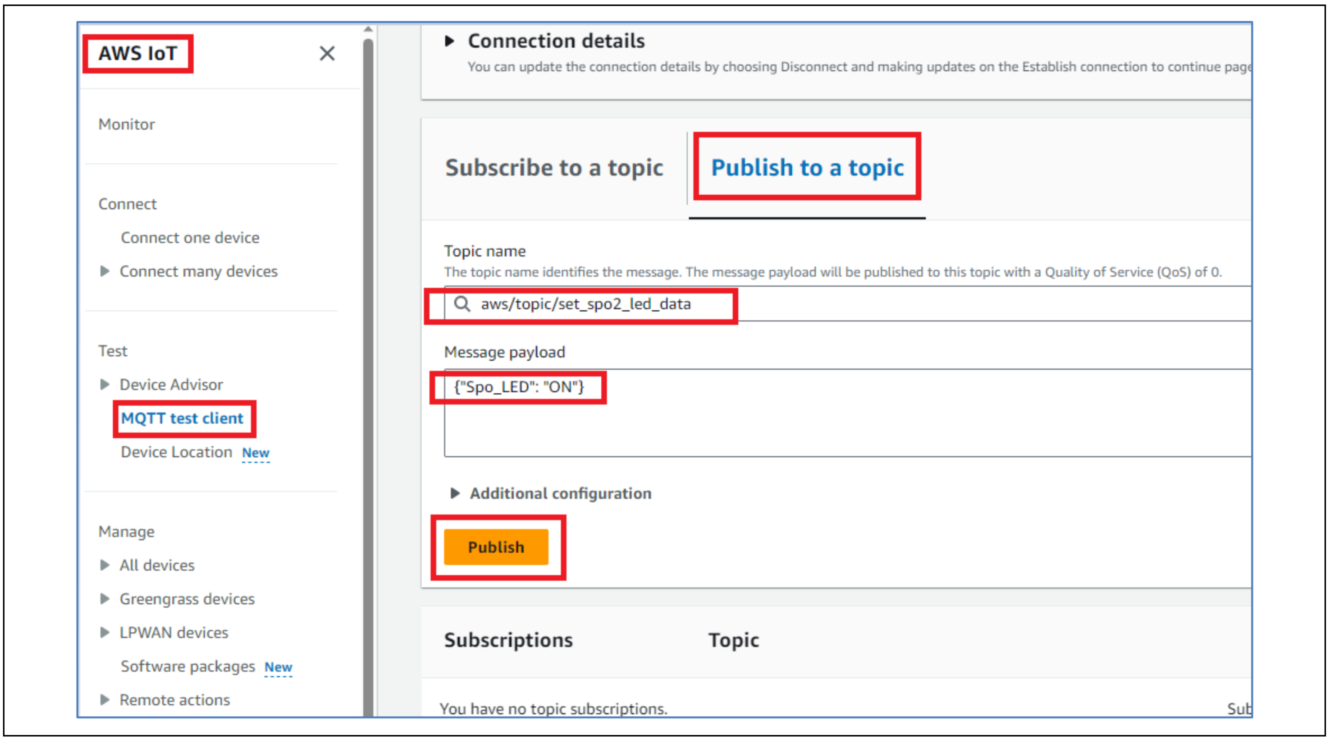
Figure 74. Publish the MQTT Message (1/2)

OB1203 SPO2 alerts:

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

Topic: `aws/topic/set_spo2_led_data`

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



Connection details

You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page

Subscribe to a topic | Publish to a topic

Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

aws/topic/set_spo2_led_data

Message payload

{"Spo_LED": "ON"}

Additional configuration

Publish

Subscriptions Topic

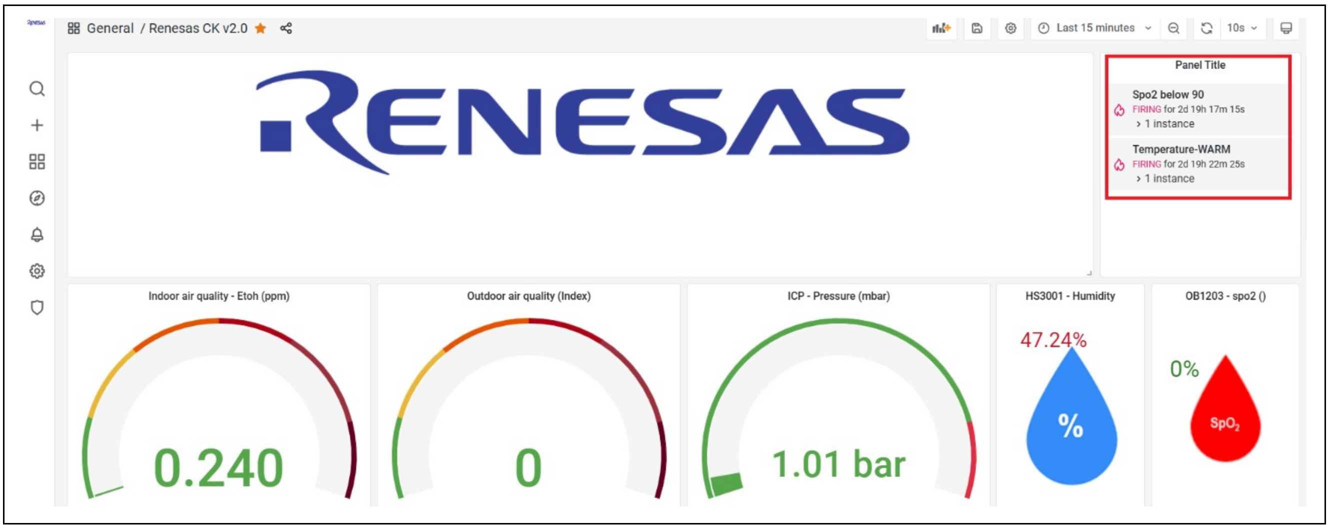
You have no topic subscriptions.

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

5.5.2.2 With Renesas Dashboard

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

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



Panel Title

Spo2 below 90
FIRING for 2d 19h 17m 15s
› 1 instance

Temperature-WARM
FIRING for 2d 19h 22m 25s
› 1 instance

Figure 76. Sensor Status Feedback

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

6. OTA over MQTT

6.1 Overview OTA

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

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

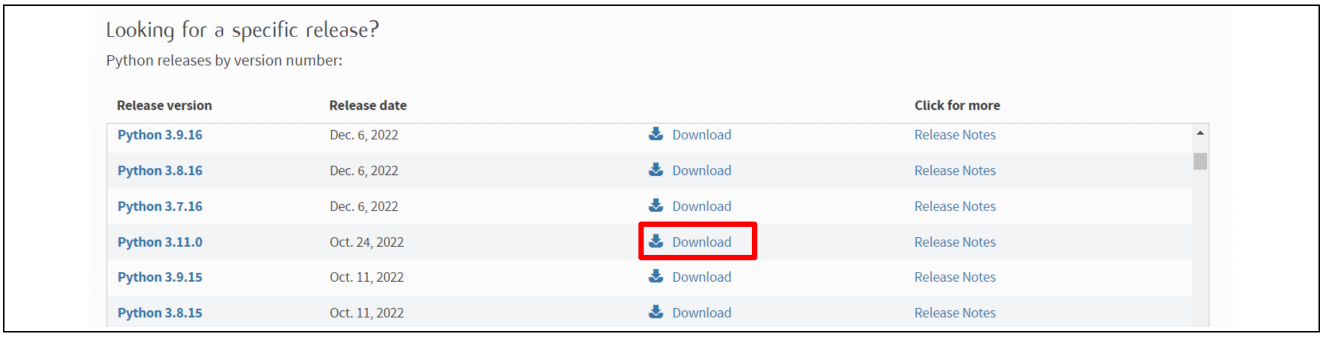
Please refer to the document about OTA: OTA-using-Amazon-Web-Services-in-RX65N-FreeRTOS-for-v202210.01-LTS-rx-1.1.0

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

6.2 Prerequisites

6.2.1 Installing Python

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



Looking for a specific release?

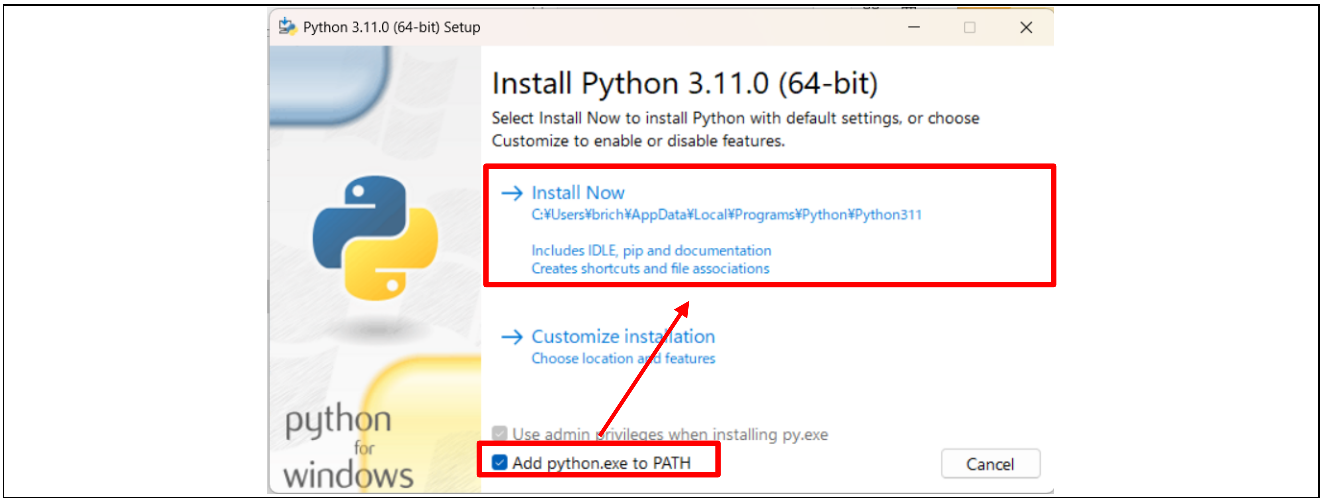
Python releases by version number:

Release version	Release date		Click for more
Python 3.9.16	Dec. 6, 2022	Download	Release Notes
Python 3.8.16	Dec. 6, 2022	Download	Release Notes
Python 3.7.16	Dec. 6, 2022	Download	Release Notes
Python 3.11.0	Oct. 24, 2022	Download	Release Notes
Python 3.9.15	Oct. 11, 2022	Download	Release Notes
Python 3.8.15	Oct. 11, 2022	Download	Release Notes

Figure 77. Python Download Website

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

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



Python 3.11.0 (64-bit) Setup

Install Python 3.11.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:¥Users¥brich¥AppData¥Local¥Programs¥Python¥Python311

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

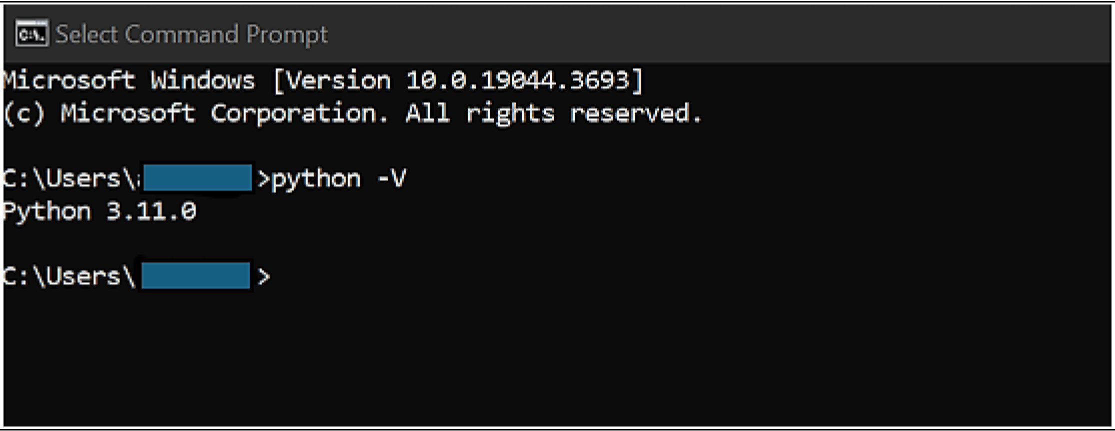
☑ Use admin privileges when installing py.exe

☑ Add python.exe to PATH

Figure 78. Python 3.11.0 installer

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

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

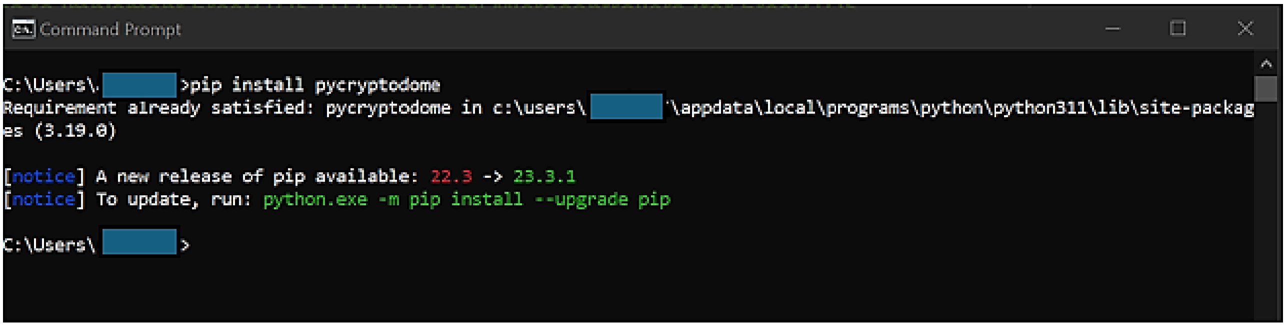


```
Microsoft Windows [Version 10.0.19044.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>python -V
Python 3.11.0

C:\Users\[redacted]>
```

Figure 79. Checking version of Python

4. Install the Python encryption library (pycryptodome)

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

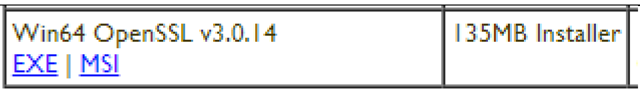


```
C:\Users\[redacted]>pip install pycryptodome
Requirement already satisfied: pycryptodome in c:\users\[redacted]\appdata\local\programs\python\python311\lib\site-packag
es (3.19.0)

[notice] A new release of pip available: 22.3 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\[redacted]>
```

Figure 80. Installing Python encryption library

6.2.2 Installing OpenSSL

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



Win64 OpenSSL v3.0.14 EXE | MSI	135MB Installer

Figure 81. OpenSSL Download Website

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

3. From the Start Menu, open the Win64 OpenSSL Command Prompt and confirm that OpenSSL is installed.

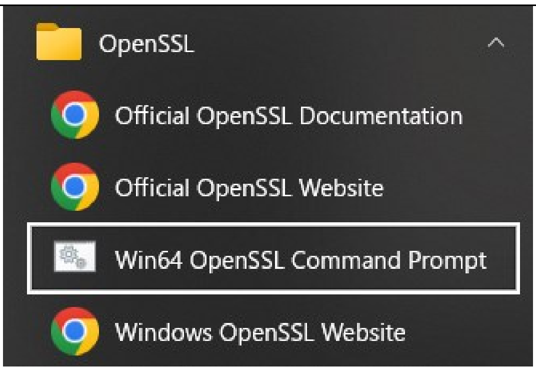


Figure 82. Open the Win64 OpenSSL Command Prompt

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

6.2.3 Installing Renesas Image Generator

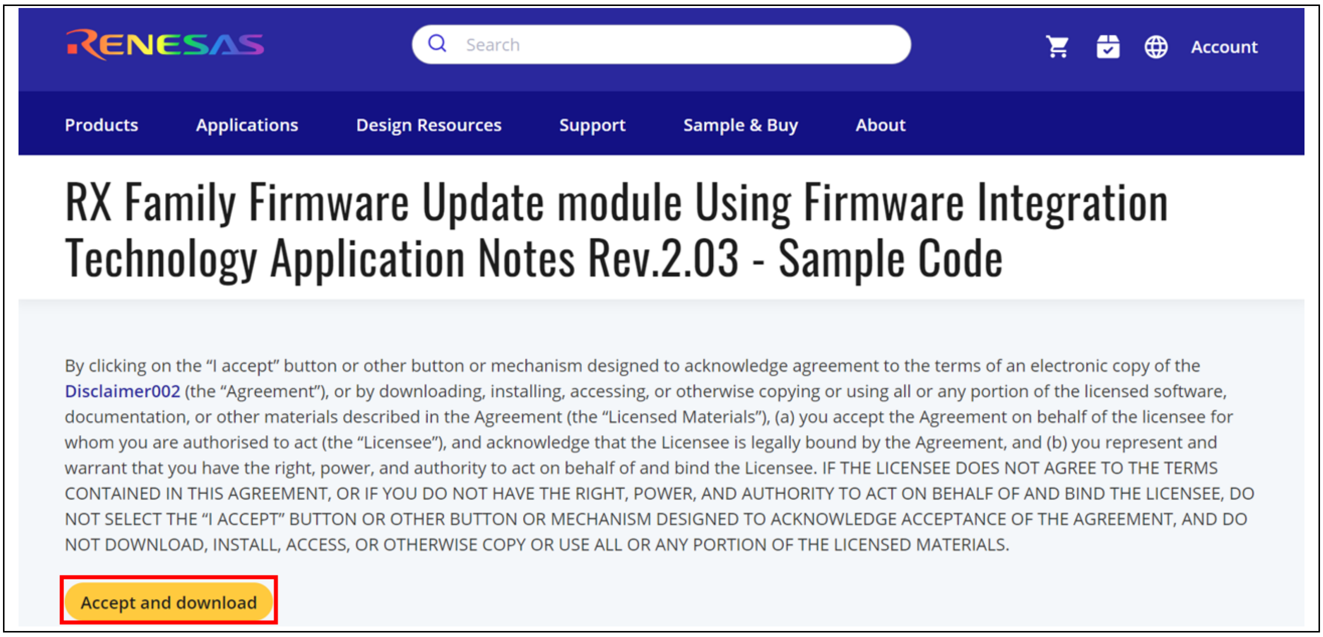
Renesas Image Generator is a tool that generates the firmware images used by the firmware update module. Renesas Image Generator can generate the following images for use by the firmware update module:

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

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

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

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



RX Family Firmware Update module Using Firmware Integration Technology Application Notes Rev.2.03 - Sample Code

By clicking on the "I accept" button or other button or mechanism designed to acknowledge agreement to the terms of an electronic copy of the Disclaimer002 (the "Agreement"), or by downloading, installing, accessing, or otherwise copying or using all or any portion of the licensed software, documentation, or other materials described in the Agreement (the "Licensed Materials"), (a) you accept the Agreement on behalf of the licensee for whom you are authorised to act (the "Licensee"), and acknowledge that the Licensee is legally bound by the Agreement, and (b) you represent and warrant that you have the right, power, and authority to act on behalf of and bind the Licensee. IF THE LICENSEE DOES NOT AGREE TO THE TERMS CONTAINED IN THIS AGREEMENT, OR IF YOU DO NOT HAVE THE RIGHT, POWER, AND AUTHORITY TO ACT ON BEHALF OF AND BIND THE LICENSEE, DO NOT SELECT THE "I ACCEPT" BUTTON OR OTHER BUTTON OR MECHANISM DESIGNED TO ACKNOWLEDGE ACCEPTANCE OF THE AGREEMENT, AND DO NOT DOWNLOAD, INSTALL, ACCESS, OR OTHERWISE COPY OR USE ALL OR ANY PORTION OF THE LICENSED MATERIALS.

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

2. Extract the downloaded firmware update module.

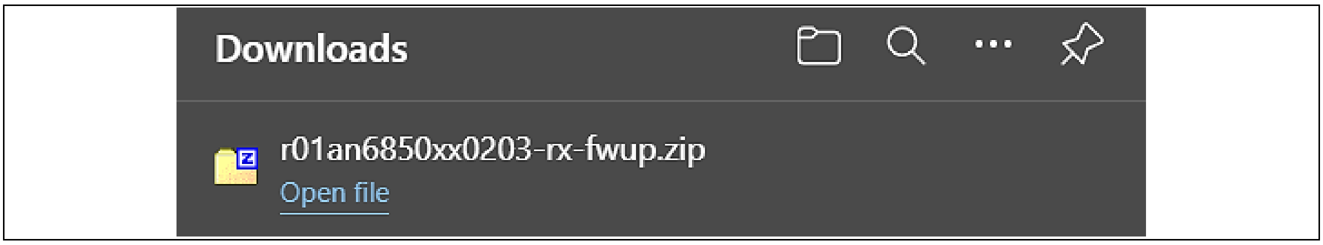
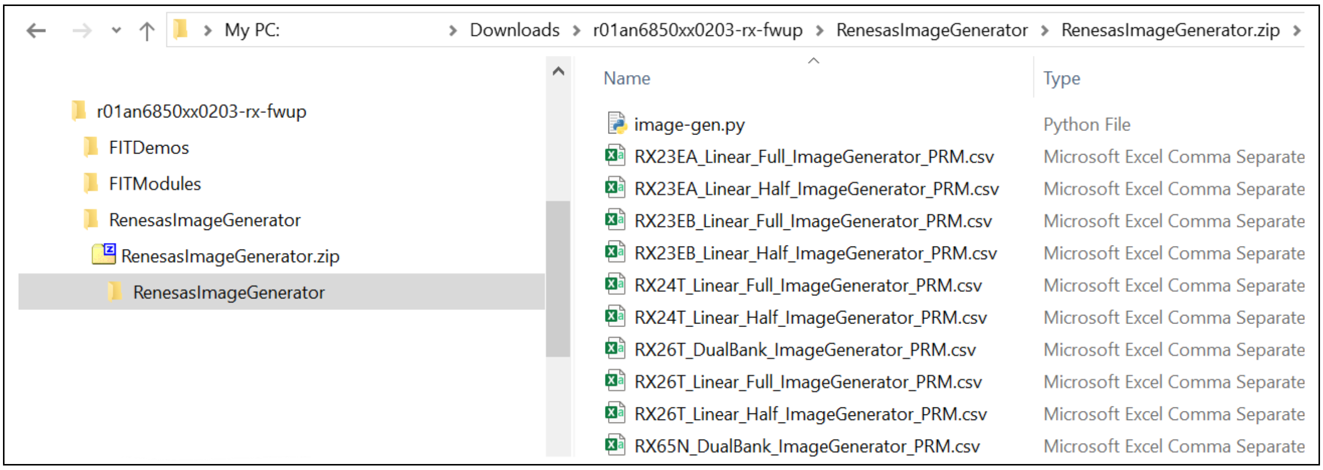


Figure 84. Renesas Image Generator Downloading (2/2)

3. Extract Renesas Image Generator.

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



My PC: > Downloads > r01an6850xx0203-rx-fwup > RenesasImageGenerator > RenesasImageGenerator.zip >

- r01an6850xx0203-rx-fwup
  - FITDemos
  - FITModules
  - RenesasImageGenerator
    - RenesasImageGenerator.zip
      - RenesasImageGenerator

Name	Type
image-gen.py	Python File
RX23EA_Linear_Full_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX23EA_Linear_Half_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX23EB_Linear_Full_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX23EB_Linear_Half_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX24T_Linear_Full_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX24T_Linear_Half_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX26T_DualBank_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX26T_Linear_Full_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX26T_Linear_Half_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate
RX65N_DualBank_ImageGenerator_PRM.csv	Microsoft Excel Comma Separate

Figure 85. Renesas Image Generator package

6.3 Setting up AWS for OTA

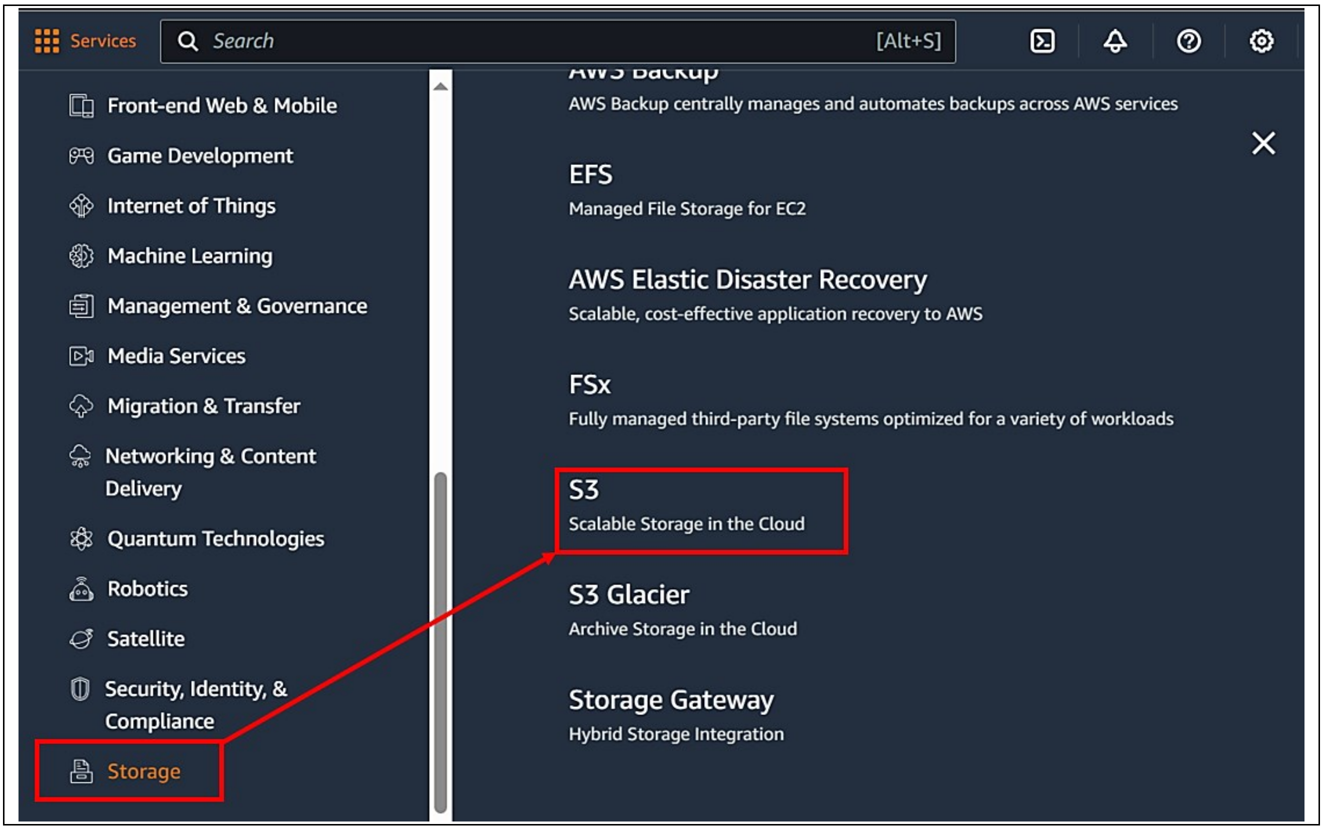
6.3.1 Register your Device in AWS

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

6.3.2 Creating an Amazon S3 bucket

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

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



AWS Backup
AWS Backup centrally manages and automates backups across AWS services

EFS
Managed File Storage for EC2

AWS Elastic Disaster Recovery
Scalable, cost-effective application recovery to AWS

FSx
Fully managed third-party file systems optimized for a variety of workloads

S3
Scalable Storage in the Cloud

S3 Glacier
Archive Storage in the Cloud

Storage Gateway
Hybrid Storage Integration

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

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

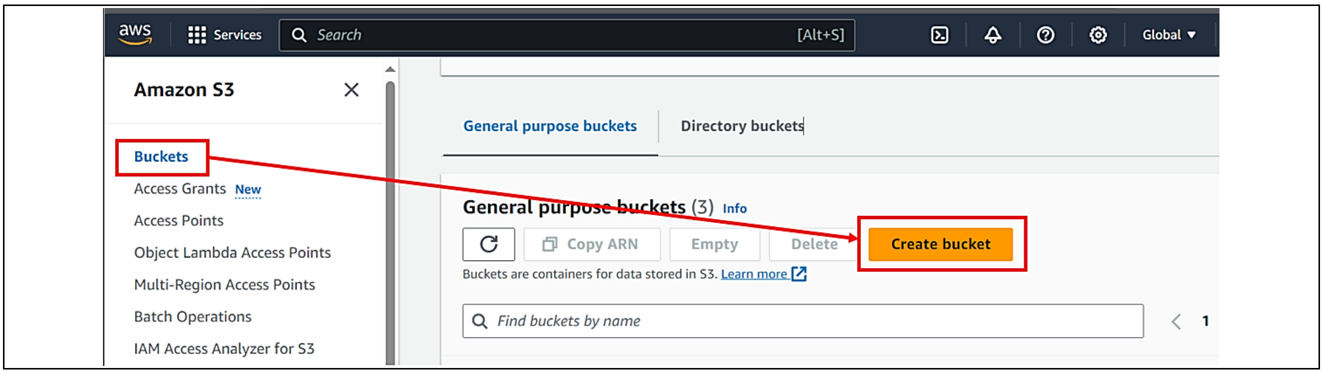
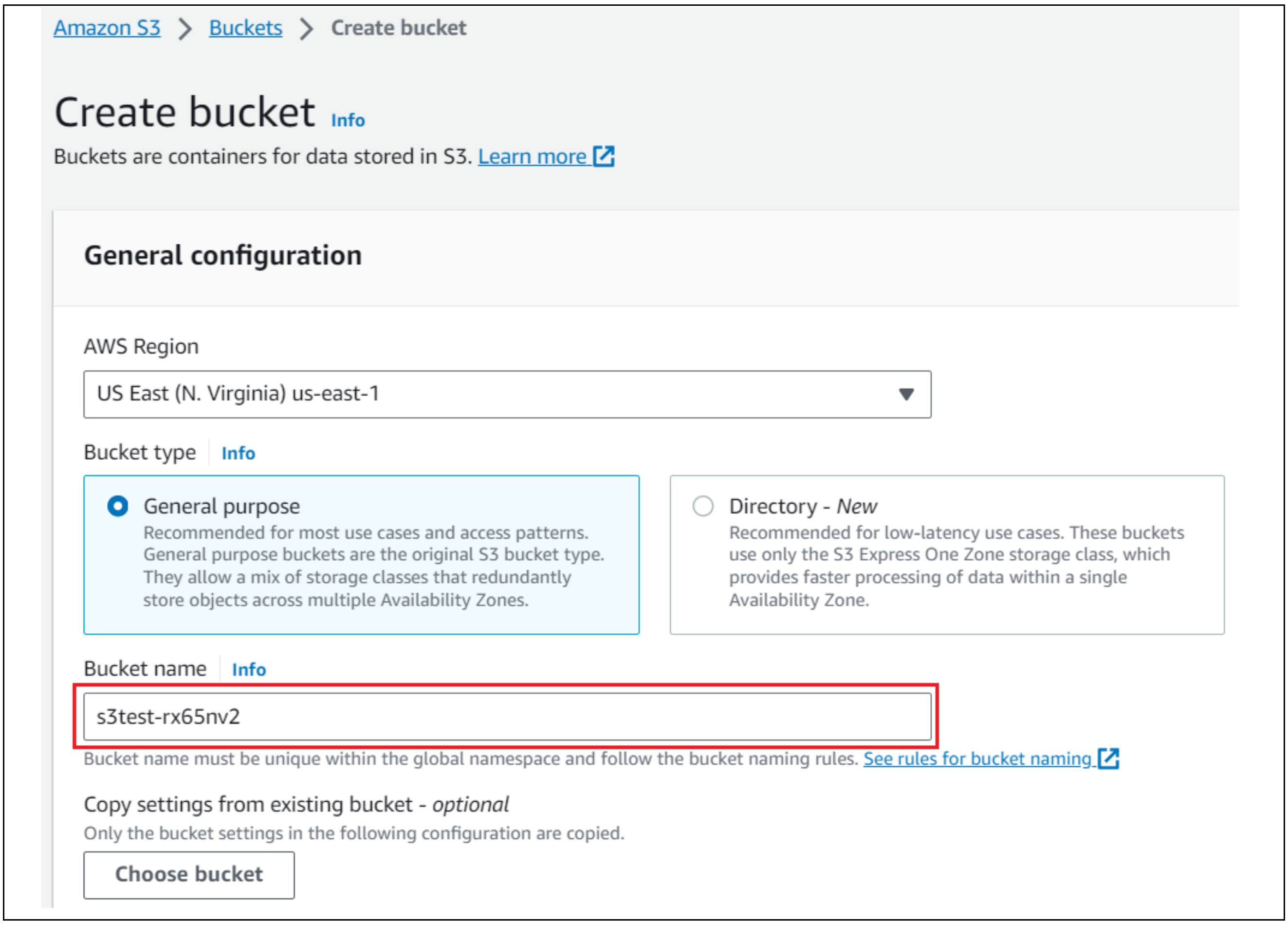


Figure 87. Create AWS S3 bucket (2/2)

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



Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type Info

● General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

○ Directory - *New*
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

s3test-rx65nv2

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

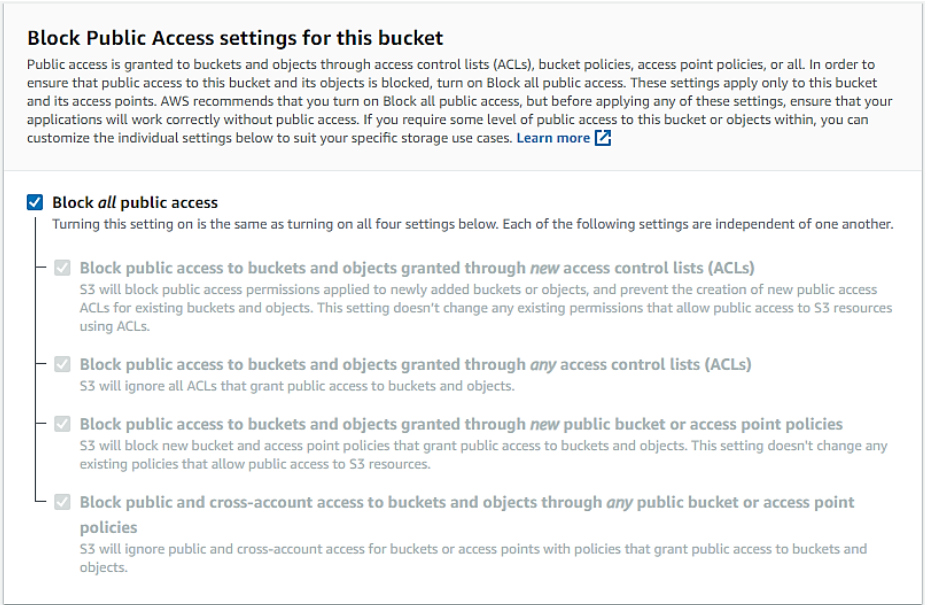
Figure 88. Naming for bucket

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

4. Create the bucket.

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

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



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

☑ Block *all* public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☑ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☑ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

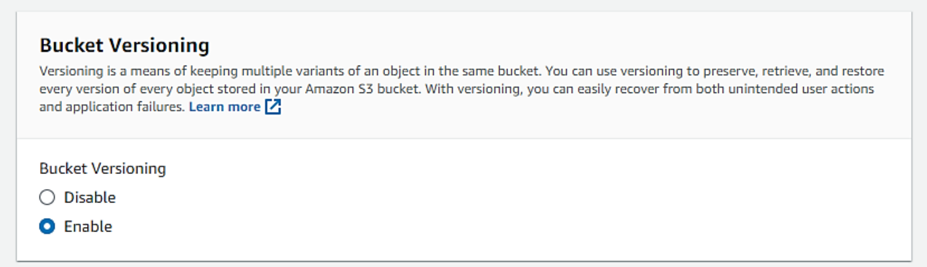
☑ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☑ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Figure 89. Bucket setting (1/2)



Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

○ Disable

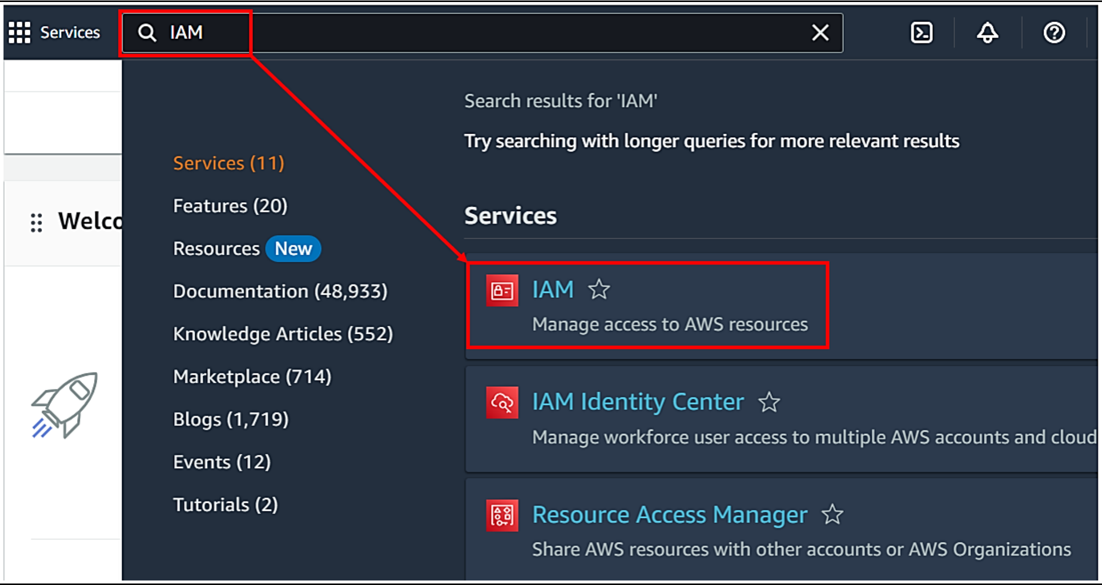
● Enable

Figure 90. Bucket setting (2/2)

6.3.3 Allocating OTA execution permission to IAM users

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

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



Search results for 'IAM'

Try searching with longer queries for more relevant results

Services

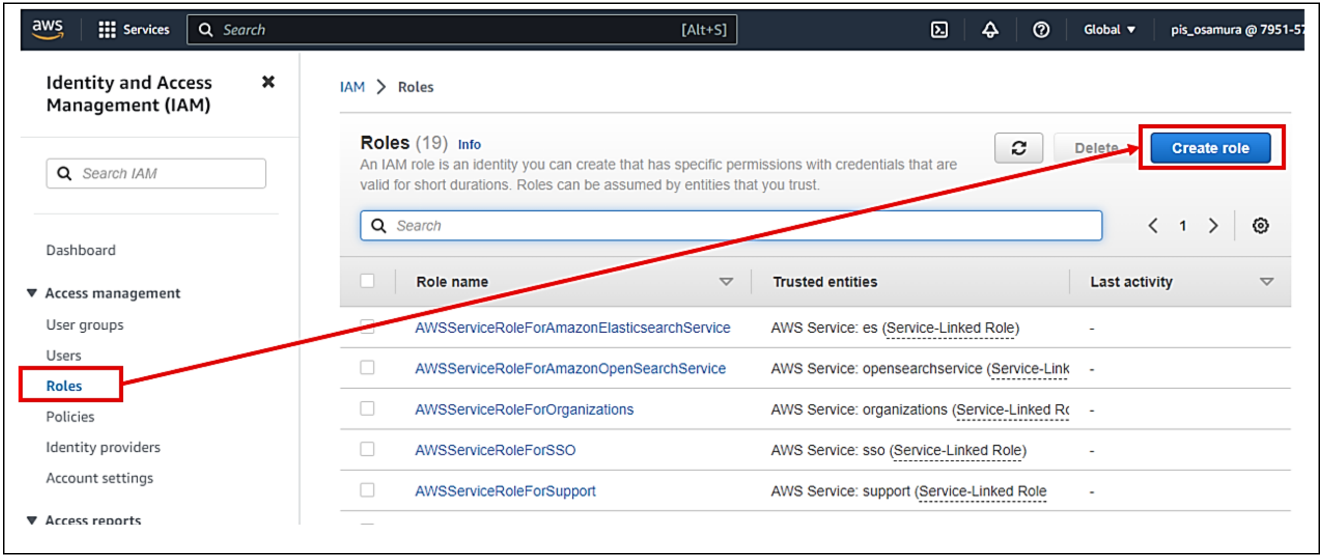
IAM ☆
Manage access to AWS resources

IAM Identity Center ☆
Manage workforce user access to multiple AWS accounts and cloud

Resource Access Manager ☆
Share AWS resources with other accounts or AWS Organizations

Figure 91. IAM search box

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



Roles (19)

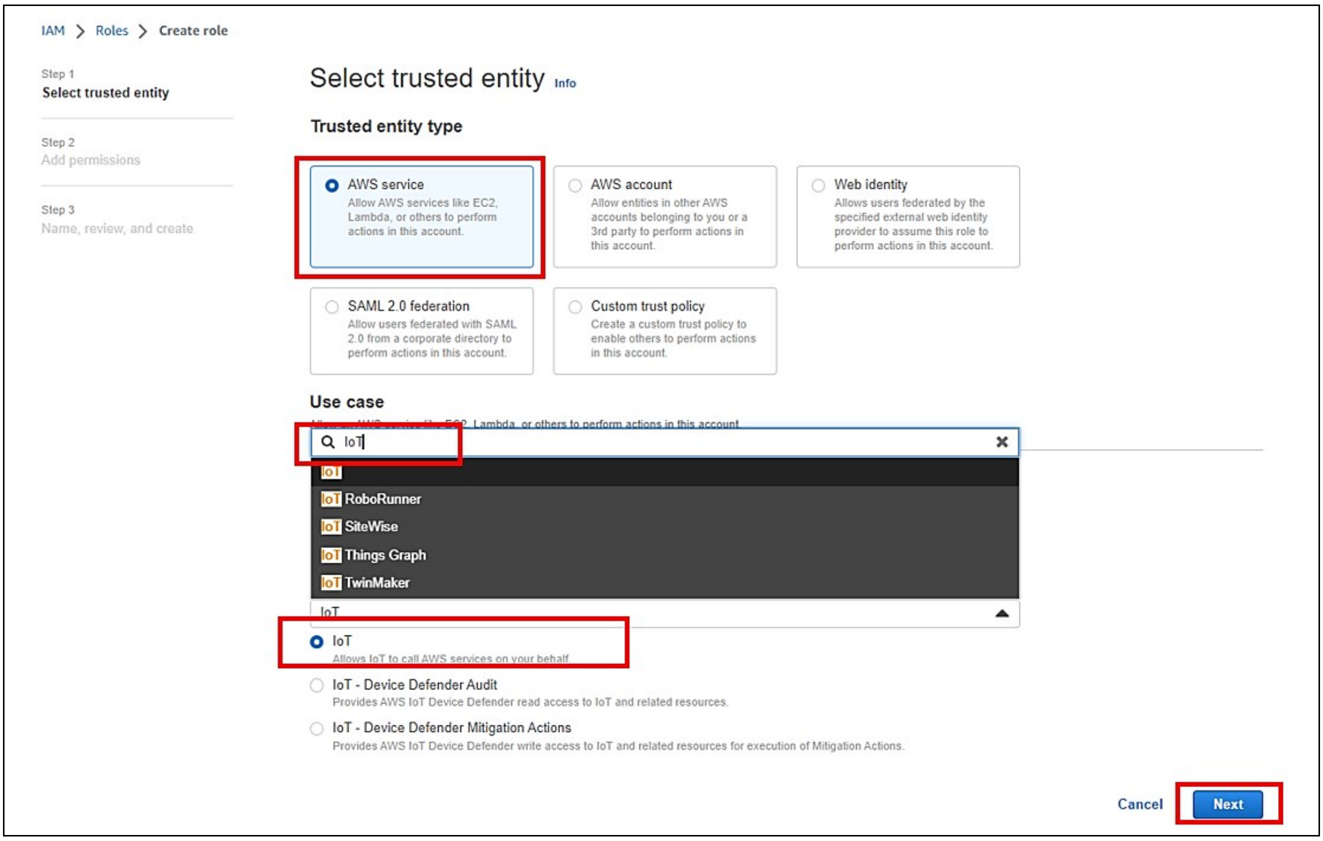
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

□	Role name	Trusted entities	Last activity
□	AWSServiceRoleForAmazonElasticsearchService	AWS Service: es (Service-Linked Role)	-
□	AWSServiceRoleForAmazonOpenSearchService	AWS Service: opensearchservice (Service-Link	-
□	AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Rc	-
□	AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	-
□	AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role	-

Figure 92. Creating a role

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

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



Select trusted entity

Trusted entity type

● AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

○ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

IoT

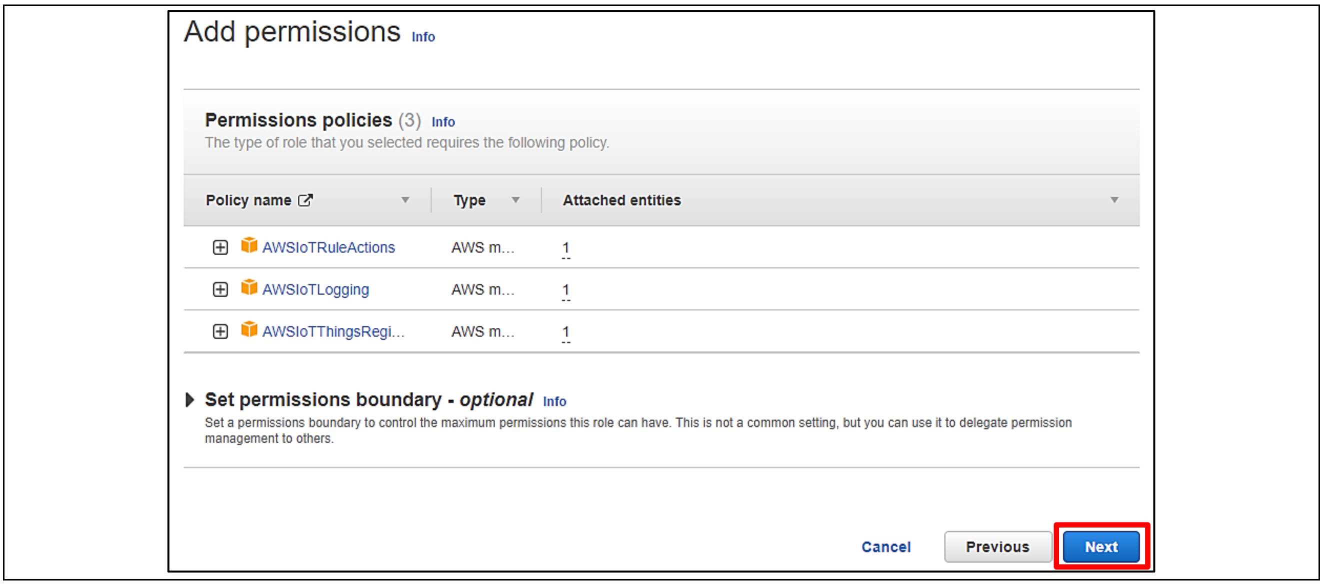
● IoT
Allows IoT to call AWS services on your behalf.

