RX Family

Provisioning Procedure for IoT Devices

Introduction

IoT device provisioning is required in order to connect to AWS IoT, a cloud service provided as part of Amazon Web Services™ (AWS). As used here, the term “provisioning” refers to the process of generating, utilizing, and managing authentication information such as things, private keys, and device certificates. Provisioning requires consideration of matters such as how to write authentication information to products as part of the manufacturing process (initial installation) and how to manage (protect) and update key data. These types of data are stored in the on-chip flash memory of RX Family MCUs. Since it is extremely difficult to modify the provisioning method for IoT devices afterward, the above-mentioned consideration must begin at the product development stage so that verification can be completed by the mass production phase.

Of the various provisioning methods provided by AWS, this document describes the “fleet provisioning method,” which automates provisioning during the manufacturing process and when the device is initially used.

Deploying the fleet provisioning method eliminates the need to devote time and effort to cumbersome provisioning procedures while making the provisioning process more secure and convenient.

What you will learn in this application note

- ✓ Overview of provisioning methods provided by AWS.
- ✓ How to realize fleet provisioning using demo and confirm operation. The steps to run the demo will be explained from "4 Running the Fleet Provisioning Demo".

The contents of this document are sufficient to implement provisioning, but if followed unmodified will result in important data saved as part of the provisioning processing, such as the private key and device certificate, being stored as “clear text” (unencrypted text) in the on-chip flash memory of the RX Family MCU. This means that if there is a security hole in a user program programmed to the RX Family MCU that allows arbitrary areas of memory to be read, the provisioning data in the flash memory could be accessed, possibly allowing an attacker to perform an unauthorized login to the user’s AWS account.

Using the Trusted Secure IP (TSIP) module of the RX Family MCU enables the private key and device certificate to be stored in encrypted form, greatly reducing the danger of unauthorized access to the provisioning data. For details of the TSIP module, see the page linked to below.

https://www.renesas.com/software-tool/trusted-secure-ip-driver

We strongly encourage using the TSIP module to boost security.

It is possible to reduce the risk of unauthorized access to provisioning data by improving software quality, but this approach can never completely eliminate it. In particular, if there are defects in the software of IoT devices, which are vulnerable to threats posed by attackers, it is recommended that firmware update functionality be used to apply corrections in a timely manner. For more information on firmware updates, please refer to the application note Renesas MCU Firmware Update Design Policy (R01AN5548).

Note : This application note shows an implementation example based on the operating environment of the CK-RX65N v1 board and the RYZ014A PMOD module,
but it can also be utilized with other boards and communication control combinations.
For each board and communication control combination, please see:

[GitHub] iot-reference-rx/Getting_Started_Guide.md at main · renesas/iot-reference-rx (github.com)

Note : Renesas announces to discontinue the existing Sequans-sourced LTE module known as the part number RYZ014A and will no longer be shipping this product.
With the discontinuation of RYZ014A, the CK-RX65N v1 board will also be discontinued.
If you are using RYZ014A in a current design or production, the Sequans part numbers, GM01Q is a pin and functionally compatible replacement for RYZ014A.

Below Cellular driver of RX family works the below alternate product combination.
- RYZ014A Cellular Module Control Module ： Sequans GM01Q is the compatible module.

Regarding EOL notice of the RYZ014A, please see ：
[The link] https://www.renesas.com/document/eln/plc-240004-end-life-eol-process-select-part-numbers?r=1503996
[The product page] https://www.renesas.com/products/wireless-connectivity/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module

Operating Environment

The operation described in this application note has been confirmed on the following environment.

Integrated development environment	e^2 studio 2024-04
Board	CK-RX65N
Toolchain	CC-RX Compiler v3.05.00
Emulator	E2OB (E2 Lite On Board) module of CK-RX65N

Before applying the contents of this application note to another MCU, a review of product-specific settings matching the specifications of the MCU should be made and adequate evaluation performed.

Related Application Notes

Information on documents related to this application note is provided below. Refer to these documents as necessary.

- Renesas MCU Firmware Update Design Policy (R01AN5548)
- RX Family How to implement FreeRTOS OTA by using Amazon Web Services on RX65N (R01AN5549)
- Firmware Integration Technology User’s Manual (R01AN1833)
- RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)

Information about boards, related programs, and development tools needed to develop RX cloud solutions is summarized on the page linked to below.

https://www.renesas.com/rx-cloud

Also, the following information publicly released by AWS may be of use. (The first two items are only available in Japanese.)

- Provisioning authentication information to devices in AWS IoT
  Video: https://youtu.be/gcJwNEQ2eLY
  Document: https://pages.awscloud.com/rs/112-TZM-766/images/EV_iot-deepdive-aws2_Sep-2020.pdf
- Document on fleet provisioning templates
  https://docs.aws.amazon.com/iot/latest/developerguide/provision-template.html
- Document on AWS IoT Core policies
  https://docs.aws.amazon.com/iot/latest/developerguide/iot-policies.html
- AWS IoT API reference document: CreateCertificateFromCsr
  https://docs.aws.amazon.com/iot/latest/apireference/API_CreateCertificateFromCsr.html
- Provisioning devices that don’t have device certificates using fleet provisioning
  https://docs.aws.amazon.com/iot/latest/developerguide/provision-wo-cert.html
- How to automate onboarding of IoT devices to AWS IoT Core at scale with Fleet Provisioning
  https://aws.amazon.com/blogs/iot/how-to-automate-onboarding-of-iot-devices-to-aws-iot-core-at-scale-with-fleet-provisioning/

Contents

- AWS™ is a trademark of Amazon.com, Inc. or its affiliates. (https://aws.amazon.com/trademark-guidelines/)
- FreeRTOS™ is a trademark of Amazon Web Services, Inc. (https://freertos.org/copyright.html)

1. Terminology

The following terms are used in this document.

Table 1.1 List of Terms

Term	Meaning
AWS	A suite of cloud computing services provided by Amazon Web Services, Inc.
FreeRTOS	An open-source real-time operating system for embedded systems.
Provisioning	Device provisioning. Certification of a device to enable communication with AWS IoT Core.
Fleet provisioning	Functionality that implements automated provisioning of IoT devices when they are turned on for the first time.

2. Device Provisioning

IoT device provisioning refers to the process of generating a unique ID (such as an X.509 certificate or private key) for a device, registering the unique ID with an AWS IoT endpoint, and linking the necessary access privileges (IoT policies, etc.) to enable the device to connect securely to AWS IoT and other cloud-based applications. (See Figure 2.2)

Device provisioning on AWS IoT makes use of AWS IoT Core functionality such as just-in-time-registration (JITR) and just-in-time-provisioning (JITP) to automate the process of registering the identity of each device in the AWS cloud and linking it with the necessary permissions, making it easy the perform provisioning for multiple devices. However, the process of securely generating a unique ID and writing it to each device is the responsibility of the user, and for OEM vendors manufacturing large numbers of devices, this process can involve manual operations and be quite time consuming.

Fleet provisioning, which is described in this document, is one way to deal with this issue.