○ IoT - Device Defender Audit
Provides AWS IoT Device Defender read access to IoT and related resources.

○ IoT - Device Defender Mitigation Actions
Provides AWS IoT Device Defender write access to IoT and related resources for execution of Mitigation Actions.

Figure 93. Selecting trusted entity

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



Add permissions Info

Permissions policies (3) Info

The type of role that you selected requires the following policy.

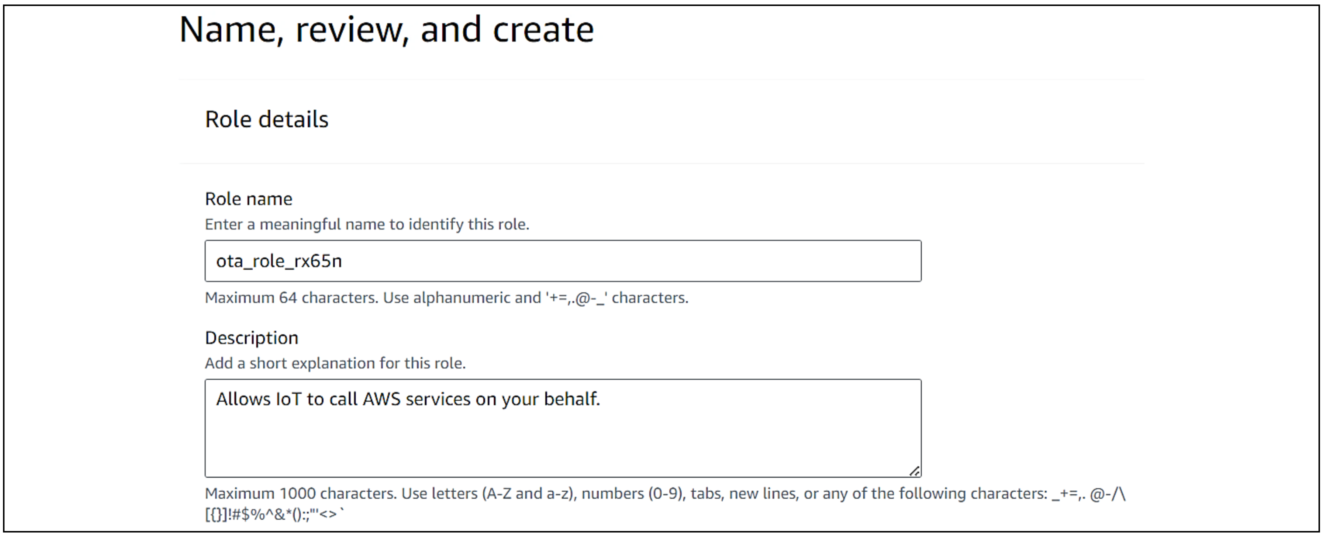
Policy name	Type	Attached entities
AWSIoTRuleActions	AWS m...	1
AWSIoTLogging	AWS m...	1
AWSIoTThingsRegi...	AWS m...	1

Set permissions boundary - *optional* Info

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.

Figure 94. Add Permissions for Role

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



Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

ota_role_rx65n

Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description

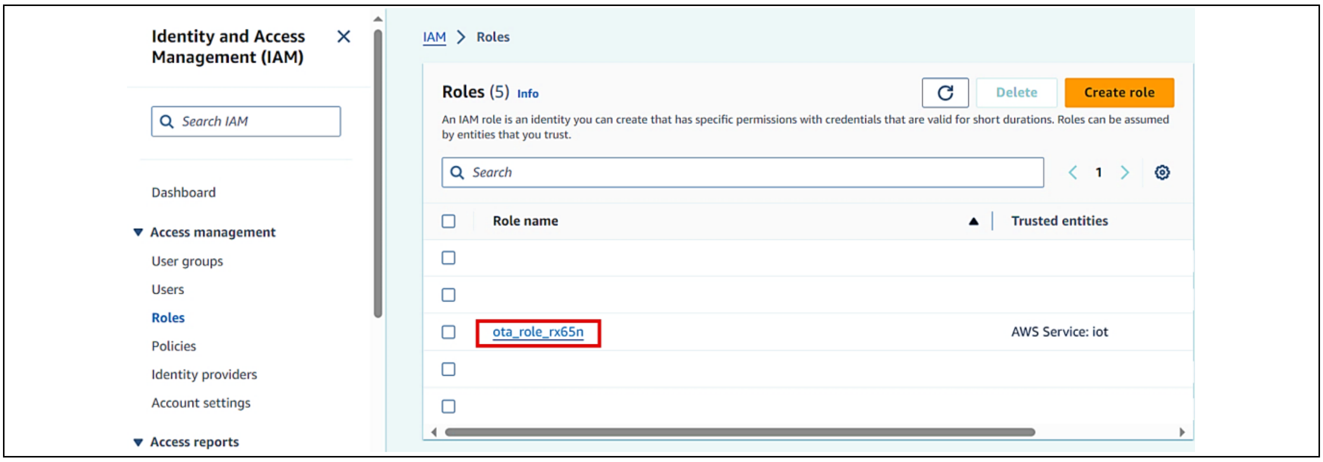
Add a short explanation for this role.

Allows IoT to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^&*();:"'<>`

Figure 95. Naming for Created Role

6. Click on the role you created.



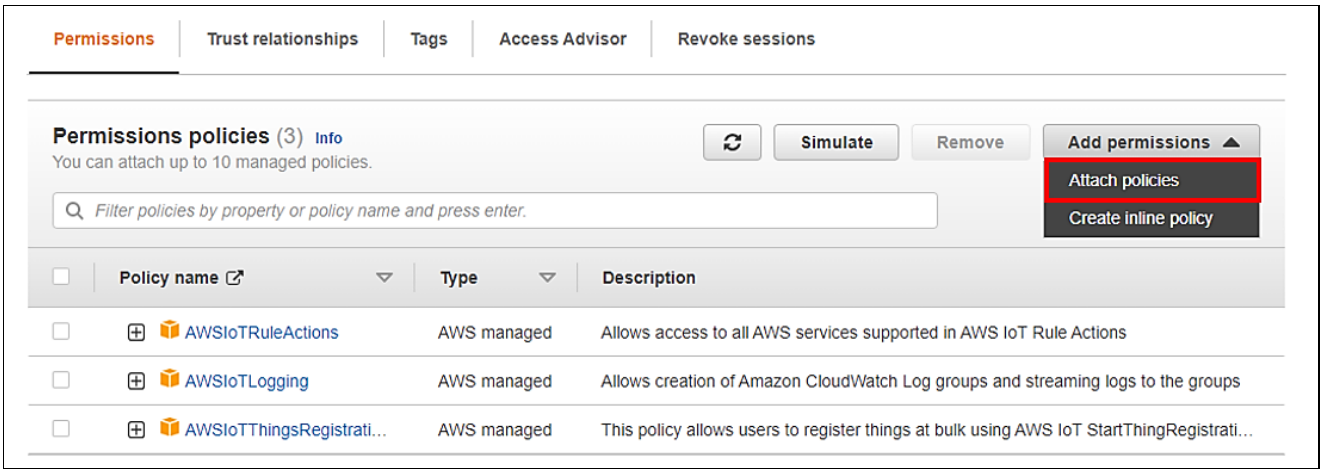
Roles (5) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

	Role name	Trusted entities
□		
□		
□	ota_role_rx65n	AWS Service: iot
□		
□		

Figure 96. Created Role

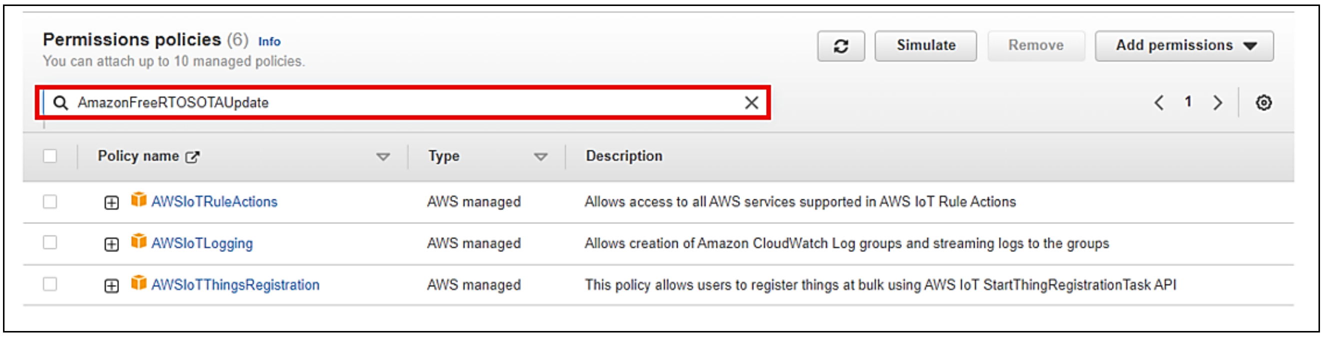
7. Select Attach policies.


Permissions
Trust relationships
Tags
Access Advisor
Revoke sessions
Permissions policies (3) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Attach policies
Create inline policy
Filter policies by property or policy name and press enter.

Policy name	Type	Description
AWSIoTRuleActions	AWS managed	Allows access to all AWS services supported in AWS IoT Rule Actions
AWSIoTLogging	AWS managed	Allows creation of Amazon CloudWatch Log groups and streaming logs to the groups
AWSIoTThingsRegistrati...	AWS managed	This policy allows users to register things at bulk using AWS IoT StartThingRegistrati...

Figure 97. Add permission (1/3)

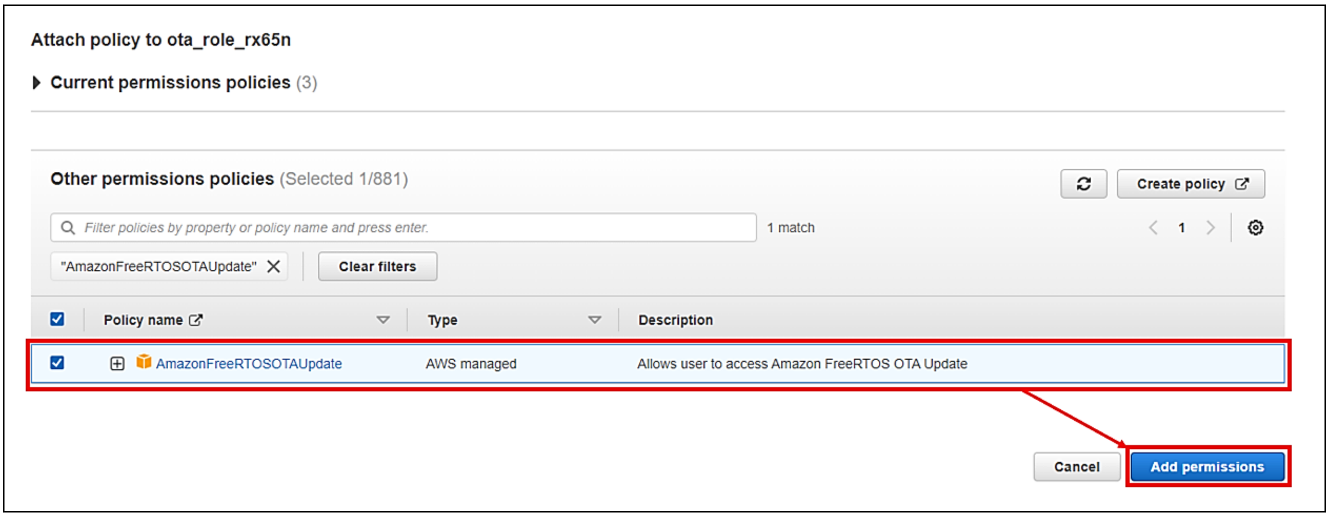
8. Enter AmazonFreeRTOSOTAUpdate in the **Permissions policies** search box, and then press the **Enter** key.


Permissions policies (6) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
AmazonFreeRTOSOTAUpdate
1

Policy name	Type	Description
AWSIoTRuleActions	AWS managed	Allows access to all AWS services supported in AWS IoT Rule Actions
AWSIoTLogging	AWS managed	Allows creation of Amazon CloudWatch Log groups and streaming logs to the groups
AWSIoTThingsRegistration	AWS managed	This policy allows users to register things at bulk using AWS IoT StartThingRegistrationTask API

Figure 98. Add permission (2/3)

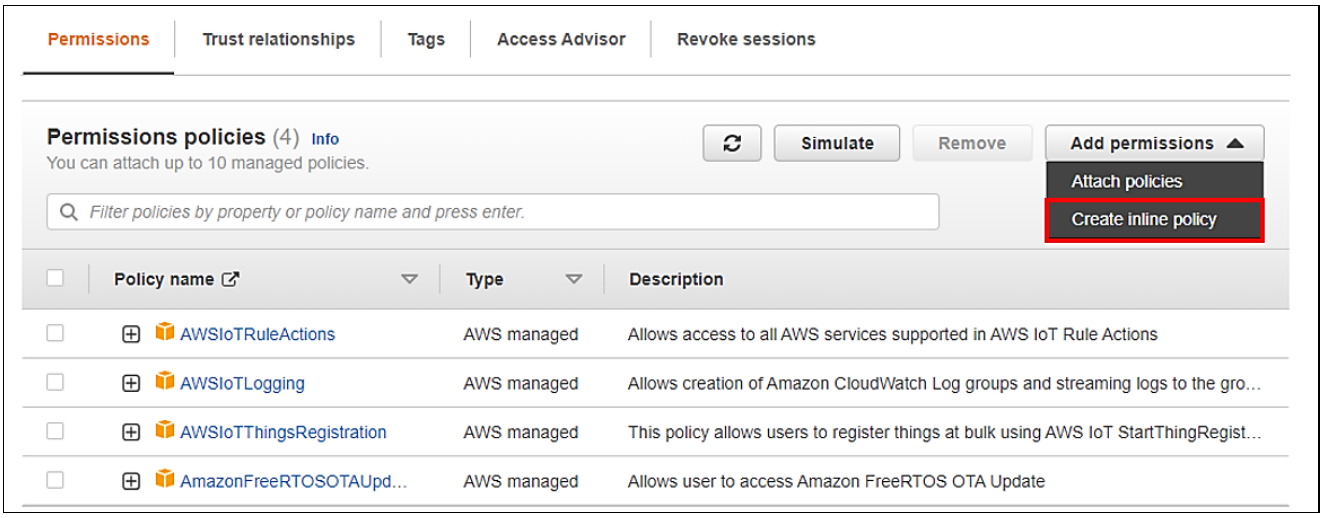
9. Select the check box beside the AmazonFreeRTOSOTAUpdate policy, and then click the **Add permissions** button.


Attach policy to ota_role_rx65n
Current permissions policies (3)
Other permissions policies (Selected 1/881)
Create policy
Filter policies by property or policy name and press enter.
1 match
1
"AmazonFreeRTOSOTAUpdate"
Clear filters
Cancel
Add permissions

	Policy name	Type	Description
☑	AmazonFreeRTOSOTAUpdate	AWS managed	Allows user to access Amazon FreeRTOS OTA Update

Figure 99. Add permission (3/3)

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



Permissions policies (4) Info

You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

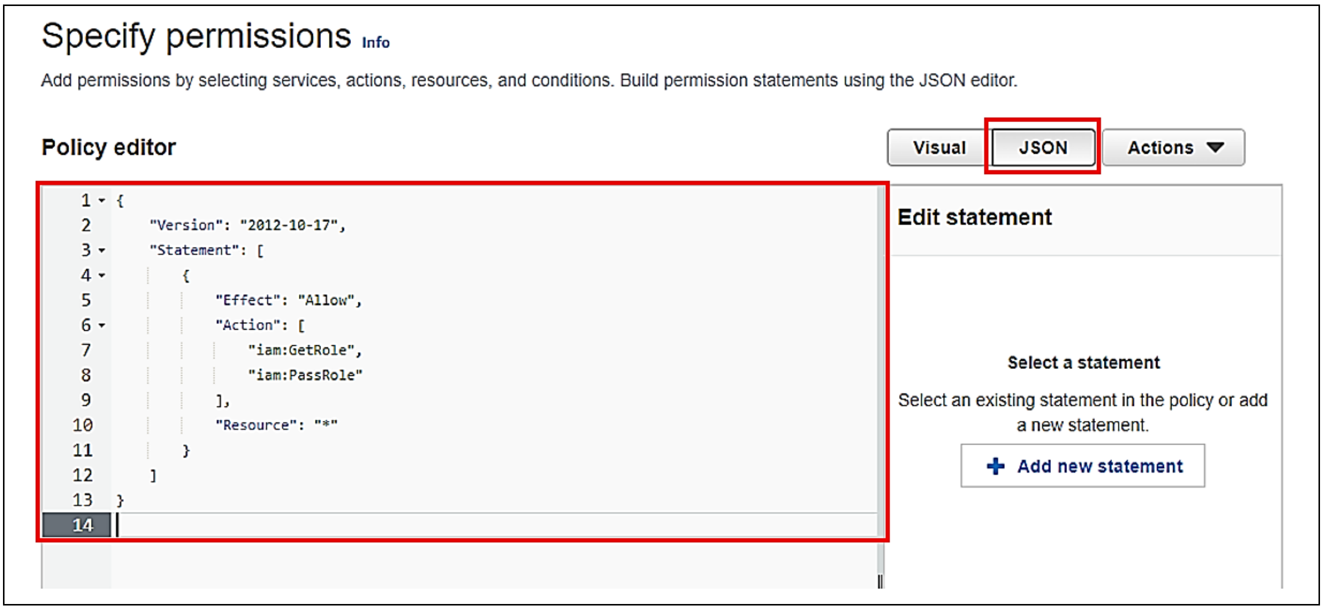
	Policy name	Type	Description
□	AWSIoTRuleActions	AWS managed	Allows access to all AWS services supported in AWS IoT Rule Actions
□	AWSIoTLogging	AWS managed	Allows creation of Amazon CloudWatch Log groups and streaming logs to the gro…
□	AWSIoTThingsRegistration	AWS managed	This policy allows users to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

Figure 100. Create inline policy (1/3)

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

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

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



Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

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

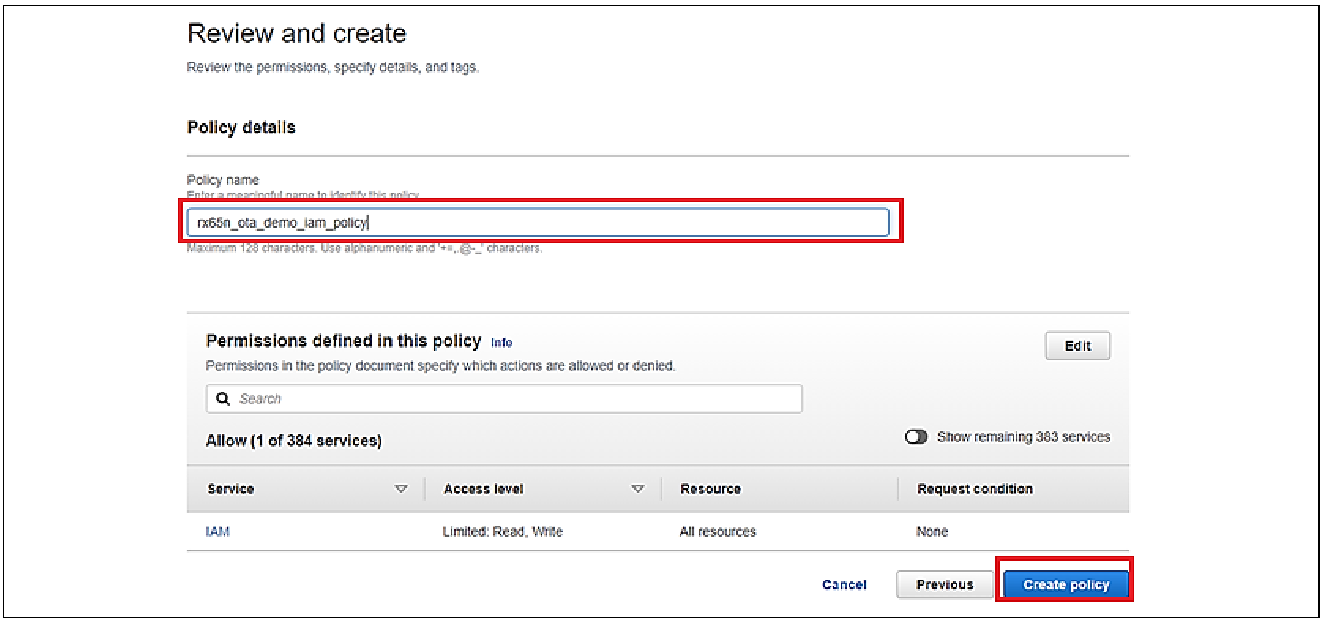
Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

Figure 101. Create inline policy (2/3)

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



Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

rx65n_ota_demo_iam_policy

Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

Permissions defined in this policy Info

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

Search

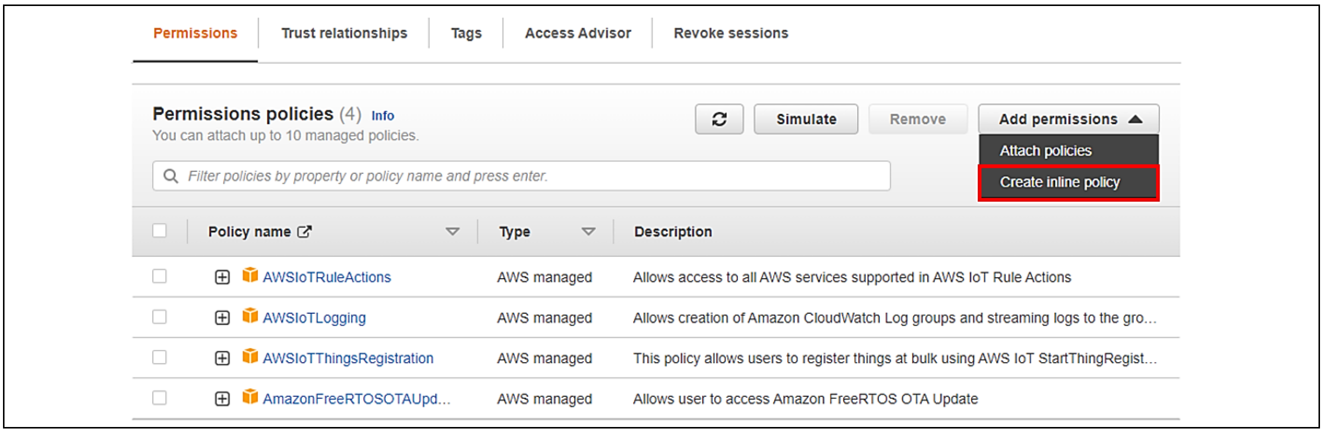
Allow (1 of 384 services)

Show remaining 383 services

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

Figure 102. Create inline policy (3/3)

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



Permissions policies (4) Info

You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

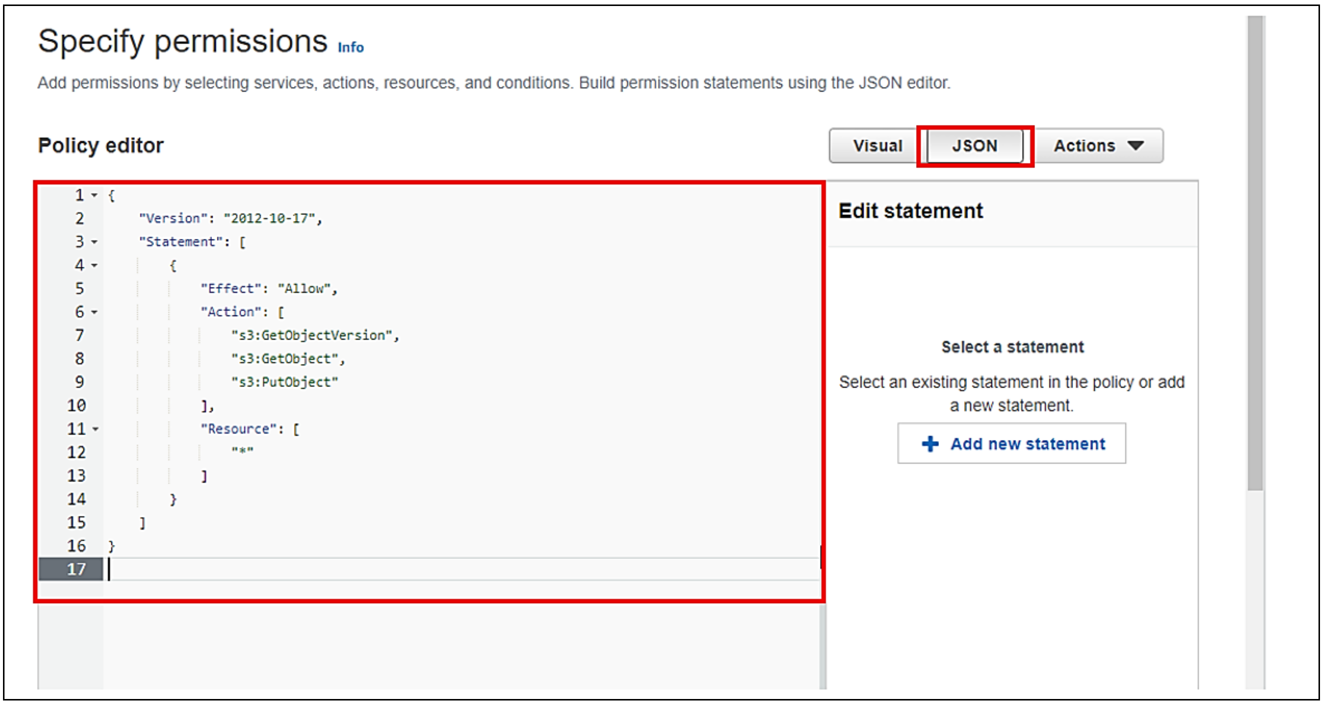
	Policy name	Type	Description
□	AWSIoTRuleActions	AWS managed	Allows access to all AWS services supported in AWS IoT Rule Actions
□	AWSIoTLogging	AWS managed	Allows creation of Amazon CloudWatch Log groups and streaming logs to the gro...
□	AWSIoTThingsRegistration	AWS managed	This policy allows users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

Figure 103. Create inline policy (1/3)

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

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

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



Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

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

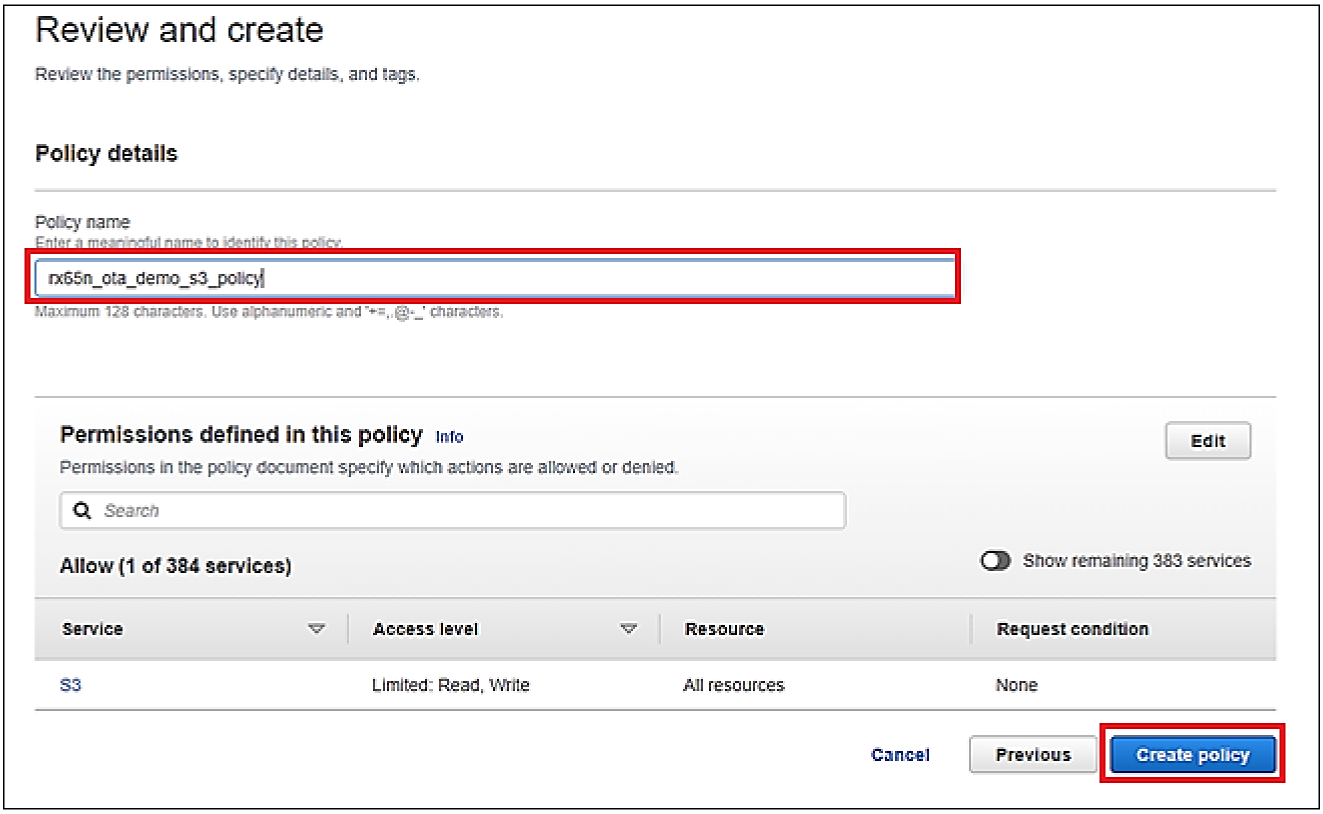
Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

Figure 104. Create inline policy (2/3)

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



Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

rx65n_ota_demo_s3_policy

Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

Permissions defined in this policy Info

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

Allow (1 of 384 services)

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

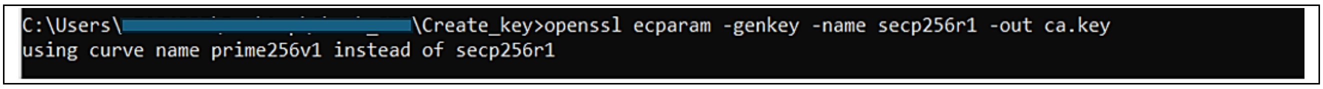
Figure 105. Create inline policy (3/3)

6.4 Setting up the Device

6.4.1 Generating Key Pairs and Certificates

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

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

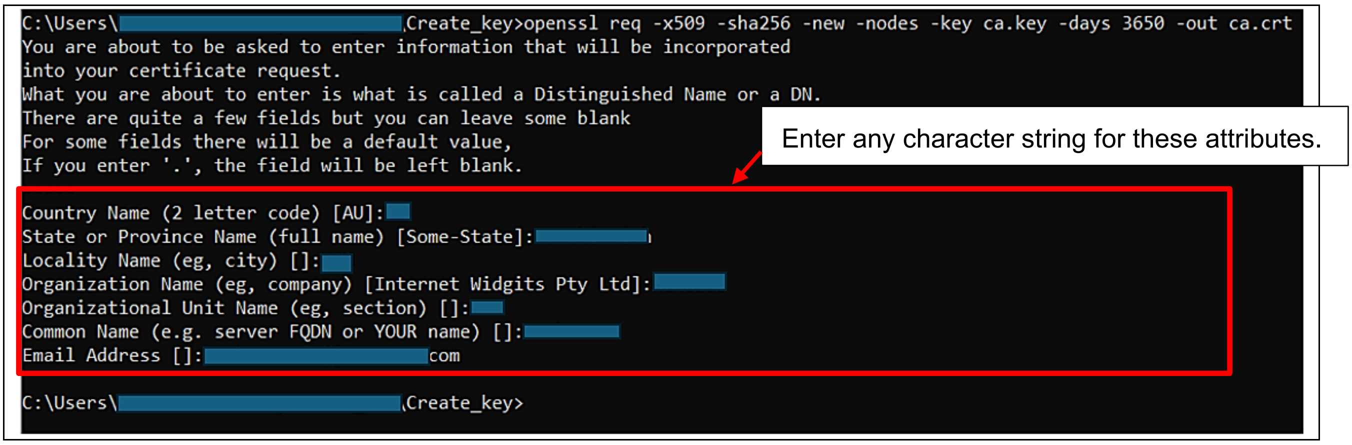


```
C:\Users\[redacted]\Create_key>openssl ecparam -genkey -name secp256r1 -out ca.key
using curve name prime256v1 instead of secp256r1
```

Figure 106. Creating CA private key

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

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



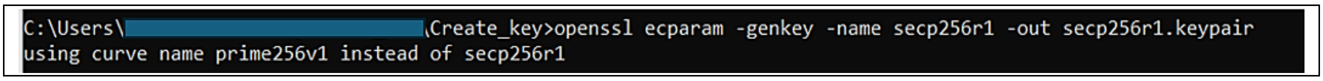
```
C:\Users\[redacted]Create_key>openssl req -x509 -sha256 -new -nodes -key ca.key -days 3650 -out ca.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite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

Country Name (2 letter code) [AU]:[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]com

C:\Users\[redacted]Create_key>
```

Enter any character string for these attributes.

Figure 107. Creating CA certificate

4. Create an ECDSA key pair.

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



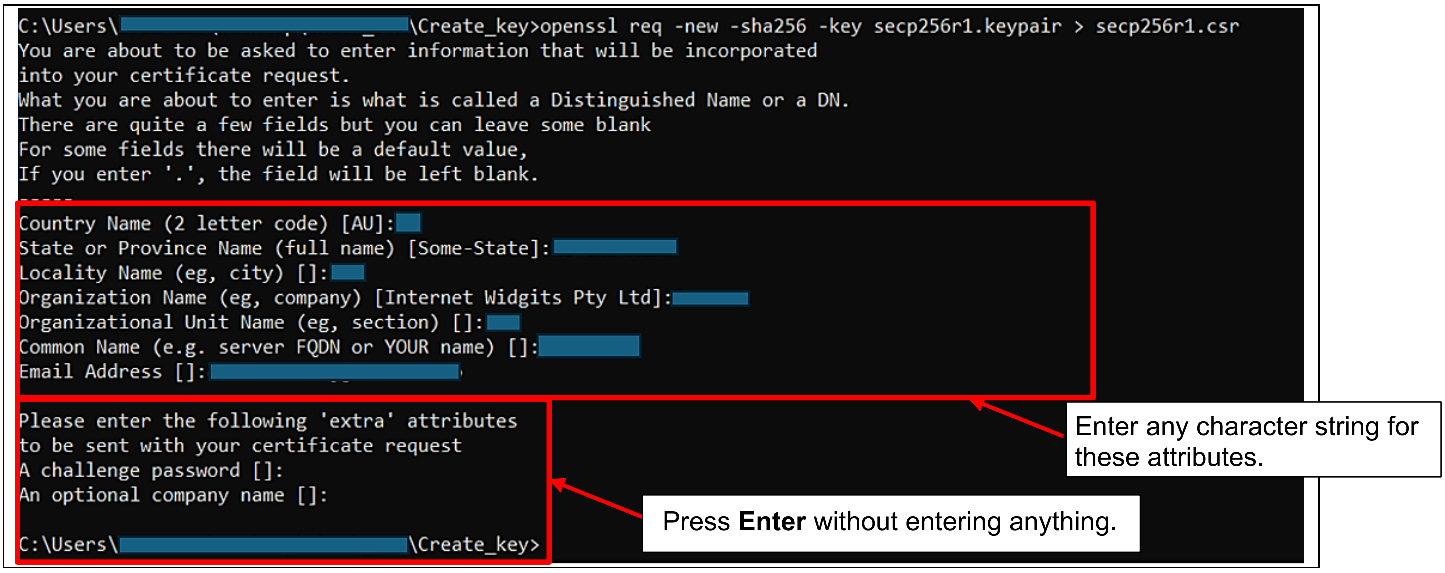
```
C:\Users\[redacted]\Create_key>openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1
```

Figure 108. Creating ECDSA key pairs

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

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

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



```
C:\Users\[redacted]\Create_key>openssl req -new -sha256 -key secp256r1.keypair > secp256r1.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:[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 []:

C:\Users\[redacted]\Create_key>
```

Enter any character string for these attributes.

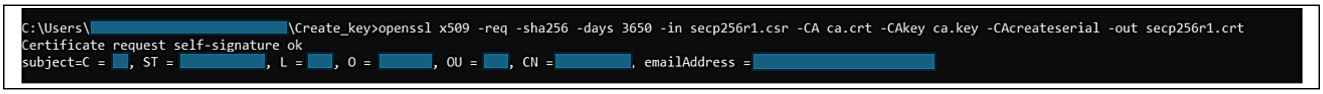
Press **Enter** without entering anything.

Figure 109. Creating certificate signing request

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

Execute the following command:

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



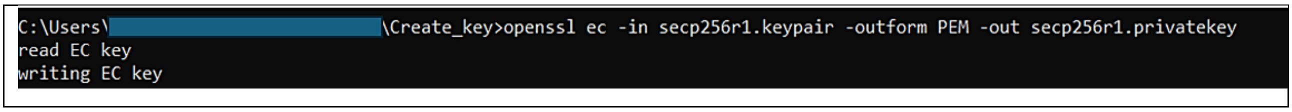
```
C:\Users\[redacted]\Create_key>openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out secp256r1.crt
Certificate request self-signature ok
subject=C = [redacted], ST = [redacted], L = [redacted], O = [redacted], OU = [redacted], CN = [redacted], emailAddress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

Figure 110. Creating code signing certificate

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

Execute the following command:

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



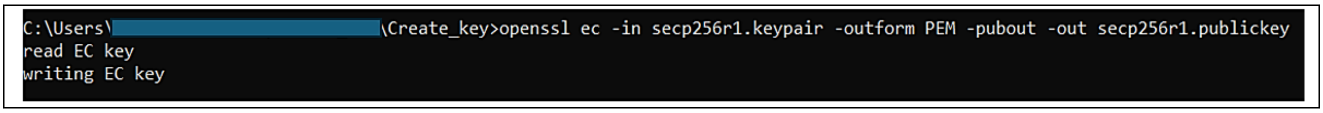
```
C:\Users\[redacted]\Create_key>openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
read EC key
writing EC key
```

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

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

Execute the following command:

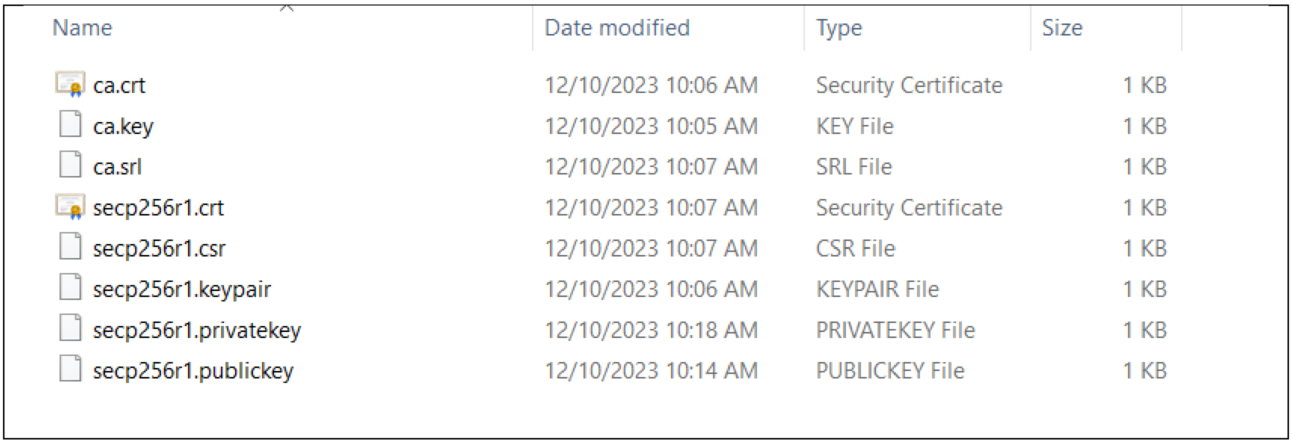
"openssl ec -in secp256r1.keypair -outform PEM -pubout -out secp256r1.publickey"



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

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

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



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

Figure 113. Total created files

6.4.2 Setting up the project

6.4.2.1 Creating initial firmware

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

1. Additional import "**[Project Root folder]\Projects\boot_loader_ck_rx65n_v2\e2studio_gcc** project:

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

Table 7. Components Configuration of boot loader project

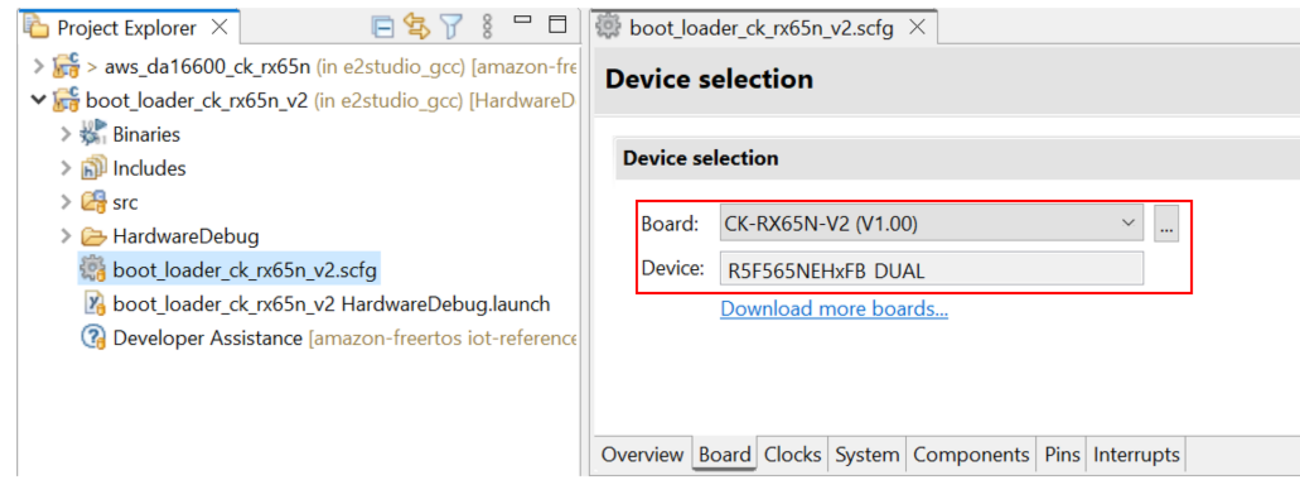
No	Component	Configuration
1	**Startup→Generic→r_bsp (v7.42)**	Enable user stdio charput function: **Use user charput() function**
		User stdio charput function name: **my_sw_charput_function**
		Select whether it is bootloader project: **bootloader project**
2	**Drivers→Memory→r_flash_rx (v5.11)**	Enable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

3	**Drivers→Communications→r_sci_rx (v5.00)**	Include software support for channel 5: **Include**
4	**Middleware→Generic→r_byteq (v2.10)**	Memory allocation for queue control blocks: **Static memory allocation**
		Number of static queue control block: **32**
5	**Middleware→Generic→r_fwup (v2.03)**	Select the update mode: **Dual bank**
		Select the function mode: **user for Boot Loader**
		Main area start address: **0xFFF00000**
		Buffer area start address: **0xFFE00000**
		Install area size: **0xF0000**
		Select the algorithm of signature verification: **ECDSA + SHA256**
		Enable user disable interrupt function: **Use FWUP r_fwup_wrap_disable_interrupt() function**
		Enable user enable interrupt function: **Use FWUP r_fwup_wrap_enable_interrupt() function**
		Enable user software delay function: **Use FWUP r_fwup_wrap_software_delay() function**
		Enable user software reset function: **Use FWUP r_fwup_wrap_software_reset() function**
		Enable user sha256 init function: **Use FWUP r_fwup_wrap_sha256_init() function**
		Enable user sha256 update function: **Use FWUP r_fwup_wrap_sha256_update() function**
		Enable user sha256 final function: **Use FWUP r_fwup_wrap_sha256_final() function**
		Enable user verify ecdsa function: **Use FWUP r_fwup_wrap_verify_ecdsa() function**
		Enable user get crypt context function: **Use FWUP r_fwup_wrap_get_crypt_context() function**
		Enable user flash open function: **Use user r_fwup_wrap_flash_open() function**
		User flash open function name: **my_flash_open_function**
		Enable user flash close function: **Use user r_fwup_wrap_flash_close() function**
		User flash close function name: **my_flash_close_function**
		Enable user flash erase function: **Use user r_fwup_wrap_flash_erase() function**
		User flash erase function name: **my_flash_erase_function**
		Enable user flash write function: **Use user r_fwup_wrap_flash_write() function**
		User flash write function name: **my_flash_write_function**

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

Check the boot_loader device.

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


Project Explorer
> aws_da16600_ck_rx65n (in e2studio_gcc) [amazon-fre
boot_loader_ck_rx65n_v2 (in e2studio_gcc) [HardwareD
Binaries
Includes
src
HardwareDebug
boot_loader_ck_rx65n_v2.scfg
boot_loader_ck_rx65n_v2 HardwareDebug.launch
Developer Assistance [amazon-freertos iot-reference
boot_loader_ck_rx65n_v2.scfg
Device selection
Download more boards...
Overview
Board
Clocks
System
Components
Pins
Interrupts

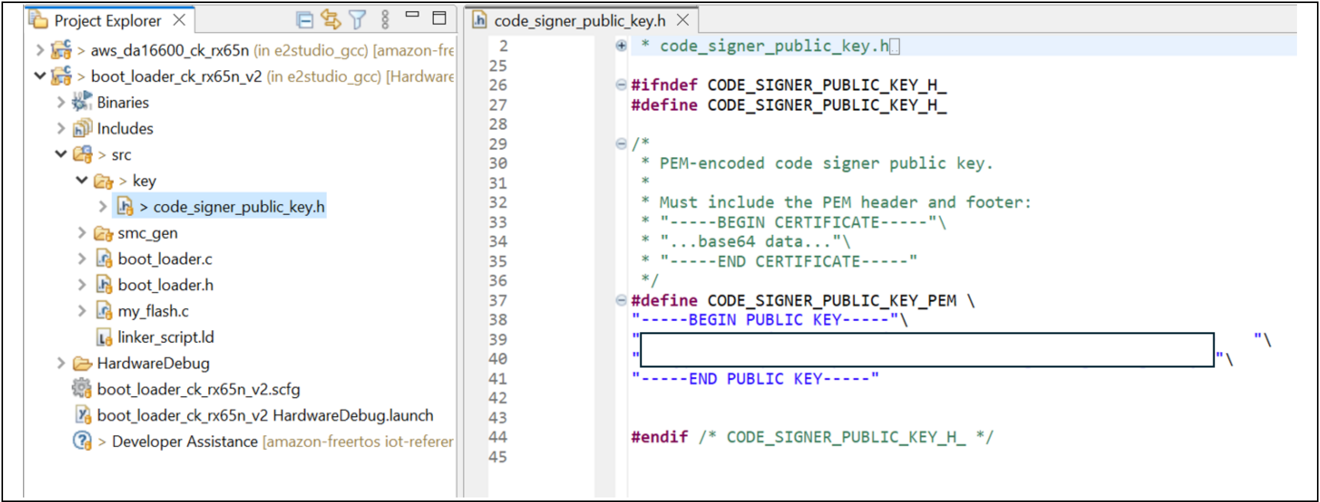
Device selection

Board: CK-RX65N-V2 (V1.00)

Device: R5F565NEHxFB_DUAL

Figure 114. bootloader's device selection

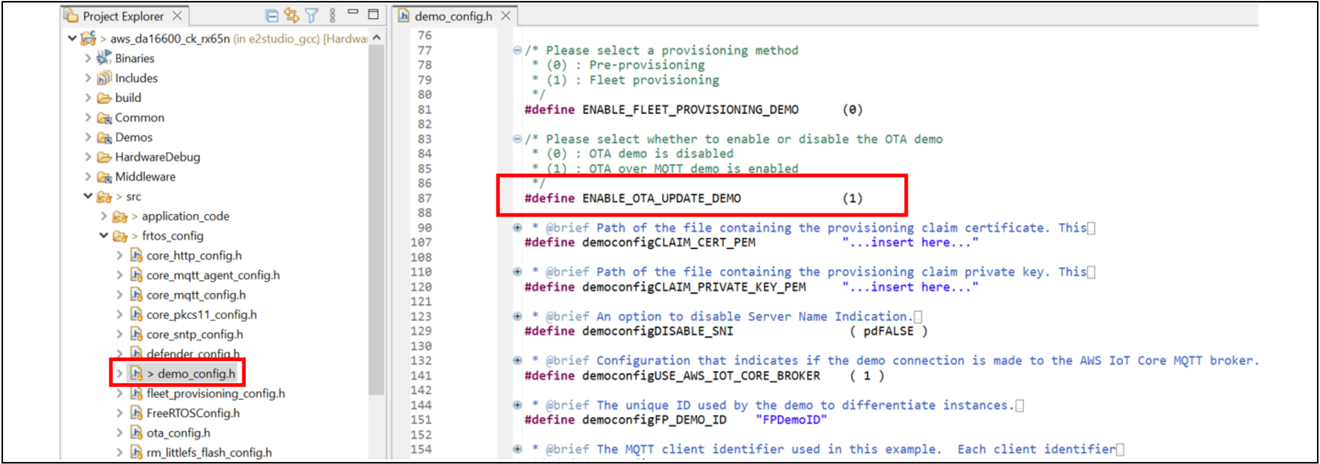
Assign public key: Copy the contents of the secp256r1.publickey file you created in **6.4**, and paste the contents into **CODE_SIGNER_PUBLIC_KEY_PEM** defined in the following files: boot_loader_ck_rx65n_v2\src\key\code_signer_public_key.h


Project Explorer
> aws_da16600_ck_rx65n (in e2studio_gcc) [amazon-fre
> boot_loader_ck_rx65n_v2 (in e2studio_gcc) [Hardware
Binaries
Includes
> src
> key
> code_signer_public_key.h
smc_gen
boot_loader.c
boot_loader.h
my_flash.c
linker_script.ld
HardwareDebug
boot_loader_ck_rx65n_v2.scfg
boot_loader_ck_rx65n_v2 HardwareDebug.launch
> Developer Assistance [amazon-freertos iot-referer
code_signer_public_key.h

```
 * code_signer_public_key.h

#ifndef CODE_SIGNER_PUBLIC_KEY_H_
#define CODE_SIGNER_PUBLIC_KEY_H_

/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"\
 * "...base64 data..."\
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM \
"-----BEGIN PUBLIC KEY-----"\
"                                                            "\
"                                                            "\
"-----END PUBLIC KEY-----"



#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

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

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



```
/* Please select a provisioning method
 * (0) : Pre-provisioning
 * (1) : Fleet provisioning
 */
#define ENABLE_FLEET_PROVISIONING_DEMO      (0)

/* Please select whether to enable or disable the OTA demo
 * (0) : OTA demo is disabled
 * (1) : OTA over MQTT demo is enabled
 */
#define ENABLE_OTA_UPDATE_DEMO              (1)

 * @brief Path of the file containing the provisioning claim certificate. This
#define democonfigCLAIM_CERT_PEM            "...insert here..."

 * @brief Path of the file containing the provisioning claim private key. This
#define democonfigCLAIM_PRIVATE_KEY_PEM     "...insert here..."

 * @brief An option to disable Server Name Indication.
#define democonfigDISABLE_SNI                ( pdFALSE )

 * @brief Configuration that indicates if the demo connection is made to the AWS IoT Core MQTT broker.
#define democonfigUSE_AWS_IOT_CORE_BROKER    ( 1 )

 * @brief The unique ID used by the demo to differentiate instances.
#define democonfigFP_DEMO_ID    "FPDemoID"

 * @brief The MQTT client identifier used in this example.  Each client identifier
```

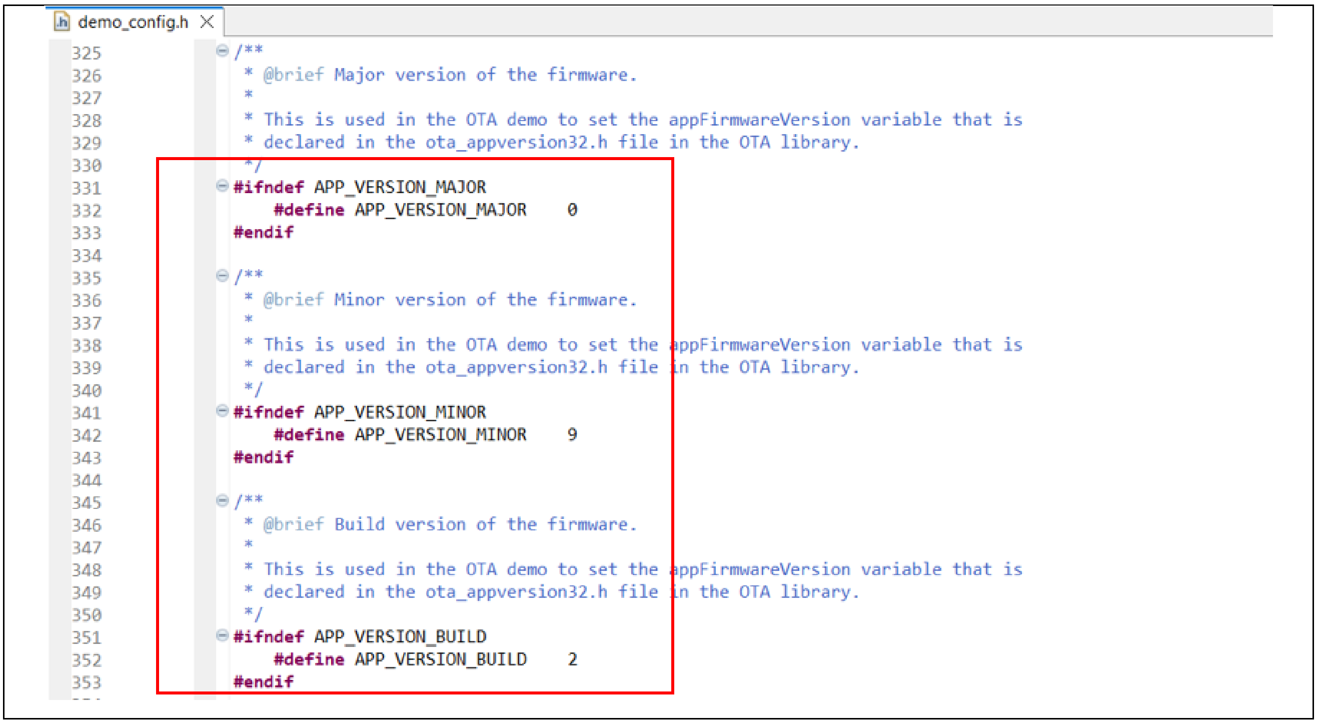
Figure 116. Enable OTA Demo

3. Confirm the initial project version:

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

- APP_VERSION_MAJOR
- APP_VERSION_MINOR
- APP_VERSION_BUILD

Example: The initial project version is 0.9.2



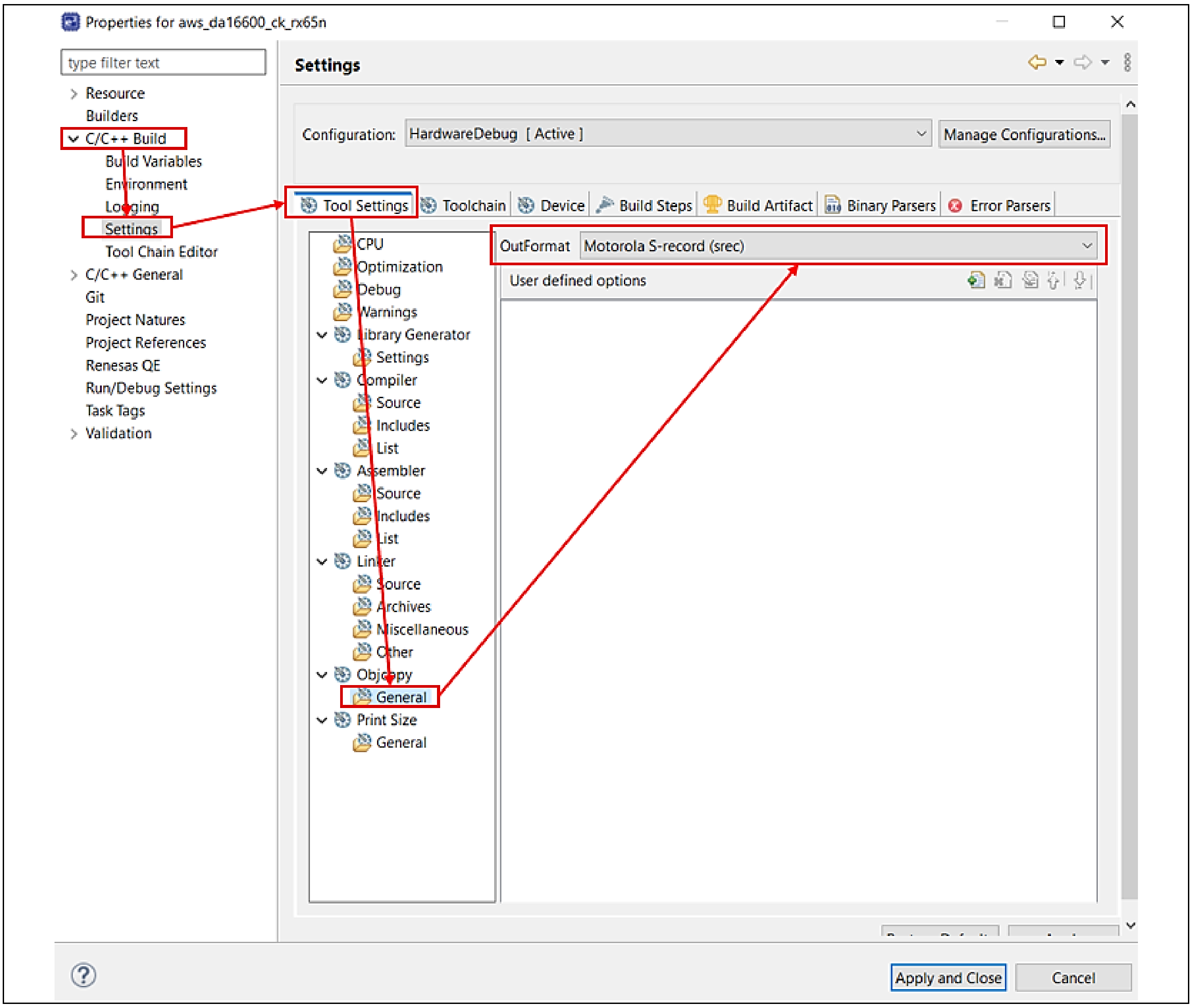
```
/**
 * @brief Major version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MAJOR
    #define APP_VERSION_MAJOR    0
#endif

/**
 * @brief Minor version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MINOR
    #define APP_VERSION_MINOR    9
#endif

/**
 * @brief Build version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    2
#endif
```

Figure 117 The initial project version 0.9.2

4. Checking the project environment settings:

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

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

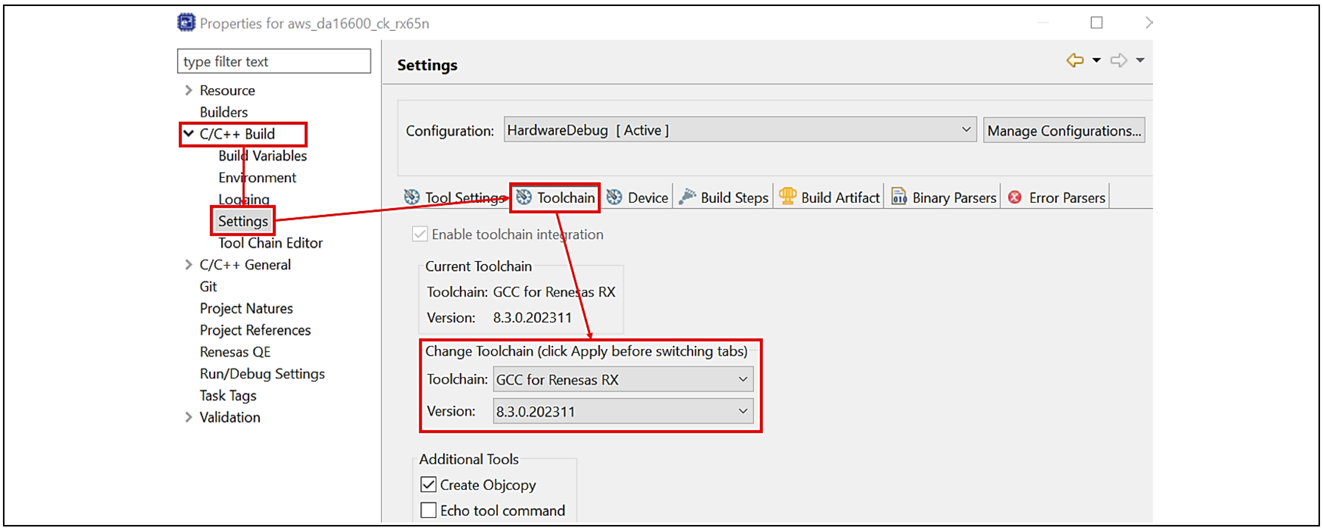


Configuration: HardwareDebug [Active]

OutFormat: Motorola S-record (srec)

Figure 118. Output format

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



Configuration: HardwareDebug [Active]

☑ Enable toolchain integration

Current Toolchain
Toolchain: GCC for Renesas RX
Version: 8.3.0.202311

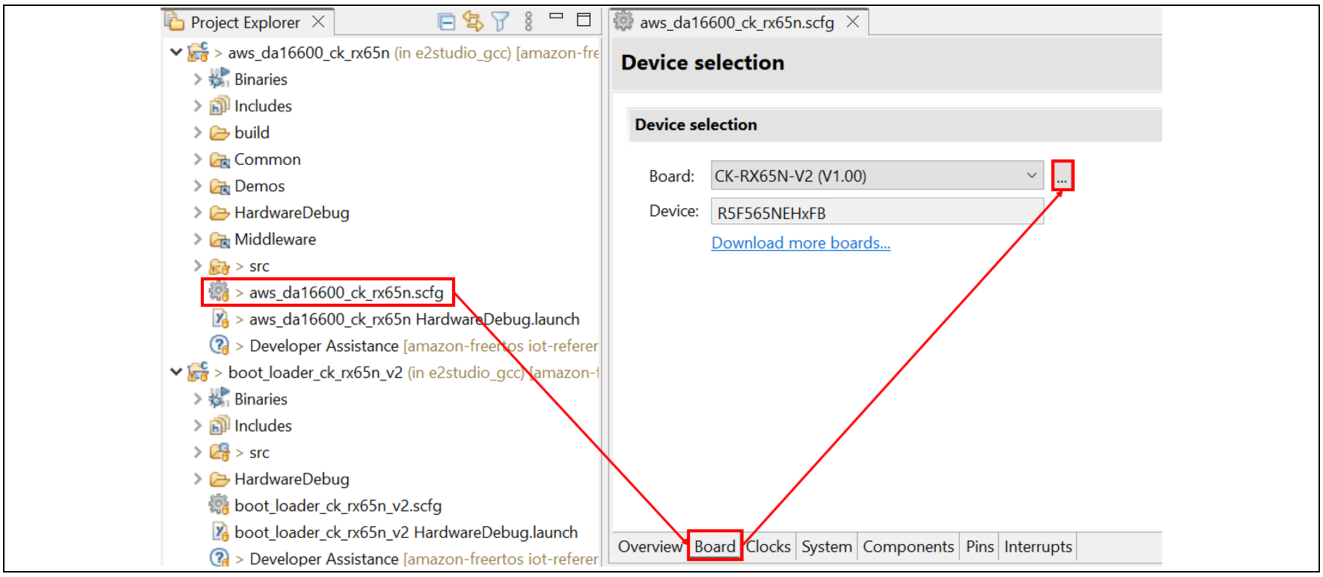
Change Toolchain (click Apply before switching tabs)
Toolchain: GCC for Renesas RX
Version: 8.3.0.202311

Additional Tools
☑ Create Objcopy
☐ Echo tool command

Figure 119. Project toolchain

5. Device selection setting

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



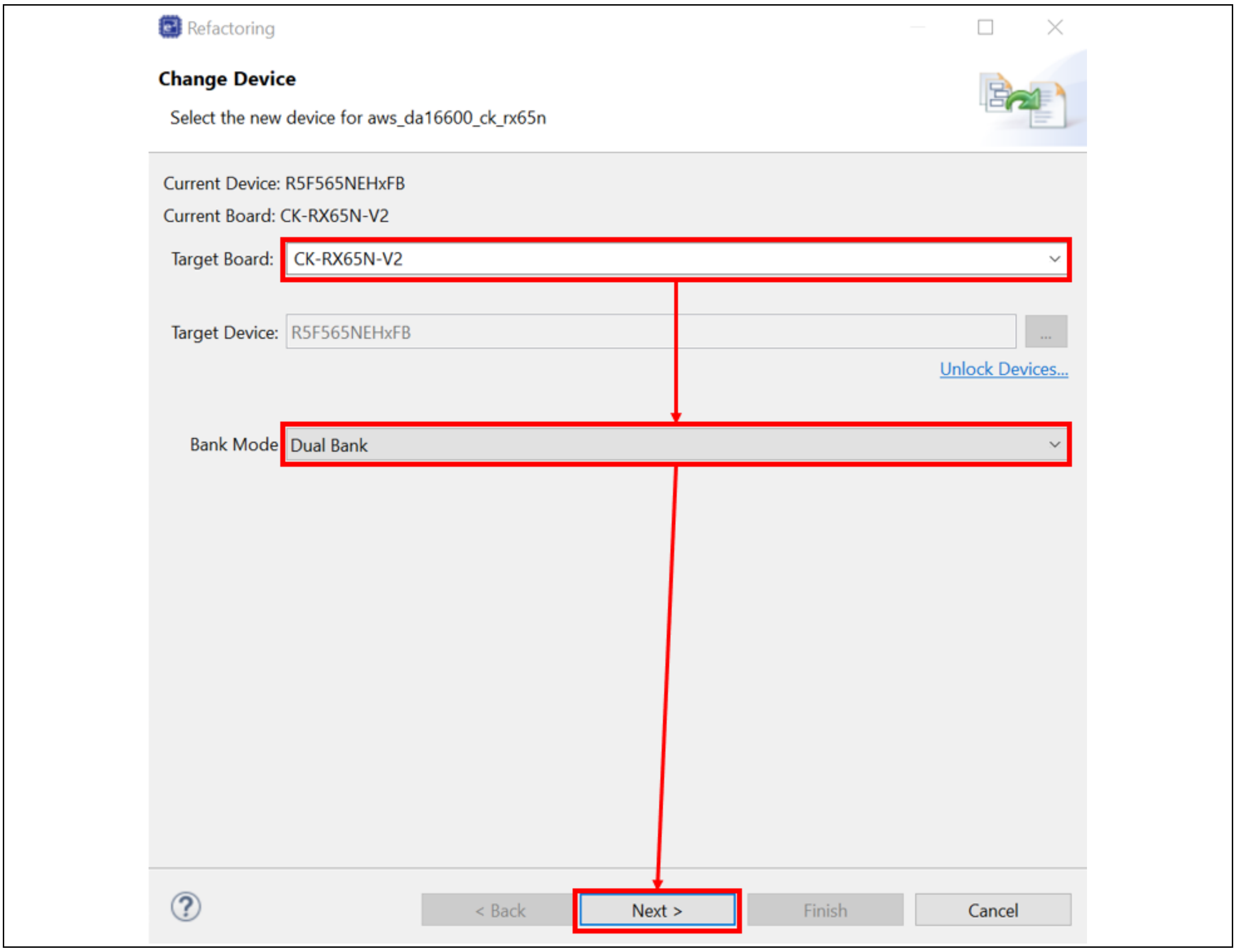
Board: CK-RX65N-V2 (V1.00)

Device: R5F565NEHxFB

Project Explorer
aws_da16600_ck_rx65n (in e2studio_gcc) [amazon-fre
Binaries
Includes
build
Common
Demos
HardwareDebug
Middleware
src
aws_da16600_ck_rx65n.scfg
aws_da16600_ck_rx65n HardwareDebug.launch
Developer Assistance [amazon-freertos iot-referer
boot_loader_ck_rx65n_v2 (in e2studio_gcc) [amazon-
Binaries
Includes
src
HardwareDebug
boot_loader_ck_rx65n_v2.scfg
boot_loader_ck_rx65n_v2 HardwareDebug.launch
Developer Assistance [amazon-freertos iot-referer
aws_da16600_ck_rx65n.scfg
Device selection
Device selection
...
Download more boards...
Overview
Board
Clocks
System
Components
Pins
Interrupts

Figure 120. Device selection (1/3)

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



Change Device

Select the new device for aws_da16600_ck_rx65n

Current Device: R5F565NEHxFB

Current Board: CK-RX65N-V2

Target Board: CK-RX65N-V2

Target Device: R5F565NEHxFB

Bank Mode: Dual Bank

Figure 121. Device selection (2/3)

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

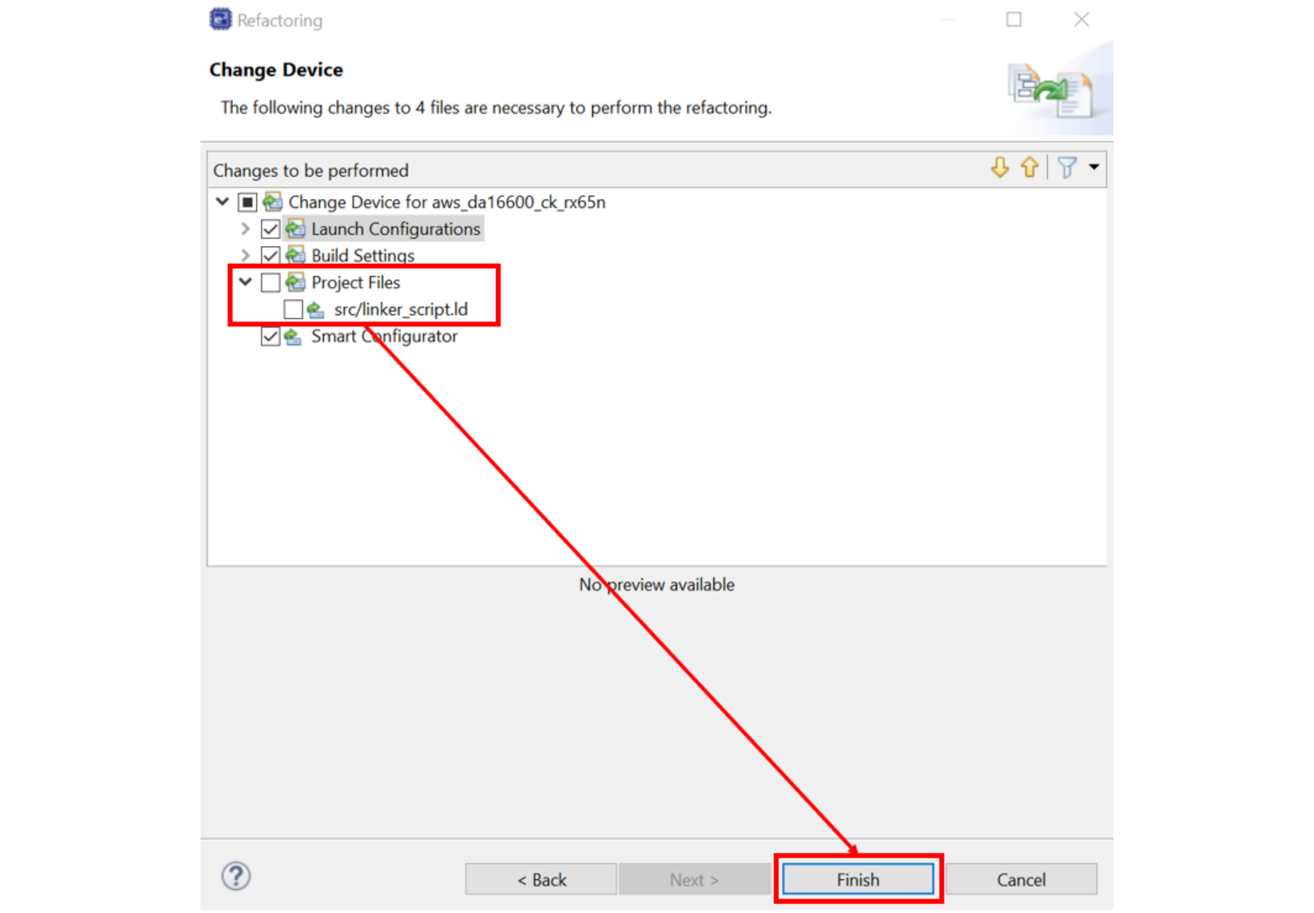
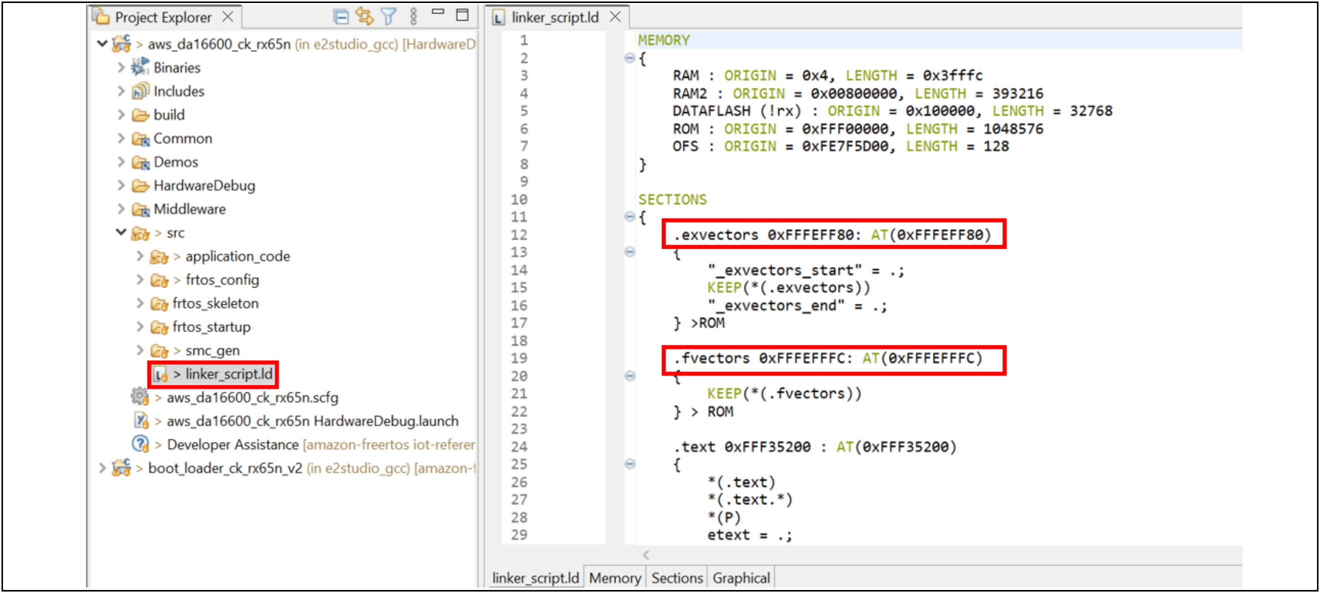


Figure 122. Device selection (3/3)

6. Change the firmware (aws_da16600_ck_rx65n) vector

Open the **aws_da16600_ck_rx65n\src\linker_script.ld** file then allocate **.exvectors** to **0xFFFEFF80: AT(0xFFFEFF80)** and **.fvectors** to **0xFFFEFFFC: AT(0xFFFEFFFC)**


Project Explorer
> aws_da16600_ck_rx65n (in e2studio_gcc) [HardwareD
Binaries
Includes
build
Common
Demos
HardwareDebug
Middleware
> src
> application_code
> frtos_config
frtos_skeleton
frtos_startup
> smc_gen
> linker_script.ld
> aws_da16600_ck_rx65n.scfg
> aws_da16600_ck_rx65n HardwareDebug.launch
> Developer Assistance [amazon-freertos iot-referer
> boot_loader_ck_rx65n_v2 (in e2studio_gcc) [amazon-
linker_script.ld
linker_script.ld
Memory
Sections
Graphical

```
MEMORY
{
    RAM : ORIGIN = 0x4, LENGTH = 0x3fffc
    RAM2 : ORIGIN = 0x00800000, LENGTH = 393216
    DATAFLASH (!rx) : ORIGIN = 0x100000, LENGTH = 32768
    ROM : ORIGIN = 0xFFF00000, LENGTH = 1048576
    OFS : ORIGIN = 0xFE7F5D00, LENGTH = 128
}

SECTIONS
{
    .exvectors 0xFFFEFF80: AT(0xFFFEFF80)
    {
        "_exvectors_start" = .;
        KEEP(*(.exvectors))
        "_exvectors_end" = .;
    } >ROM

    .fvectors 0xFFFEFFFC: AT(0xFFFEFFFC)
    {
        KEEP(*(.fvectors))
    } > ROM

    .text 0xFFF35200 : AT(0xFFF35200)
    {
        *(.text)
        *(.text.*)
        *(P)
        etext = .;
```

Figure 123. Vector setting

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

7. Generate the initial firmware

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

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

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

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

6.4.2.2 Running OTA project

1. Start Renesas Flash Programmer and create new project:

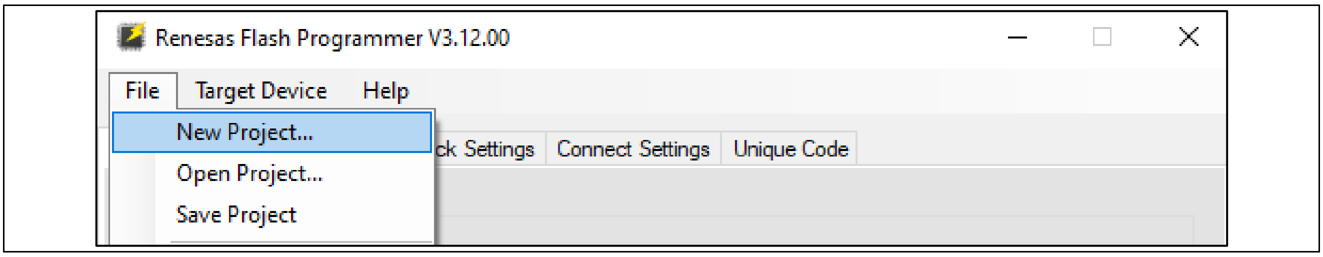
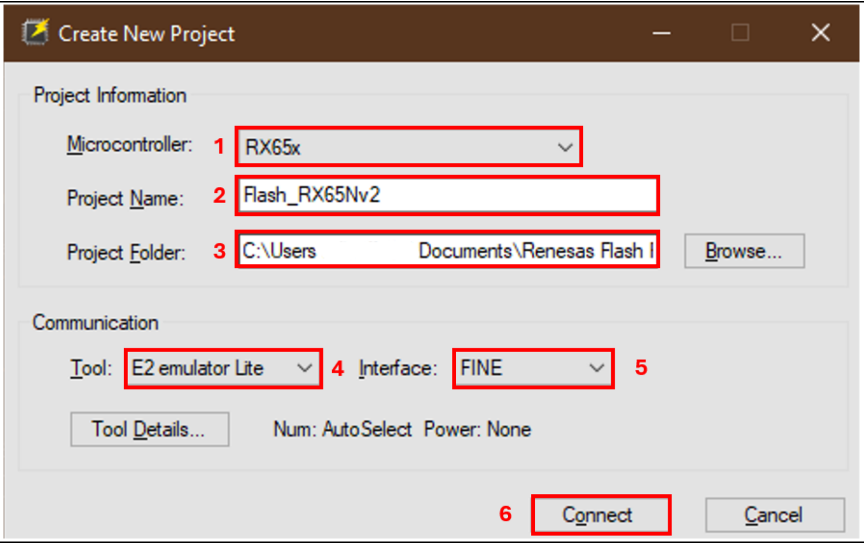


Figure 124. Create New Flash Project (1/4)

After that:

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

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


Create New Project

Project Information

Microcontroller: 1 RX65x

Project Name: 2 Flash_RX65Nv2

Project Folder: 3 C:\Users Documents\Renesas Flash I

Communication

Tool: E2 emulator Lite 4 Interface: FINE 5

Num: AutoSelect Power: None

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

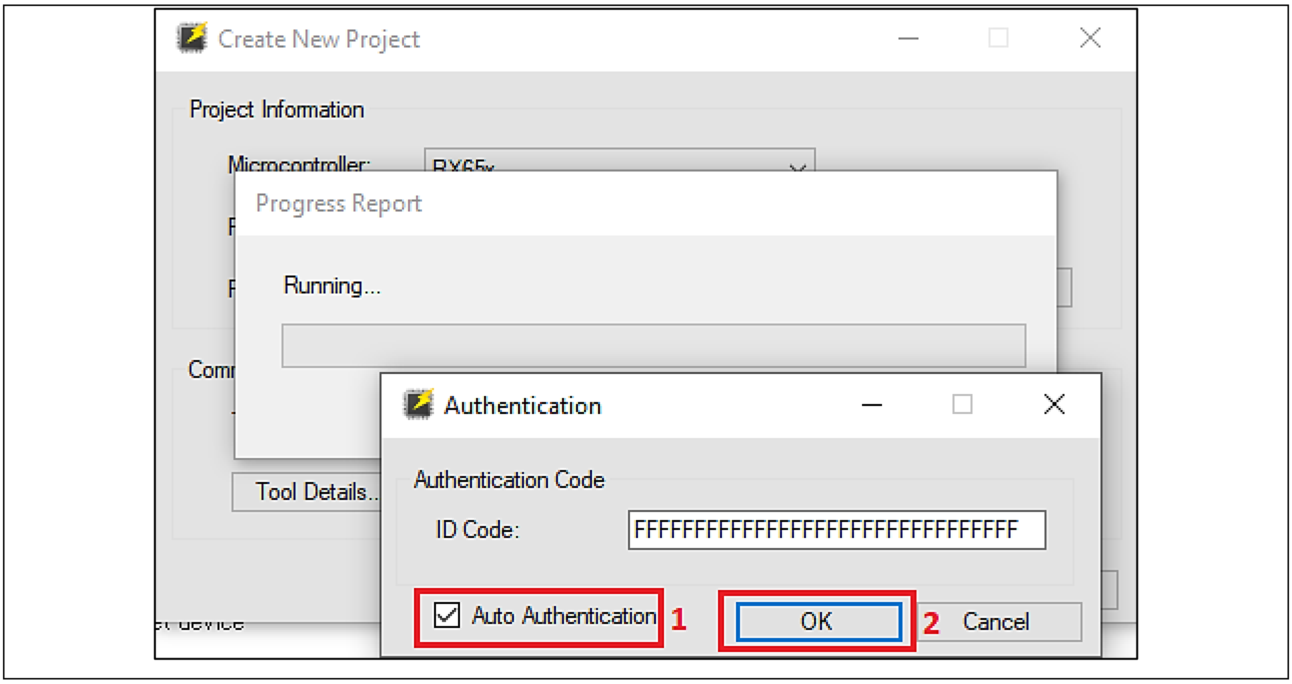
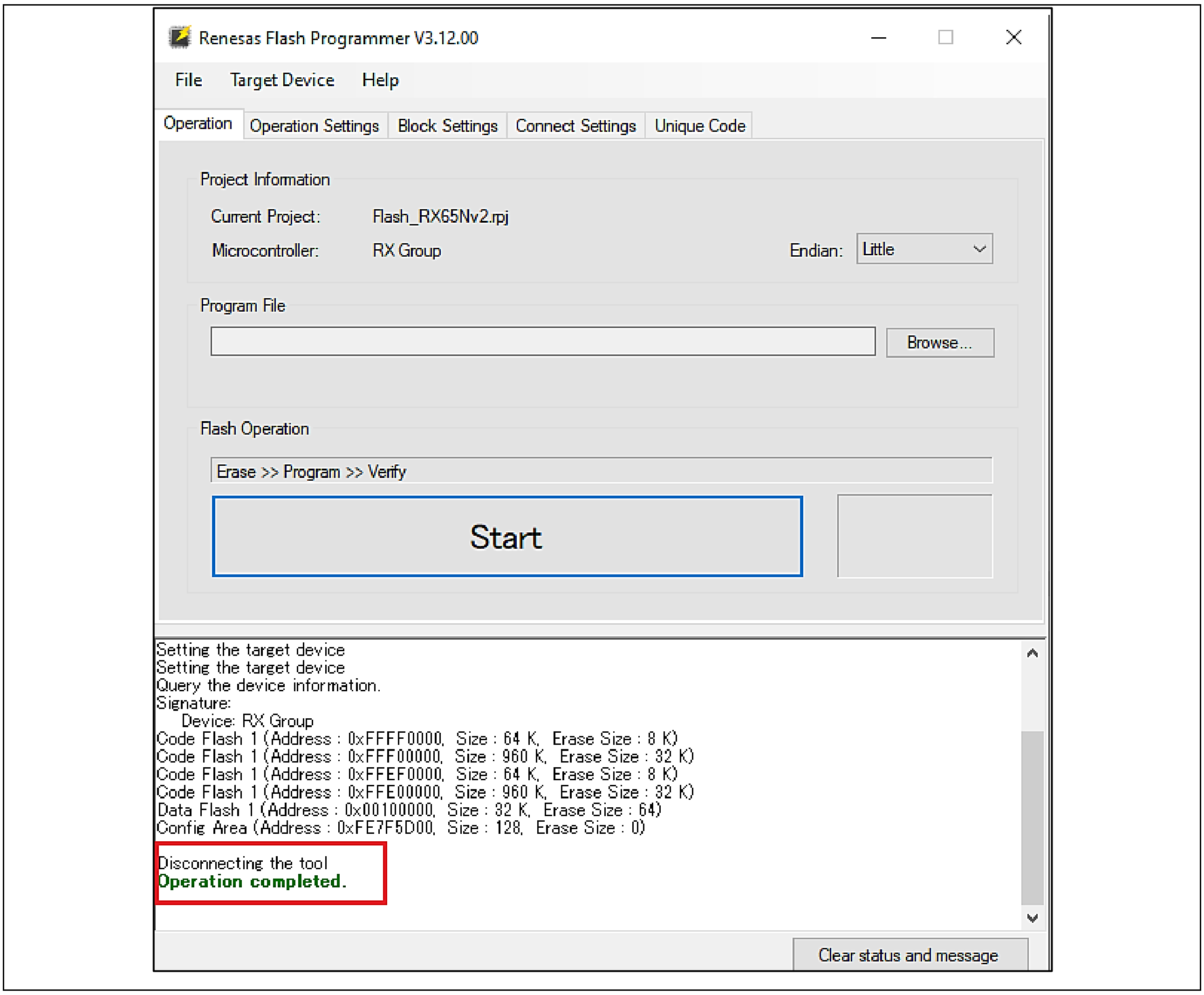


Figure 126. Create New Flash Project (3/4)



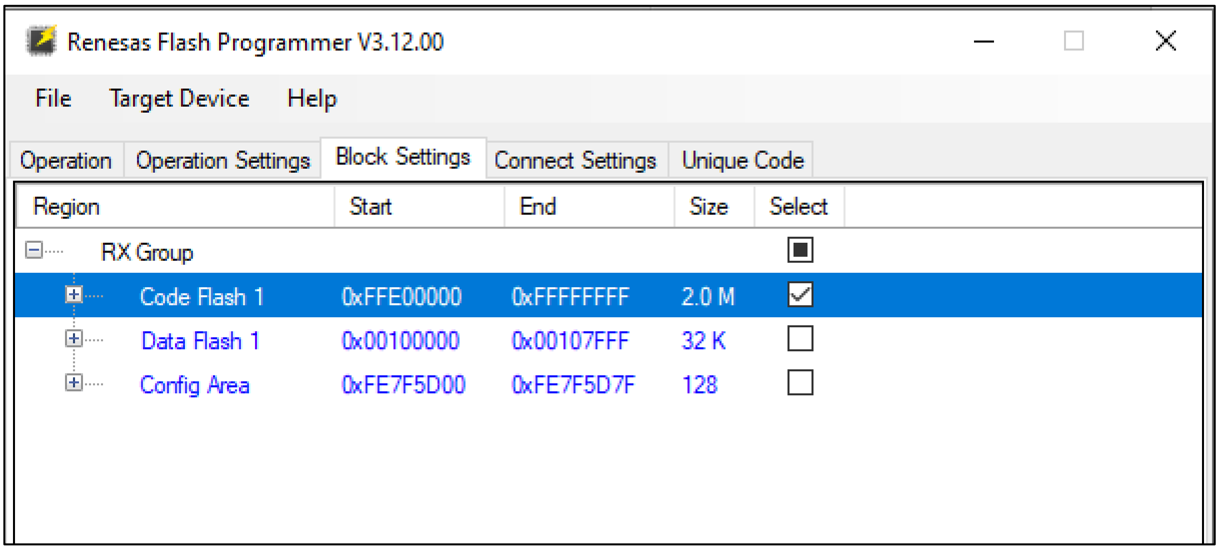
Current Project: Flash_RX65Nv2.rpj
Microcontroller: RX Group
Endian: Little
Erase >> Program >> Verify

```
Setting the target device
Setting the target device
Query the device information.
Signature:
    Device: RX Group
Code Flash 1 (Address : 0xFFFF0000,  Size : 64 K,  Erase Size : 8 K)
Code Flash 1 (Address : 0xFFF00000,  Size : 960 K,  Erase Size : 32 K)
Code Flash 1 (Address : 0xFFEF0000,  Size : 64 K,  Erase Size : 8 K)
Code Flash 1 (Address : 0xFFE00000,  Size : 960 K,  Erase Size : 32 K)
Data Flash 1 (Address : 0x00100000,  Size : 32 K,  Erase Size : 64)
Config Area (Address : 0xFE7F5D00,  Size : 128,  Erase Size : 0)

Disconnecting the tool
Operation completed.
```