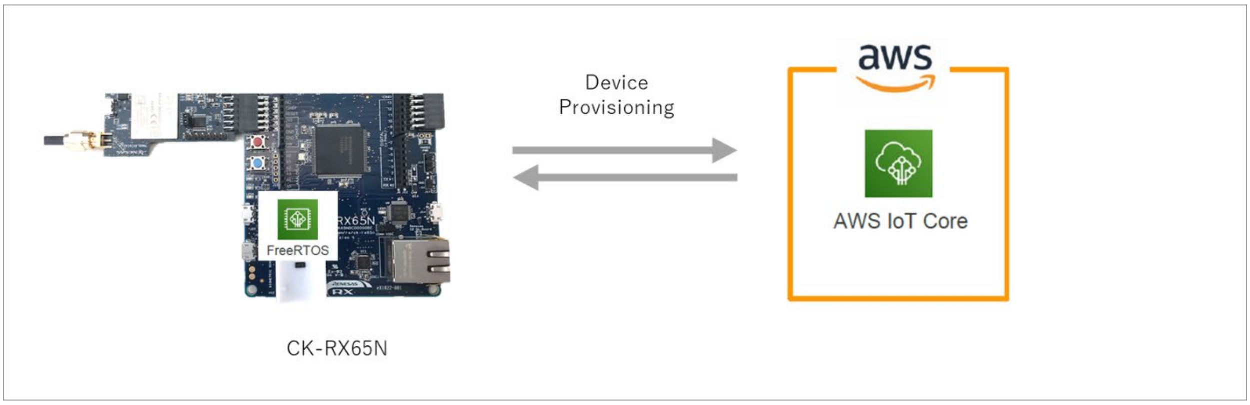


Figure 2.1 Device Provisioning

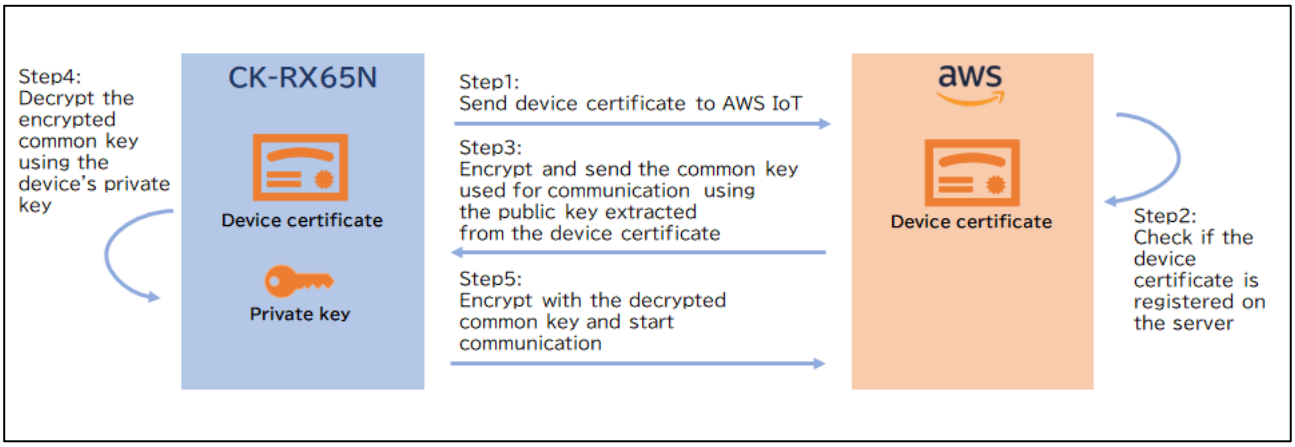

Step4:
Decrypt the encrypted common key using the device's private key
CK-RX65N
Device certificate
Private key
Step1:
Send device certificate to AWS IoT
Step3:
Encrypt and send the common key used for communication using the public key extracted from the device certificate
Step5:
Encrypt with the decrypted common key and start communication
aws
Device certificate
Step2:
Check if the device certificate is registered on the server

Figure 2.2 IoT Device Provisioning

2.1 Provisioning Methods of AWS IoT

AWS IoT allows the user to select from the provisioning methods listed below.

AWS allows the user to select the device provisioning method that best matches their application. Multiple provisioning methods are available to accommodate market demand and a variety of use cases. The following document describes now the various provisioning methods work as well as their advantages and disadvantages in order to assist users in making a selection. We recommend referencing this document when considering the different provisioning methods.

https://pages.awscloud.com/rs/112-TZM-766/images/EV_iot-deepdive-aws2_Sep-2020.pdf#page=115

[Provisioning Methods of AWS IoT]

1. Private key and certificate issuance and pre-registration by AWS IoT (registration at time of device kitting)
2. Certificate issuance and pre-registration by AWS IoT (registration at time of device kitting)
3. Fleet provisioning registration (Described in this document.)
4. Certificate issuance by your own certification authority and pre-registration on AWS IoT
5. Certificate issuance by your own certification authority and registration by JITR
6. Certificate issuance by your own certification authority and registration by JITP
7. Registration of a certificate from an unregistered certification authority (multi-account registration)

When confirming the operation of FreeRTOS at the preliminary stages when considering mass production, the simplest approach is “private key and certificate issuance and pre-registration by AWS IoT.” In this case a private key certificate is issued and the source code is converted on AWS, and the resulting source code is embedded in the source code of FreeRTOS. However, it is difficult to embed individual certificates during manufacturing using this method. For this reason, this document focuses on fleet provisioning, which does not require use of a certification authority and imposes the lightest workload during mass production.

Note: A part of RX Family MCUs incorporate a Trusted Secure IP (TSIP) module. When the TSIP is used, an on-chip random number generator is used to generate an RSA or elliptic curve cryptosystem key pair, and the public key is extracted and sent to a user-specified certification authority, which appends and returns a certificate. This enables implementation of JITR or JITP. This method provides strong security while reducing the implementation cost, and it should be considered for practical use moving forward.

2.2 Fleet Provisioning Method

Fleet provisioning is a procedure in which provisioning takes place when each IoT device is started for the first time. Generally speaking, it can be implemented in either of the following two ways.

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

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

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

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

This document describes the implementation of a fleet provisioning demo that combines 1. and B). (See Figure 2.6.) The provisioning method presented in this document provides the following advantages.

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.

2.3 Provisioning by Claim (Approach Using Provisioning Claim Certificates)

Each device can be manufactured with an embedded provisioning claim certificate and private key. If these credentials have been registered with AWS IoT, AWS IoT can exchange them for a unique device certificate that can then be used in the normal operation of the device. This process consists of the steps listed below.

The design of provisioning by claim assumes a scenario in which all the devices are manufactured using a common provisioning claim certificate. The provisioning claim certificate only allows each device to do the following.

1. Establish an initial connection to AWS IoT Core.
2. Verify identity.
3. Use data communication as described below to request an ID to which the necessary permissions have been assigned.

The provisioning claim certificate common to all the devices is written to each device, along with the initial software, at a site such as the manufacturing plant. If the device already contains an individual private key, it can send a provisioning claim certificate to be signed by AWS IoT Core and a certificate signature request (CSR). (See Figure 2.6.)

In addition to the provisioning claim certificate presented by each device, fleet provisioning can make use of Lambda-based provisioning hooks to verify the attributes of devices. Examples of device attributes include serial number, MAC ID, and device location. We recommend that you consider making use of Lambda functions in provisioning transactions as a way to automate acceptance or rejection of the provisioning status of individual devices based on the custom attributes sent during this process.

(The demo project described in this application note does not make use of Lambda functions.)

Refer to the page linked to below for information on using AWS Lambda for provisioning.

https://docs.aws.amazon.com/iot/latest/developerguide/provision-wo-cert.html

"Using pre-provisioning hooks with the AWS CLI"

2.3.1 Overview of Provisioning by Claim (Using Provisioning Claim Certificate)

When the device is powered on and capable of establishing a network connection, one of the following workflows is executed.

Figure 2.5 and Figure 2.6 show the workflows for the CreateKeysAndCertificate method and CreateCertificateFromCsr method, respectively.

Also, you can confirm the details of the AWS IoT Fleet Provisioning Demo workflow (CreateCertificateFromCsr method), on which the fleet provisioning demo described in this document is based, by visiting the page linked to below.

https://aws.github.io/aws-iot-device-sdk-embedded-C/latest/docs/doxygen/output/html/fleet_provisioning_demo.html

1. Using the claim certificate written to the device beforehand, the device connects to AWS IoT Core via a secure TLS 1.2 connection. If the device contains a CSR, this is presented along with the provisioning claim certificate.

2. The certificate is linked to an extremely restrictive policy that only provides access to IoT topics linked to the fleet provisioning process.

3. The fleet provisioning service returns a token providing “proof of ownership” to securely isolate the transaction and a valid certificate and private key payload. The token will be called later to activate the certificate. If a CSR was presented, it is used to generate the certificate.

4. The device sends a MQTT request to AWS IoT Core and presents the ownership token, the name of the fleet provisioning template created by the account owner, and (optionally) device attributes for provisioning validation. It is recommended that Lambda-based provisioning hooks be used to enable additional validation, such as checking the device’s serial number or MAC ID against a pre-approved list.

5. The fleet provisioning template is acted upon, the provisioning transaction takes place, and the results are returned. Typically, these results may include verification by Lambda function of device attributes, certificate activation, production policy attachment, and thing or group creation (optional).

6. Based on the results of the provisioning transaction, the status of the new certificate is returned. If the transaction was successful, the provisioning claim certificate is deprecated or rotated for the “production” certificate. If the transaction is denied, an “access denied” error is returned to the device.

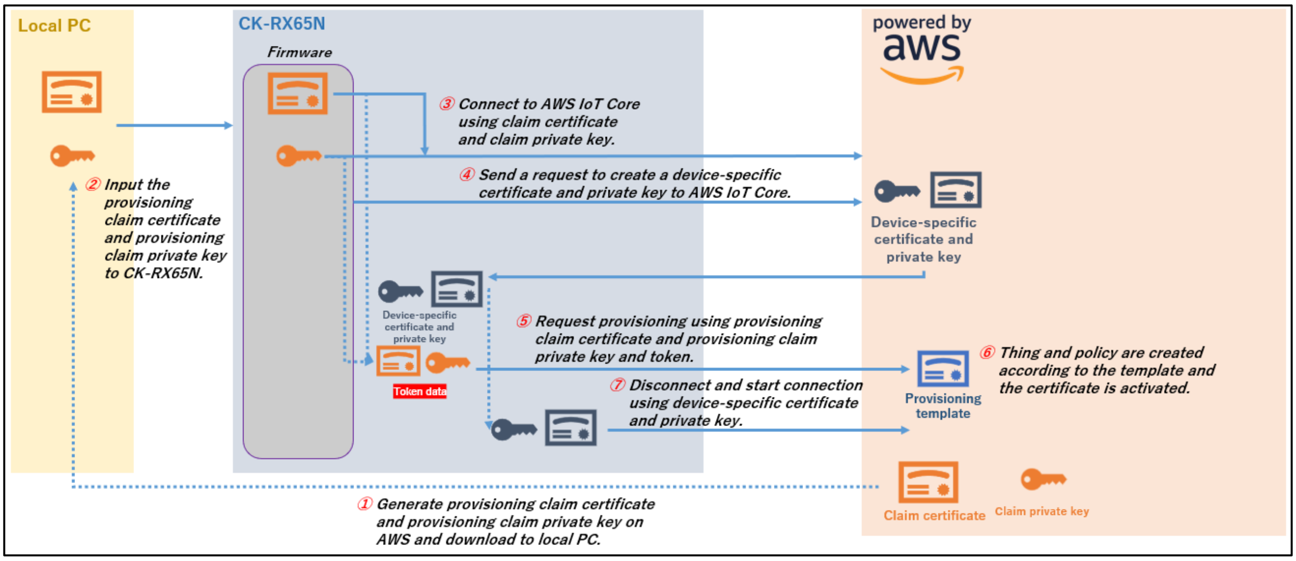

Local PC
CK-RX65N
Firmware
powered by aws
③ Connect to AWS IoT Core using claim certificate and claim private key.
④ Send a request to create a device-specific certificate and private key to AWS IoT Core.
Device-specific certificate and private key
② Input the provisioning claim certificate and provisioning claim private key to CK-RX65N.
Device-specific certificate and private key
⑤ Request provisioning using provisioning claim certificate and provisioning claim private key and token.
Token data
⑥ Thing and policy are created according to the template and the certificate is activated.
⑦ Disconnect and start connection using device-specific certificate and private key.
Provisioning template
① Generate provisioning claim certificate and provisioning claim private key on AWS and download to local PC.
Claim certificate
Claim private key

Figure 2.3 Workflow of Provisioning by Claim Using CreateKeysAndCertificate Method

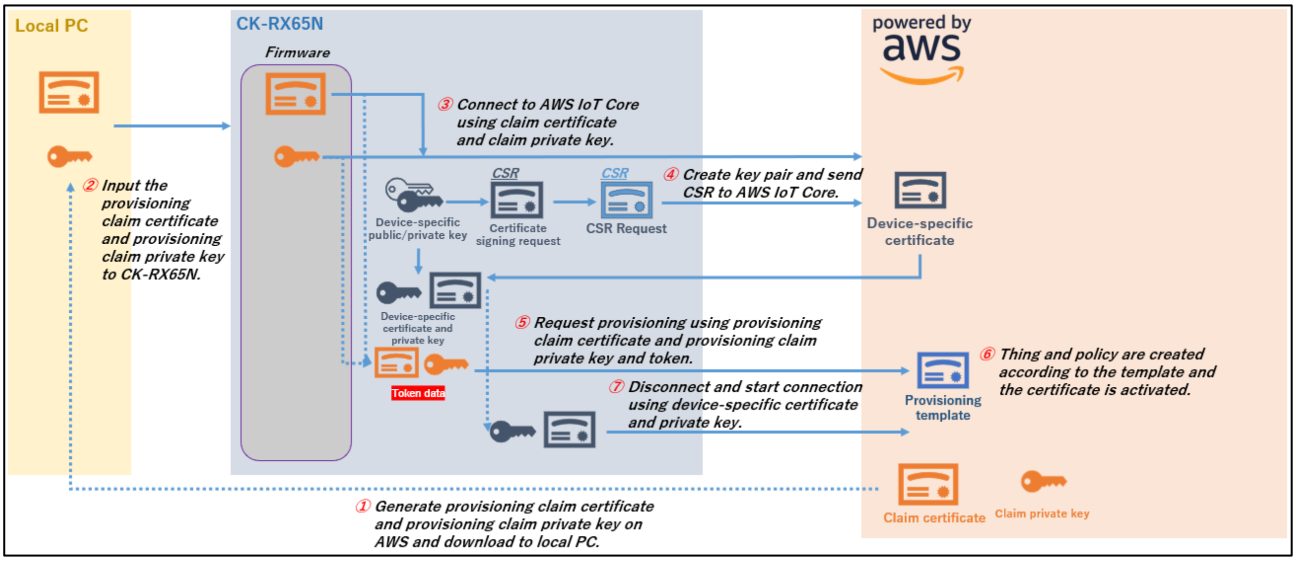

Local PC
CK-RX65N
Firmware
powered by aws
③ Connect to AWS IoT Core using claim certificate and claim private key.
CSR
CSR
④ Create key pair and send CSR to AWS IoT Core.
Device-specific public/private key
Certificate signing request
CSR Request
Device-specific certificate
② Input the provisioning claim certificate and provisioning claim private key to CK-RX65N.
Device-specific certificate and private key
⑤ Request provisioning using provisioning claim certificate and provisioning claim private key and token.
Token data
⑥ Thing and policy are created according to the template and the certificate is activated.
⑦ Disconnect and start connection using device-specific certificate and private key.
Provisioning template
① Generate provisioning claim certificate and provisioning claim private key on AWS and download to local PC.
Claim certificate
Claim private key

Figure 2.4 Workflow of Provisioning by Claim Using CreateCertificateFromCsr Method

2.3.2 Determining a Unique Thing Name

When sending a request to MQTT during fleet provisioning, the device's serial number can be included in the payload to ensure that each device has a unique thing name that does not duplicate an existing one.

Generally speaking, one of the following two methods is used to determine the serial number.

1. A random value generated by a random number generator or an ID value unique to the device is used as the serial number.
2. A Lambda-based provisioning hook and Amazon S3 or a user-specified database are used to change a temporarily assigned serial number to a unique serial number.

The example described in this document makes use of method 1. The unique ID assigned to each RX Family MCU is used to prevent duplication of thing names.