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

2. Choose code flash area in configuration:

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

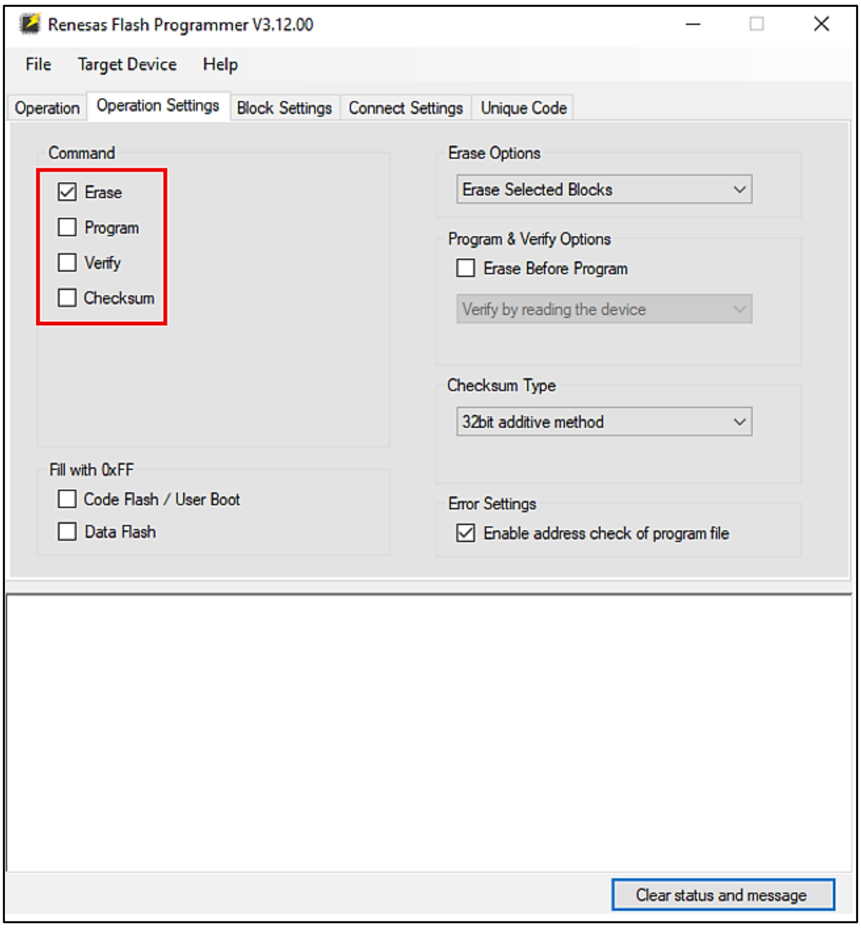


Region	Start	End	Size	Select
RX Group				■
Code Flash 1	0xFFE00000	0xFFFFFFFF	2.0 M	☑
Data Flash 1	0x00100000	0x00107FFF	32 K	□
Config Area	0xFE7F5D00	0xFE7F5D7F	128	□

Figure 128. Choose Code Flash Area

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

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



Command
☑ Erase
□ Program
□ Verify
□ Checksum

Fill with 0xFF
□ Code Flash / User Boot
□ Data Flash

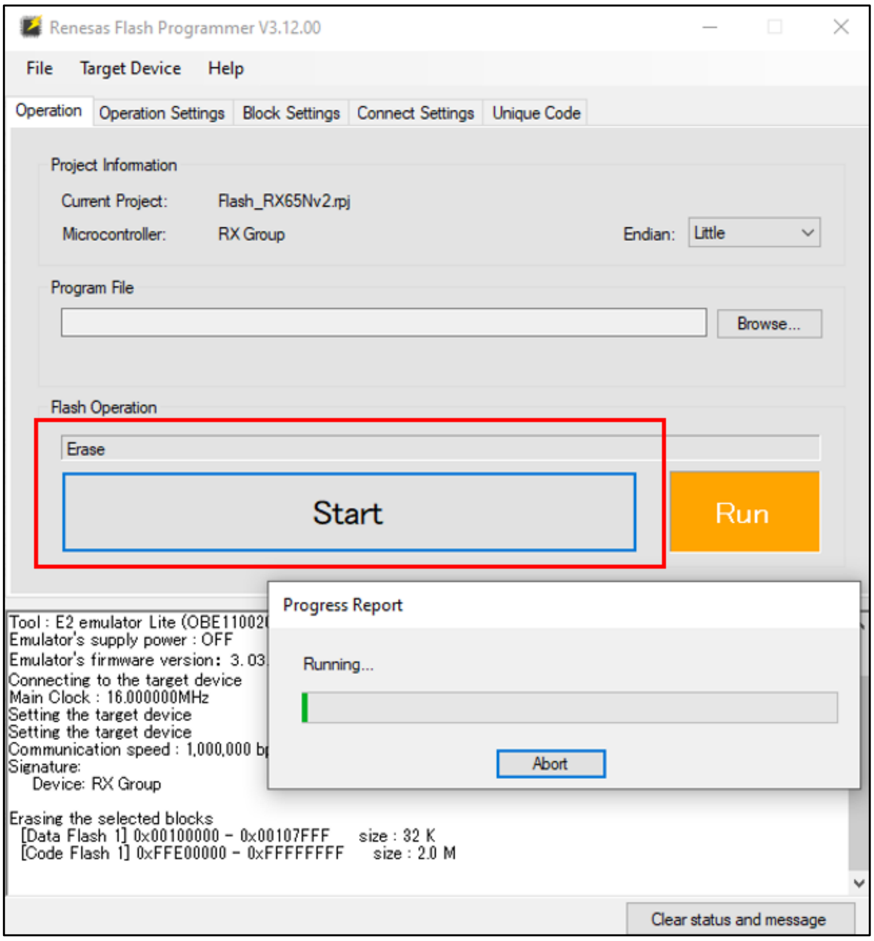
Erase Options
Erase Selected Blocks

Program & Verify Options
□ Erase Before Program
Verify by reading the device

Checksum Type
32bit additive method

Error Settings
☑ Enable address check of program file

Figure 129. Erase Code Flash (1/2)



Project Information

Current Project: Flash_RX65Nv2.rpj

Microcontroller: RX Group

Endian: Little

Program File

Flash Operation

Erase

Progress Report

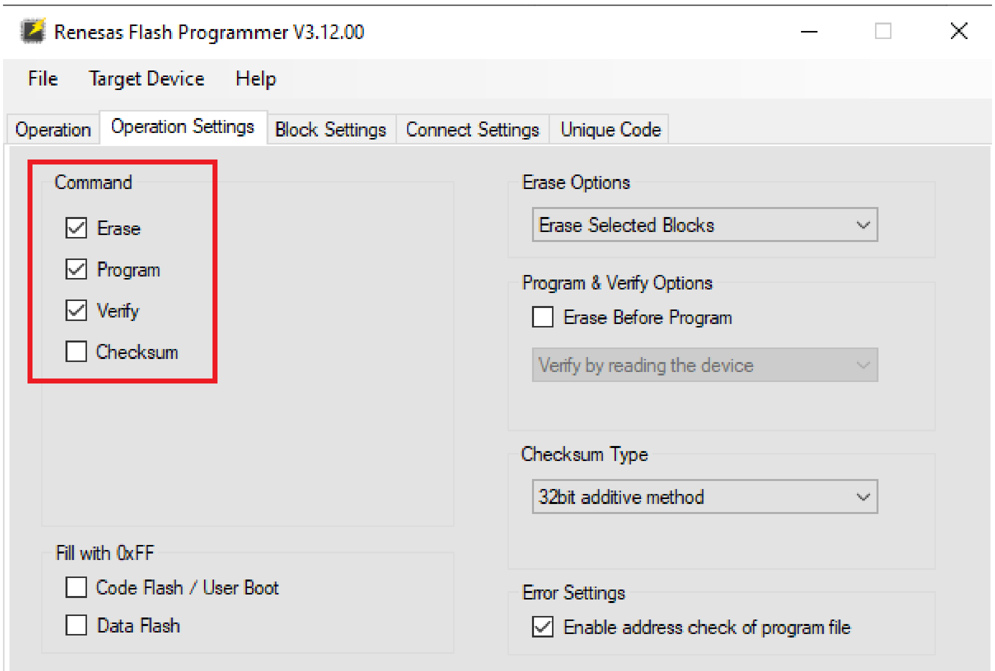
Running...

```
Tool : E2 emulator Lite (OBE11002
Emulator's supply power : OFF
Emulator's firmware version: 3. 03
Connecting to the target device
Main Clock : 16.000000MHz
Setting the target device
Setting the target device
Communication speed : 1,000,000 b
Signature:
   Device: RX Group

Erasing the selected blocks
  [Data Flash 1] 0x00100000 - 0x00107FFF    size : 32 K
  [Code Flash 1] 0xFFE00000 - 0xFFFFFFFF    size : 2.0 M
```

Figure 130. Erase Code Flash (2/2)

4. Write the initial firmware (userprog.mot)

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



Command

☑ Erase

☑ Program

☑ Verify

☐ Checksum

Fill with 0xFF

☐ Code Flash / User Boot

☐ Data Flash

Erase Options

Erase Selected Blocks

Program & Verify Options

☐ Erase Before Program

Verify by reading the device

Checksum Type

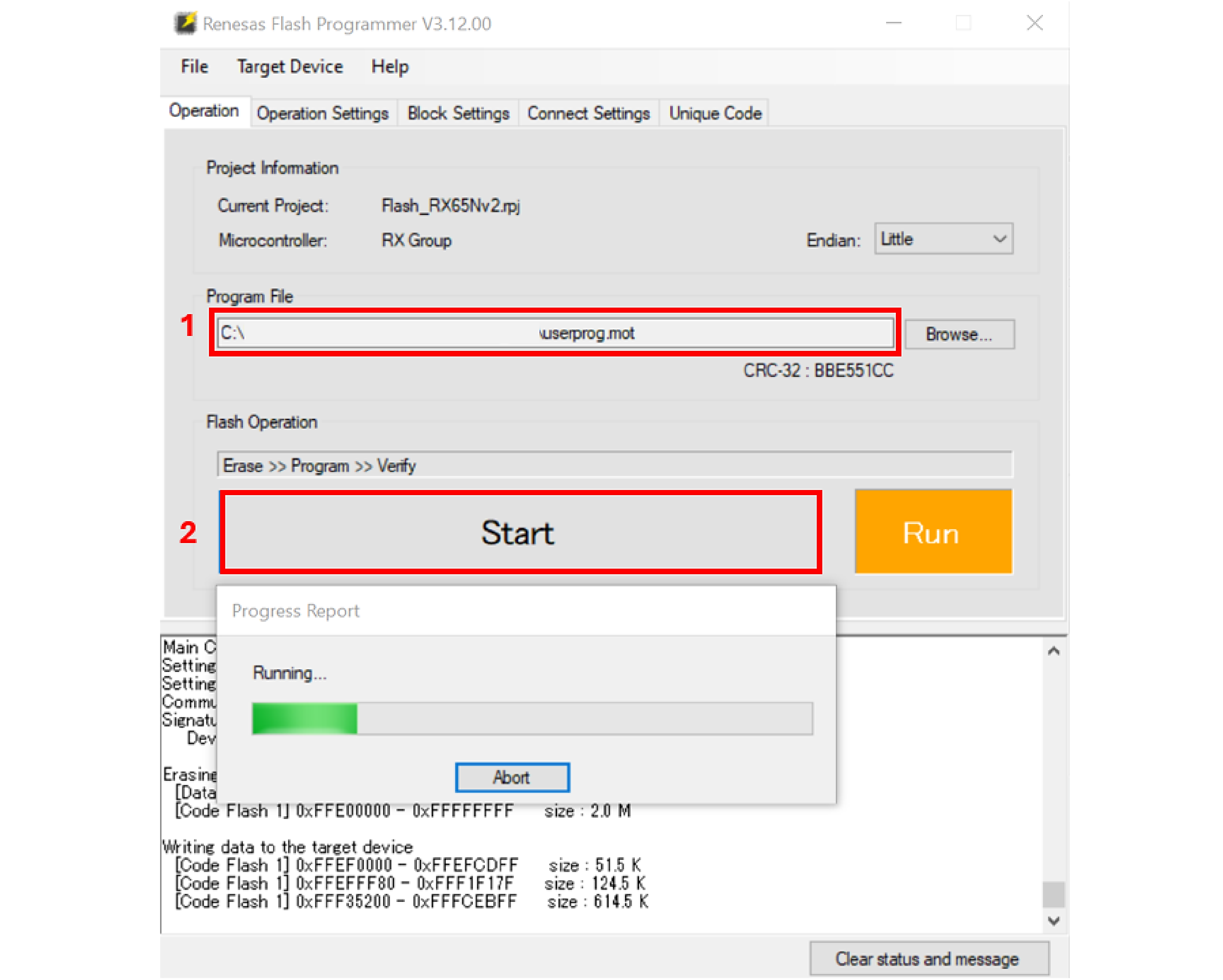
32bit additive method

Error Settings

☑ Enable address check of program file

Figure 131. Flash the Firmware (1/3)

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



Project Information

Current Project: Flash_RX65Nv2.rpj

Microcontroller: RX Group

Endian: Little

Program File

C:\ \userprog.mot

CRC-32 : BBE551CC

Flash Operation

Erase >> Program >> Verify

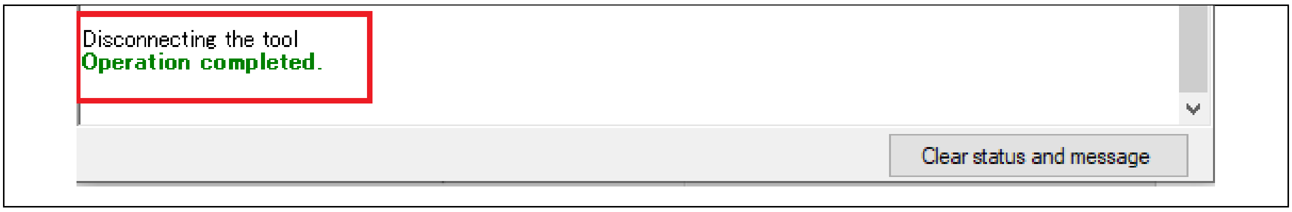
Progress Report

Running...

```
Main C
Setting
Setting
Commu
Signatu
  Dev

Erasing
  [Data
  [Code Flash 1] 0xFFE00000 - 0xFFFFFFFF    size : 2.0 M

Writing data to the target device
  [Code Flash 1] 0xFFEF0000 - 0xFFEFCDFF    size : 51.5 K
  [Code Flash 1] 0xFFEFFF80 - 0xFFF1F17F    size : 124.5 K
  [Code Flash 1] 0xFFF35200 - 0xFFFCEBFF    size : 614.5 K
```

Figure 132. Flash the Firmware (2/3)

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

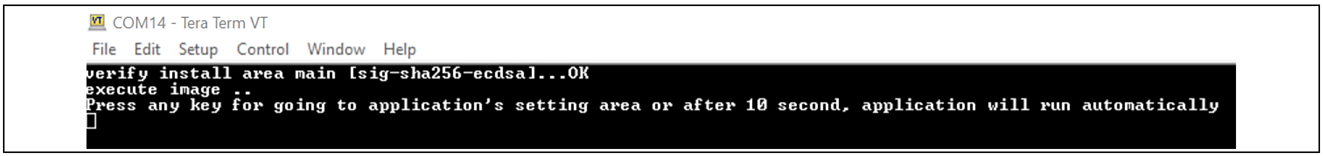


```
Disconnecting the tool
Operation completed.
```

Figure 133. Flash the Firmware (3/3)

5. Running the application:

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



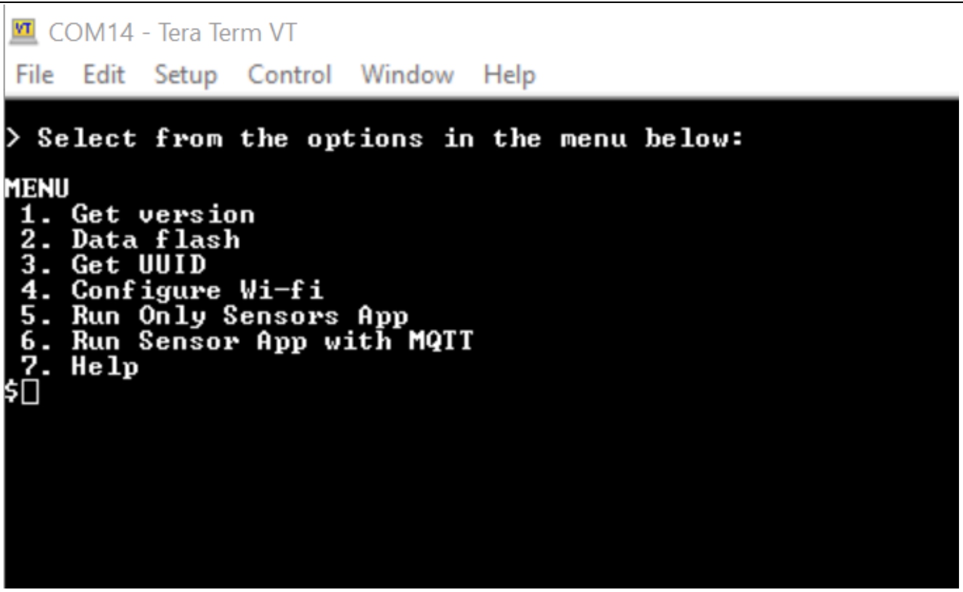
```
verify install area main [sig-sha256-ecdsa]...OK
execute image ..
Press any key for going to application's setting area or after 10 second, application will run automatically
```

Figure 134. Start Application with OTA

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

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

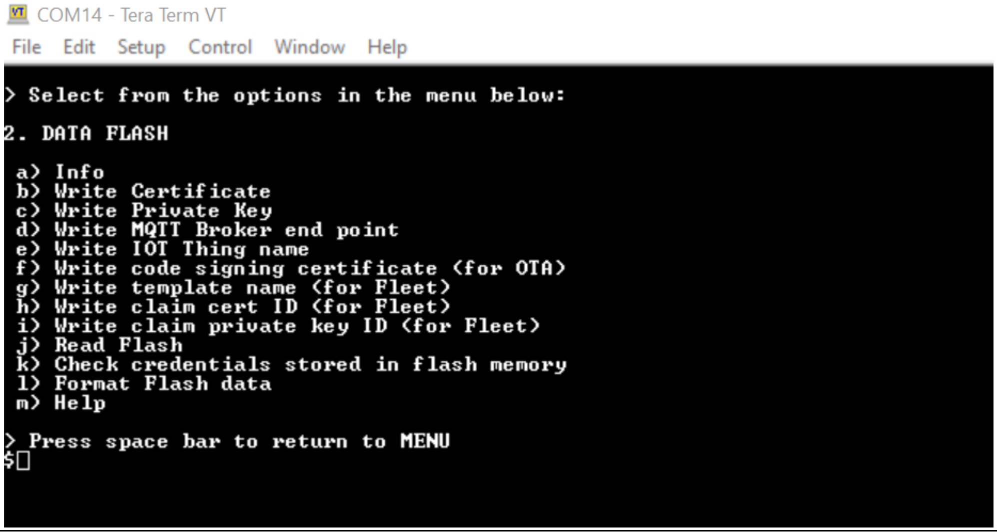
Press ‘2’ on the Main Menu to display Data Flash.



```
> Select from the options in the menu below:

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

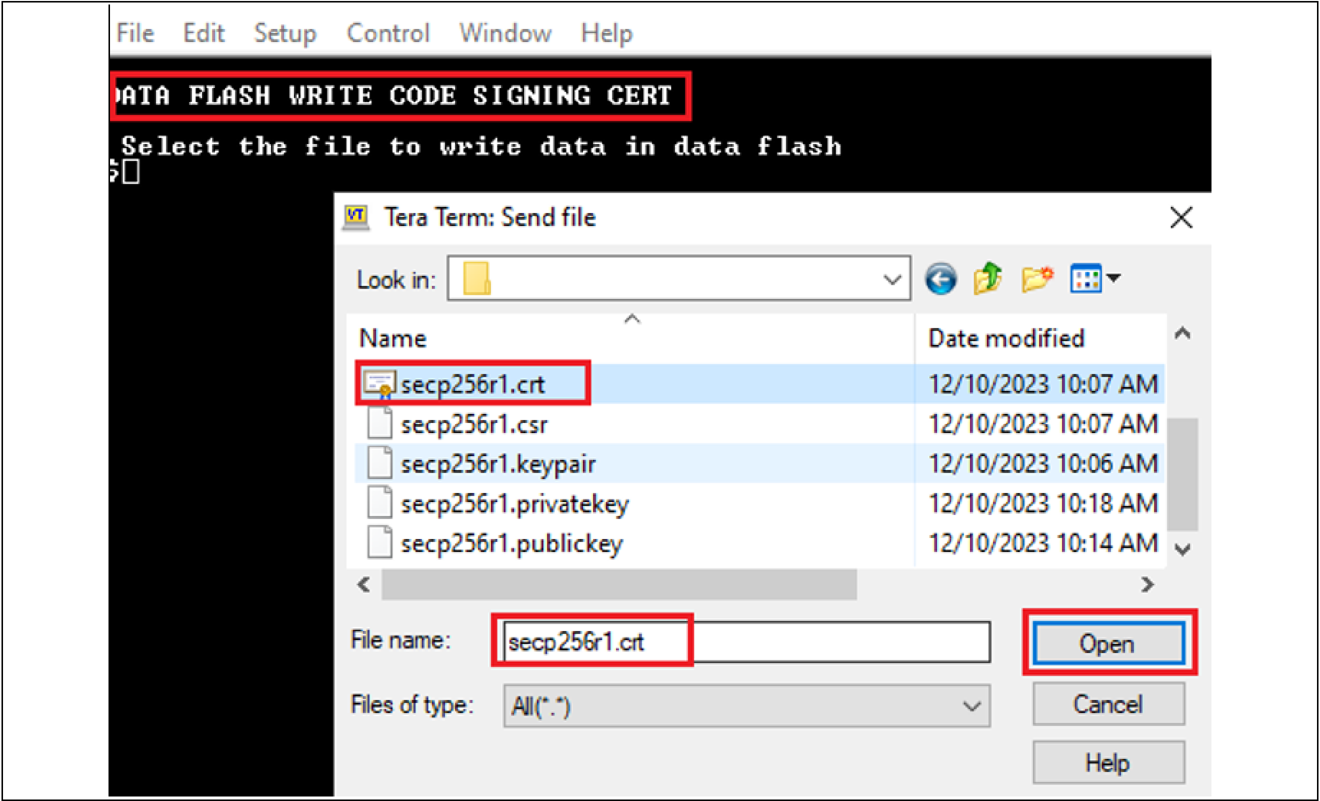
Figure 135. Main Menu

Press ‘f’ for storing code signing certificate:



```
> Select from the options in the menu below:

2. DATA FLASH

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

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

Figure 136. Data Flash Menu



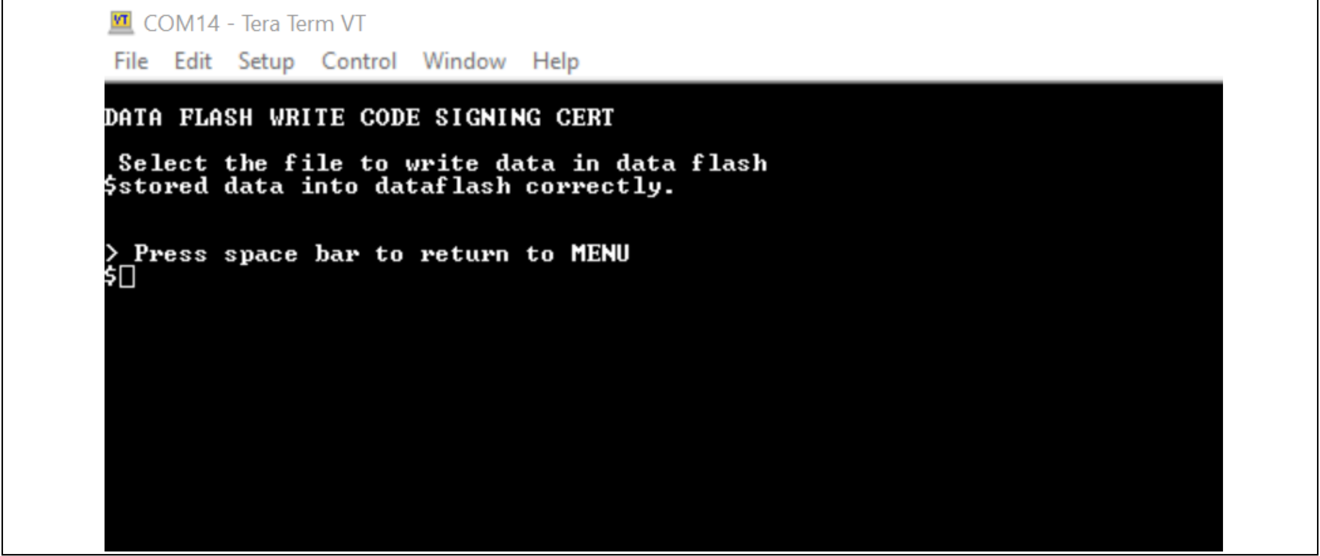
```
DATA FLASH WRITE CODE SIGNING CERT

 Select the file to write data in data flash
$
```

Name	Date modified
secp256r1.crt	12/10/2023 10:07 AM
secp256r1.csr	12/10/2023 10:07 AM
secp256r1.keypair	12/10/2023 10:06 AM
secp256r1.privatekey	12/10/2023 10:18 AM
secp256r1.publickey	12/10/2023 10:14 AM

File name: secp256r1.crt

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

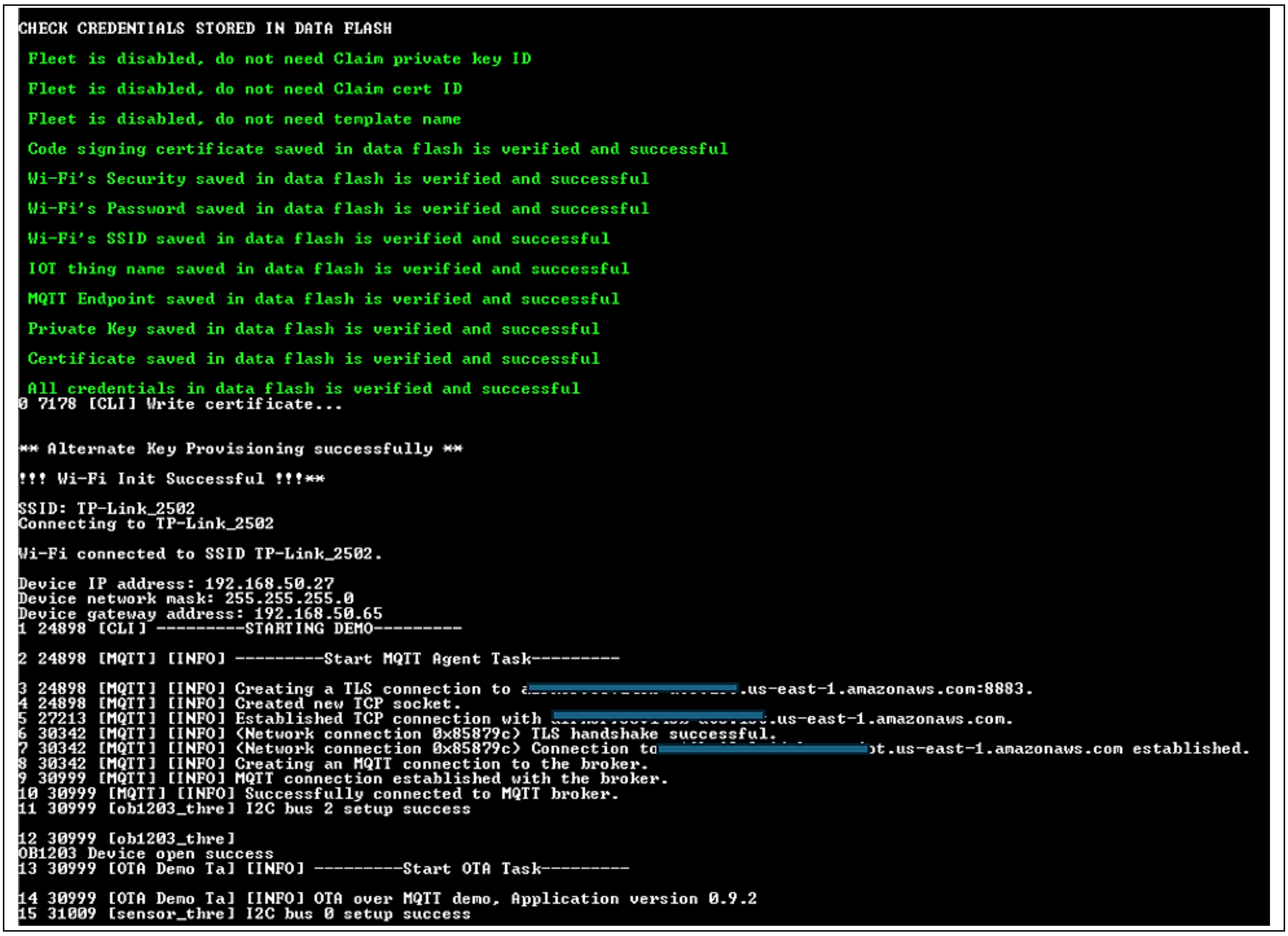
Note: please check the EOL of secp256r1.crt and convert it to LF before saving it into data flash.



```
DATA FLASH WRITE CODE SIGNING CERT

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


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

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

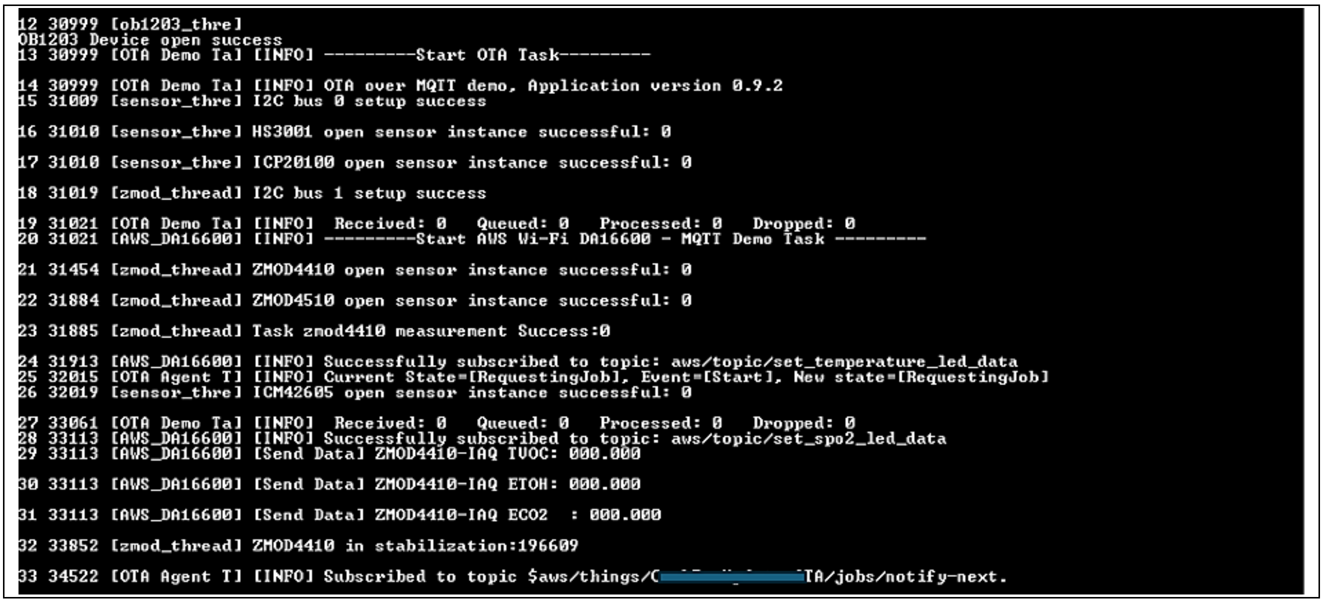
After saving all Cloud credentials, and Wi-Fi credentials, the user starts the application by choosing option ‘6’ on the Main Menu:



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

** Alternate Key Provisioning successfully **

!!! Wi-Fi Init Successful !!!**

SSID: TP-Link_2502
Connecting to TP-Link_2502

Wi-Fi connected to SSID TP-Link_2502.

Device IP address: 192.168.50.27
Device network mask: 255.255.255.0
Device gateway address: 192.168.50.65
1 24898 [CLI] ---------STARTING DEMO---------

2 24898 [MQTT] [INFO] ---------Start MQTT Agent Task---------

3 24898 [MQTT] [INFO] Creating a TLS connection to [redacted].us-east-1.amazonaws.com:8883.
4 24898 [MQTT] [INFO] Created new TCP socket.
5 27213 [MQTT] [INFO] Established TCP connection with [redacted].us-east-1.amazonaws.com.
6 30342 [MQTT] [INFO] (Network connection 0x85879c) TLS handshake successful.
7 30342 [MQTT] [INFO] (Network connection 0x85879c) Connection to[redacted]t.us-east-1.amazonaws.com established.
8 30342 [MQTT] [INFO] Creating an MQTT connection to the broker.
9 30999 [MQTT] [INFO] MQTT connection established with the broker.
10 30999 [MQTT] [INFO] Successfully connected to MQTT broker.
11 30999 [ob1203_thre] I2C bus 2 setup success

12 30999 [ob1203_thre]
OB1203 Device open success
13 30999 [OTA Demo Ta] [INFO] ---------Start OTA Task---------

14 30999 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.2
15 31009 [sensor_thre] I2C bus 0 setup success
```

Figure 139. Application with OTA (1/2)

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



```
12 30999 [ob1203_thre]
OB1203 Device open success
13 30999 [OTA Demo Ta] [INFO] ---------Start OTA Task---------

14 30999 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.2
15 31009 [sensor_thre] I2C bus 0 setup success

16 31010 [sensor_thre] HS3001 open sensor instance successful: 0

17 31010 [sensor_thre] ICP20100 open sensor instance successful: 0

18 31019 [zmod_thread] I2C bus 1 setup success

19 31021 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
20 31021 [AWS_DA16600] [INFO] ---------Start AWS Wi-Fi DA16600 - MQTT Demo Task ---------

21 31454 [zmod_thread] ZMOD4410 open sensor instance successful: 0

22 31884 [zmod_thread] ZMOD4510 open sensor instance successful: 0

23 31885 [zmod_thread] Task zmod4410 measurement Success:0

24 31913 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
25 32015 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
26 32019 [sensor_thre] ICM42605 open sensor instance successful: 0

27 33061 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
28 33113 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
29 33113 [AWS_DA16600] [Send Data] ZMOD4410-IAQ TVOC: 000.000

30 33113 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ETOH: 000.000

31 33113 [AWS_DA16600] [Send Data] ZMOD4410-IAQ ECO2  : 000.000

32 33852 [zmod_thread] ZMOD4410 in stabilization:196609

33 34522 [OTA Agent T] [INFO] Subscribed to topic $aws/things/[redacted]TA/jobs/notify-next.
```

Figure 140. Application with OTA (2/2)

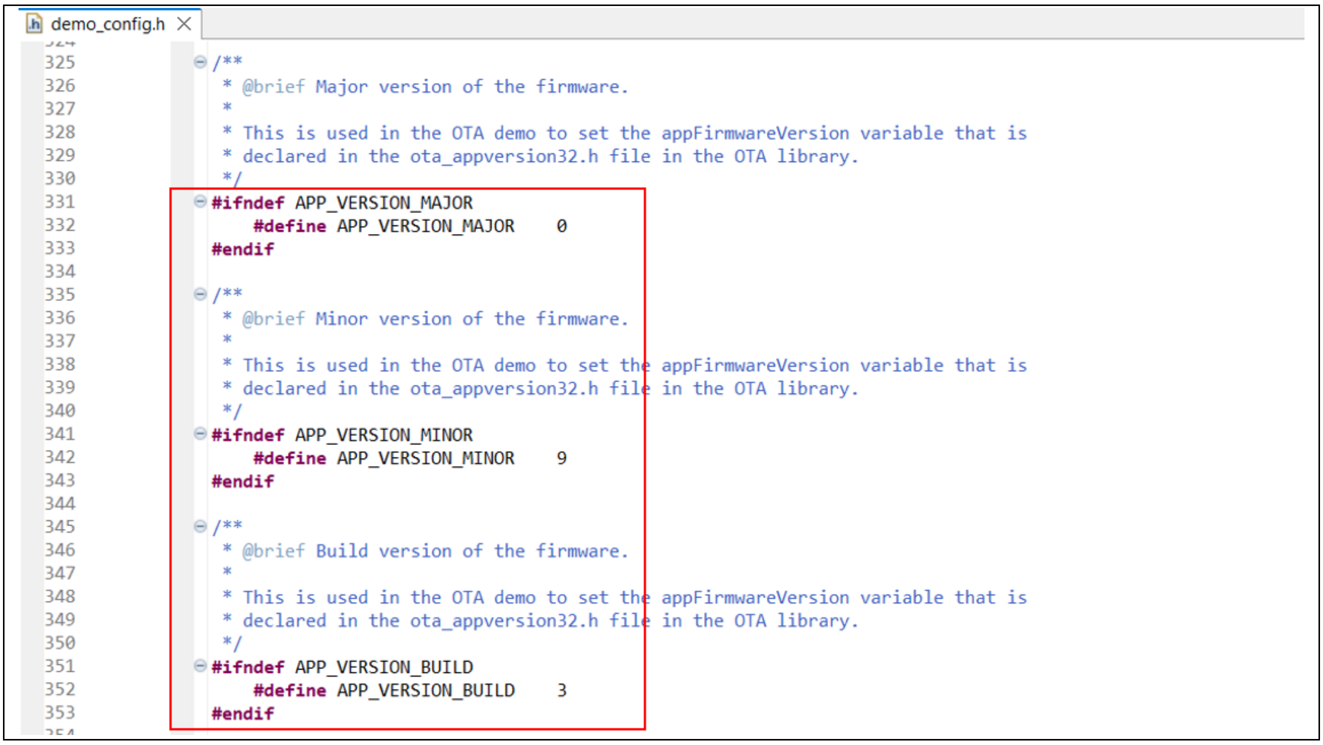
6.5 Updating the Firmware

6.5.1 Creating the updated firmware

6.5.1.1 Changing the firmware version

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

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



```
/**
 * @brief Major version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MAJOR
    #define APP_VERSION_MAJOR    0
#endif

/**
 * @brief Minor version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MINOR
    #define APP_VERSION_MINOR    9
#endif

/**
 * @brief Build version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    3
#endif
```

Figure 141. Setting New Version for Firmware

6.5.1.2 Use Renesas Image Generator to Generate the Updated Firmware

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

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

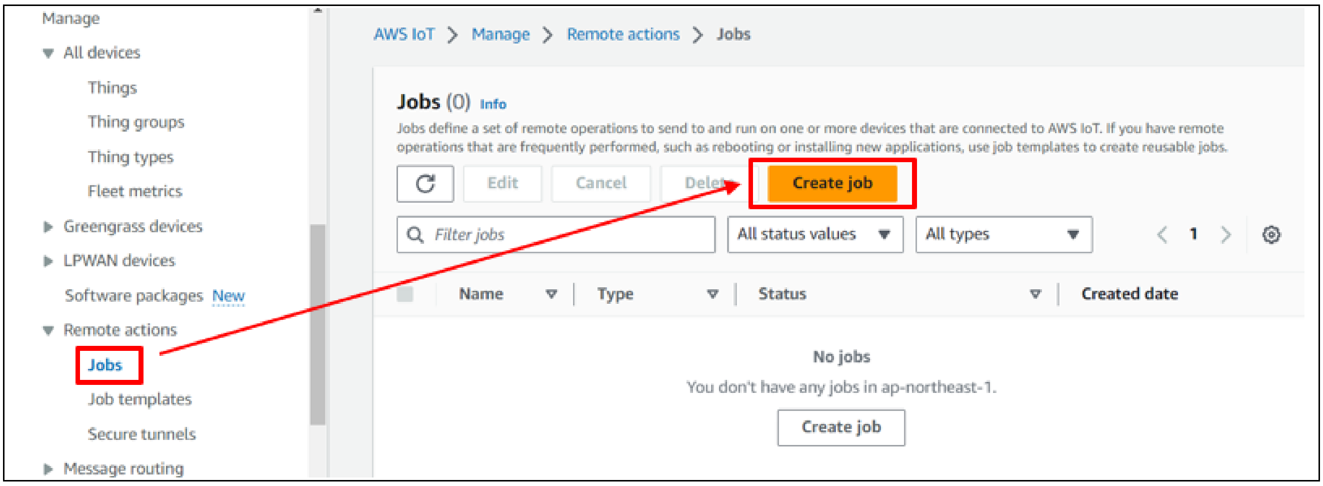
This command generates a file named user_093.rsu.

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.



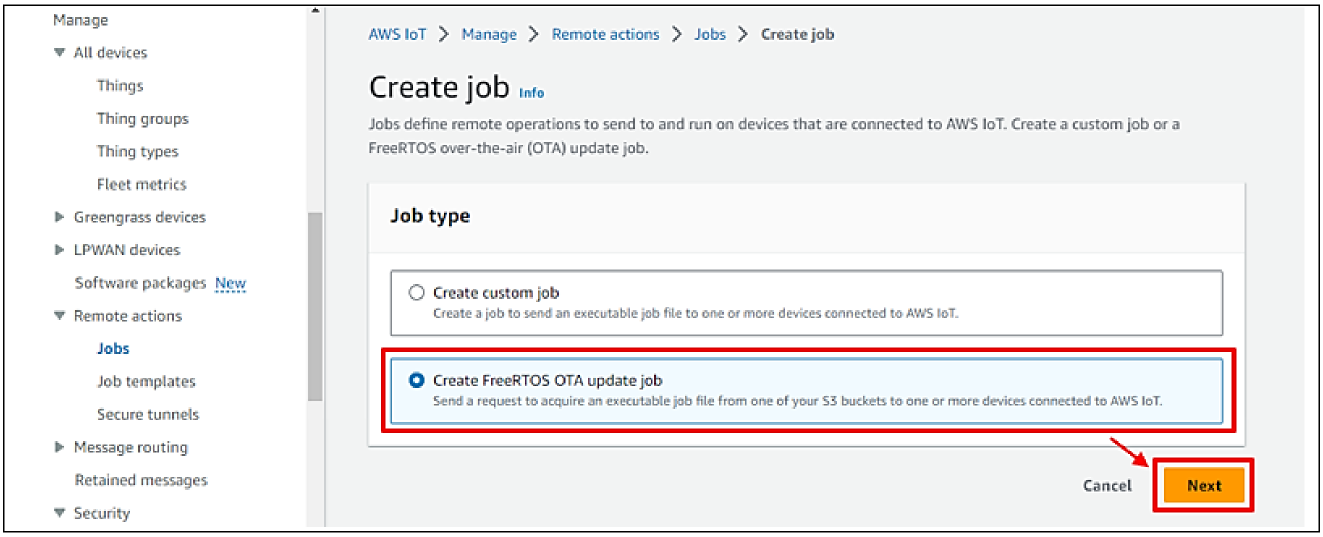
Jobs define a set of remote operations to send to and run on one or more devices that are connected to AWS IoT. If you have remote operations that are frequently performed, such as rebooting or installing new applications, use job templates to create reusable jobs.

You don't have any jobs in ap-northeast-1.

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

6.5.2.2 Creating FreeRTOS OTA Job Update

Select Create FreeRTOS OTA update job and then click Next.



Jobs define remote operations to send to and run on devices that are connected to AWS IoT. Create a custom job or a FreeRTOS over-the-air (OTA) update job.

○ Create custom job
Create a job to send an executable job file to one or more devices connected to AWS IoT.

● Create FreeRTOS OTA update job
Send a request to acquire an executable job file from one of your S3 buckets to one or more devices connected to AWS IoT.