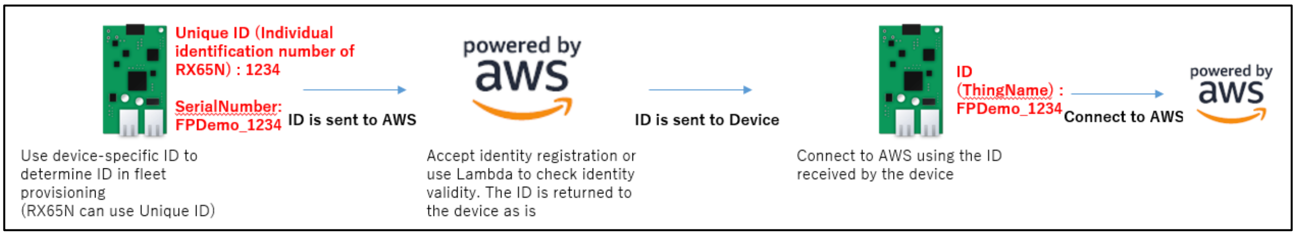
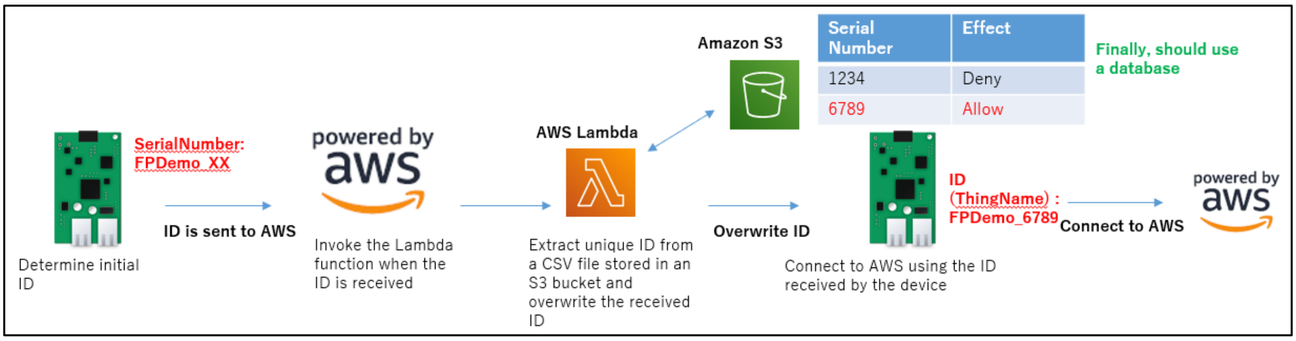

Unique ID (Individual identification number of RX65N) : 1234
SerialNumber: FPDemo_1234
ID is sent to AWS
powered by aws
ID is sent to Device
ID (ThingName) : FPDemo_1234
Connect to AWS
powered by aws
Use device-specific ID to determine ID in fleet provisioning (RX65N can use Unique ID)
Accept identity registration or use Lambda to check identity validity. The ID is returned to the device as is
Connect to AWS using the ID received by the device

Figure 2.5 Using a Random Value or Unique ID to Determine the Thing Name


Amazon S3
Finally, should use a database
SerialNumber: FPDemo_XX
powered by aws
AWS Lambda
ID is sent to AWS
Overwrite ID
ID (ThingName) : FPDemo_6789
Connect to AWS
powered by aws
Determine initial ID
Invoke the Lambda function when the ID is received
Extract unique ID from a CSV file stored in an S3 bucket and overwrite the received ID
Connect to AWS using the ID received by the device

Serial Number	Effect
1234	Deny
6789	Allow

Figure 2.6 Using Amazon S3 or a Database to Determine the Thing Name

3. Preparation

This section and those that follow describe the sequence of steps from importing the project accompanying this application note to running the fleet provisioning demo on the CK-RX65N board.

3.1 Hardware Environment

The components of the hardware environment for the demo project are listed below.

Table 3.1 Hardware Components

Item	Product Name	Provider	Description
Board	CK-RX65N	Renesas Electronics Corporation	RX65N Cloud Kit
PC	PC running Windows 10 (recommended)	—	Host PC for demo

3.2 Software Environment

The components of the software environment for the demo project are listed below.

Table 3.2 Components

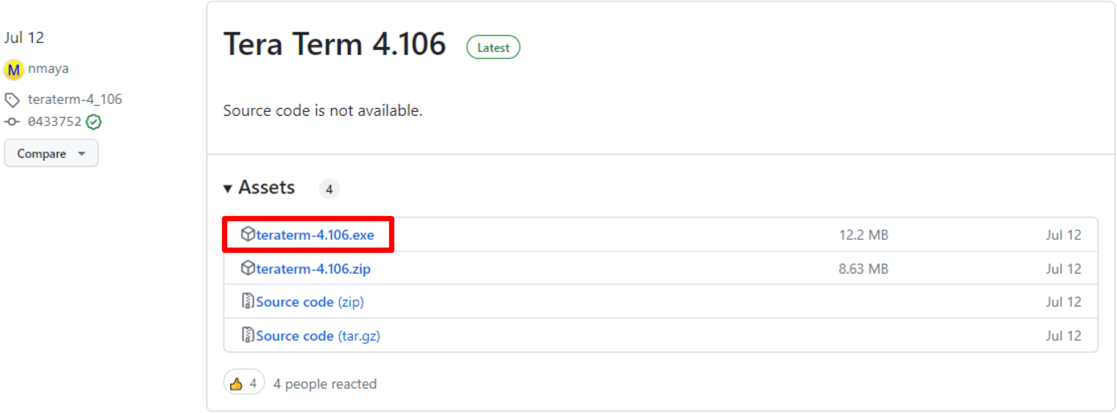
Item	Product Name	Version	Description
Integrated development environment	e^2 studio	2024-04	—
Toolchain	CC-RX	v3.05.00	—
Communication software	Tera Term	Ver 4.106	For displaying logs
FreeRTOS	v202210.01-LTS-rx	V1.2.1	
Emulator	E2OB (E2 Lite On Board) module of CK-RX65N	—	—

3.3 Tera Term Installation and Settings

The demo uses Tera Term to display log output.

1. Access the Tera Term download page.
   Tera Term download page (GitHub)

2. Download the Tera Term installer.



Jul 12
nmaya
teraterm-4_106
0433752
Compare

Tera Term 4.106 Latest

Source code is not available.

Assets 4

teraterm-4.106.exe	12.2 MB	Jul 12
teraterm-4.106.zip	8.63 MB	Jul 12
Source code (zip)		Jul 12
Source code (tar.gz)		Jul 12

👍 4 4 people reacted

Figure 3.1 Downloading Tera Term

3. Launch the installer, and follow the instructions that appear to install Tera Term.
4. In the Start menu, click the Tera Term icon and confirm that Tera Term starts.
5. Configure the following settings in Tera Term.

Table 3.3 Tera Term Settings

Item	Setting
Baud rate	115,200
Data length	8 bits
Parity	None
Stop bits	1 bit
Flow control	None

3.4 FreeRTOS Project

Figure 3.2 shows the software components of the demo project.

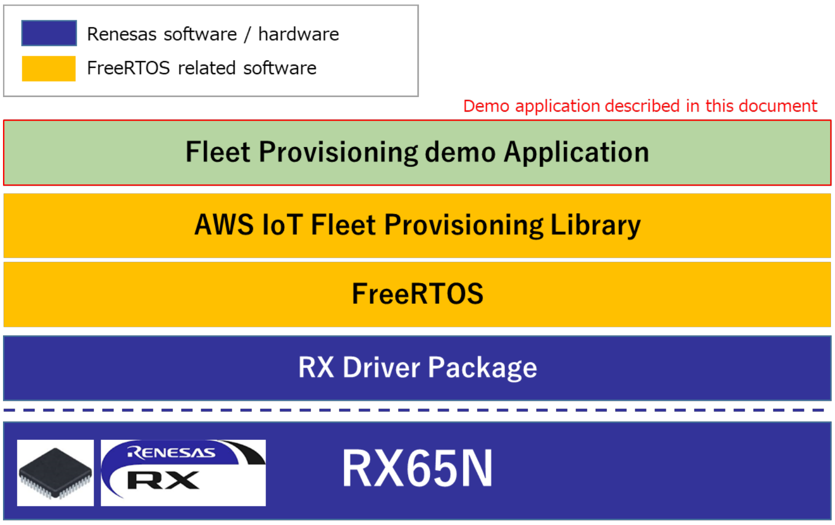

Renesas software / hardware
FreeRTOS related software
Demo application described in this document
Fleet Provisioning demo Application
AWS IoT Fleet Provisioning Library
FreeRTOS
RX Driver Package
RENESAS RX
RX65N

Figure 3.2 Components of Demo Project Accompanying This Application Note

The AWS IoT Fleet Provisioning Library for FreeRTOS is used to implement fleet provisioning functionality. RX Driver Package, FreeRTOS, AWS IoT Fleet Provisioning Library, and the demo application are available from the repository linked to below.

Demo application: iot-reference-rx : FreeRTOS reference repository

4. Running the Fleet Provisioning Demo

How to run the fleet provisioning demo application is described below.

4.1 Preparing the Running Environment

First, prepare the environment on which the demo will run. Figure 4.1 shows an example using the CK-RX65N board. Either a wired (Ethernet) or wireless (cellular) communication interface can be used to connect to AWS.