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

6.5.2.3 Entering a Job Name

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

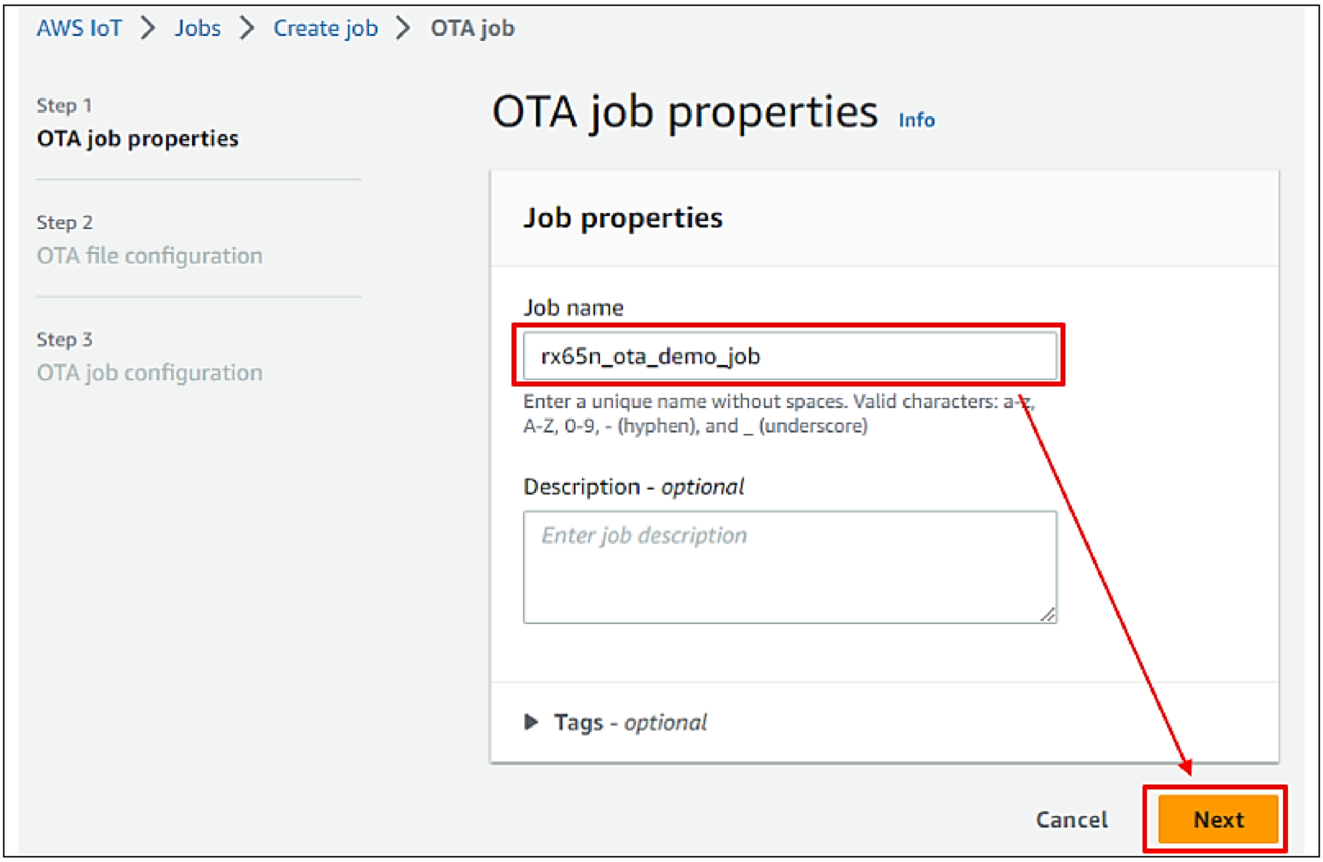


Figure 144. Enter OTA Job Name

6.5.2.4 Updating Devices

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

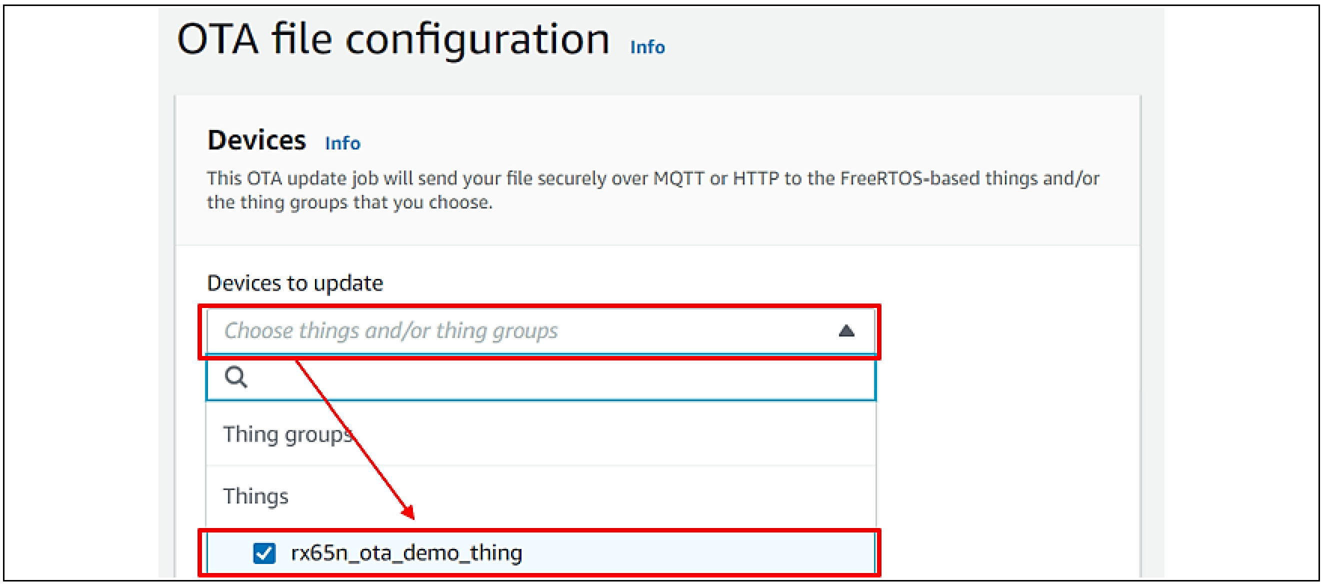
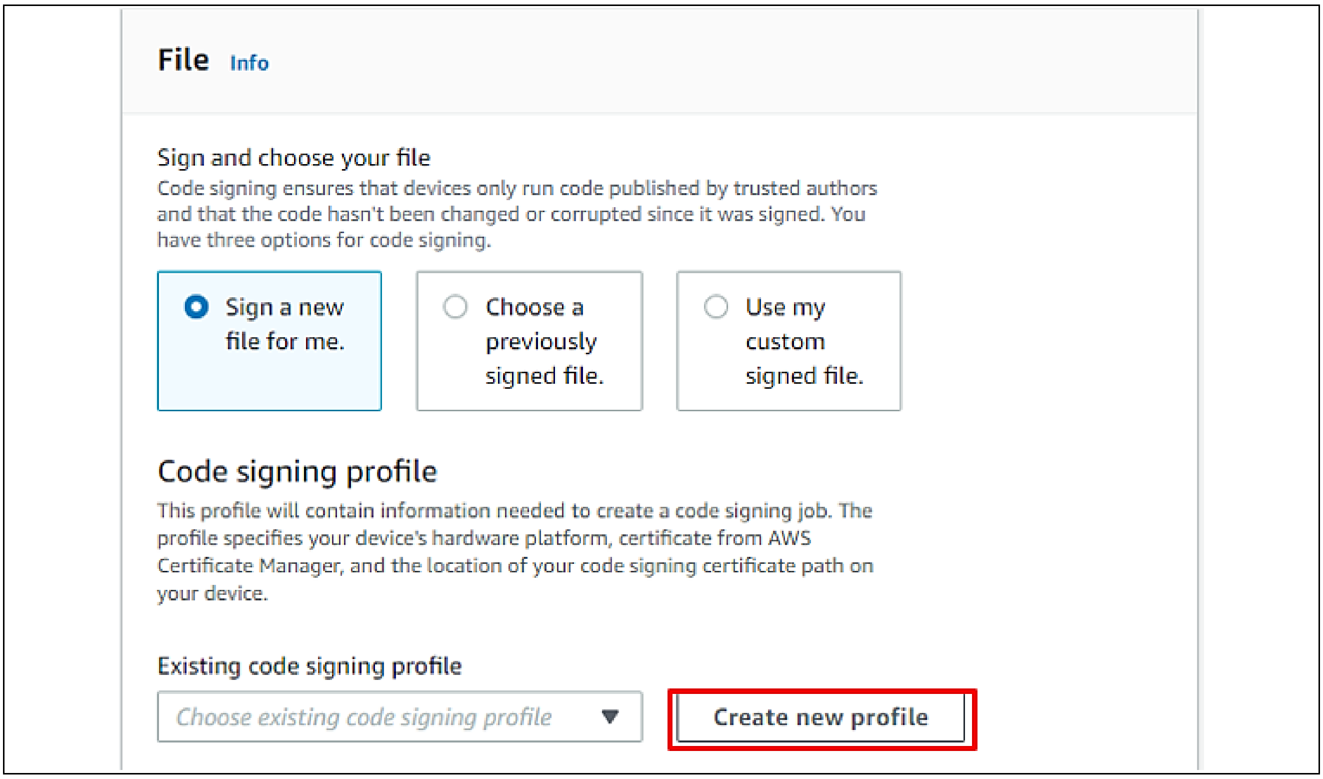

OTA file configuration Info
Devices Info
This OTA update job will send your file securely over MQTT or HTTP to the FreeRTOS-based things and/or the thing groups that you choose.
Devices to update
Choose things and/or thing groups
Thing groups
Things
☑ rx65n_ota_demo_thing

Figure 145. Choose Device to Update

6.5.2.5 Creating New Profile

Click **Create new profile**.



File Info

Sign and choose your file

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

● Sign a new file for me. ○ Choose a previously signed file. ○ Use my custom signed file.

Code signing profile

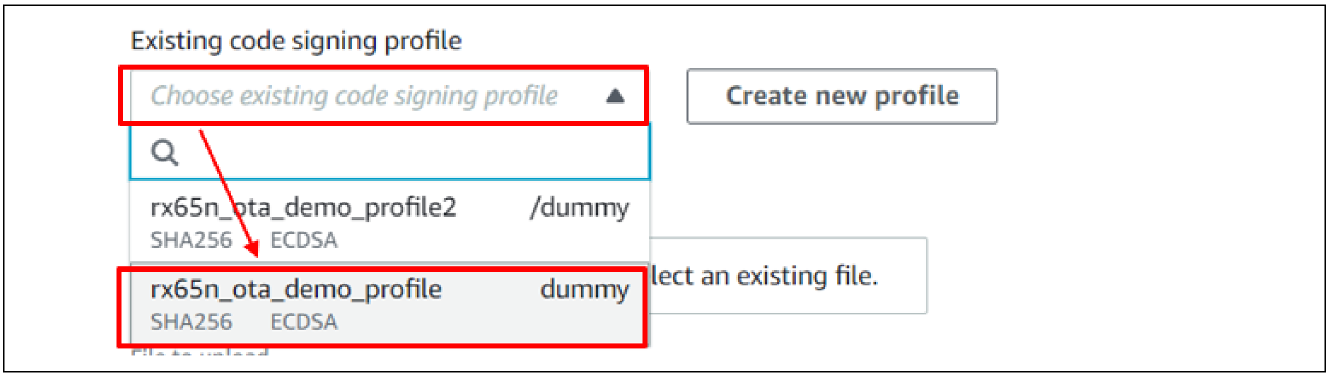
This profile will contain information needed to create a code signing job. The profile specifies your device's hardware platform, certificate from AWS Certificate Manager, and the location of your code signing certificate path on your device.

Existing code signing profile

Choose existing code signing profile ▼ | Create new profile

Figure 146. Create New Code Signing Profile

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



Existing code signing profile

Choose existing code signing profile ▲ | Create new profile

rx65n_ota_demo_profile2 /dummy
SHA256 ECDSA

rx65n_ota_demo_profile dummy
SHA256 ECDSA

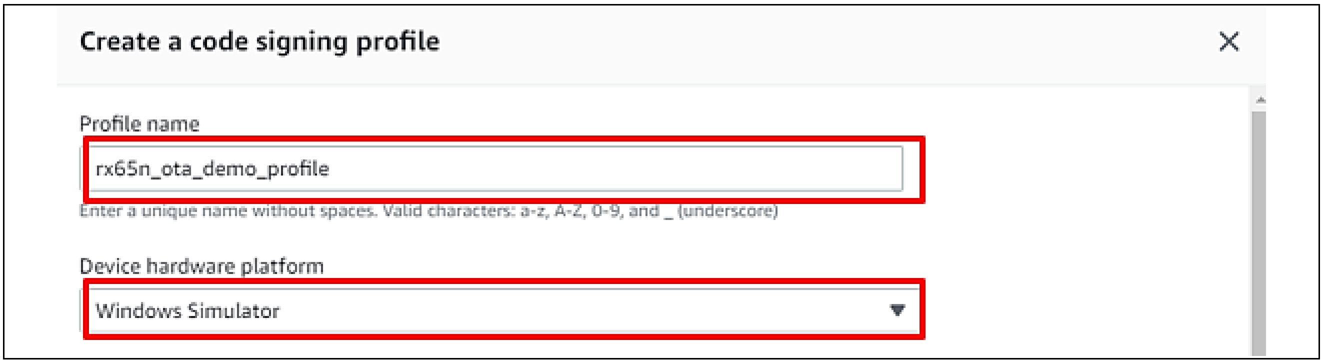
lect an existing file.

Figure 147. Choose Existing Code Signing Profile

6.5.2.6 Creating a Profile Naming

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

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



Create a code signing profile

Profile name

rx65n_ota_demo_profile

Enter a unique name without spaces. Valid characters: a-z, A-Z, 0-9, and _ (underscore)

Device hardware platform

Windows Simulator

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

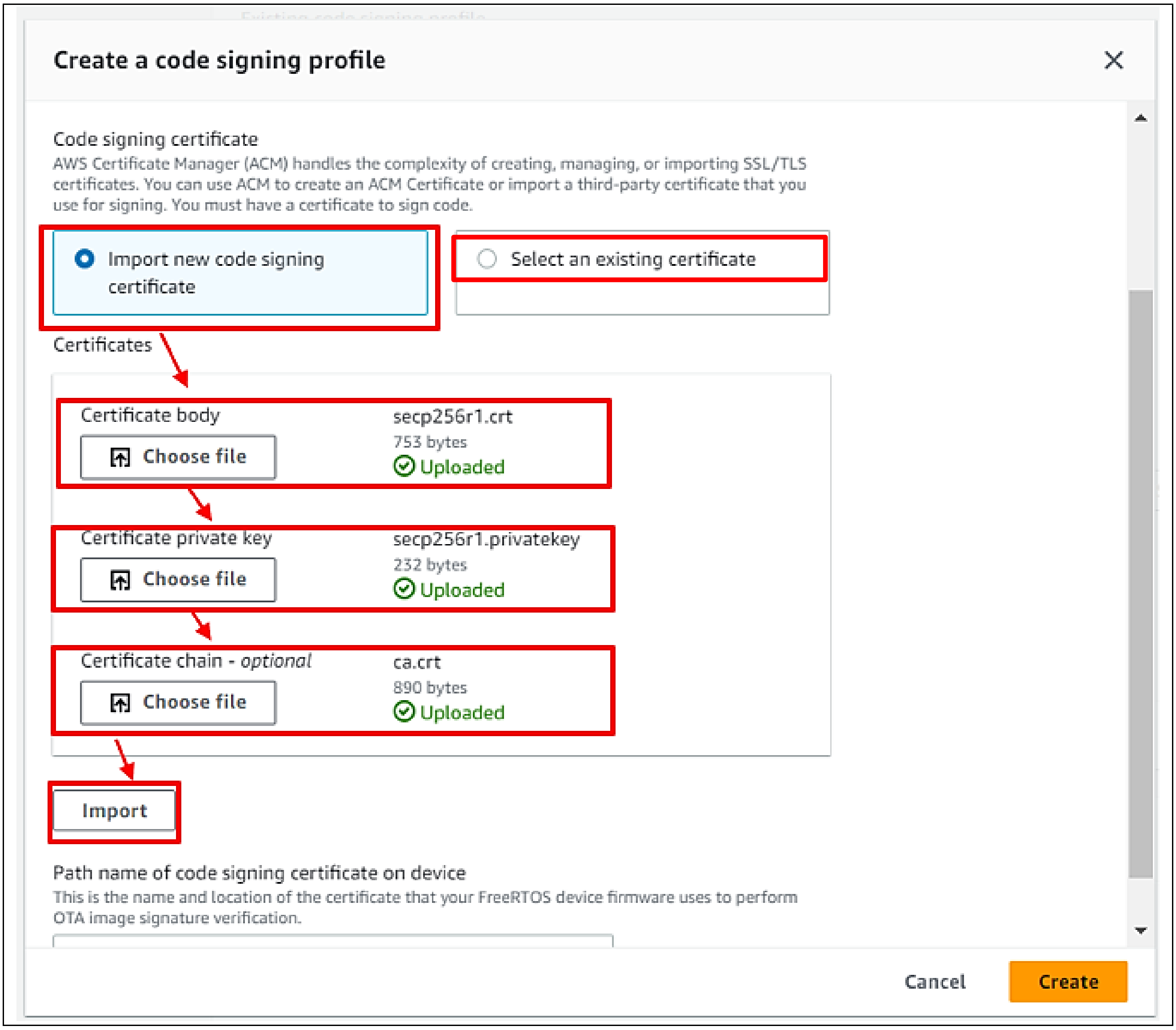
6.5.2.7 Creating a Profile: Importing Certificate

Create a profile (2): Import a certificate.

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

  Note: You have created the above files in **6.4**.

- Click **Import**



Create a code signing profile

Code signing certificate

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

● Import new code signing certificate

○ Select an existing certificate

Certificates

Certificate body: secp256r1.crt, 753 bytes, Uploaded

Certificate private key: secp256r1.privatekey, 232 bytes, Uploaded

Certificate chain - *optional*: ca.crt, 890 bytes, Uploaded

Path name of code signing certificate on device

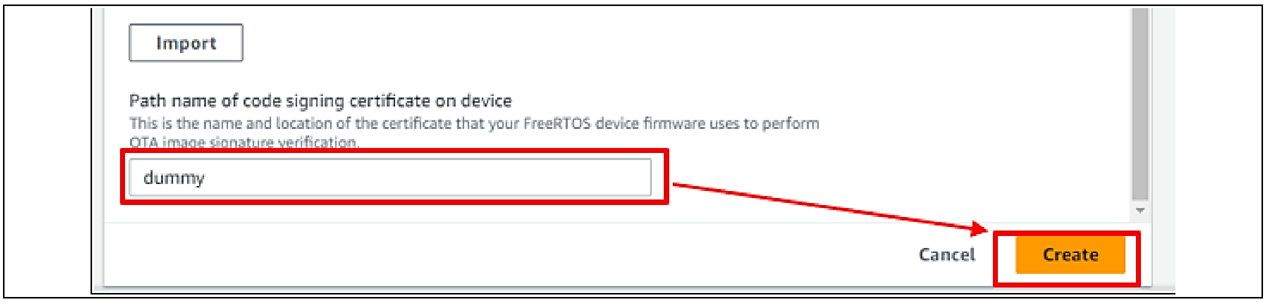
This is the name and location of the certificate that your FreeRTOS device firmware uses to perform OTA image signature verification.

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

6.5.2.8 Creating a Profile: Entering Path

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

You can enter any path. (Example: dummy)



Path name of code signing certificate on device

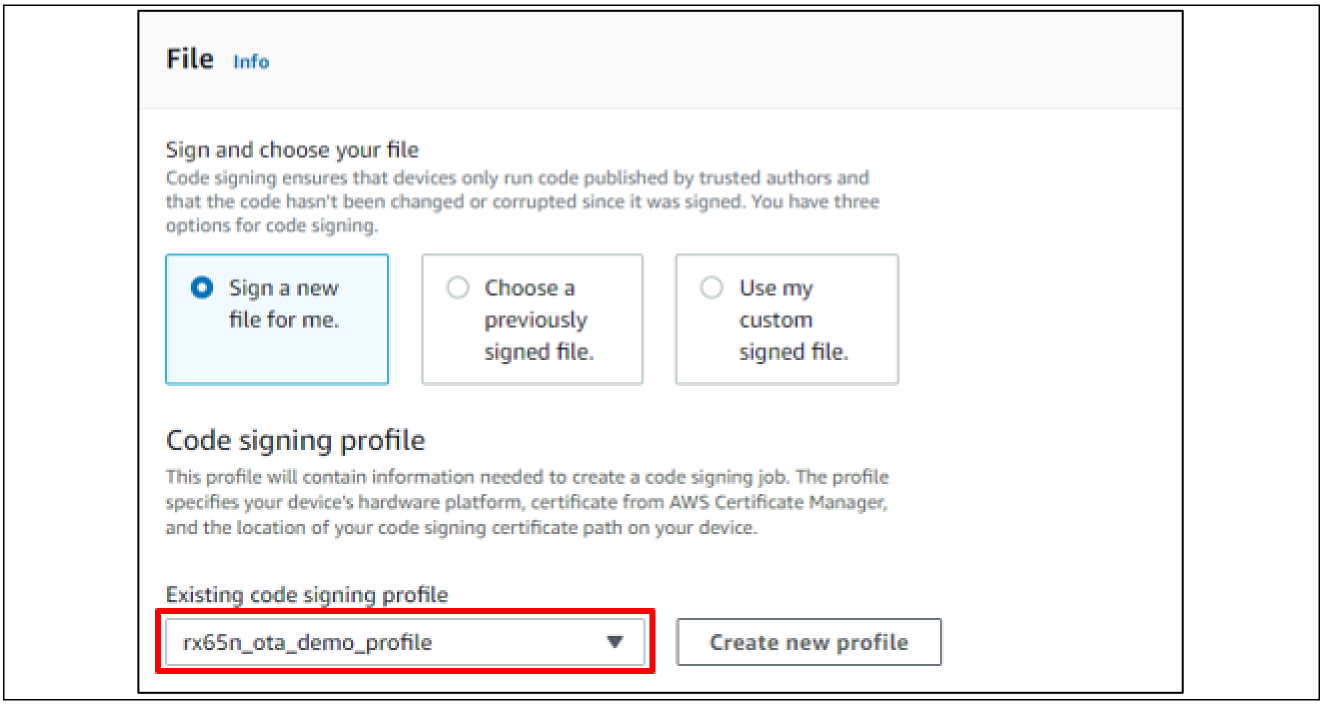
This is the name and location of the certificate that your FreeRTOS device firmware uses to perform OTA image signature verification.

dummy

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

6.5.2.9 Confirming Profile Name

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



File Info

Sign and choose your file

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

● Sign a new file for me.

○ Choose a previously signed file.

○ Use my custom signed file.

Code signing profile

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

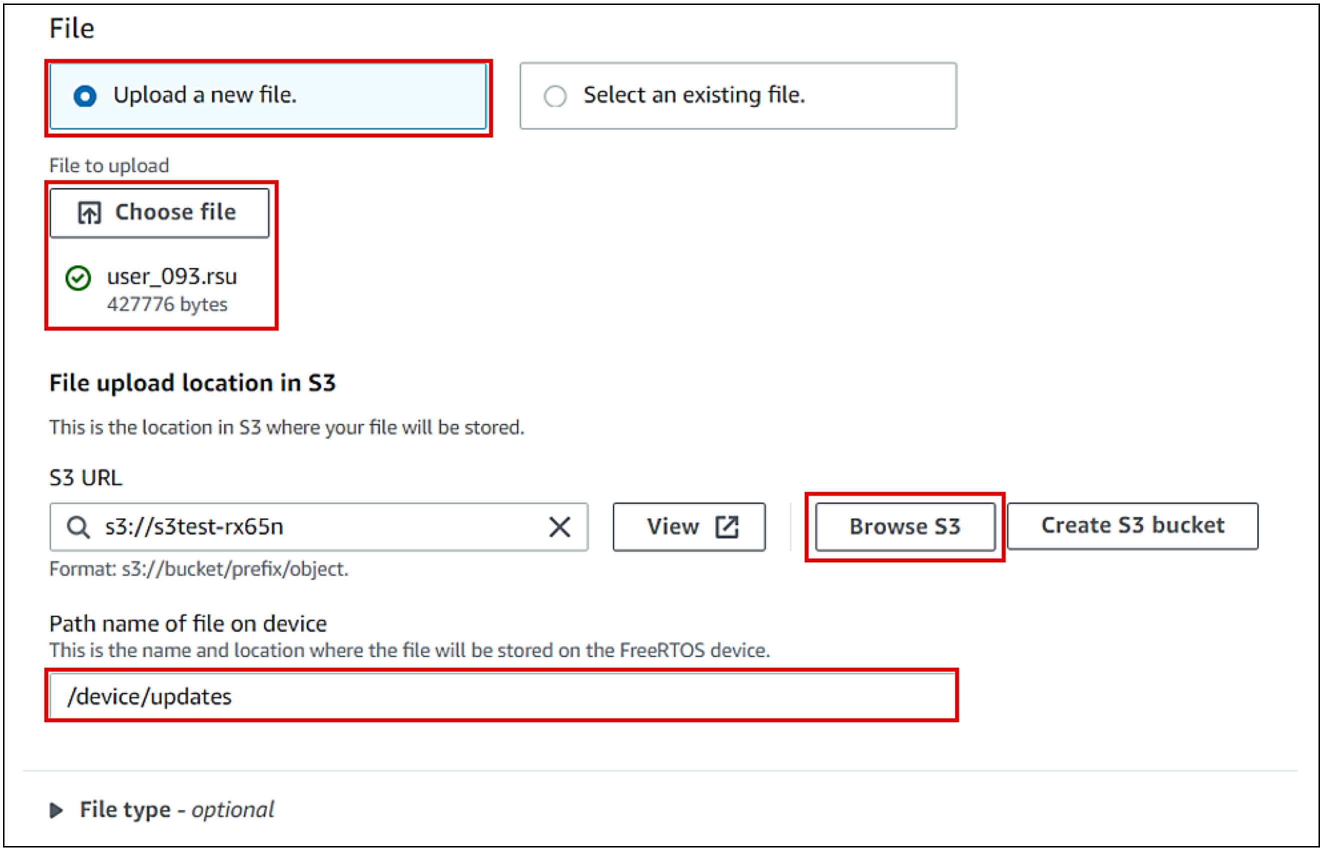
Existing code signing profile

rx65n_ota_demo_profile ▼

Figure 151. Choose Existing Code Signing Profile

6.5.2.10 Updating the Firmware

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


File

● Upload a new file.

○ Select an existing file.

File to upload

Choose file

user_093.rsu
427776 bytes

File upload location in S3

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

S3 URL

s3://s3test-rx65n

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

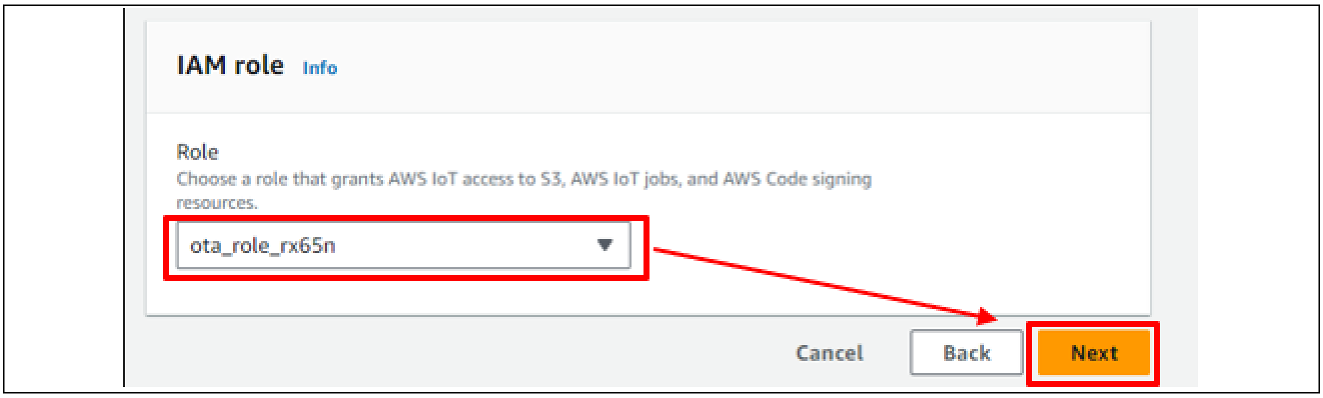
Path name of file on device
This is the name and location where the file will be stored on the FreeRTOS device.

/device/updates

Figure 152. Choose Firmware to Update

6.5.2.11 Choosing the Role for the Job

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

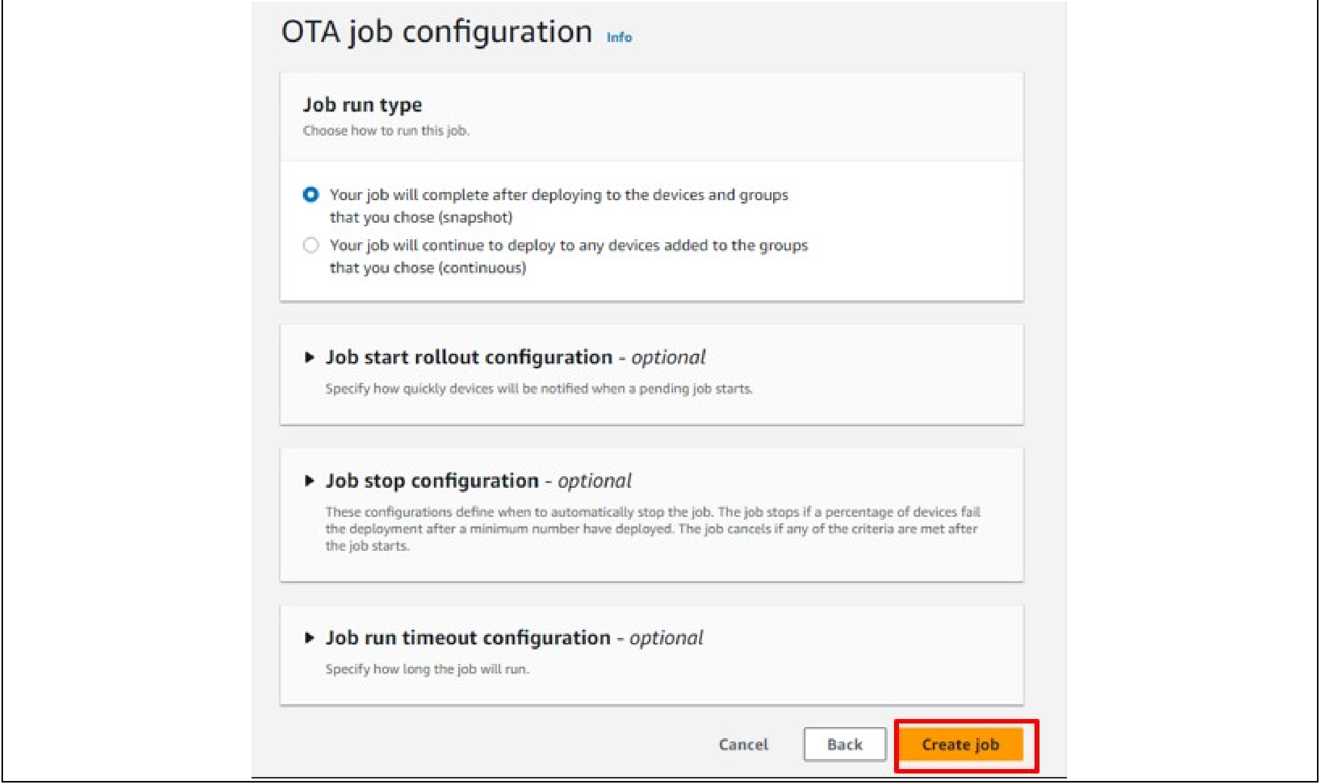

IAM role Info

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

ota_role_rx65n

Figure 153. Choose Role for Creating Job

Click **Create job**.



OTA job configuration Info

Job run type

Choose how to run this job.

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

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

▸ Job start rollout configuration - *optional*

Specify how quickly devices will be notified when a pending job starts.

▸ Job stop configuration - *optional*

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

▸ Job run timeout configuration - *optional*

Specify how long the job will run.

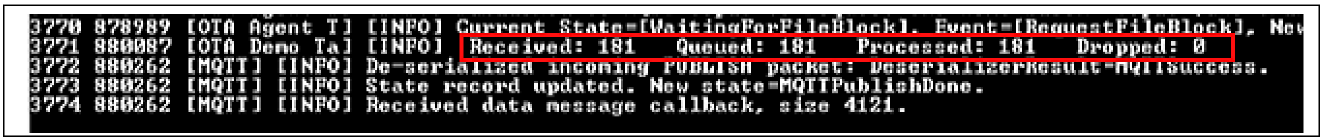
Figure 154. Create Job

6.5.2.12 Waiting until Firmware Reception is Complete

Wait until firmware reception is complete.

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

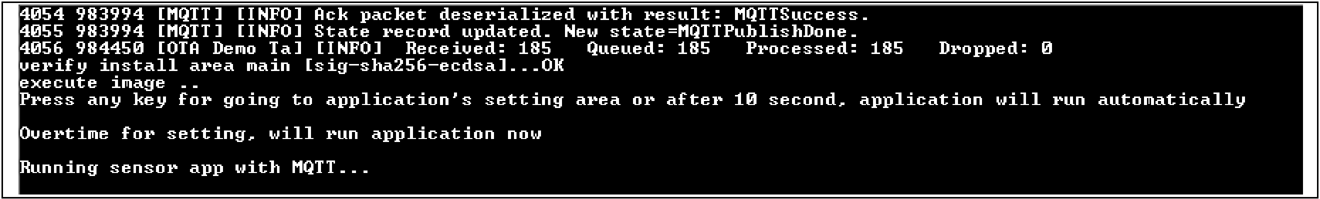
The Received counter is incremented when the reception starts.



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

Figure 155. OTA Job is Processed

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



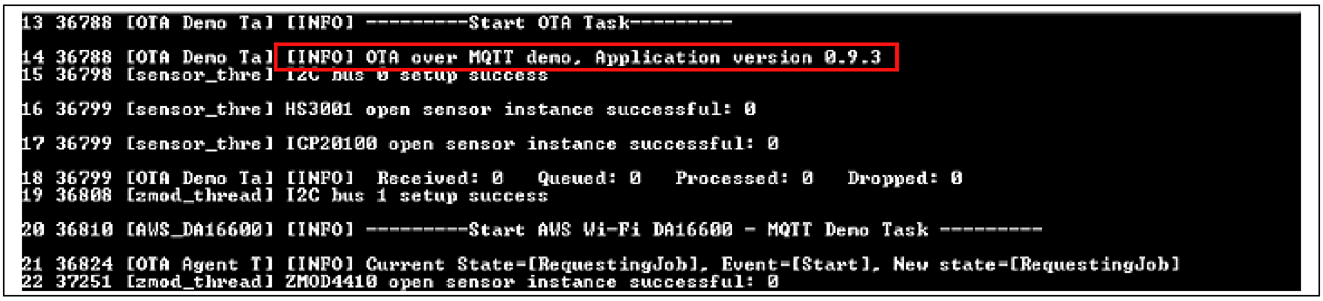
```
4054 983994 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
4055 983994 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
4056 984450 [OTA Demo Ta] [INFO]  Received: 185   Queued: 185   Processed: 185   Dropped: 0
verify install area main [sig-sha256-ecdsa]...OK
execute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

Overtime for setting, will run application now

Running sensor app with MQTT...
```

Figure 156. Device Reset and Update New Firmware

6.5.2.13 Confirming that the Firmware Version is a New Version

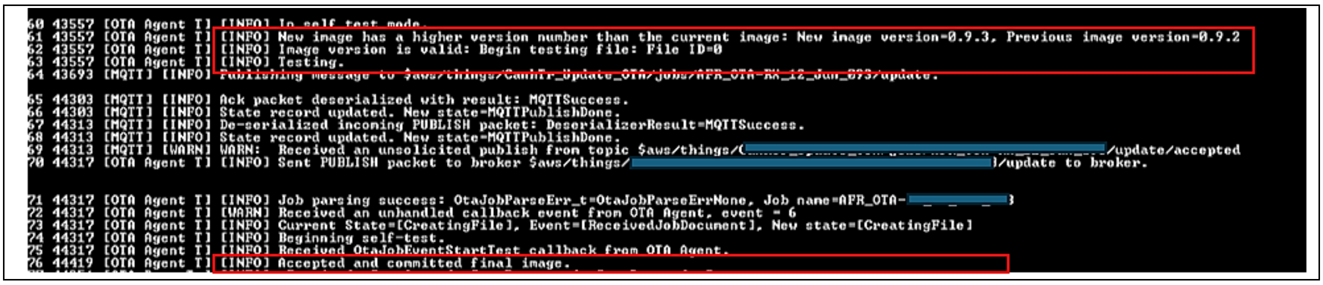
Example: 0.9.3 (updated at **6.5.1.1**)

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



```
13 36788 [OTA Demo Ta] [INFO] ---------Start OTA Task---------

14 36788 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.3
15 36798 [sensor_thre] I2C bus 0 setup success

16 36799 [sensor_thre] HS3001 open sensor instance successful: 0

17 36799 [sensor_thre] ICP20100 open sensor instance successful: 0

18 36799 [OTA Demo Ta] [INFO] Received: 0   Queued: 0   Processed: 0   Dropped: 0
19 36808 [zmod_thread] I2C bus 1 setup success

20 36810 [AWS_DA16600] [INFO] ---------Start AWS Wi-Fi DA16600 - MQTT Demo Task ---------

21 36824 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
22 37251 [zmod_thread] ZMOD4410 open sensor instance successful: 0
```

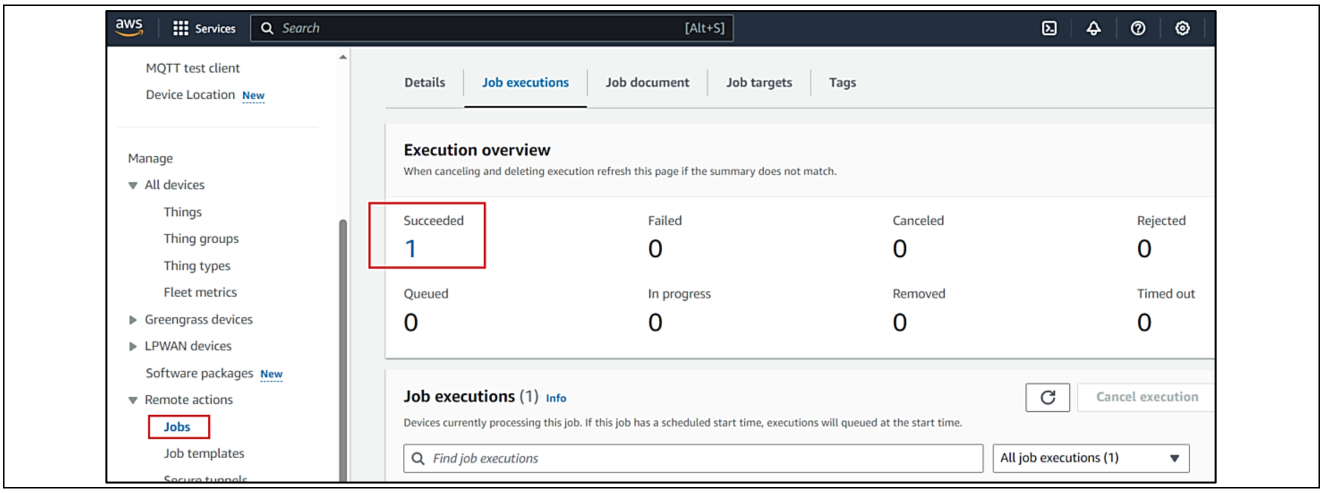
Figure 157. Firmware with New Version is Updated



```
60 43557 [OTA Agent T] [INFO] In self test mode.
61 43557 [OTA Agent T] [INFO] New image has a higher version number than the current image: New image version=0.9.3, Previous image version=0.9.2
62 43557 [OTA Agent T] [INFO] Image version is valid: Begin testing file: File ID=0
63 43557 [OTA Agent T] [INFO] Testing.
64 43693 [MQTT] [INFO] Publishing message to $aws/things/CanHr_Update_OTA/jobs/AFR_OTA-RX_12_Jun_093/update.

65 44303 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
66 44303 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
67 44313 [MQTT] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
68 44313 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
69 44313 [MQTT] [WARN] WARN:  Received an unsolicited publish from topic $aws/things/[redacted]/update/accepted
70 44317 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/[redacted]/update to broker.

71 44317 [OTA Agent T] [INFO] Job parsing success: OtaJobParseErr_t=OtaJobParseErrNone, Job name=AFR_OTA-[redacted]
72 44317 [OTA Agent T] [WARN] Received an unhandled callback event from OTA Agent, event = 6
73 44317 [OTA Agent T] [INFO] Current State=[CreatingFile], Event=[ReceivedJobDocument], New state=[CreatingFile]
74 44317 [OTA Agent T] [INFO] Beginning self-test.
75 44317 [OTA Agent T] [INFO] Received OtaJobEventStartTest callback from OTA Agent.
76 44419 [OTA Agent T] [INFO] Accepted and committed final image.
```

Figure 158 Test and accept new firmware

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



Execution overview

When canceling and deleting execution refresh this page if the summary does not match.

Succeeded	Failed	Canceled	Rejected
1	0	0	0
Queued	**In progress**	**Removed**	**Timed out**
0	0	0	0

Job executions (1)

Devices currently processing this job. If this job has a scheduled start time, executions will queued at the start time.

Figure 159. Job’s Status in AWS Portal

7. Fleet Provisioning

7.1 Overview Fleet Provisioning

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

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

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

1. Provisioning by claim (approach using provisioning claim certificates)

2. Provisioning by a trusted user (mobile or web app user, etc.)

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

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

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

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

Advantages

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

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

Disadvantages

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

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

7.2 Setting up AWS for Fleet Provisioning

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

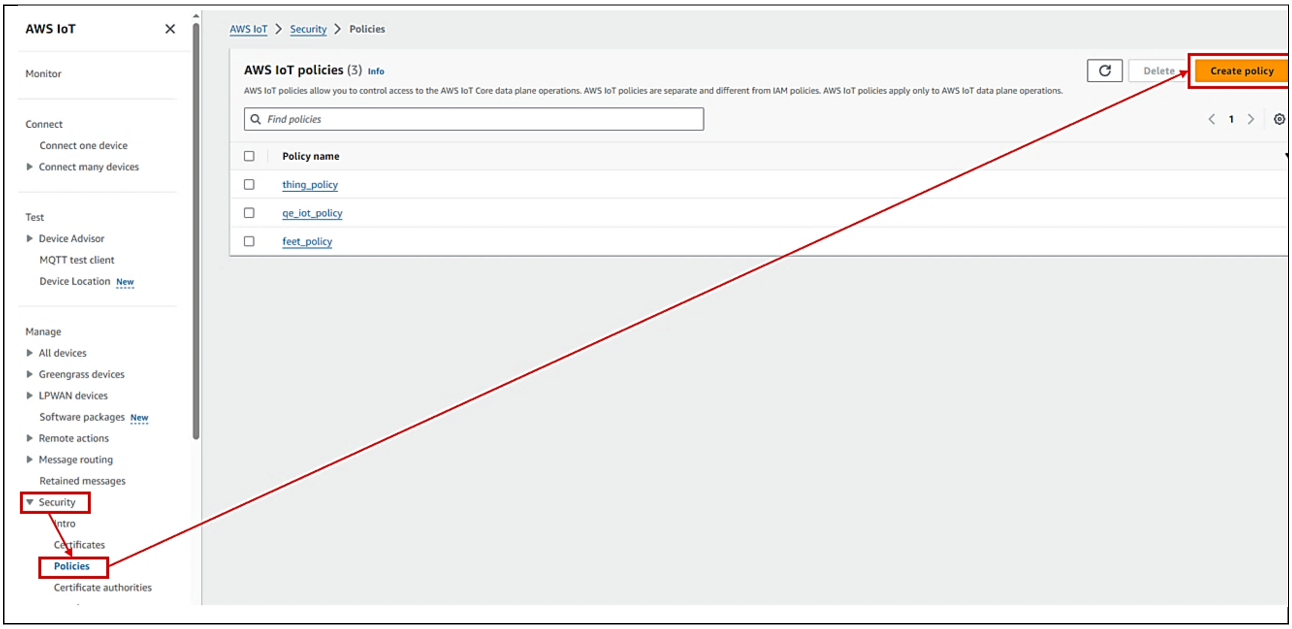
1. Policy settings

2. Generating a claim certificate and claim key pair

3. Creating a fleet provisioning template

7.2.1 Policy Settings

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



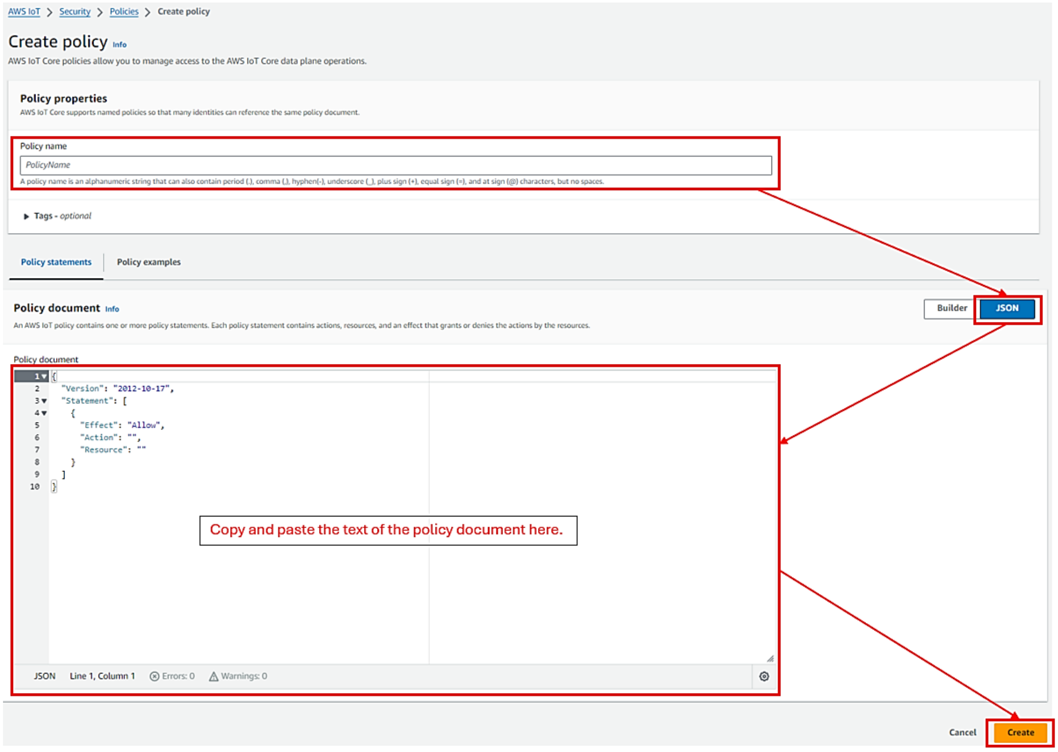
AWS IoT policies (3) Info

AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

□	Policy name
□	thing_policy
□	qe_iot_policy
□	feet_policy

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

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



Create policy

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

Policy properties

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name: PolicyName

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

Policy document

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

Policy document

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

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

Copy and paste the text of the policy document here.