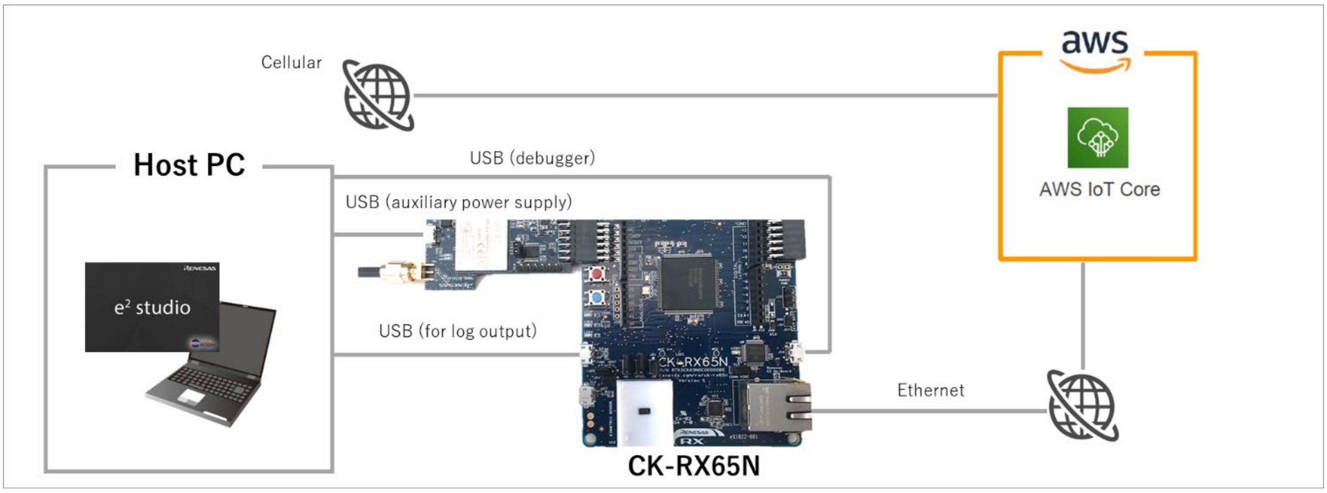


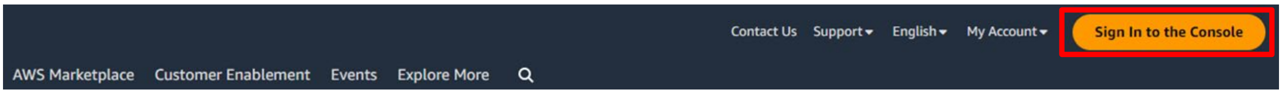
Figure 4.1 Demo Running Environment

4.2 AWS Preparation

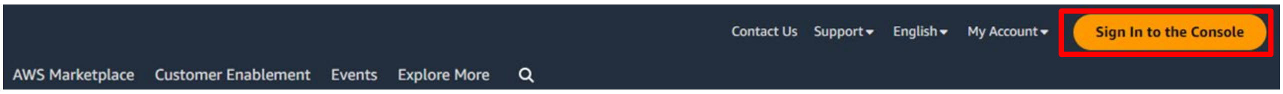
An AWS account is required to run the fleet provisioning demo application. If you do not have an account, start by creating an account and logging in to the console. Note that the screenshots of the AWS console appearing in this application note are current as of September 2023.

AWS top page (https://aws.amazon.com/)

① Select **Sign In to the Console** → **Get Started for Free** to create a new account.



② Click **Sign In to the Console** and sign in.



③ Select **Services** → **Internet of Things** → **IoT Core** to open the AWS IoT console.

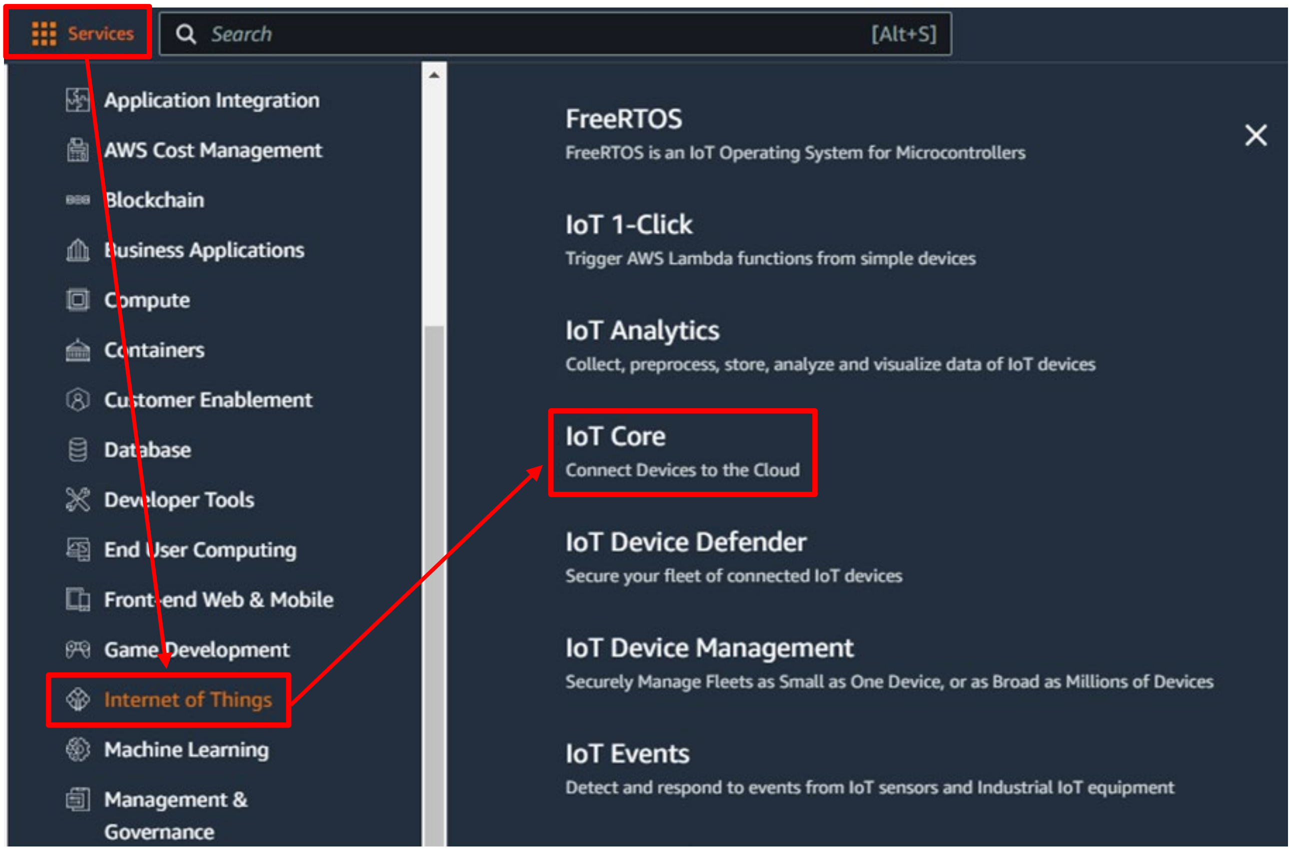

Services
Search
[Alt+S]
Application Integration
AWS Cost Management
Blockchain
Business Applications
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Management & Governance
FreeRTOS
FreeRTOS is an IoT Operating System for Microcontrollers
IoT 1-Click
Trigger AWS Lambda functions from simple devices
IoT Analytics
Collect, preprocess, store, analyze and visualize data of IoT devices
IoT Core
Connect Devices to the Cloud
IoT Device Defender
Secure your fleet of connected IoT devices
IoT Device Management
Securely Manage Fleets as Small as One Device, or as Broad as Millions of Devices
IoT Events
Detect and respond to events from IoT sensors and Industrial IoT equipment