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

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

・Change “us-east-1” to match the region used.

・Change <account id> to your account ID (account ID is the 12-digit number after @ that is displayed by clicking on the account name in the upper right corner, excluding the hyphen)

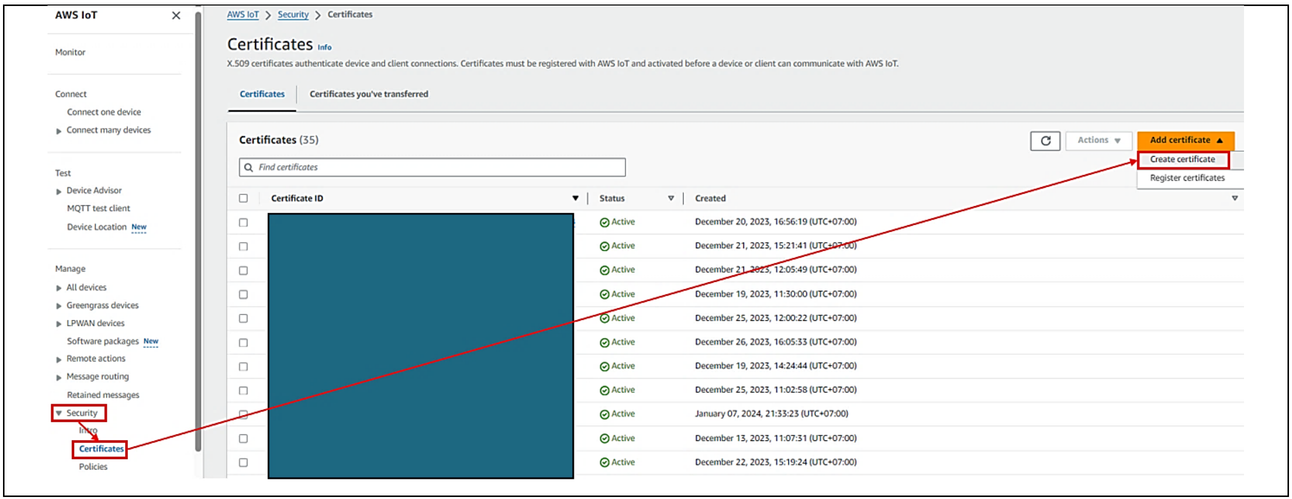
Table 8. Policy Document

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

7.2.2 Generating a Claim Certificate and Claim Key Pair

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

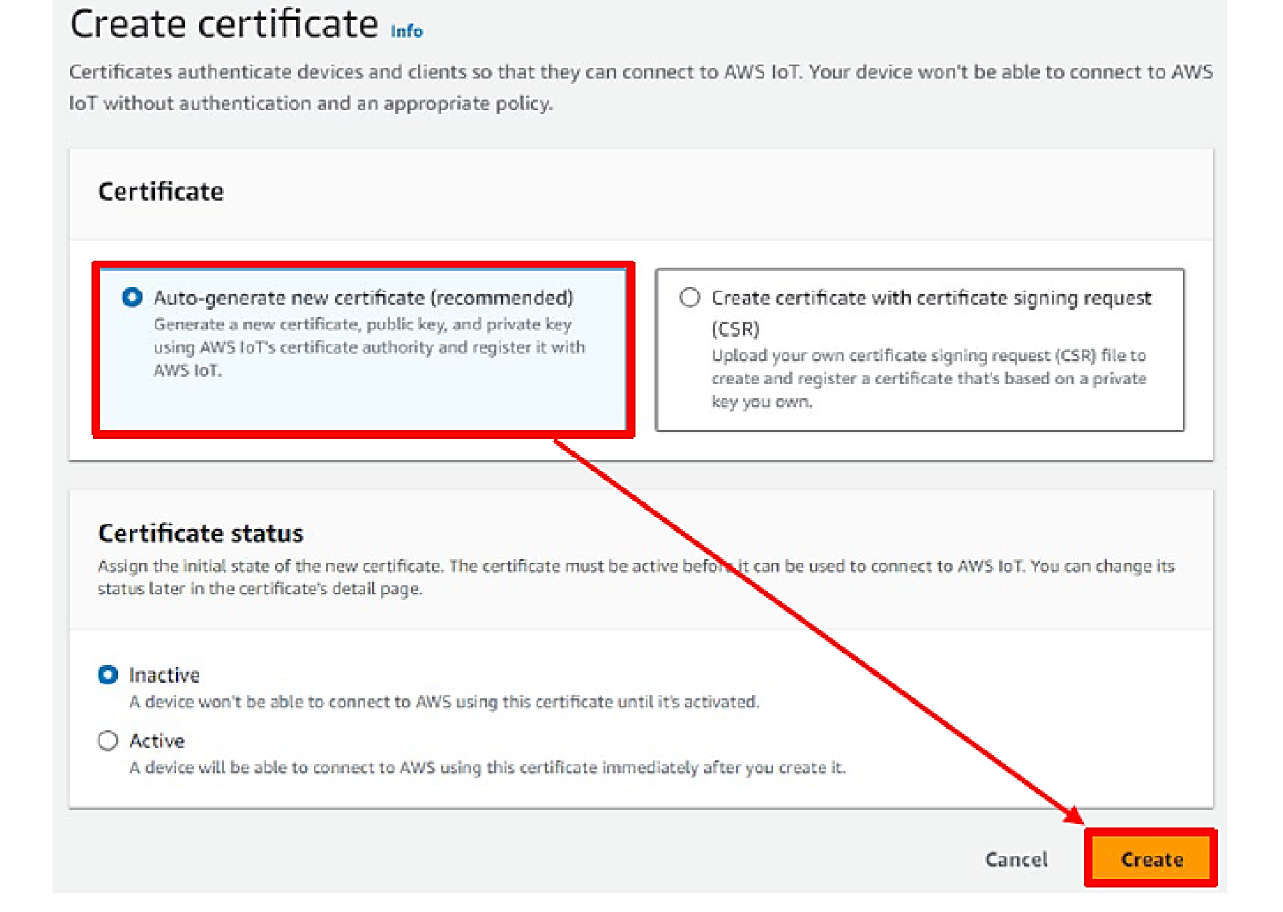
Select **Security** → **Certificates** and then click **Add certificate** → **Create certificate**.



X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with AWS IoT.

□	Certificate ID	Status	Created
□	[redacted]	Active	December 20, 2023, 16:56:19 (UTC+07:00)
□	[redacted]	Active	December 21, 2023, 15:21:41 (UTC+07:00)
□	[redacted]	Active	December 21, 2023, 12:05:49 (UTC+07:00)
□	[redacted]	Active	December 19, 2023, 11:30:00 (UTC+07:00)
□	[redacted]	Active	December 25, 2023, 12:00:22 (UTC+07:00)
□	[redacted]	Active	December 26, 2023, 16:05:33 (UTC+07:00)
□	[redacted]	Active	December 19, 2023, 14:24:44 (UTC+07:00)
□	[redacted]	Active	December 25, 2023, 11:02:58 (UTC+07:00)
□	[redacted]	Active	January 07, 2024, 21:33:23 (UTC+07:00)
□	[redacted]	Active	December 13, 2023, 11:07:31 (UTC+07:00)
□	[redacted]	Active	December 22, 2023, 15:19:24 (UTC+07:00)

Figure 162. Create a Certificate



Certificates authenticate devices and clients so that they can connect to AWS IoT. Your device won't be able to connect to AWS IoT without authentication and an appropriate policy.

● Auto-generate new certificate (recommended)
Generate a new certificate, public key, and private key using AWS IoT's certificate authority and register it with AWS IoT.

○ Create certificate with certificate signing request (CSR)
Upload your own certificate signing request (CSR) file to create and register a certificate that's based on a private key you own.

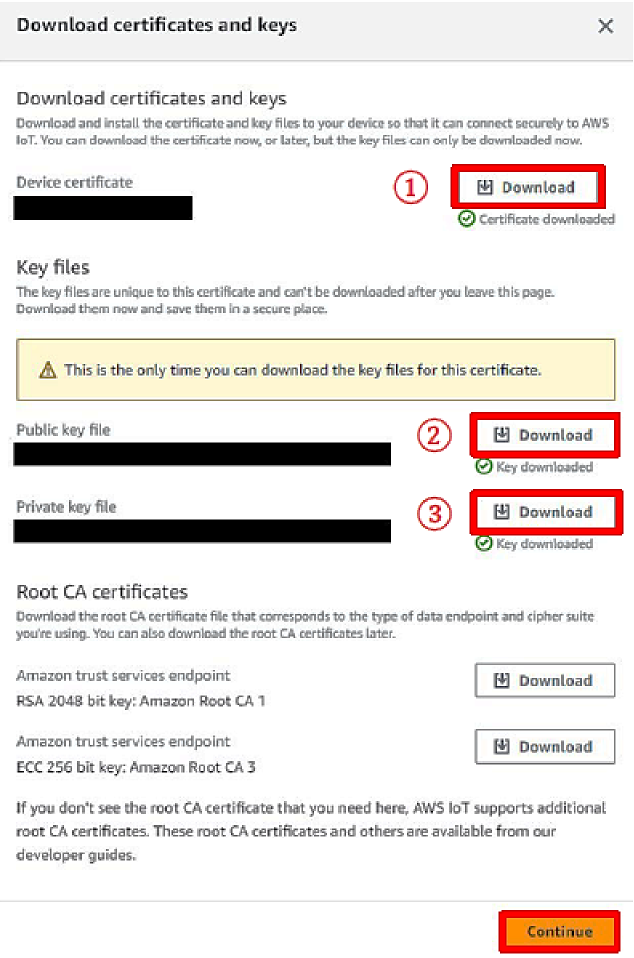
Assign the initial state of the new certificate. The certificate must be active before it can be used to connect to AWS IoT. You can change its status later in the certificate's detail page.

● Inactive
A device won't be able to connect to AWS using this certificate until it's activated.

○ Active
A device will be able to connect to AWS using this certificate immediately after you create it.

Figure 163. Creating a Certificate Automatically

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



Download certificates and keys ✕

Download certificates and keys

Download and install the certificate and key files to your device so that it can connect securely to AWS IoT. You can download the certificate now, or later, but the key files can only be downloaded now.

Device certificate

[redacted]

① Download

✓ Certificate downloaded

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

Public key file

[redacted]

② Download

✓ Key downloaded

Private key file

[redacted]

③ Download

✓ Key downloaded

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1 — Download

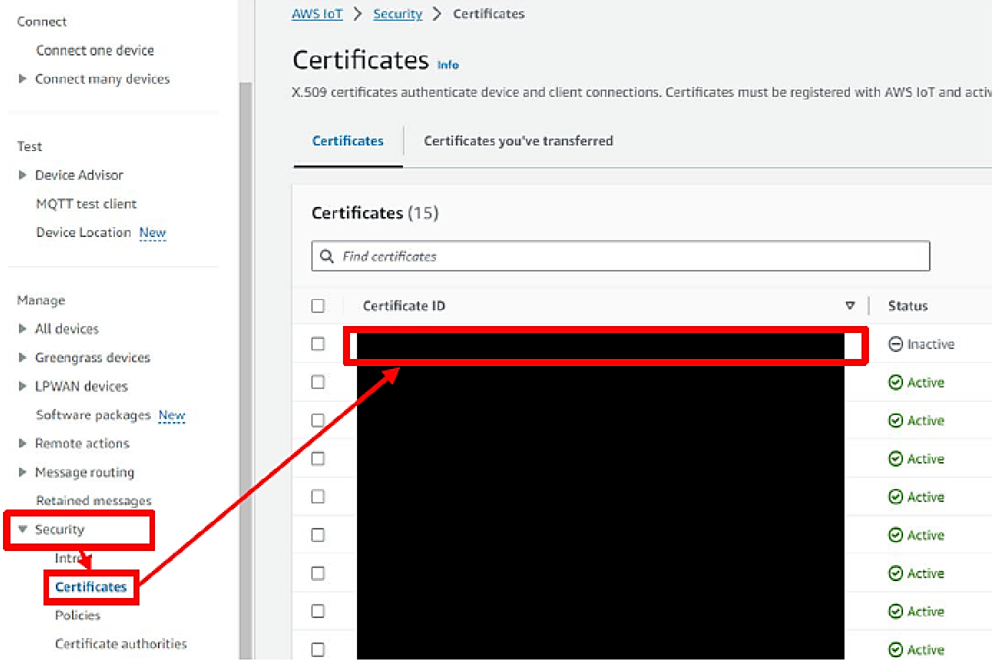
Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3 — Download

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available from our developer guides.

Continue

Figure 164. Downloading the Certificate and Key Pair

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



AWS IoT > Security > Certificates

Certificates Info

X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activ

Certificates | Certificates you've transferred

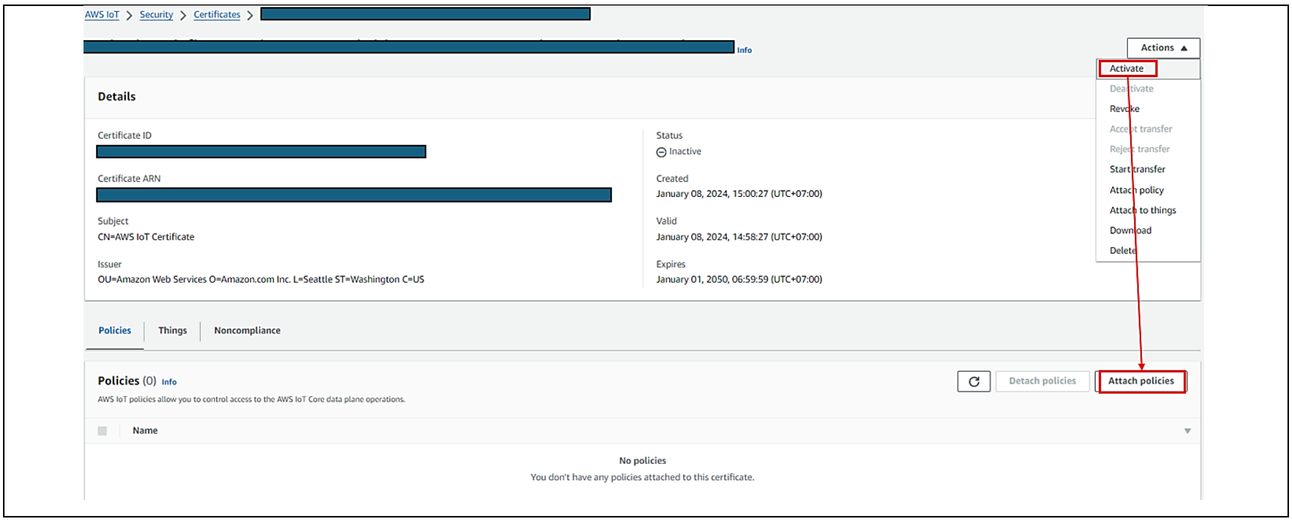
Certificates (15)

Find certificates

□	Certificate ID	Status
□	[redacted]	⊖ Inactive
□	[redacted]	✓ Active
□	[redacted]	✓ Active
□	[redacted]	✓ Active
□	[redacted]	✓ Active
□	[redacted]	✓ Active
□	[redacted]	✓ Active
□	[redacted]	✓ Active
□	[redacted]	✓ Active

Figure 165. Certificate Settings

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



Details

Certificate ID
[redacted]

Certificate ARN
[redacted]

Subject
CN=AWS IoT Certificate

Issuer
OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US

Status
⊖ Inactive

Created
January 08, 2024, 15:00:27 (UTC+07:00)

Valid
January 08, 2024, 14:58:27 (UTC+07:00)

Expires
January 01, 2050, 06:59:59 (UTC+07:00)

Policies (0) Info
AWS IoT policies allow you to control access to the AWS IoT Core data plane operations.

Name

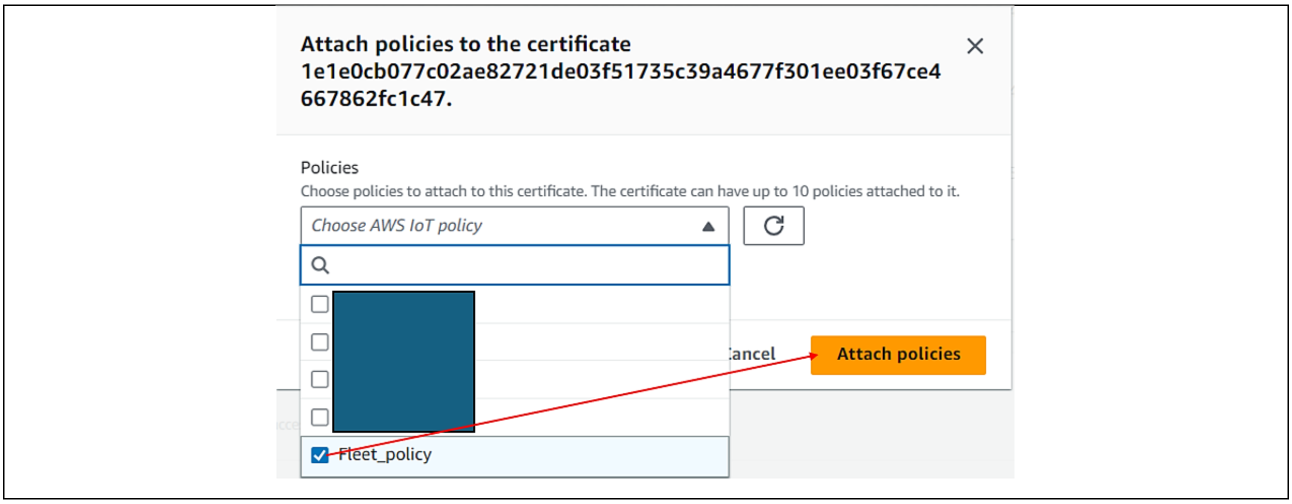
No policies
You don't have any policies attached to this certificate.

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

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

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

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



Attach policies to the certificate 1e1e0cb077c02ae82721de03f51735c39a4677f301ee03f67ce4667862fc1c47.

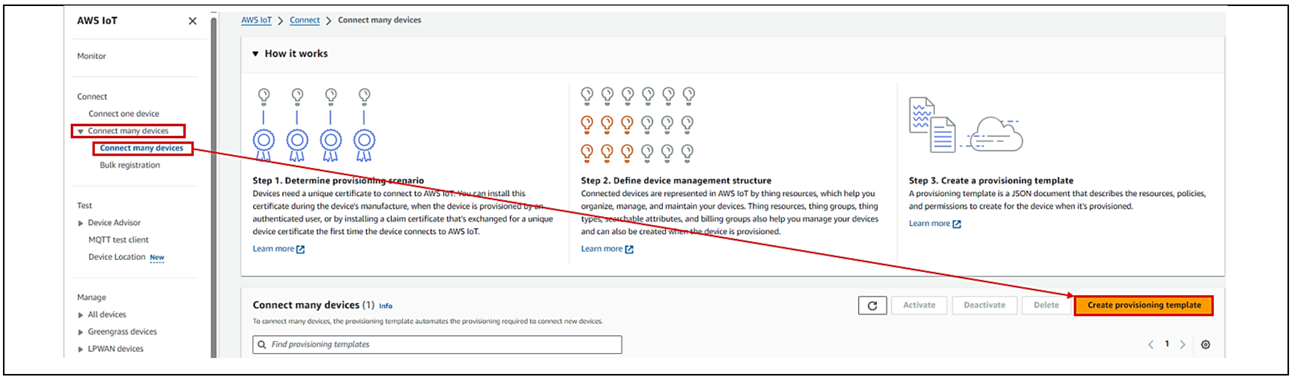
Policies
Choose policies to attach to this certificate. The certificate can have up to 10 policies attached to it.

□ [redacted]
□
□
□
☑ Fleet_policy

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

7.2.3 Creating a Fleet Provisioning Template

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



Step 1. Determine provisioning scenario
Devices need a unique certificate to connect to AWS IoT. You can install this certificate during the device's manufacture, when the device is provisioned by an authenticated user, or by installing a claim certificate that's exchanged for a unique device certificate the first time the device connects to AWS IoT.

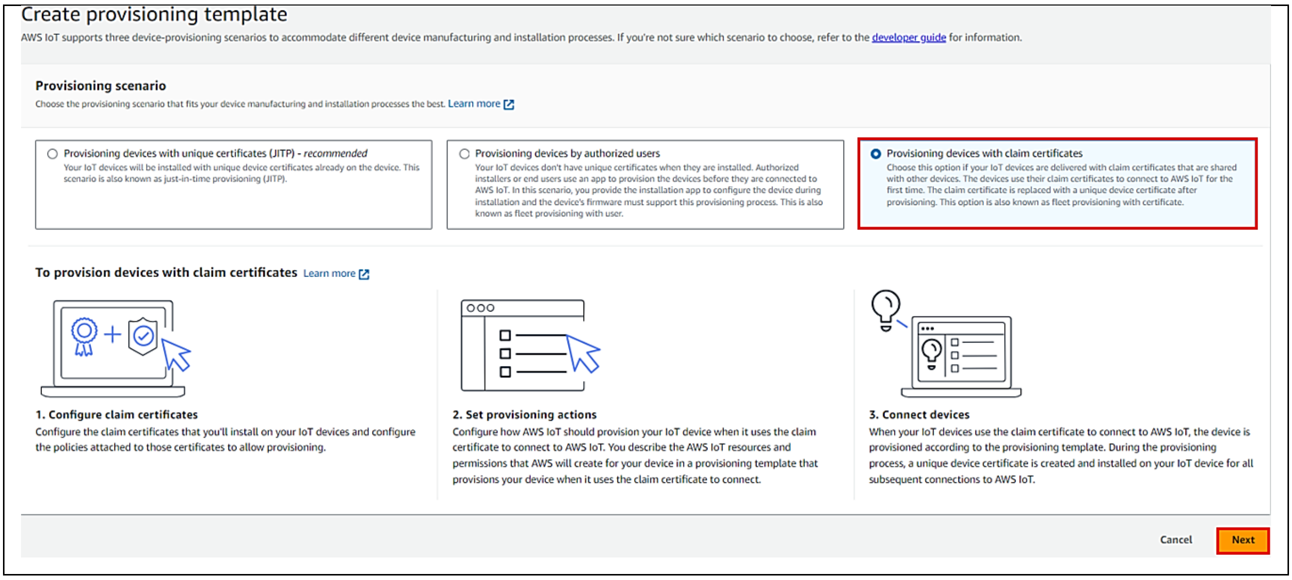
Step 2. Define device management structure
Connected devices are represented in AWS IoT by thing resources, which help you organize, manage, and maintain your devices. Thing resources, thing groups, thing types, searchable attributes, and billing groups also help you manage your devices and can also be created when the device is provisioned.

Step 3. Create a provisioning template
A provisioning template is a JSON document that describes the resources, policies, and permissions to create for the device when it's provisioned.

Connect many devices (1) Info
To connect many devices, the provisioning template automates the provisioning required to connect new devices.

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

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



Create provisioning template

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

Provisioning scenario

Choose the provisioning scenario that fits your device manufacturing and installation processes the best. Learn more

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

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

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

To provision devices with claim certificates Learn more

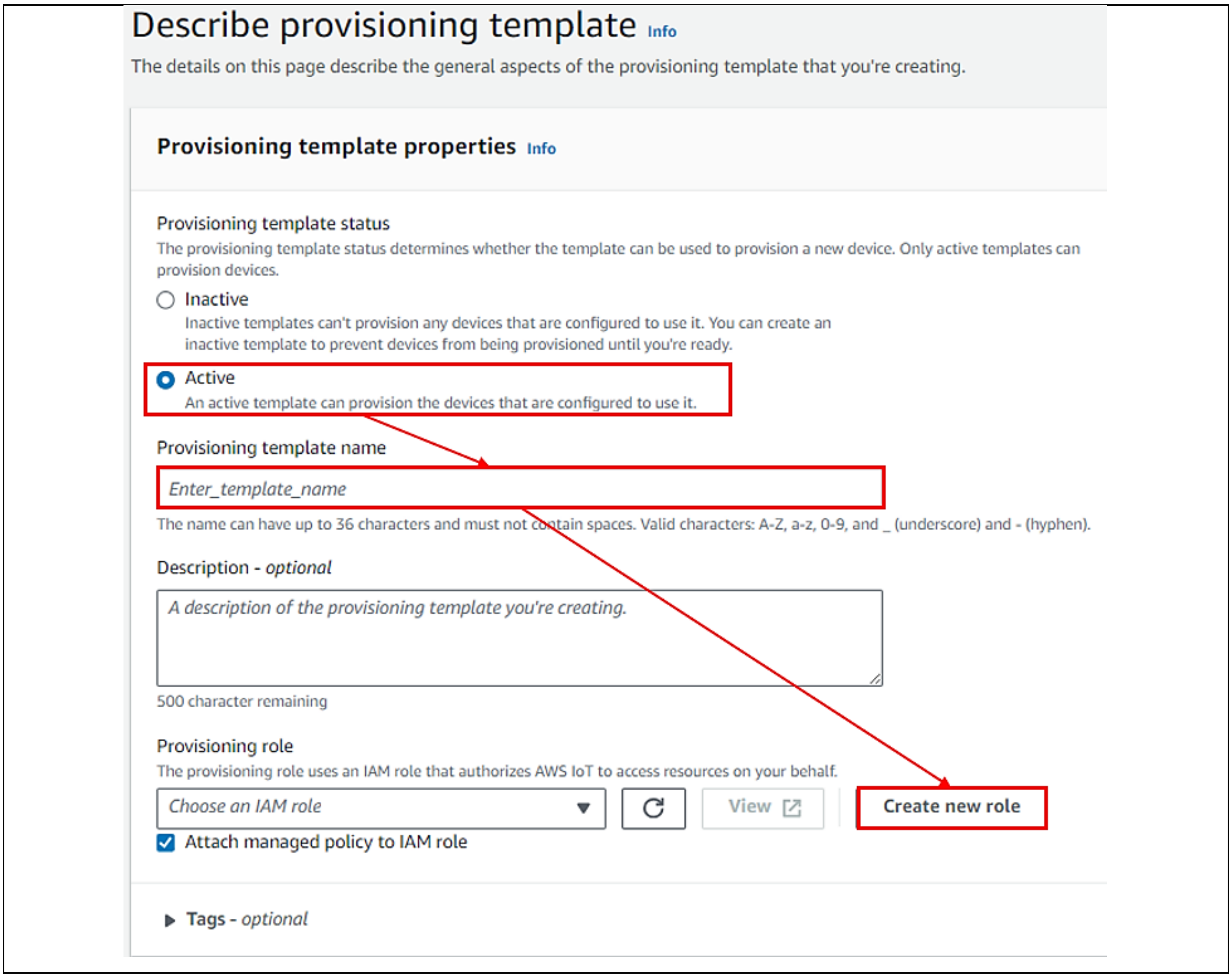
1. Configure claim certificates
Configure the claim certificates that you'll install on your IoT devices and configure the policies attached to those certificates to allow provisioning.

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

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

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

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



Describe provisioning template Info

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

Provisioning template properties Info

Provisioning template status

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

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

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

Provisioning template name

Enter_template_name

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

Description - *optional*

A description of the provisioning template you're creating.

500 character remaining

Provisioning role

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

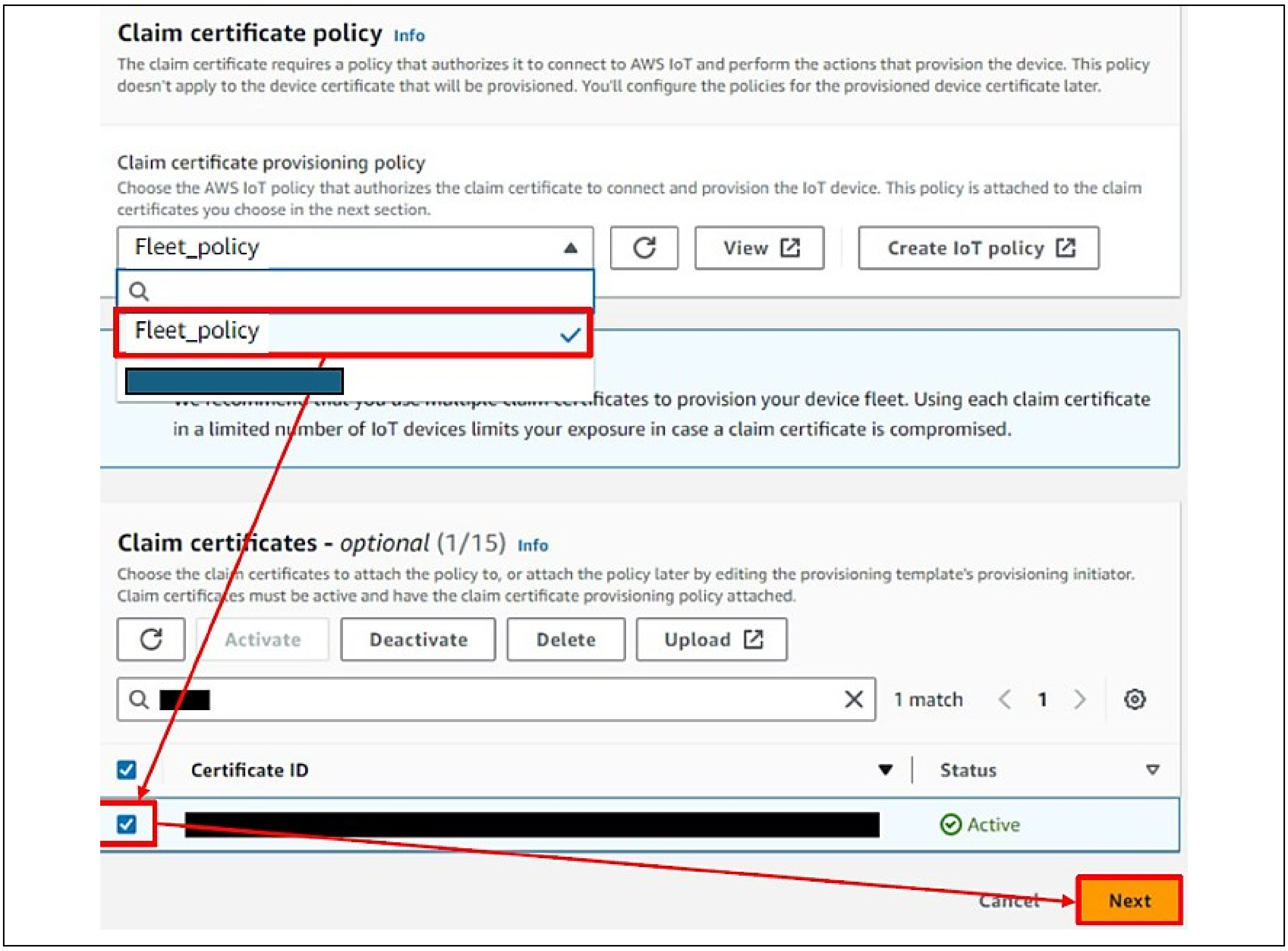
Choose an IAM role

☑ Attach managed policy to IAM role

▶ Tags - *optional*

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

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



Claim certificate policy Info

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

Claim certificate provisioning policy

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

Fleet_policy ▲ | ↻ | View | Create IoT policy

Fleet_policy ✓

[redacted]

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

Claim certificates - *optional* (1/15) Info

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

↻ | Activate | Deactivate | Delete | Upload

[redacted] ✕ 1 match < 1 >

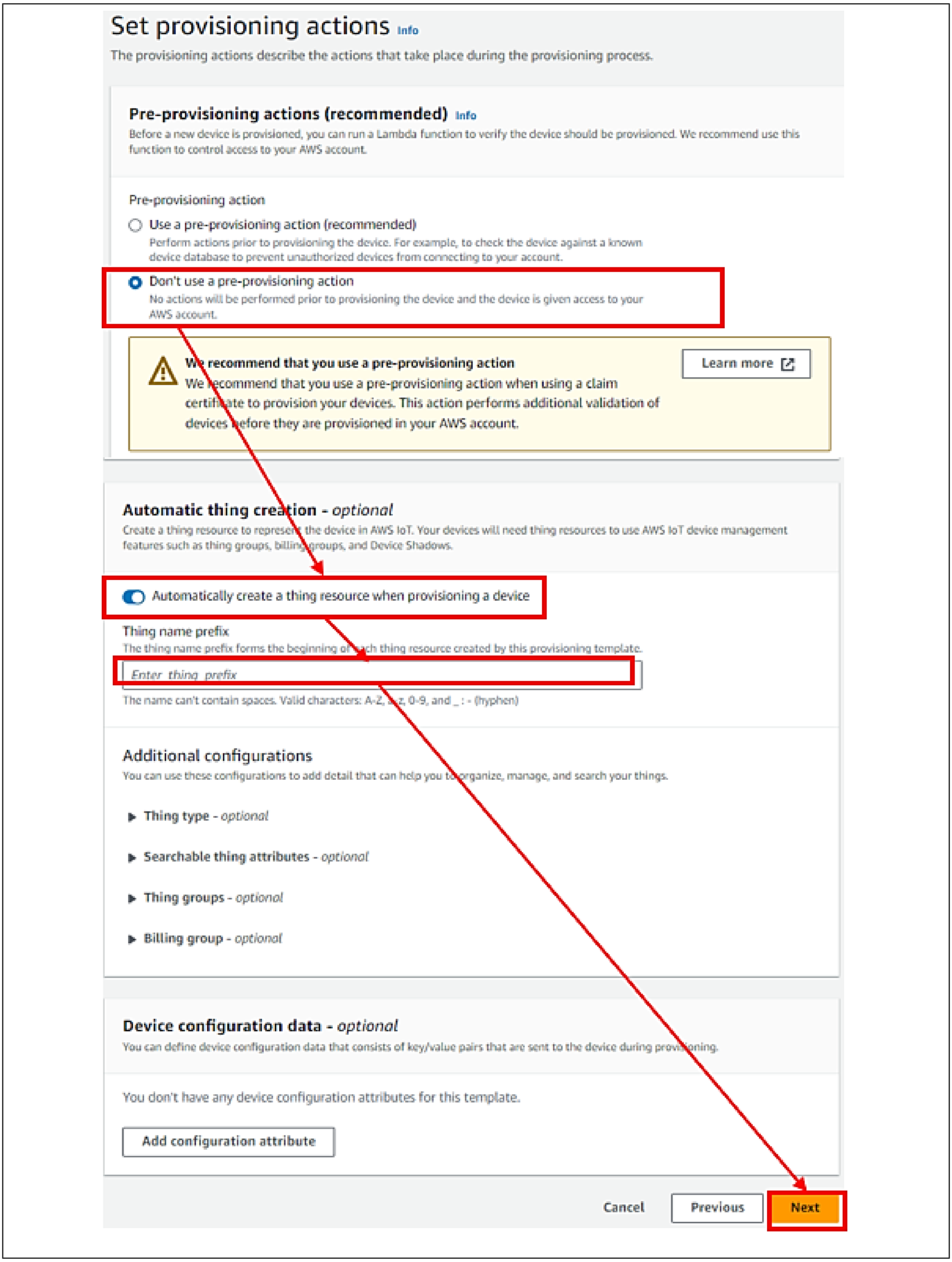
☑	Certificate ID	Status
☑	[redacted]	Active

Cancel | Next

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

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

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



Set provisioning actions Info

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

Pre-provisioning actions (recommended) Info

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

Pre-provisioning action

○ Use a pre-provisioning action (recommended)

Perform actions prior to provisioning the device. For example, to check the device against a known device database to prevent unauthorized devices from connecting to your account.

● Don't use a pre-provisioning action

No actions will be performed prior to provisioning the device and the device is given access to your AWS account.

We recommend that you use a pre-provisioning action

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

Automatic thing creation - *optional*

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

Automatically create a thing resource when provisioning a device

Thing name prefix

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

Enter thing prefix

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

Additional configurations

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

▶ Thing type - *optional*

▶ Searchable thing attributes - *optional*

▶ Thing groups - *optional*

▶ Billing group - *optional*

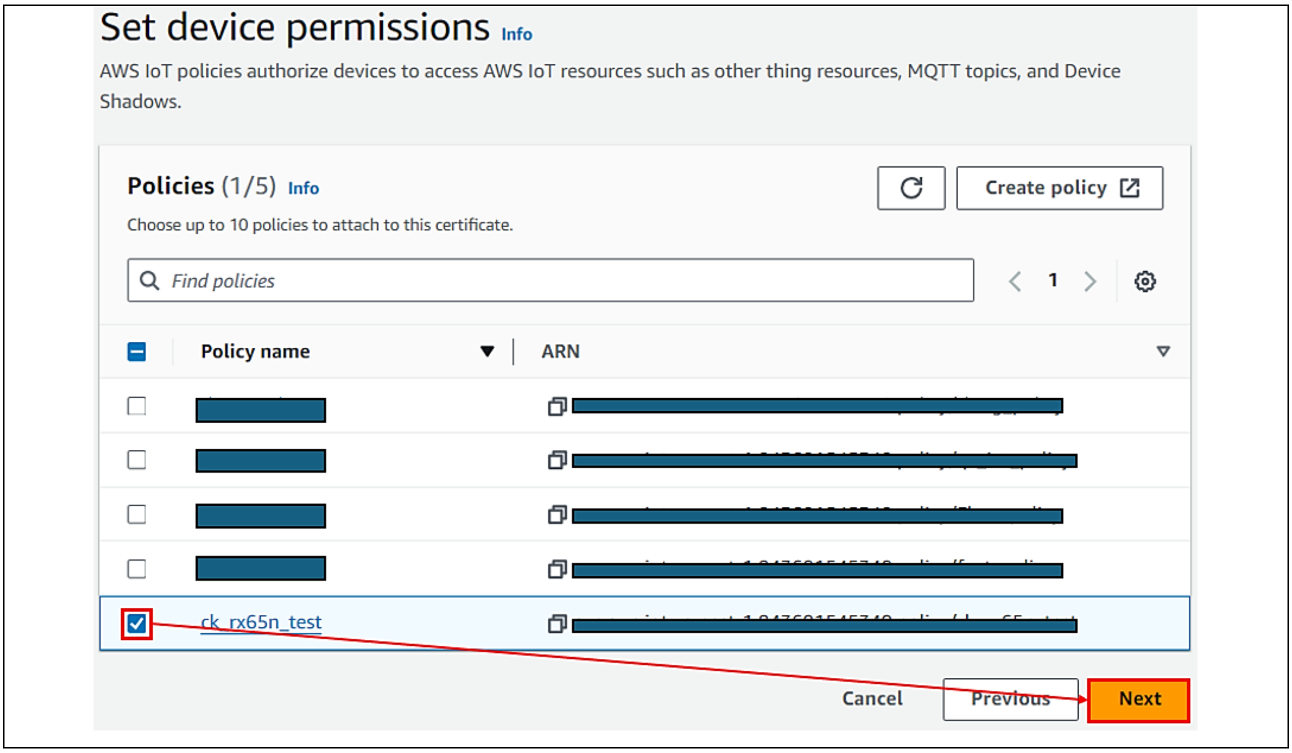
Device configuration data - *optional*

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

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

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

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



Set device permissions Info

AWS IoT policies authorize devices to access AWS IoT resources such as other thing resources, MQTT topics, and Device Shadows.

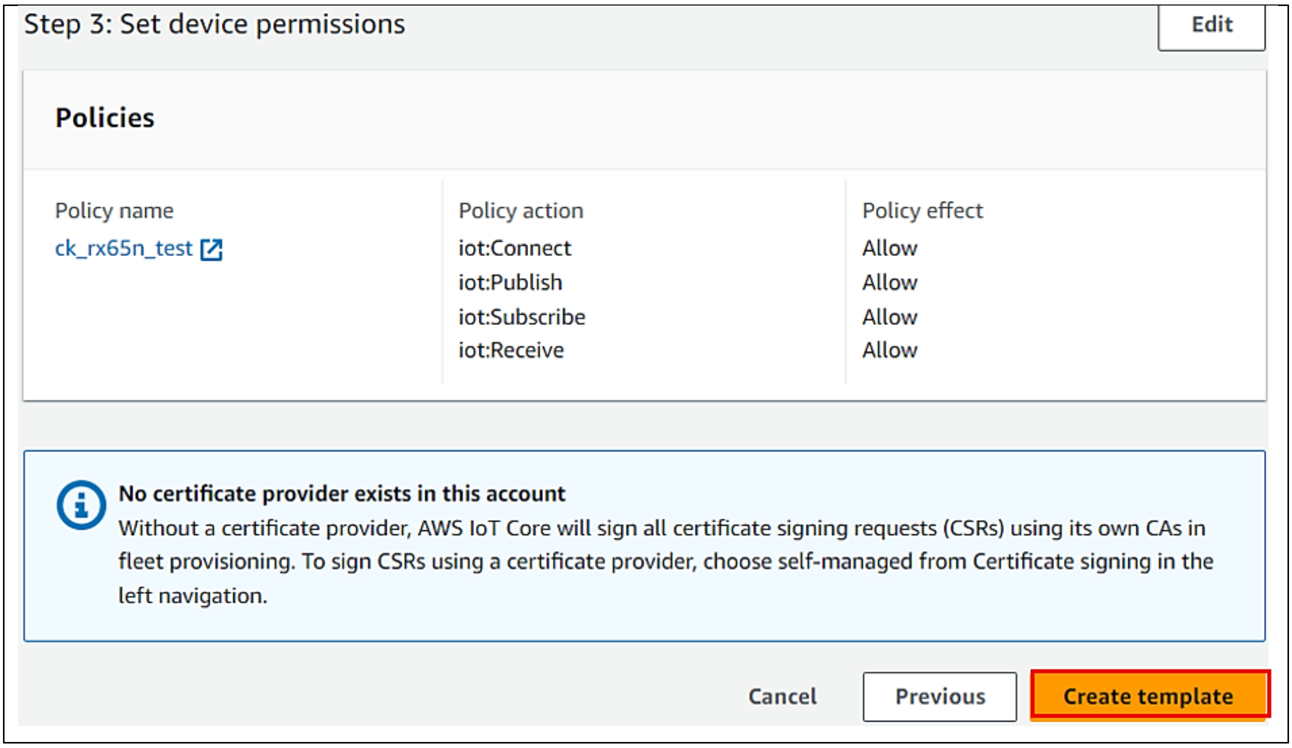
Policies (1/5) Info

Choose up to 10 policies to attach to this certificate.

	Policy name	ARN
□	[redacted]	[redacted]
□	[redacted]	[redacted]
□	[redacted]	[redacted]
□	[redacted]	[redacted]
☑	ck_rx65n_test	[redacted]

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

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



Step 3: Set device permissions

Policies

Policy name	Policy action	Policy effect
ck_rx65n_test	iot:Connect	Allow
	iot:Publish	Allow
	iot:Subscribe	Allow
	iot:Receive	Allow

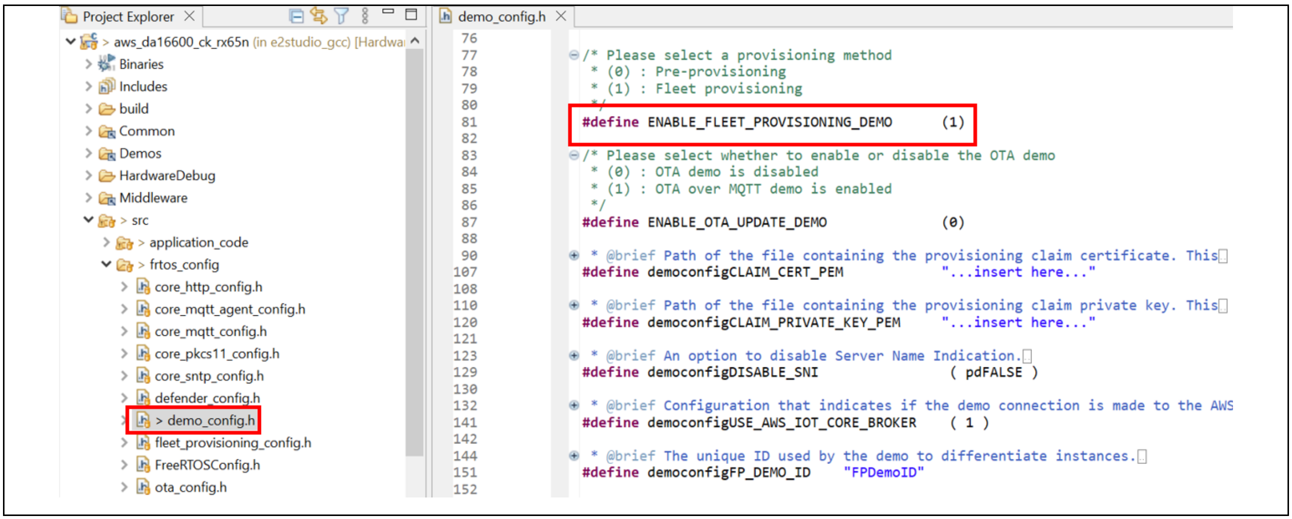
No certificate provider exists in this account

Without a certificate provider, AWS IoT Core will sign all certificate signing requests (CSRs) using its own CAs in fleet provisioning. To sign CSRs using a certificate provider, choose self-managed from Certificate signing in the left navigation.