Figure 4.2 AWS Console

4.3 AWS Settings for Fleet Provisioning

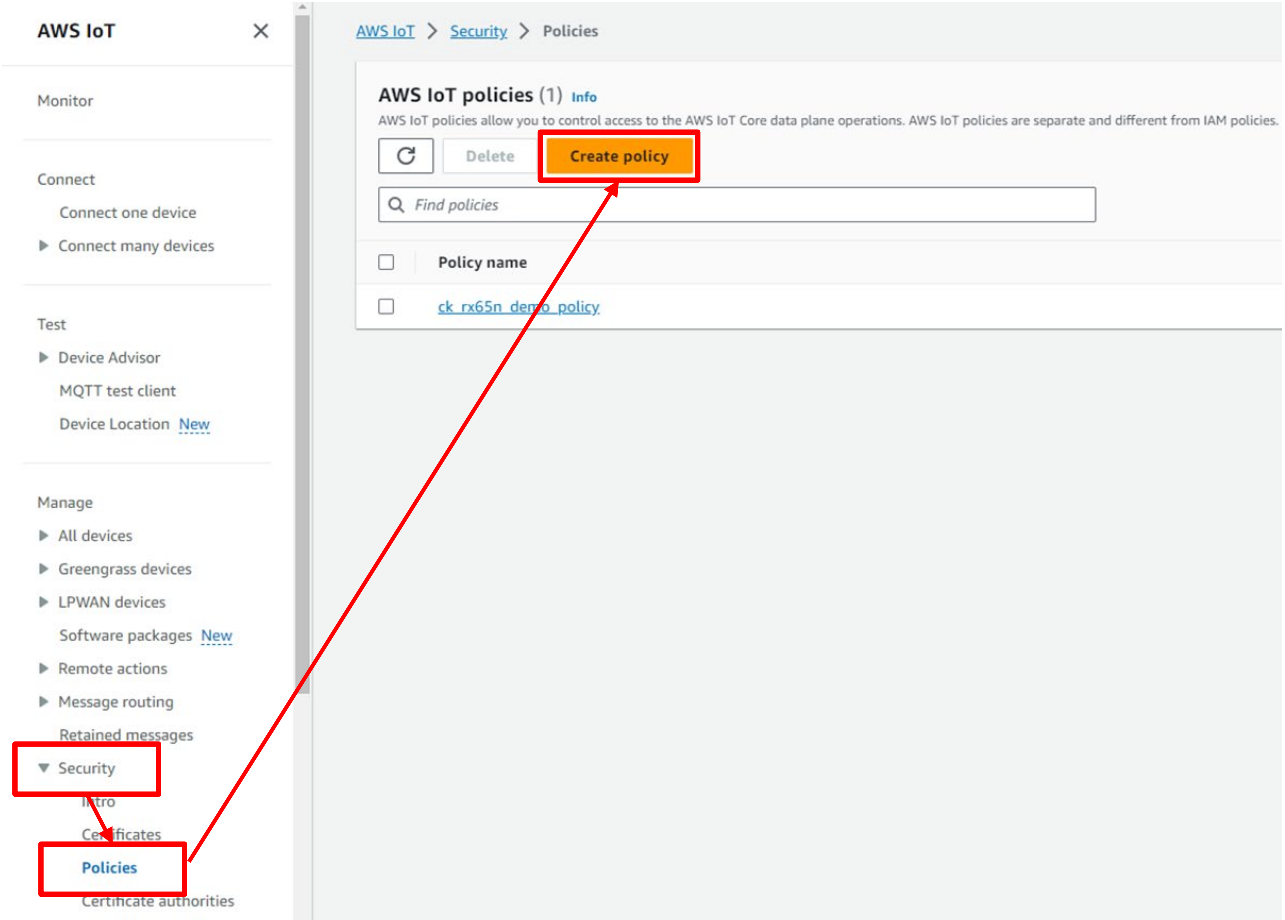
It is necessary to configure AWS settings in order to run the fleet provisioning demo.

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

4.3.1 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.

Select **Security** → **Policies** and then click the **Create policy** button.



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.

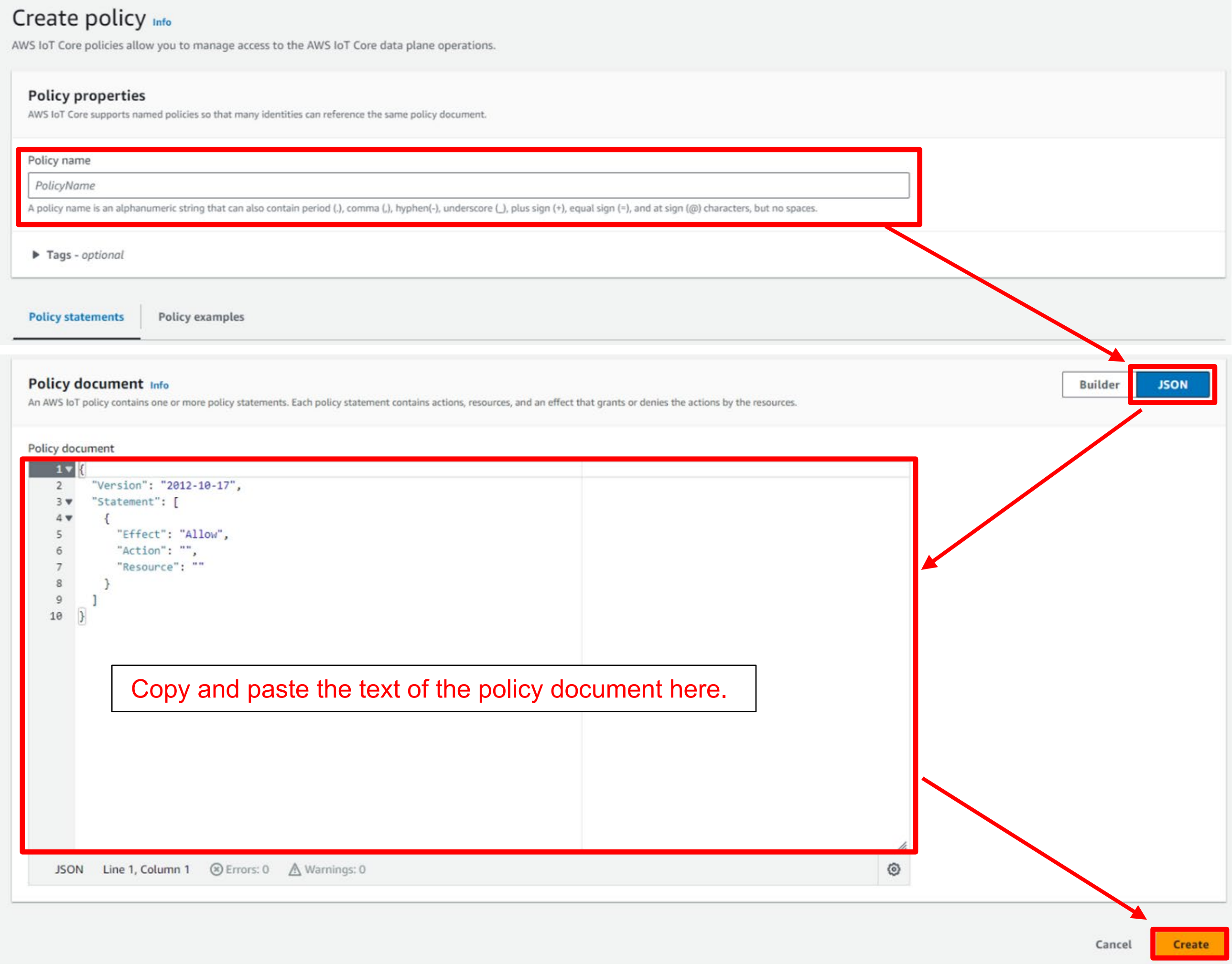
Figure 4.3 Creating an AWS IoT Policy (1)

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

Click the **JSON** button to display the policy document input field, then copy and paste the policy document shown in Figure 4.5 into the input field. When copying and pasting the policy document in Figure 4.5, make the following changes:

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

Click the **Create** button to create the 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

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.

▸ Tags - *optional*

Policy document Info

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

Policy document

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

Copy and paste the text of the policy document here.

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

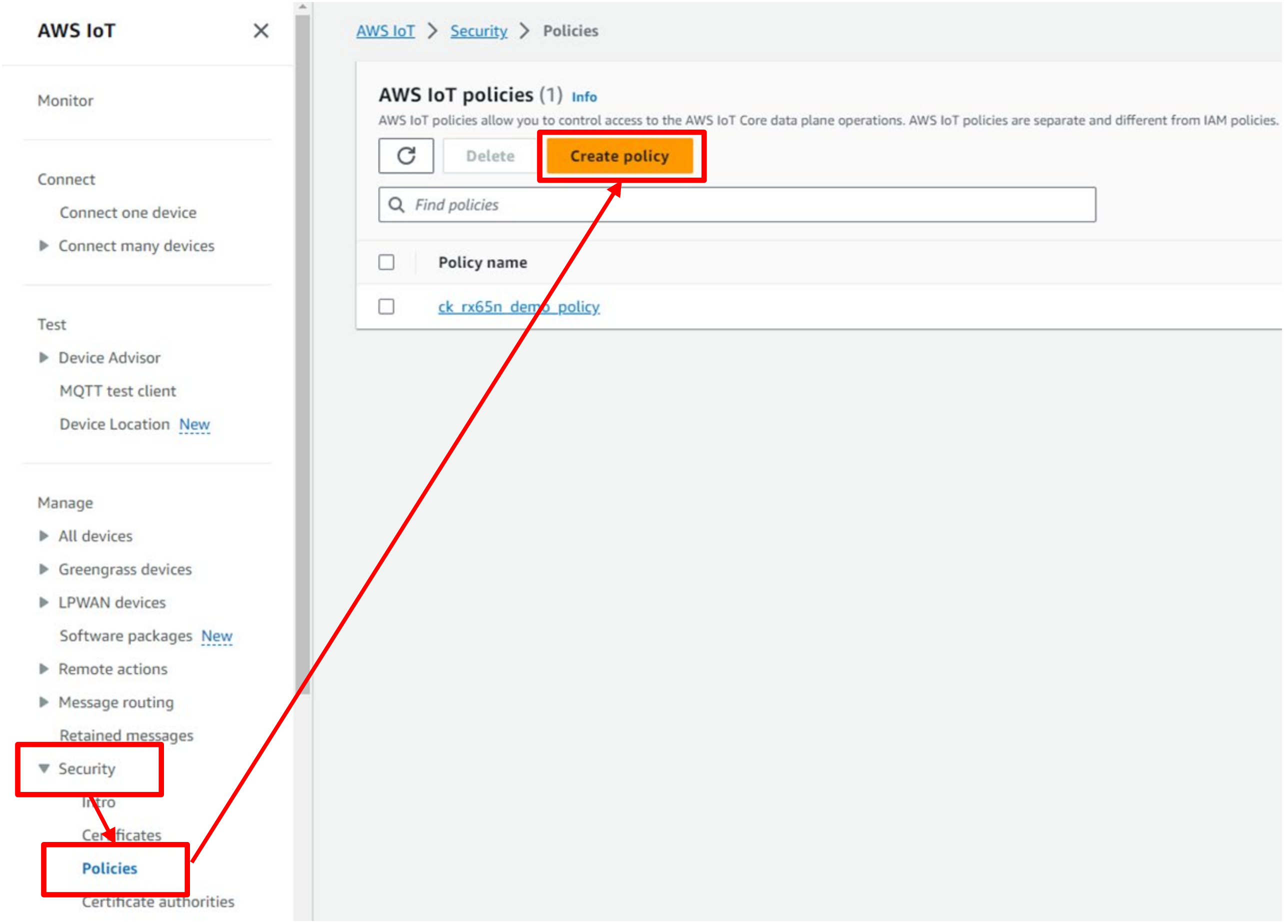
Figure 4.4 Creating an AWS IoT Policy (2)

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

Figure 4.5 Policy Document

Next, create a policy that will be attached to things created after fleet provisioning is run.

Select **Security** → **Policies** and then click the **Create policy** button.

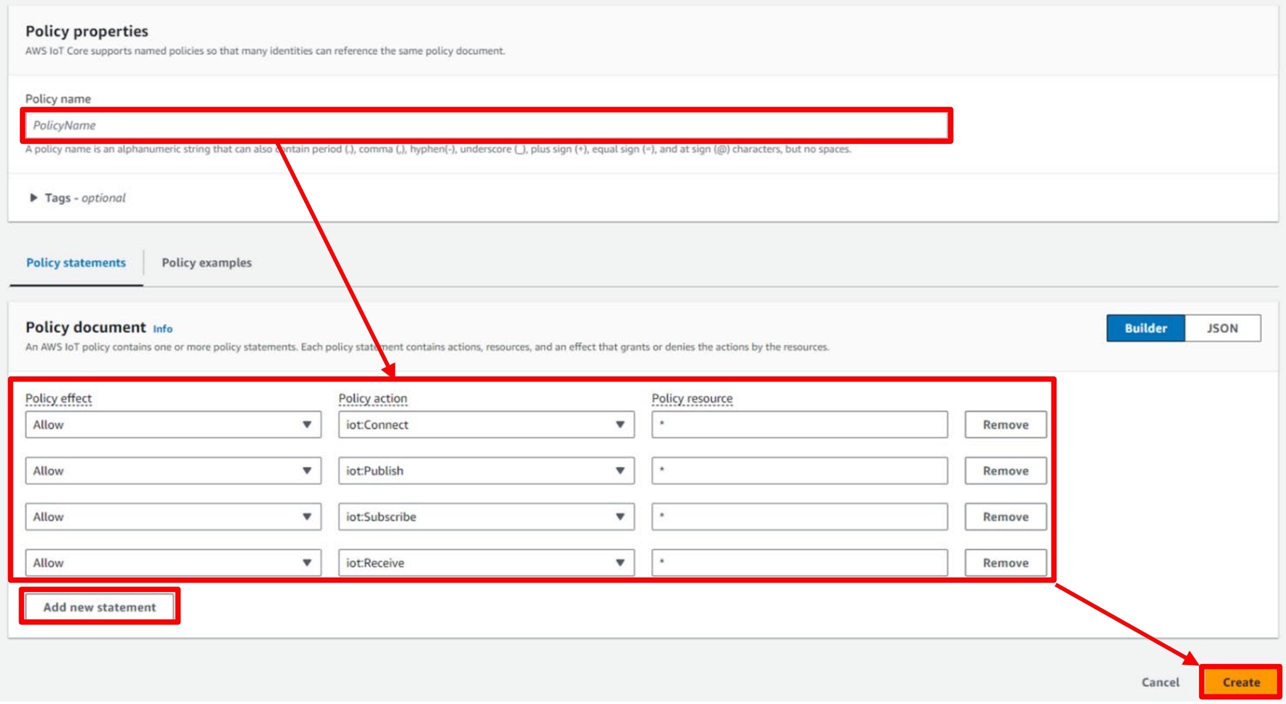


AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies.

Figure 4.6 Creating an AWS IoT Policy (1)

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

For **Policy action** under **Policy document**, select **Allow** for **iot:Connect**, **iot:Publish**, **iot:Subscribe**, and **iot:Receive**. For **Policy resource** enter the wildcard character (*) to allow all resources. By default you can configure one statement. Click the **Add new statement** button to add additional statements as needed.



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.

▸ Tags - *optional*

Policy document Info

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

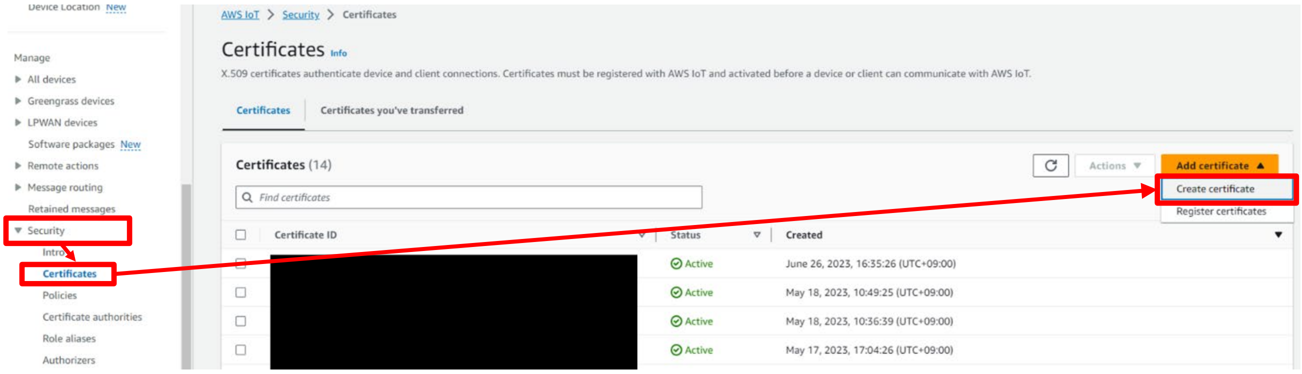
Policy effect	Policy action	Policy resource	
Allow ▾	iot:Connect ▾	*	Remove
Allow ▾	iot:Publish ▾	*	Remove
Allow ▾	iot:Subscribe ▾	*	Remove
Allow ▾	iot:Receive ▾	*	Remove

Figure 4.7 Creating an AWS IoT Policy (2)

4.3.2 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**.