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

7.3 Setting up the Project

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



```
/* Please select a provisioning method
 * (0) : Pre-provisioning
 * (1) : Fleet provisioning
 */
#define ENABLE_FLEET_PROVISIONING_DEMO      (1)

/* Please select whether to enable or disable the OTA demo
 * (0) : OTA demo is disabled
 * (1) : OTA over MQTT demo is enabled
 */
#define ENABLE_OTA_UPDATE_DEMO              (0)

 * @brief Path of the file containing the provisioning claim certificate. This
#define democonfigCLAIM_CERT_PEM            "...insert here..."

 * @brief Path of the file containing the provisioning claim private key. This
#define democonfigCLAIM_PRIVATE_KEY_PEM     "...insert here..."

 * @brief An option to disable Server Name Indication.
#define democonfigDISABLE_SNI                ( pdFALSE )

 * @brief Configuration that indicates if the demo connection is made to the AWS
#define democonfigUSE_AWS_IOT_CORE_BROKER    ( 1 )

 * @brief The unique ID used by the demo to differentiate instances.
#define democonfigFP_DEMO_ID    "FPDemoID"
```

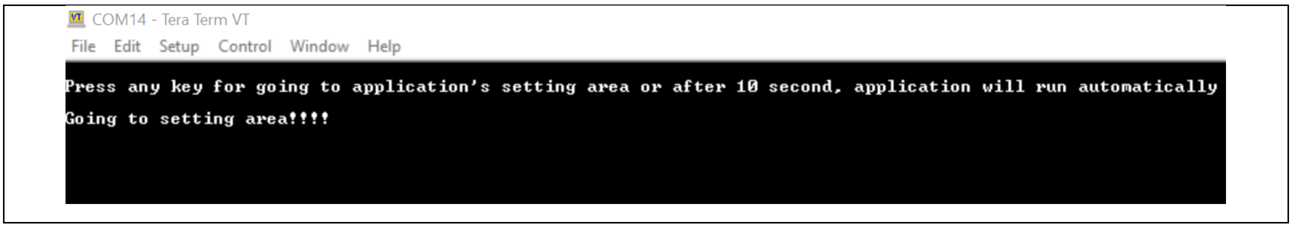
Figure 175. Enable Fleet Provisioning macro

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

7.4 Running Fleet Provisioning

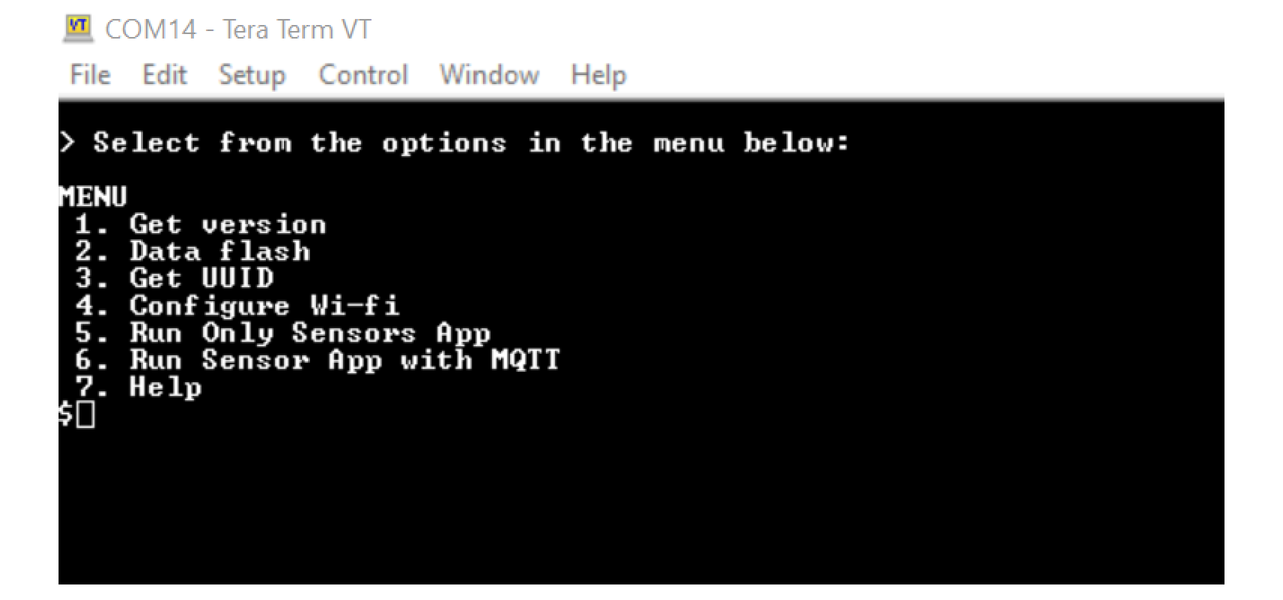
Note: User can run both Fleet Provisioning and OTA (Section **6**) in this application. This section only describes the Fleet Provisioning feature.

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



```
Press any key for going to application's setting area or after 10 second, application will run automatically
Going to setting area!!!!
```

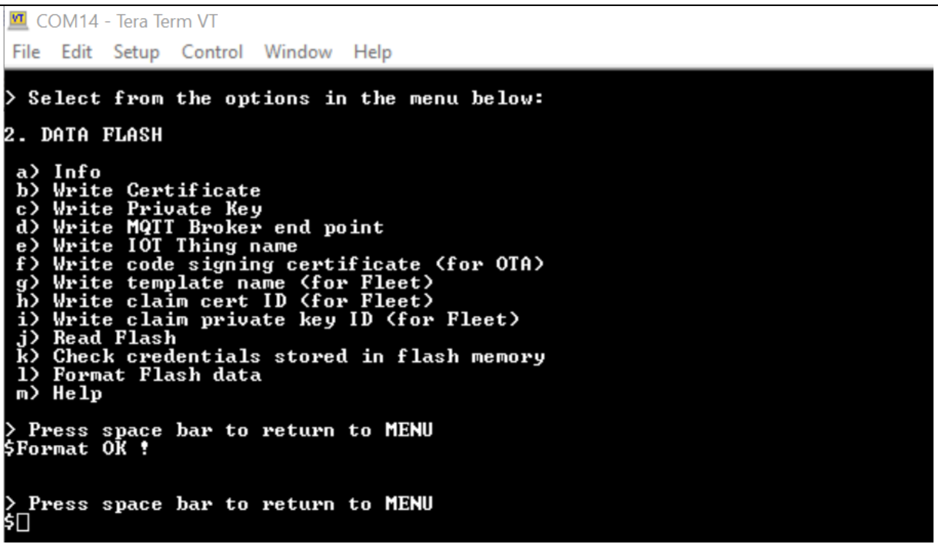
Figure 176. Save credentials for Fleet Provisioning (1/8)



```
> Select from the options in the menu below:

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

Figure 177. 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)**

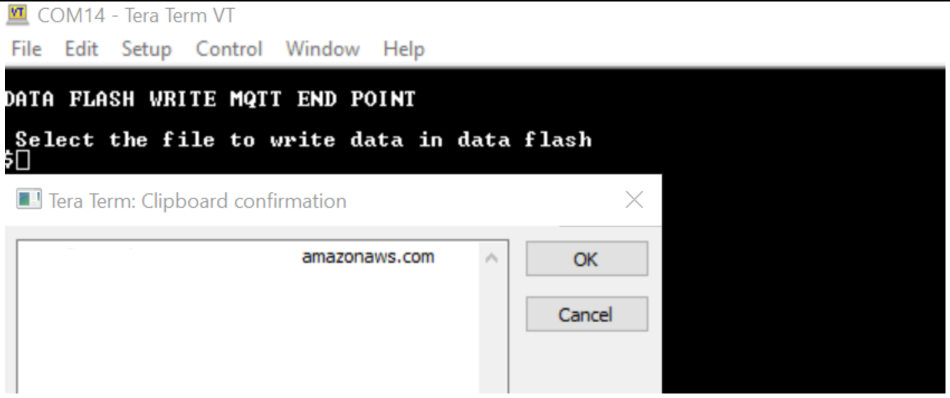


```
> Select from the options in the menu below:
2. DATA FLASH

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

> Press space bar to return to MENU
$Format OK !

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

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

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

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



COM14 - Tera Term VT
File Edit Setup Control Window Help

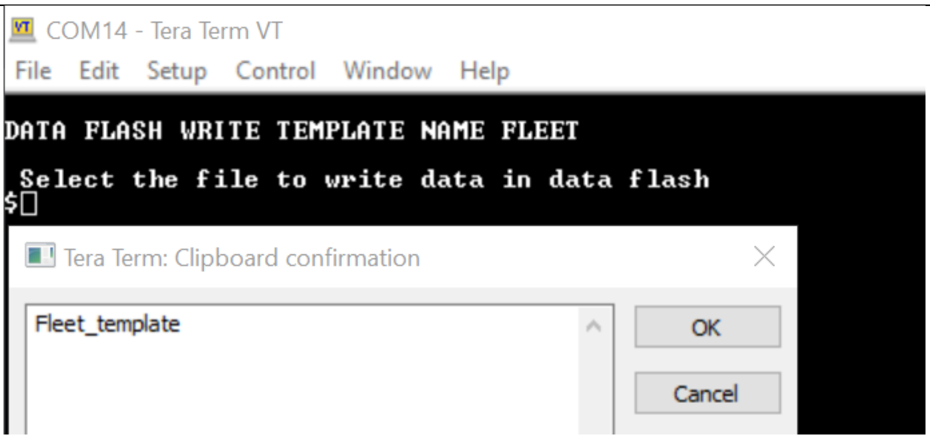
```
DATA FLASH WRITE MQTT END POINT

 Select the file to write data in data flash
$
```

Tera Term: Clipboard confirmation
OK
Cancel

amazonaws.com

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

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



COM14 - Tera Term VT
File Edit Setup Control Window Help

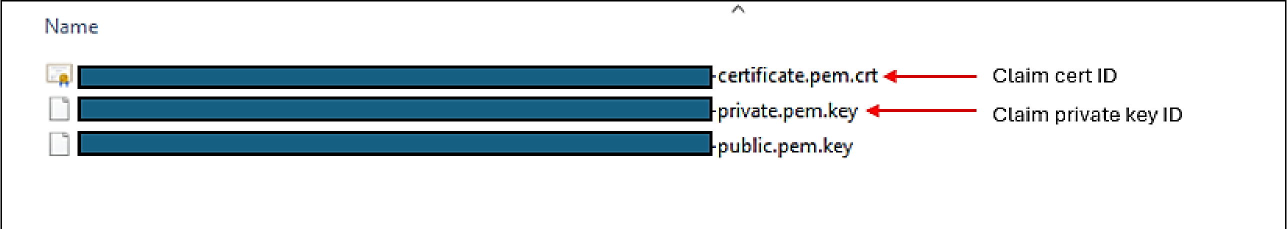
```
DATA FLASH WRITE TEMPLATE NAME FLEET

 Select the file to write data in data flash
$
```

Tera Term: Clipboard confirmation
OK
Cancel

Fleet_template

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

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



Name

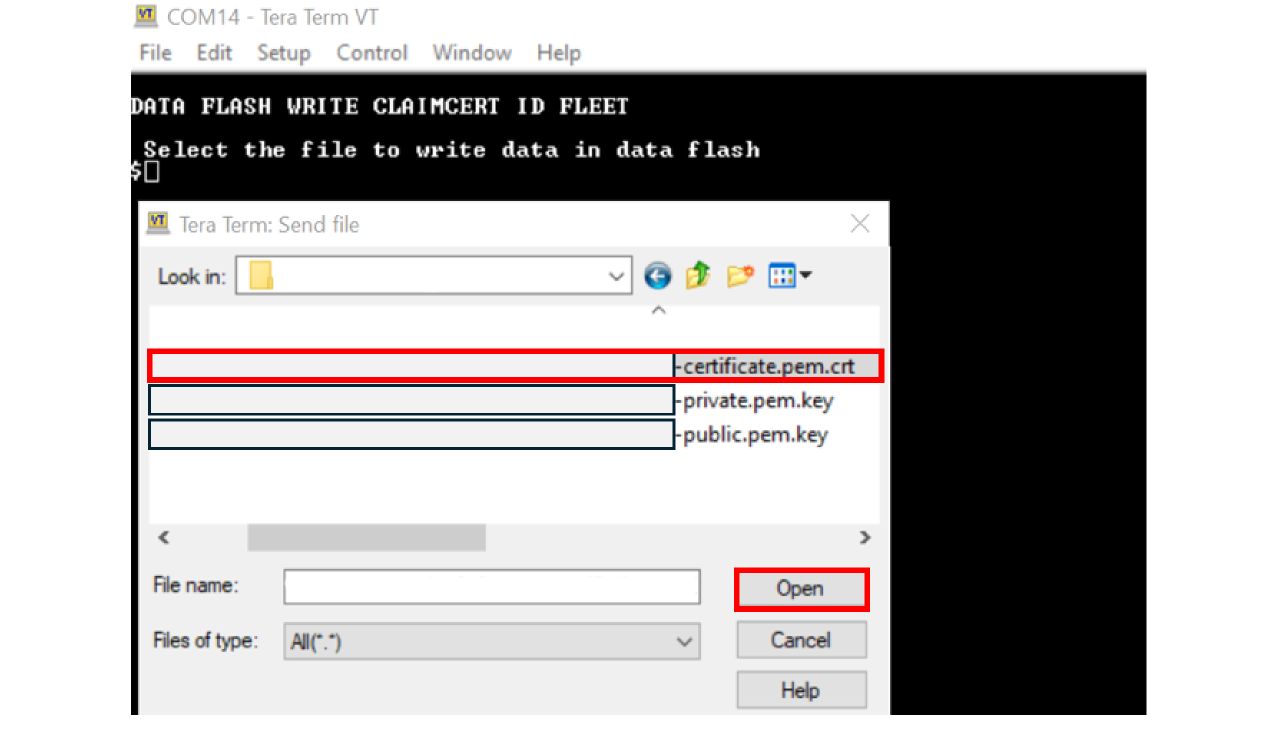
[redacted]-certificate.pem.crt ← Claim cert ID

[redacted]-private.pem.key ← Claim private key ID

[redacted]-public.pem.key

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

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



```
DATA FLASH WRITE CLAIMCERT ID FLEET

 Select the file to write data in data flash
$
```

[redacted]-certificate.pem.crt

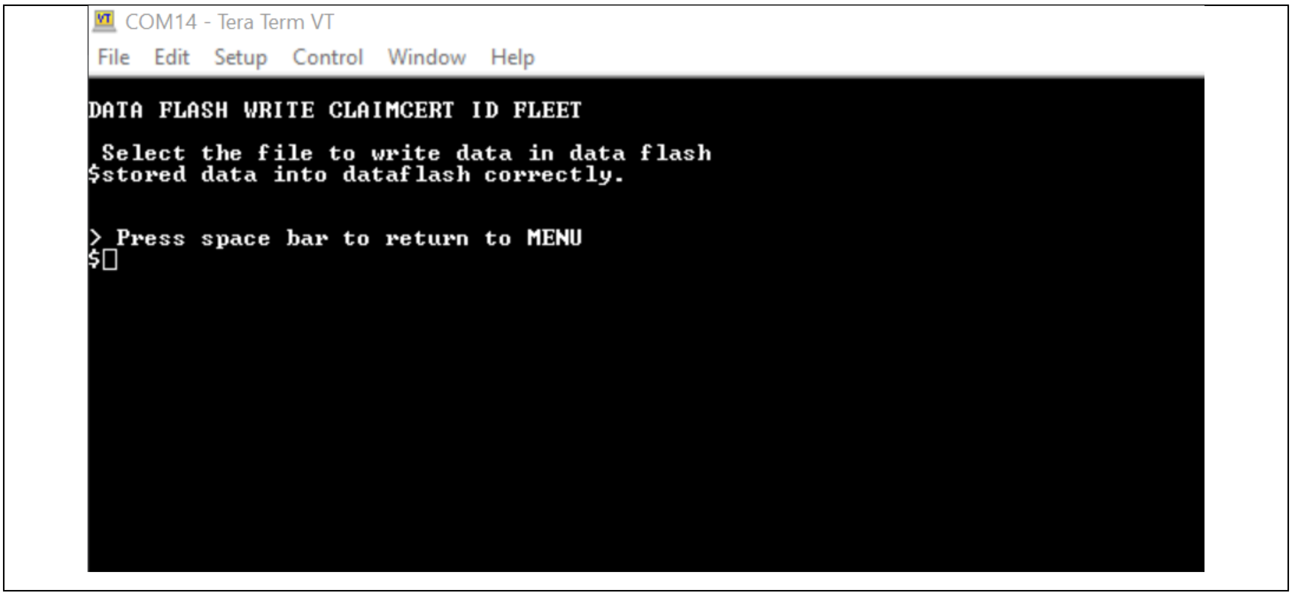
[redacted]-private.pem.key

[redacted]-public.pem.key

File name:

Files of type: All(*.*)

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



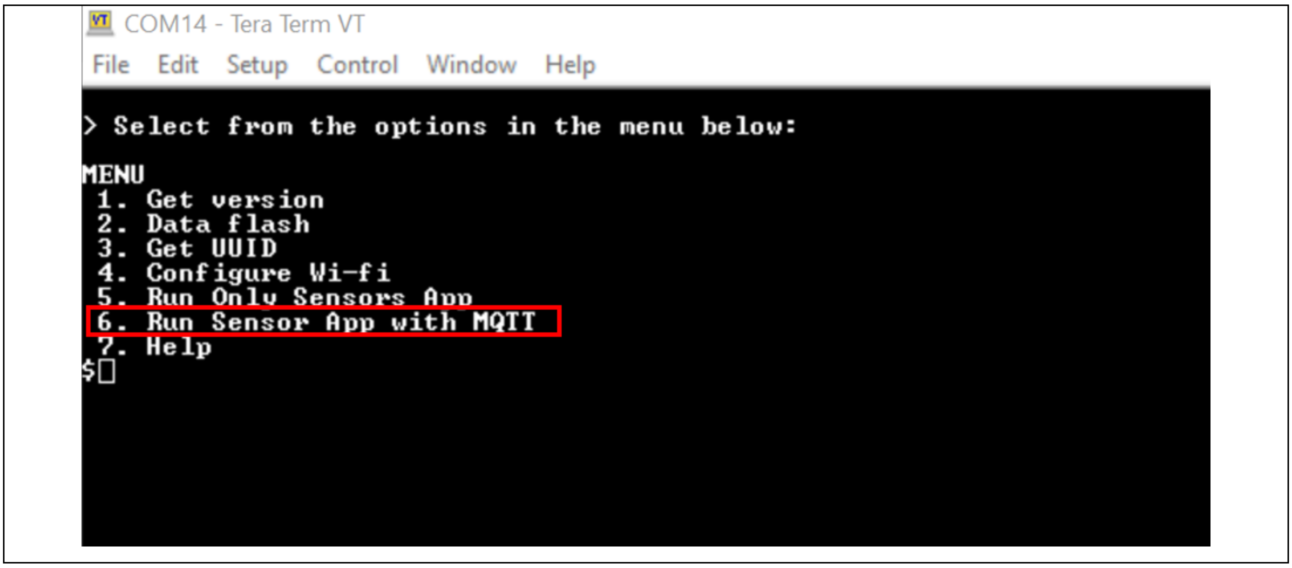
```
DATA FLASH WRITE CLAIMCERT ID FLEET

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


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

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

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

6. Store Wi-Fi’s credentials to Flash (refer to topic **5.3.2**)
7. Start application with Fleet Provisioning:

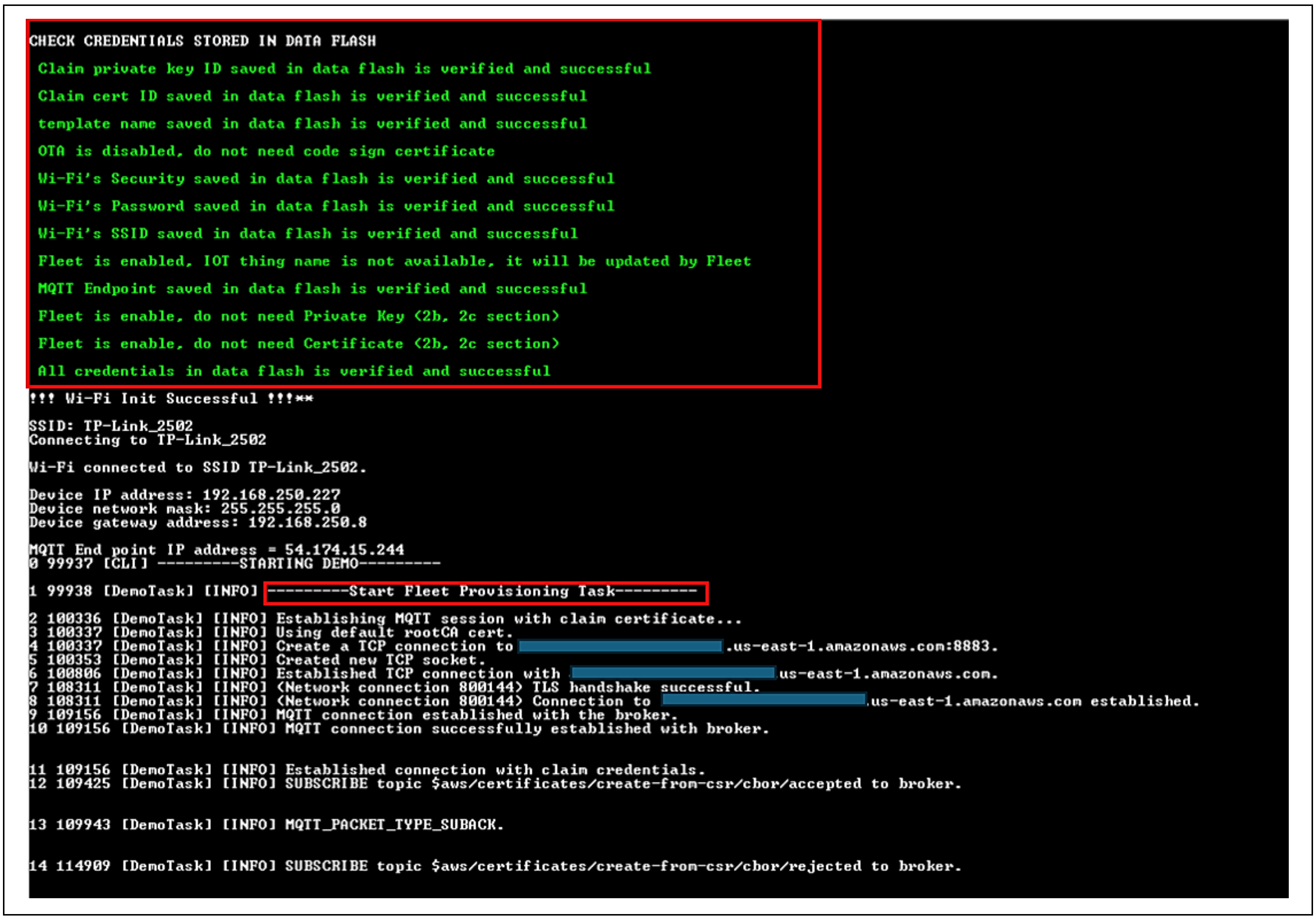
Back to the Application Menu and press **‘6’** to run application:



```
> Select from the options in the menu below:

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

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

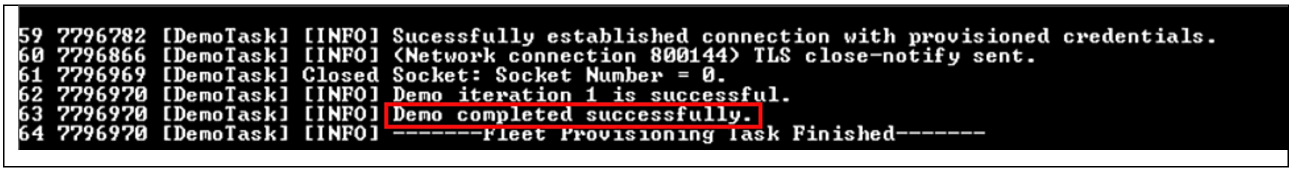
Application will check stored data flash. When it has necessary credentials enough, application will start. Otherwise, the application will send a notice to the user and the user has to save the lacking credentials for running application.



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

SSID: TP-Link_2502
Connecting to TP-Link_2502

Wi-Fi connected to SSID TP-Link_2502.

Device IP address: 192.168.250.227
Device network mask: 255.255.255.0
Device gateway address: 192.168.250.8

MQTT End point IP address = 54.174.15.244
0 99937 [CLI] ---------STARTING DEMO---------

1 99938 [DemoTask] [INFO] ---------Start Fleet Provisioning Task---------

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


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


13 109943 [DemoTask] [INFO] MQTT_PACKET_TYPE_SUBACK.


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

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

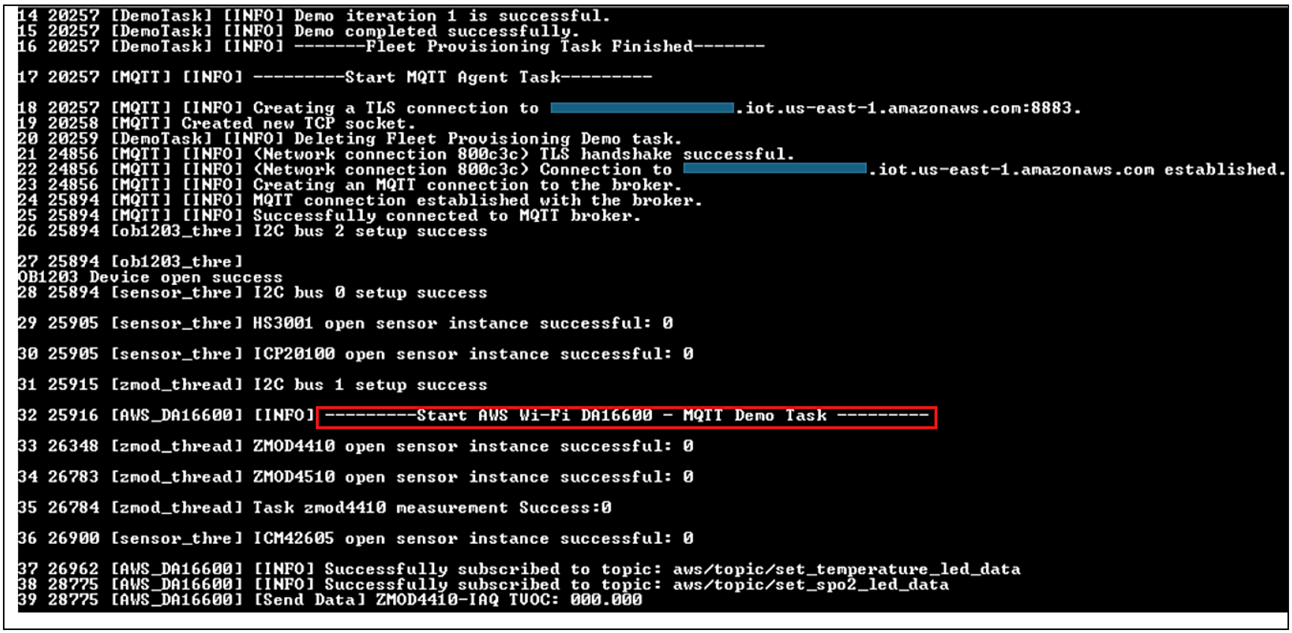
If the text string **“Demo completed successfully.”** Appears at the end of the log, the fleet provisioning demo completed successfully. Successful completion of the demo means that a new thing has been registered on AWS IoT Core and an individual device certificate assigned to it.



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

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

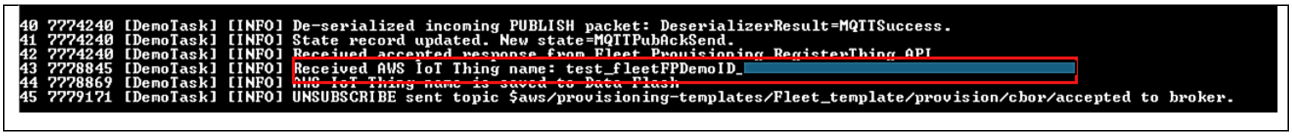
After running the fleet provisioning demo, users can use the individual device certificate and private key obtained from AWS to run the MQTT with sensors demo:



```
14 20257 [DemoTask] [INFO] Demo iteration 1 is successful.
15 20257 [DemoTask] [INFO] Demo completed successfully.
16 20257 [DemoTask] [INFO] -------Fleet Provisioning Task Finished-------

17 20257 [MQTT] [INFO] ---------Start MQTT Agent Task---------

18 20257 [MQTT] [INFO] Creating a TLS connection to [redacted].iot.us-east-1.amazonaws.com:8883.
19 20258 [MQTT] Created new TCP socket.
20 20259 [DemoTask] [INFO] Deleting Fleet Provisioning Demo task.
21 24856 [MQTT] [INFO] <Network connection 800c3c> TLS handshake successful.
22 24856 [MQTT] [INFO] <Network connection 800c3c> Connection to [redacted].iot.us-east-1.amazonaws.com established.
23 24856 [MQTT] [INFO] Creating an MQTT connection to the broker.
24 25894 [MQTT] [INFO] MQTT connection established with the broker.
25 25894 [MQTT] [INFO] Successfully connected to MQTT broker.
26 25894 [ob1203_thre] I2C bus 2 setup success

27 25894 [ob1203_thre]
OB1203 Device open success
28 25894 [sensor_thre] I2C bus 0 setup success

29 25905 [sensor_thre] HS3001 open sensor instance successful: 0

30 25905 [sensor_thre] ICP20100 open sensor instance successful: 0

31 25915 [zmod_thread] I2C bus 1 setup success

32 25916 [AWS_DA16600] [INFO] ---------Start AWS Wi-Fi DA16600 - MQTT Demo Task ---------

33 26348 [zmod_thread] ZMOD4410 open sensor instance successful: 0

34 26783 [zmod_thread] ZMOD4510 open sensor instance successful: 0

35 26784 [zmod_thread] Task zmod4410 measurement Success:0

36 26900 [sensor_thre] ICM42605 open sensor instance successful: 0

37 26962 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_temperature_led_data
38 28775 [AWS_DA16600] [INFO] Successfully subscribed to topic: aws/topic/set_spo2_led_data
39 28775 [AWS_DA16600] [Send Data] ZMOD4410-IAQ TVOC: 000.000
```

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

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

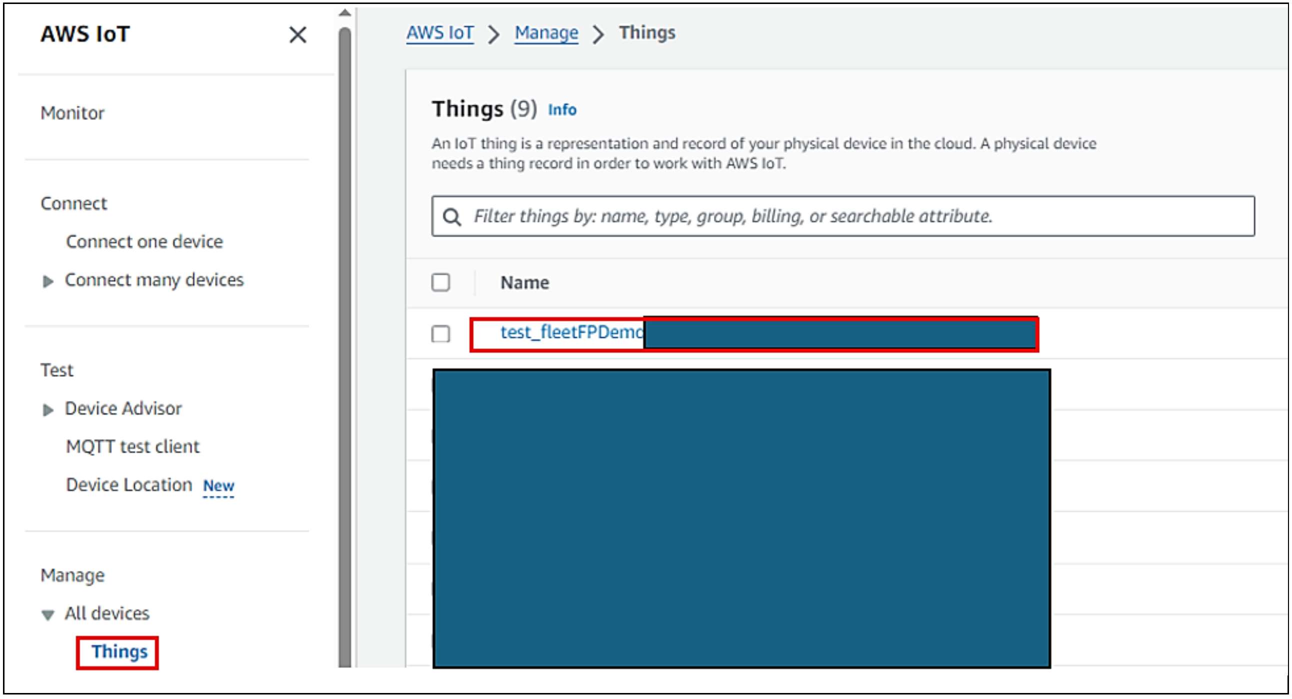


```
40 7774240 [DemoTask] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
41 7774240 [DemoTask] [INFO] State record updated. New state=MQTTPubAckSend.
42 7774240 [DemoTask] [INFO] Received accepted response from Fleet Provisioning RegisterThing API
43 7778845 [DemoTask]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

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

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

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



Things (9) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

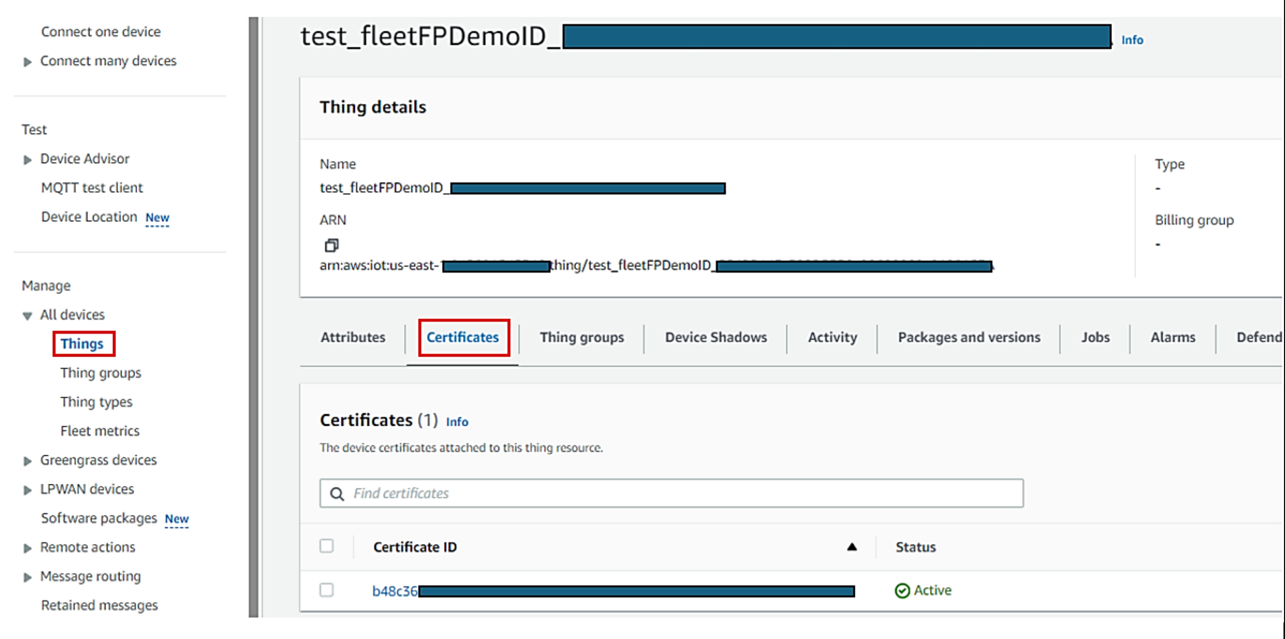
Filter things by: name, type, group, billing, or searchable attribute.

□	Name
□	test_fleetFPDemo[redacted]

[redacted]

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

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



Connect one device
▶ Connect many devices

Test
▶ Device Advisor
MQTT test client
Device Location New

Manage
▼ All devices
Things
Thing groups
Thing types
Fleet metrics
▶ Greengrass devices
▶ LPWAN devices
Software packages New
▶ Remote actions
▶ Message routing
Retained messages

test_fleetFPDemoID_[redacted] Info

Thing details

Name
test_fleetFPDemoID_[redacted]

ARN
arn:aws:iot:us-east-[redacted]thing/test_fleetFPDemoID_[redacted]

Type
-

Billing group
-

Attributes | Certificates | Thing groups | Device Shadows | Activity | Packages and versions | Jobs | Alarms | Defend

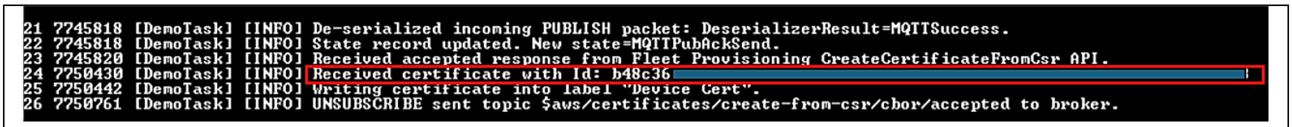
Certificates (1) Info
The device certificates attached to this thing resource.

Find certificates

□	Certificate ID ▲	Status
□	b48c36[redacted]	✓ Active

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

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



```
21 7745818 [DemoTask] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
22 7745818 [DemoTask] [INFO] State record updated. New state=MQTTPubAckSend.
23 7745820 [DemoTask] [INFO] Received accepted response from Fleet Provisioning CreateCertificateFromCsr API.
24 7750430 [DemoTask] [INFO] Received certificate with Id: b48c36[redacted]
25 7750442 [DemoTask] [INFO] Writing certificate into label "Device Cert".
26 7750761 [DemoTask] [INFO] UNSUBSCRIBE sent topic $aws/certificates/create-from-csr/cbor/accepted to broker.
```

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

8. Note and Trouble Shooting

8.1 About Stabilization Time for Sensor

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

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

Table 9. Sensor Stabilization Time

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

8.2 When Build Errors Occur

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

8.3 When Run Errors Occur

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

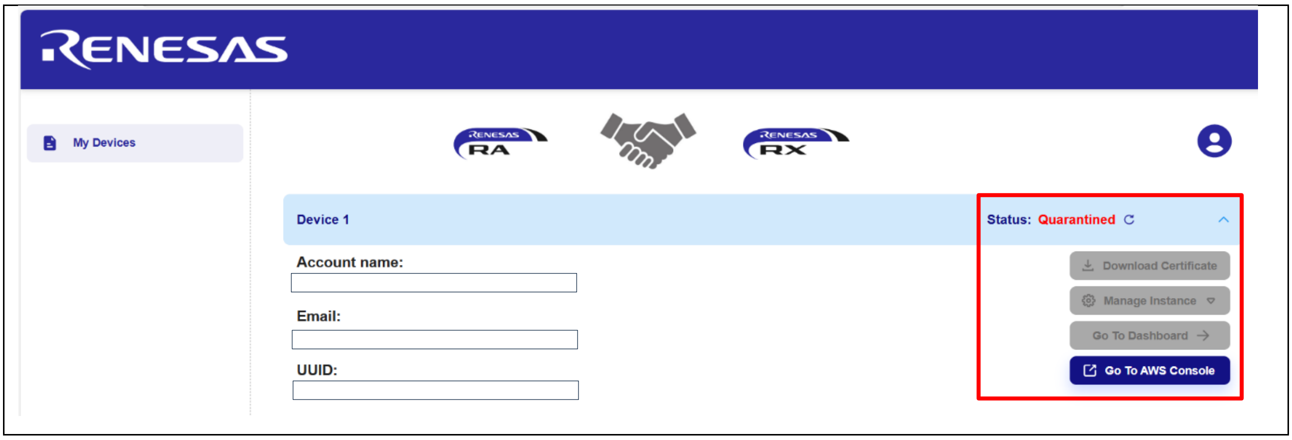
If SDK version is suitable, user need to press button **S1** to reset the board and run the application again.

8.4 Wi-Fi Access Point

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

8.5 Renesas AWS Dashboard account credits and quarantine

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



Device 1

Status: Quarantined

Account name:

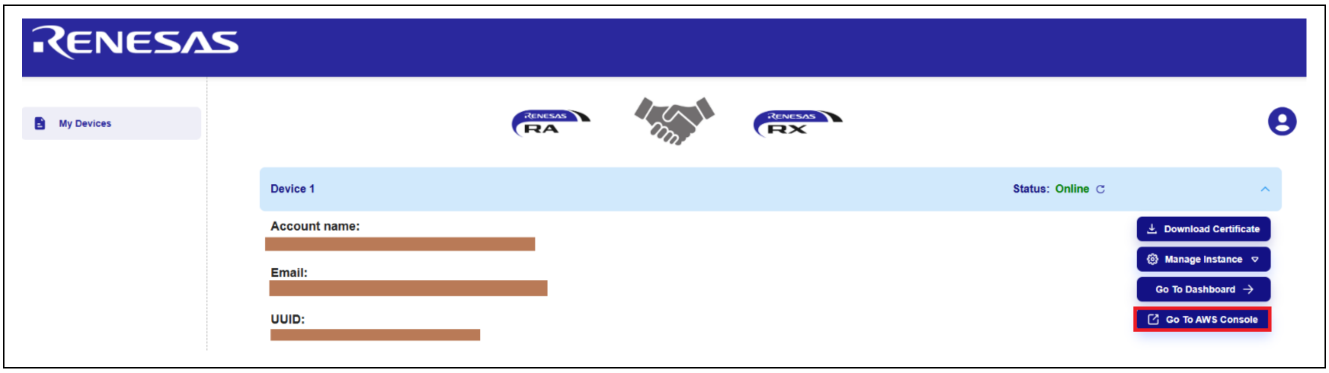
Email:

UUID:

Figure 192. Account Used Up Trial 10 USD

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

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



Device 1

Status: Online

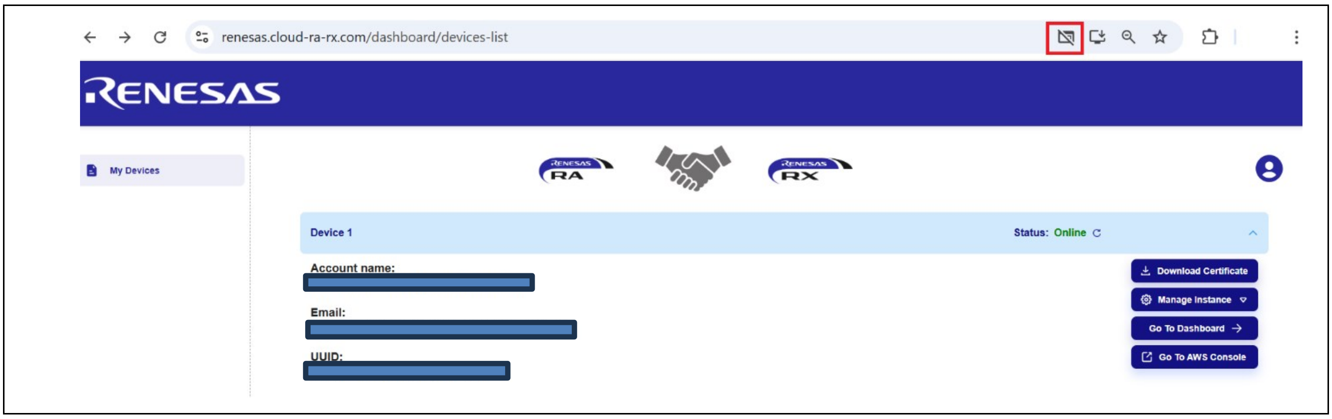
Account name: [redacted]

Email: [redacted]

UUID: [redacted]

Figure 193. Accessing AWS Account from the Dashboard

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



renesas.cloud-ra-rx.com/dashboard/devices-list

Device 1

Status: Online

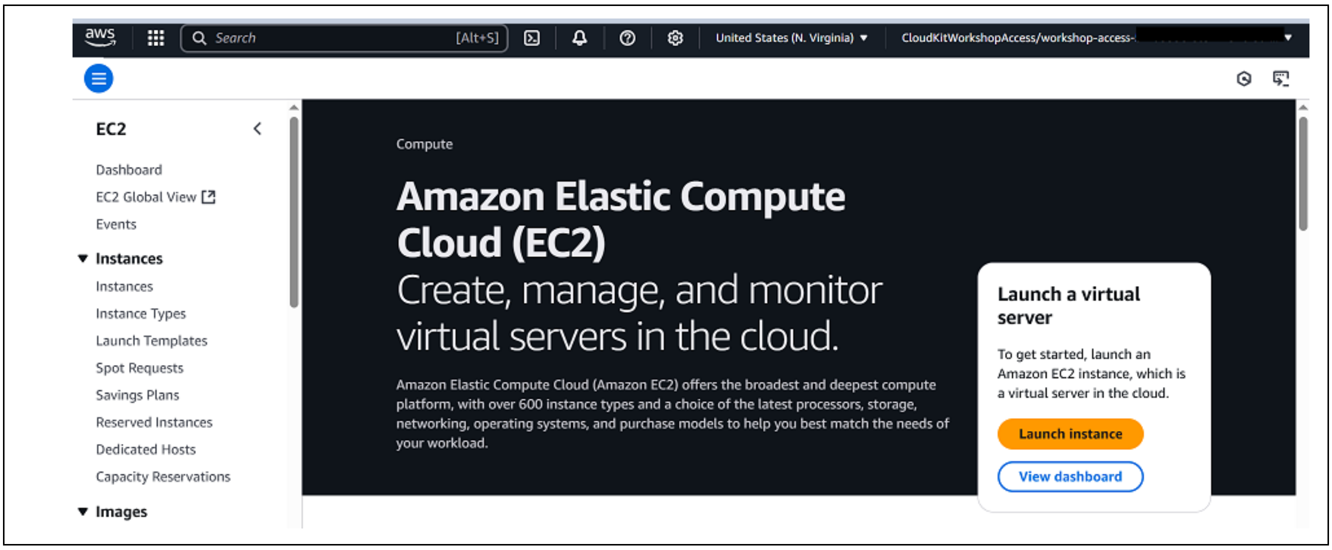
Account name: [redacted]

Email: [redacted]

UUID: [redacted]

Figure 194. Pop Ups on the browser

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



United States (N. Virginia) ▼ CloudKitWorkshopAccess/workshop-access-[redacted]

Compute

Amazon Elastic Compute Cloud (EC2)

Create, manage, and monitor virtual servers in the cloud.

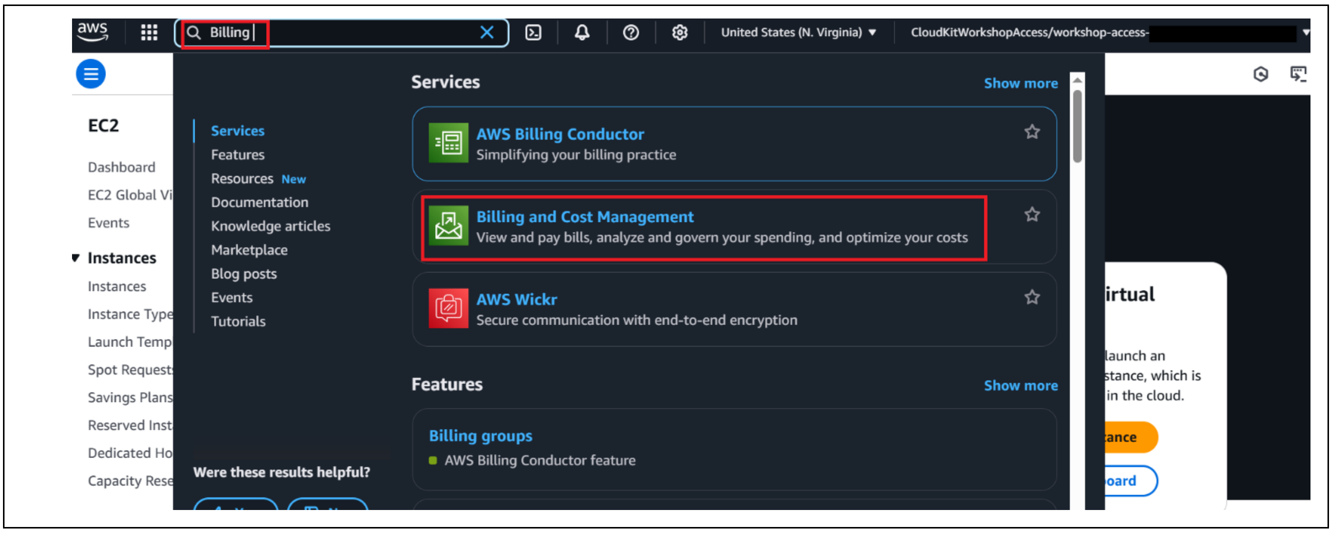
Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 600 instance types and a choice of the latest processors, storage, networking, operating systems, and purchase models to help you best match the needs of your workload.

Launch a virtual server

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Figure 195. AWS account

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



Billing

United States (N. Virginia) ▼ CloudKitWorkshopAccess/workshop-access-[redacted]

Services

AWS Billing Conductor – Simplifying your billing practice

Billing and Cost Management – View and pay bills, analyze and govern your spending, and optimize your costs

AWS Wickr – Secure communication with end-to-end encryption

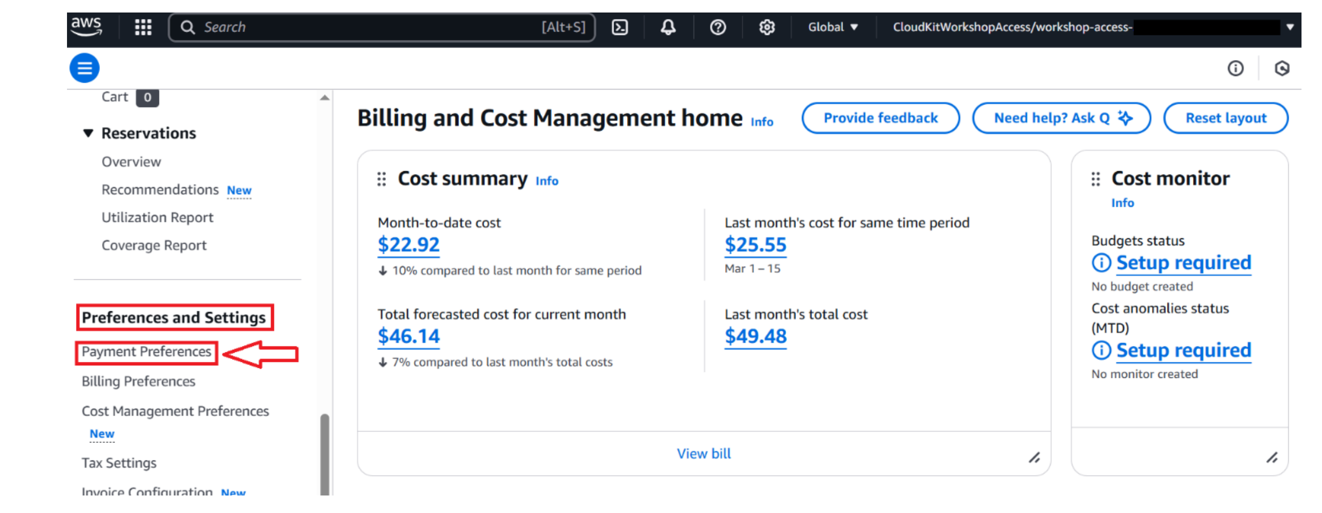
Features

Billing groups – AWS Billing Conductor feature

Were these results helpful?

Figure 196. Billing and Cost Management

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



aws Search [Alt+S] Global ▼ CloudKitWorkshopAccess/workshop-access-[redacted] ▼

Cart 0

▼ Reservations

Overview

Recommendations New

Utilization Report

Coverage Report

Preferences and Settings

Payment Preferences

Billing Preferences

Cost Management Preferences New

Tax Settings

Invoice Configuration New

Billing and Cost Management home Info

Provide feedback | Need help? Ask Q | Reset layout

Cost summary Info

Month-to-date cost
$22.92
↓ 10% compared to last month for same period

Last month's cost for same time period
$25.55
Mar 1 – 15

Total forecasted cost for current month
$46.14
↓ 7% compared to last month's total costs

Last month's total cost
$49.48

View bill

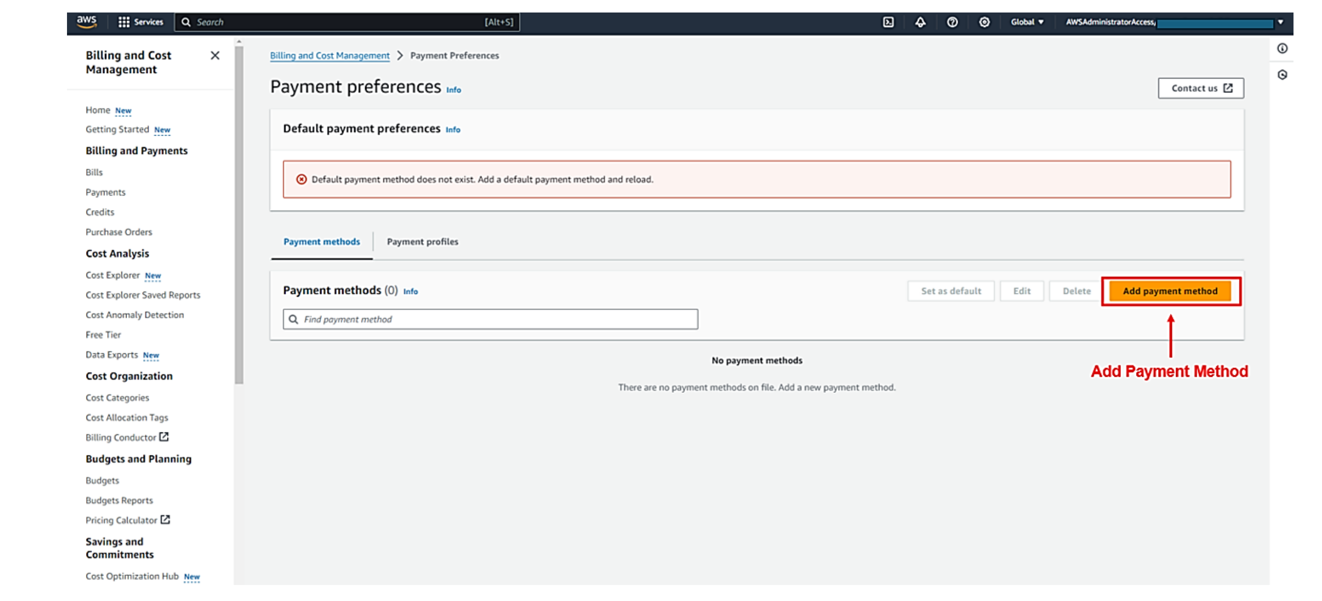
Cost monitor Info

Budgets status
ⓘ Setup required
No budget created

Cost anomalies status (MTD)
ⓘ Setup required
No monitor created

Figure 197. Payment Preferences

6. Add payment method.



aws Services Search [Alt+S] Global ▼ AWSAdministratorAccess/[redacted] ▼

Billing and Cost Management ×

Home New

Getting Started New

Billing and Payments

Bills

Payments

Credits

Purchase Orders

Cost Analysis

Cost Explorer New

Cost Explorer Saved Reports

Cost Anomaly Detection

Free Tier

Data Exports New

Cost Organization

Cost Categories

Cost Allocation Tags

Billing Conductor

Budgets and Planning

Budgets

Budgets Reports

Pricing Calculator

Savings and Commitments

Cost Optimization Hub New

Billing and Cost Management > Payment Preferences

Payment preferences Info

Contact us

Default payment preferences Info

ⓧ Default payment method does not exist. Add a default payment method and reload.

Payment methods | Payment profiles

Payment methods (0) Info

Find payment method

Set as default | Edit | Delete | Add payment method

No payment methods

There are no payment methods on file. Add a new payment method.

Add Payment Method

Figure 198. Add Payment Method

7. Go to ‘AWS Organizations’.

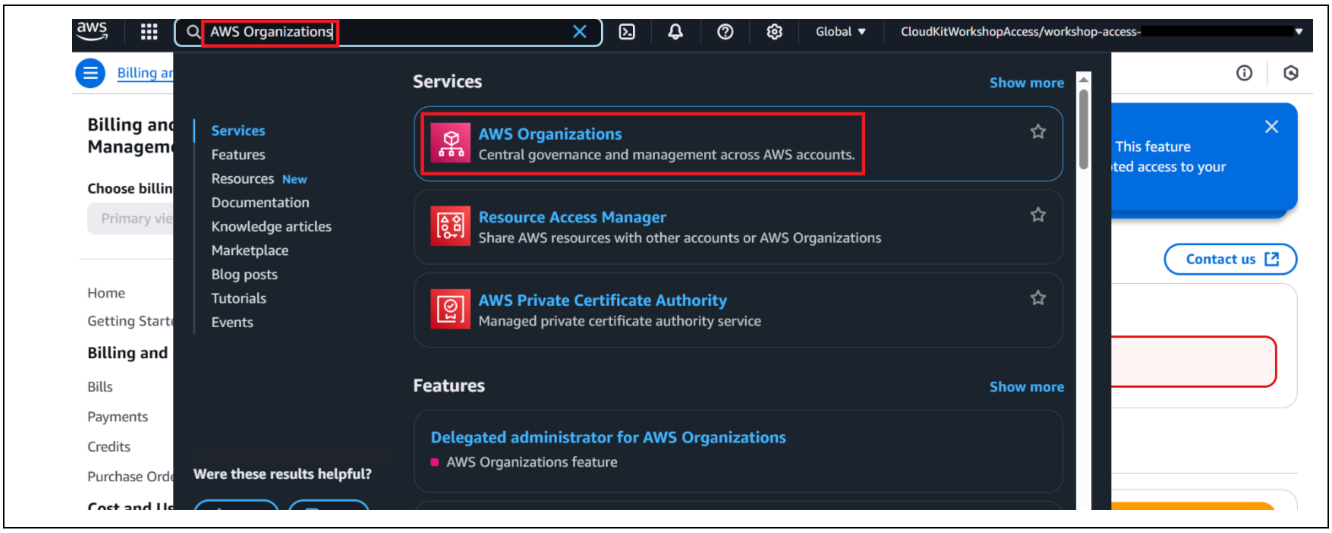
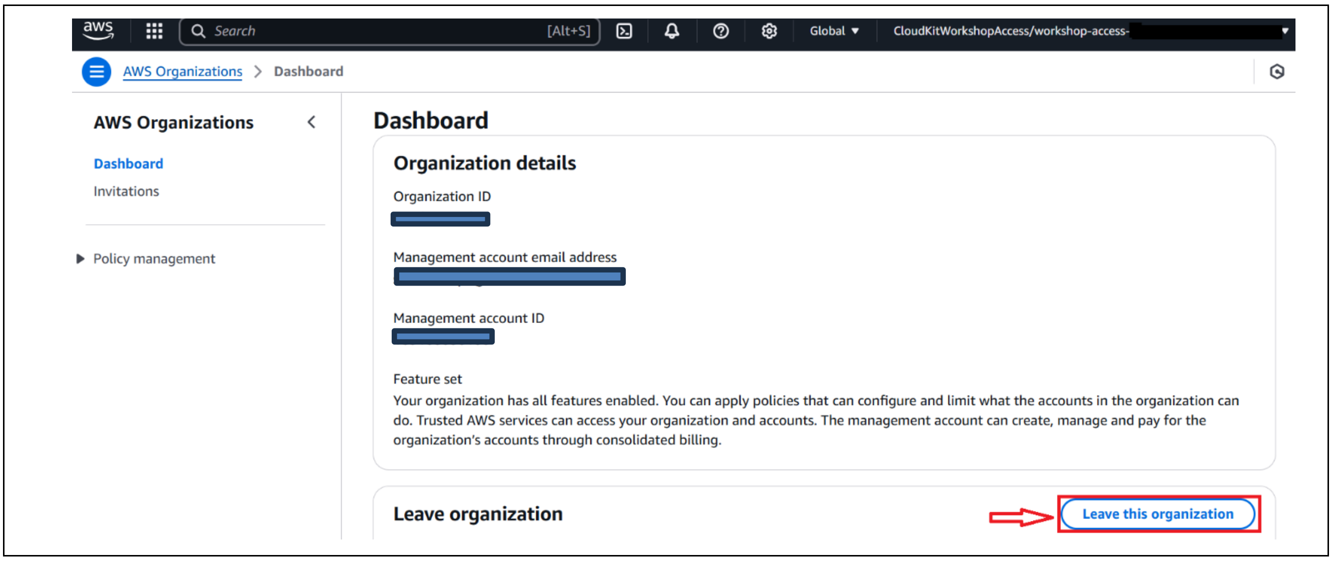


Figure 199. AWS Organizations

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



Dashboard

Organization details

Organization ID
[redacted]

Management account email address
[redacted]

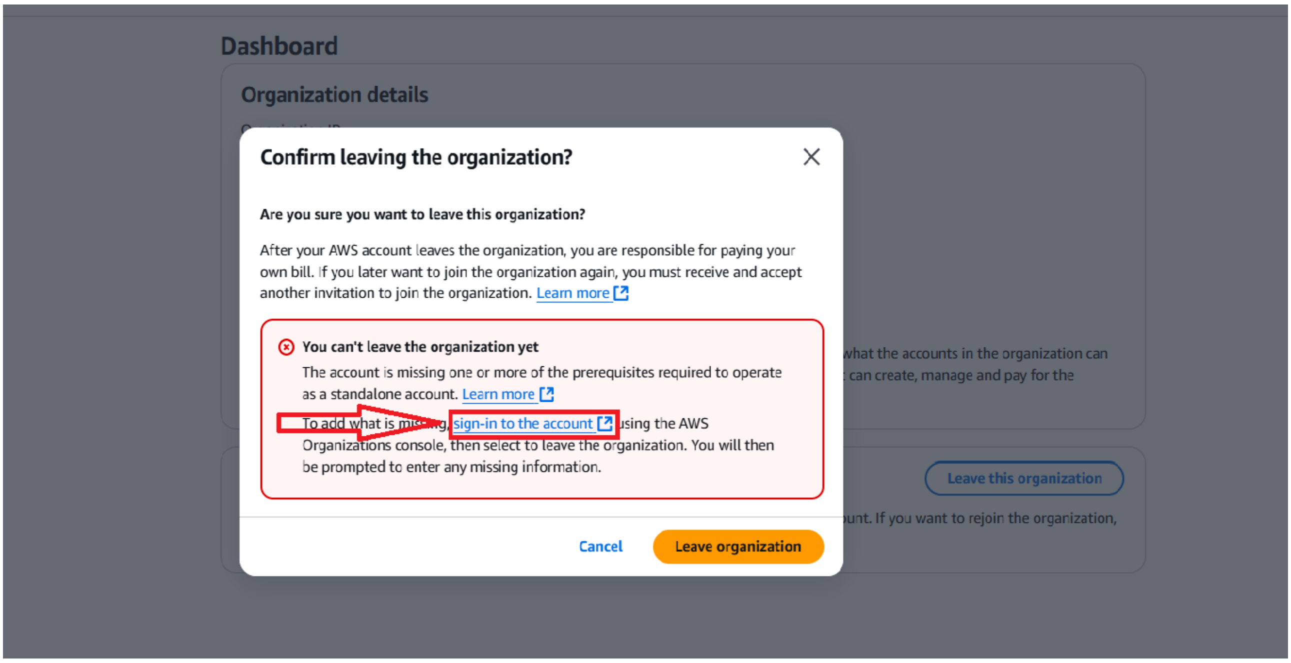
Management account ID
[redacted]

Feature set
Your organization has all features enabled. You can apply policies that can configure and limit what the accounts in the organization can do. Trusted AWS services can access your organization and accounts. The management account can create, manage and pay for the organization's accounts through consolidated billing.

Leave organization

Figure 200. Leaving the organization

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



Dashboard

Organization details

Confirm leaving the organization?

Are you sure you want to leave this organization?

After your AWS account leaves the organization, you are responsible for paying your own bill. If you later want to join the organization again, you must receive and accept another invitation to join the organization. Learn more

You can't leave the organization yet

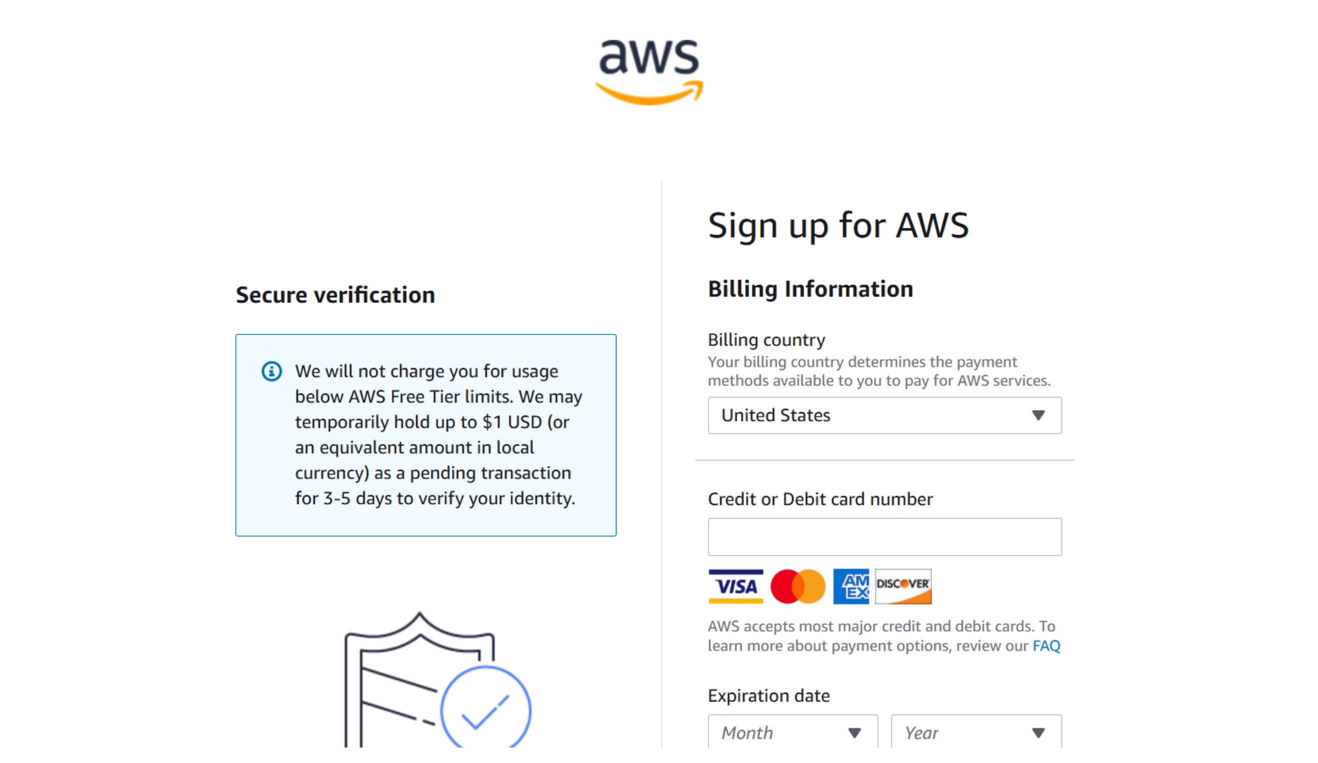
The account is missing one or more of the prerequisites required to operate as a standalone account. Learn more

To add what is missing, sign-in to the account using the AWS Organizations console, then select to leave the organization. You will then be prompted to enter any missing information.

Cancel
Leave organization
Leave this organization
what the accounts in the organization can
can create, manage and pay for the
unt. If you want to rejoin the organization,

Figure 201. AWS Sign up process

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



Sign up for AWS

Secure verification

We will not charge you for usage below AWS Free Tier limits. We may temporarily hold up to $1 USD (or an equivalent amount in local currency) as a pending transaction for 3-5 days to verify your identity.

Billing Information

Billing country
Your billing country determines the payment methods available to you to pay for AWS services.
United States

Credit or Debit card number

AWS accepts most major credit and debit cards. To learn more about payment options, review our FAQ

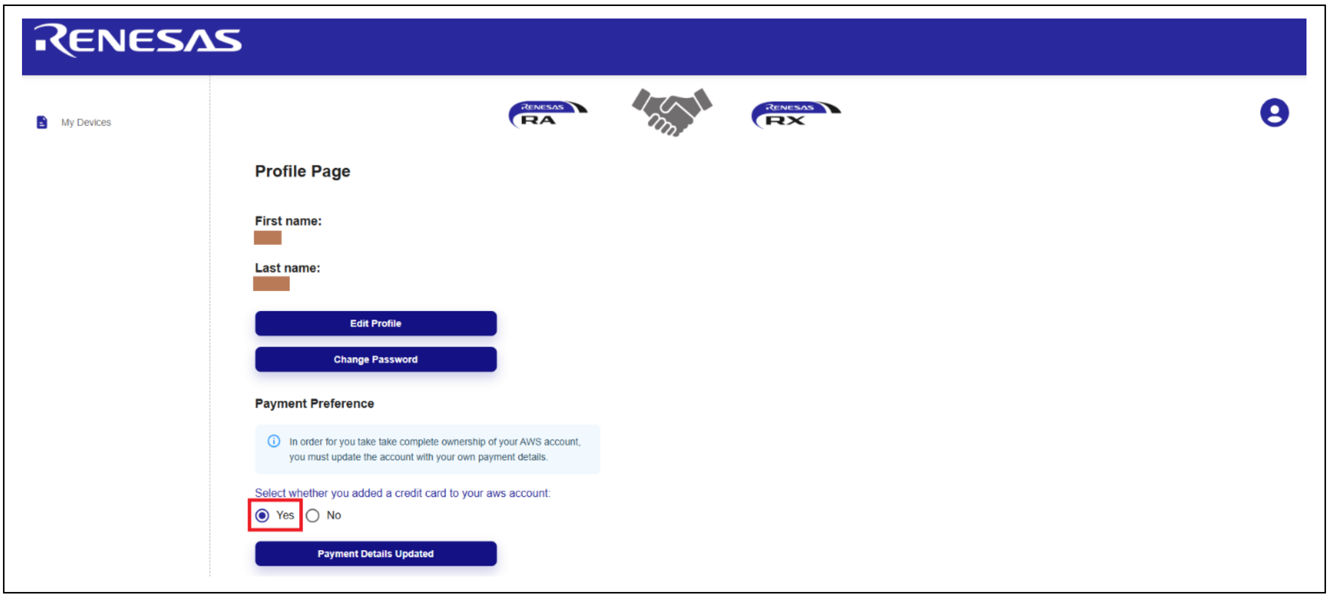
Expiration date
Month
Year

Figure 202. Prerequisites for leaving the organization

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

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

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



Profile Page

First name:
[redacted]

Last name:
[redacted]

Payment Preference

In order for you take take complete ownership of your AWS account, you must update the account with your own payment details.

Select whether you added a credit card to your aws account:
● Yes ○ No

Figure 203. Payment Preference Update

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

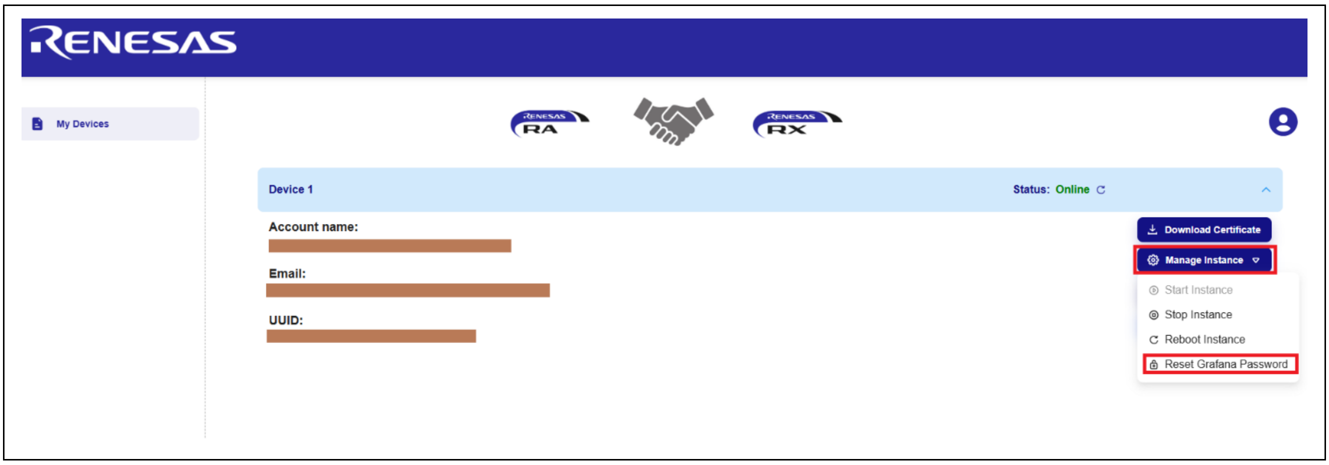
8.6 When Unable to Log in to the Dashboard (Grafana account)

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

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

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

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



Device 1 Status: Online

Account name:
[redacted]

Email:
[redacted]

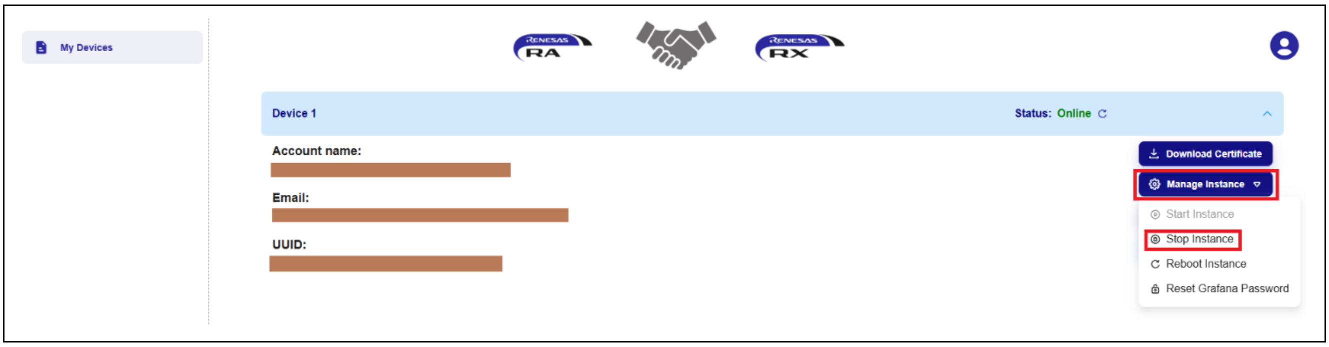
UUID:
[redacted]

Figure 204. Grafana Password Reset

8.7 How to Enable/Disable EC2 Instance

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

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



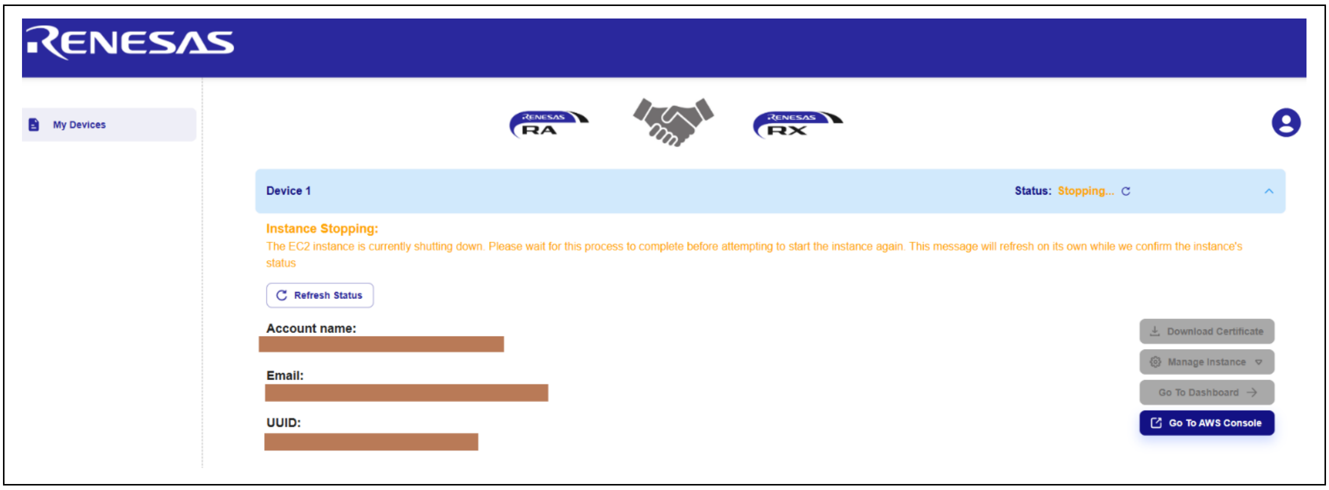
Account name: [redacted]

Email: [redacted]

UUID: [redacted]

Figure 205. Stop EC2 instance

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



Instance Stopping:
The EC2 instance is currently shutting down. Please wait for this process to complete before attempting to start the instance again. This message will refresh on its own while we confirm the instance's status

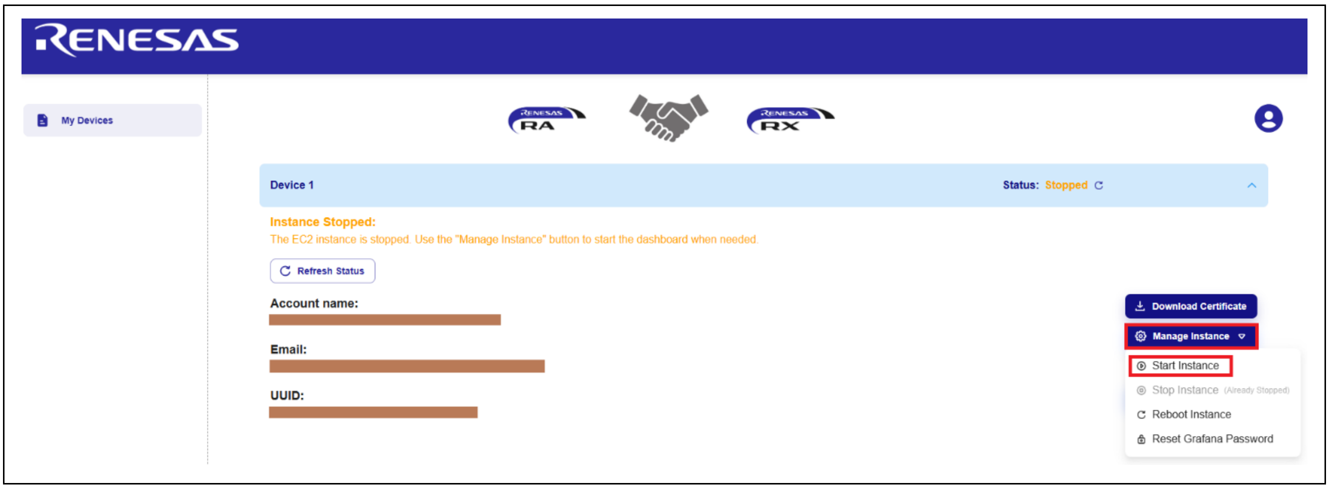
Account name: [redacted]

Email: [redacted]

UUID: [redacted]

Figure 206. EC2 instance status

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



Instance Stopped:
The EC2 instance is stopped. Use the "Manage Instance" button to start the dashboard when needed.

Account name: [redacted]

Email: [redacted]

UUID: [redacted]

Figure 207. Restarting the EC2 Instance State

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

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

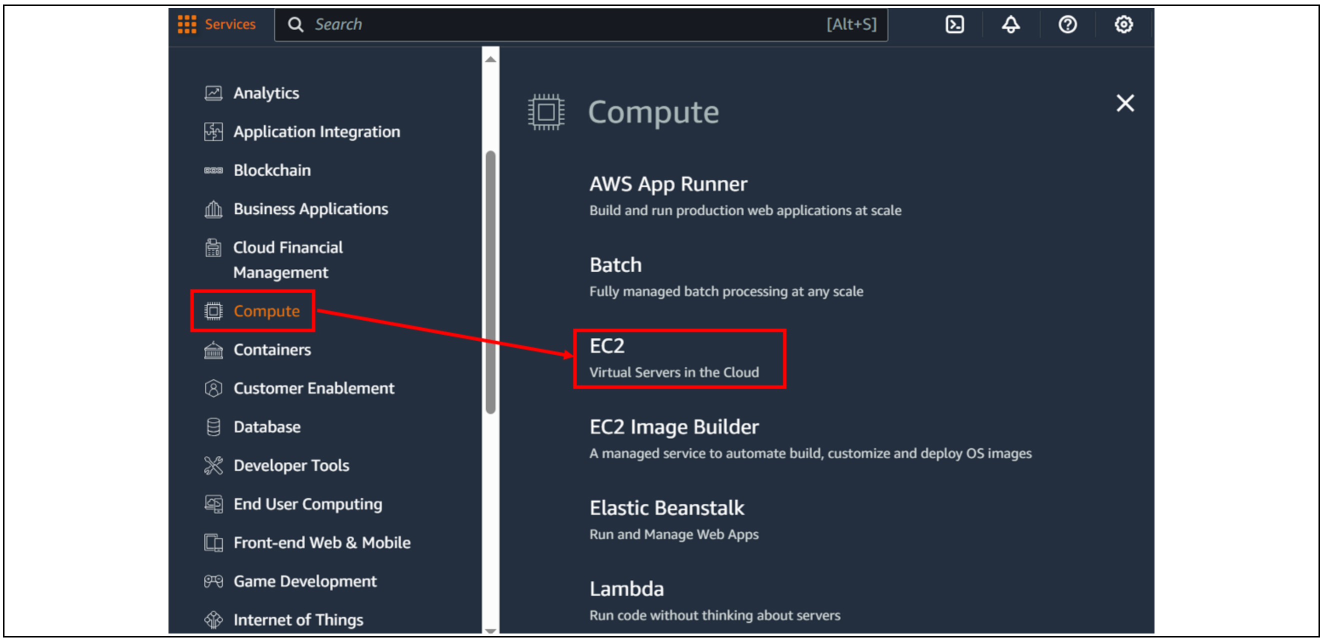


Figure 208. EC2 AWS Service

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

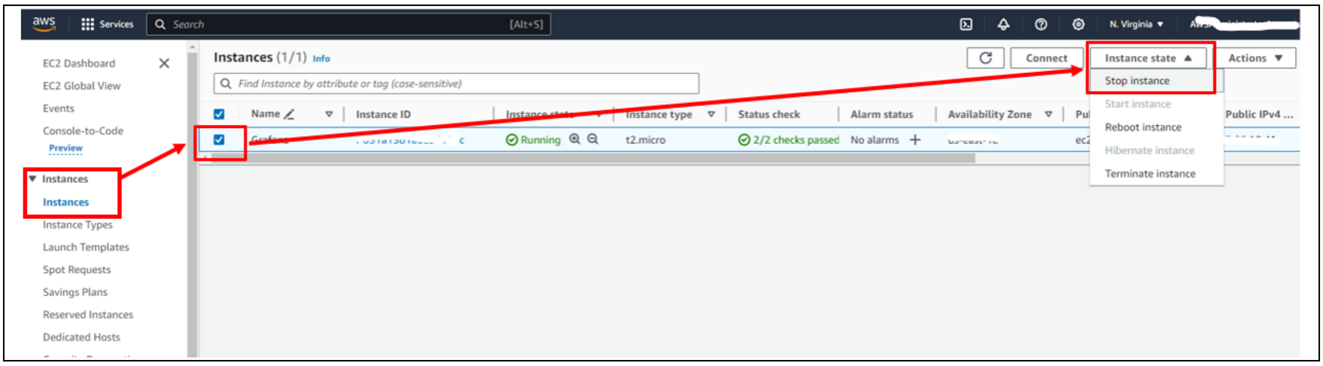


Figure 209. Disable Instance

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

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

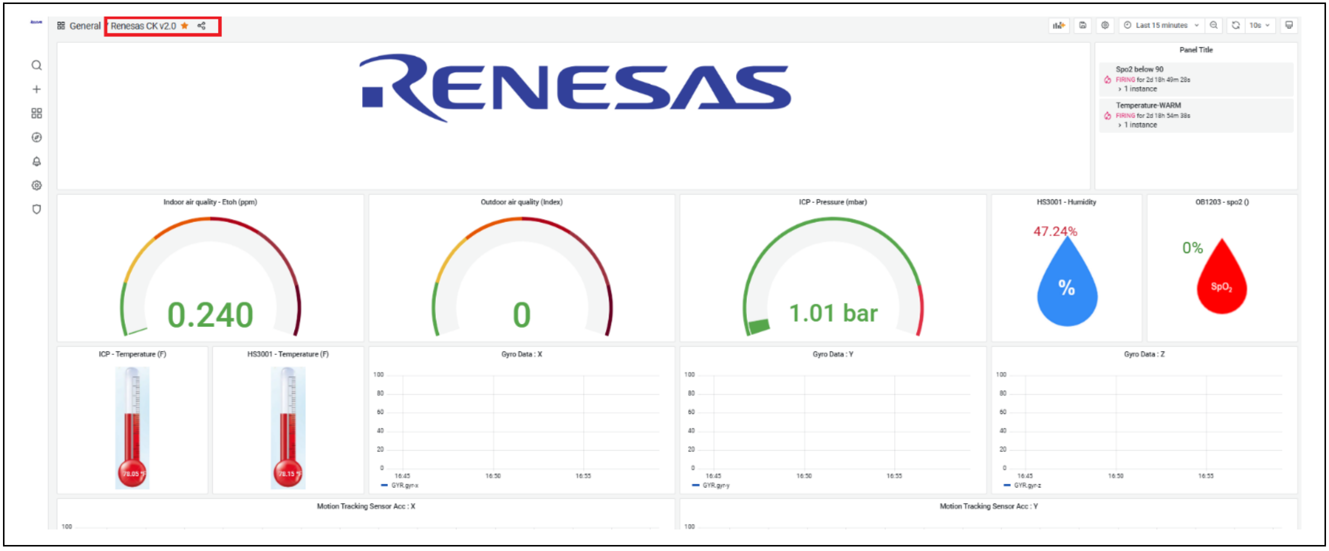


Figure 210. Renesas AWS Cloud Dashboard Types

Choose Renesas CK v2.0.

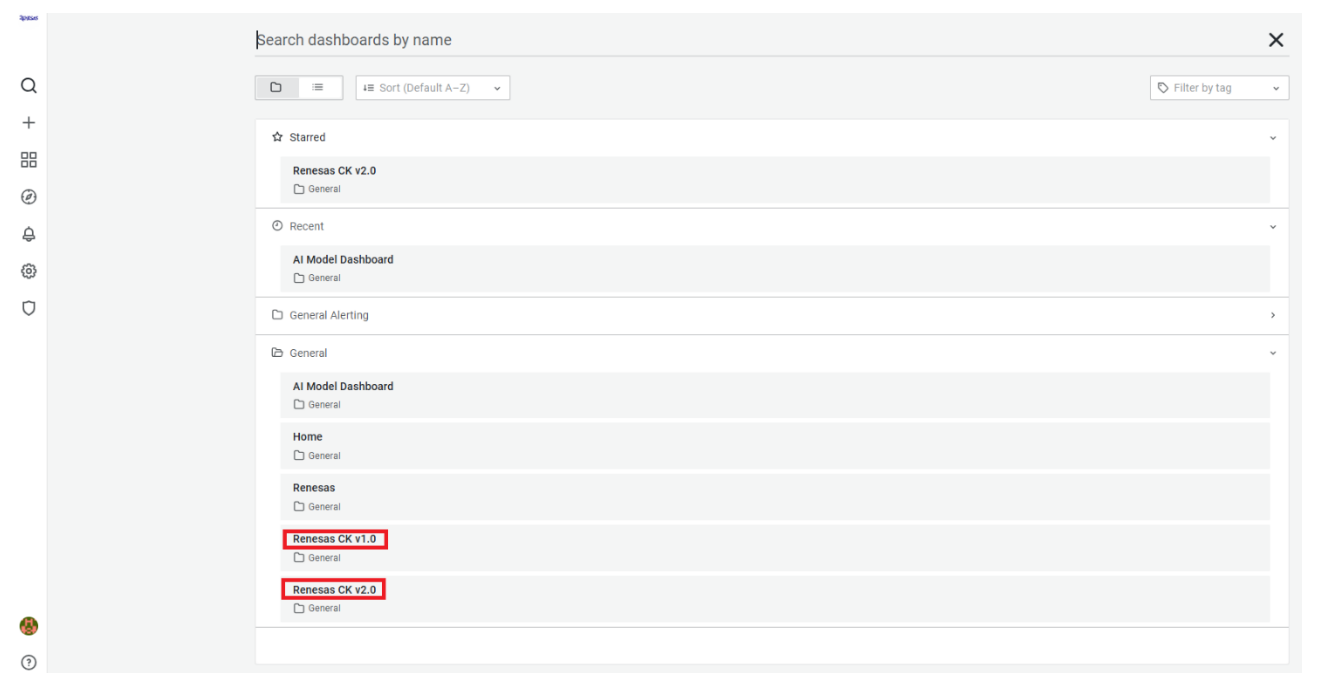


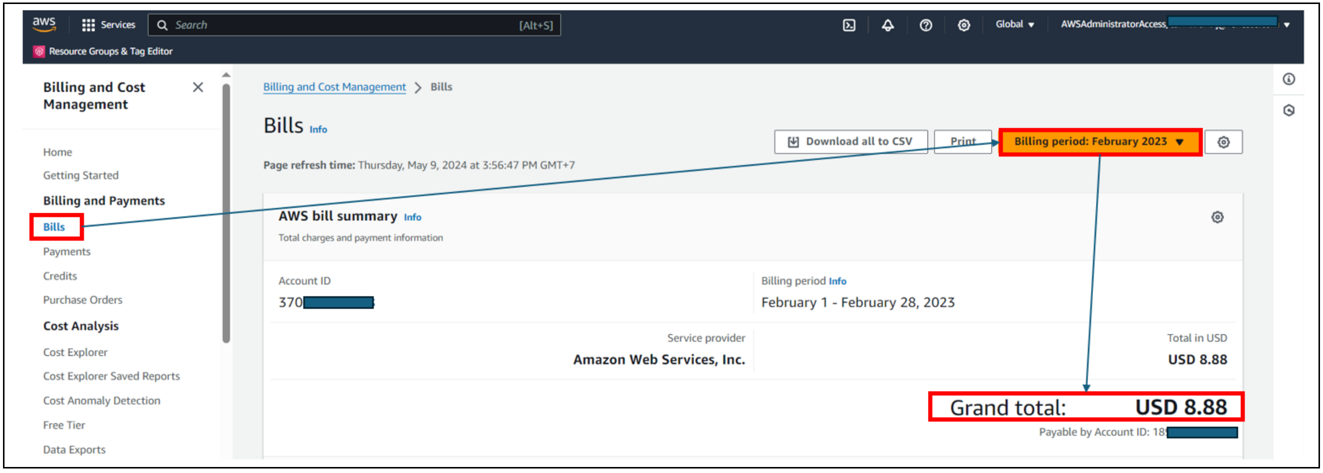
Figure 211. Choosing Renesas dashboards based on the cloud kit version

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

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

Go to **Account > Bills.**

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



Page refresh time: Thursday, May 9, 2024 at 3:56:47 PM GMT+7

Account ID: 370[redacted]

Billing period Info: February 1 - February 28, 2023

Service provider: Amazon Web Services, Inc.

Total in USD: USD 8.88

Grand total: USD 8.88

Payable by Account ID: 18[redacted]

Figure 212. Check the amount spent in AWS Account

8.10 An error occurs when connecting to AWS

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

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

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

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

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

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

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

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

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

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

Website and Support

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

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

Revision History

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

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

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/.

Trademarks

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