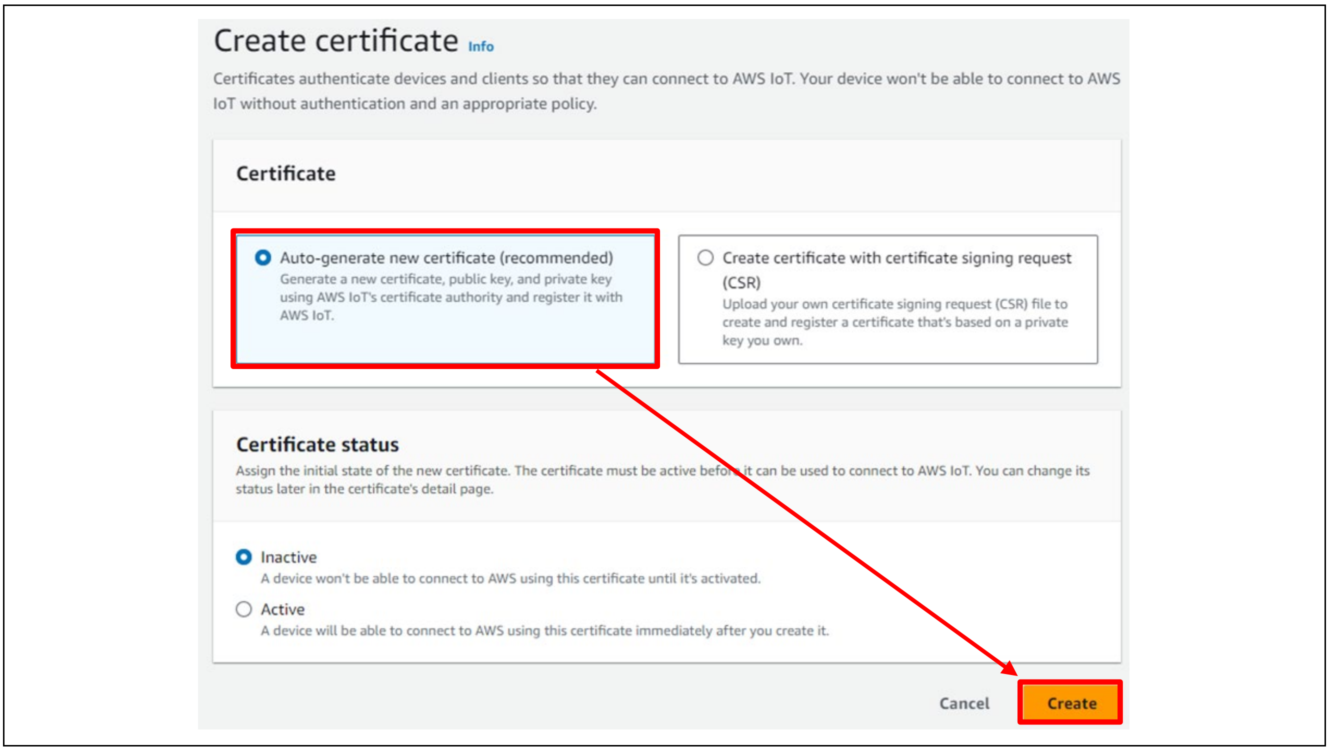


X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with AWS IoT.

□	Certificate ID	Status	Created
□	[redacted]	Active	June 26, 2023, 16:35:26 (UTC+09:00)
□	[redacted]	Active	May 18, 2023, 10:49:25 (UTC+09:00)
□	[redacted]	Active	May 18, 2023, 10:36:39 (UTC+09:00)
□	[redacted]	Active	May 17, 2023, 17:04:26 (UTC+09:00)

Figure 4.8 Creating a Certificate

Click **Auto-generate new certificate (recommended)** → **Create**.



Create certificate Info

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.

Certificate

● 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.

Certificate status

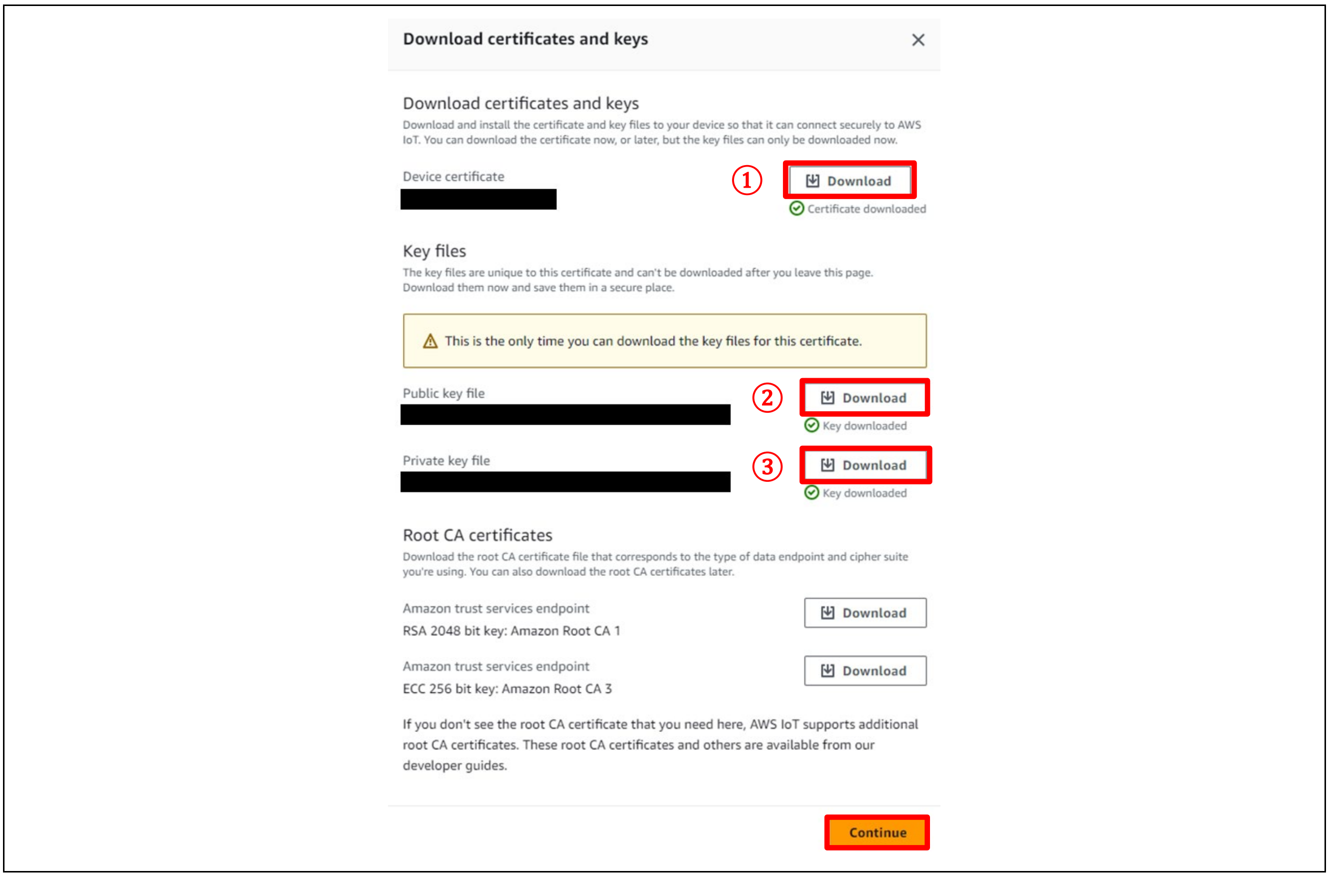
Assign the initial state of the new certificate. The certificate must be active before it can be used to connect to AWS IoT. You can change its status later in the certificate's detail page.

● 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 4.9 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]

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]

Key downloaded

Private key file
[redacted]

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.

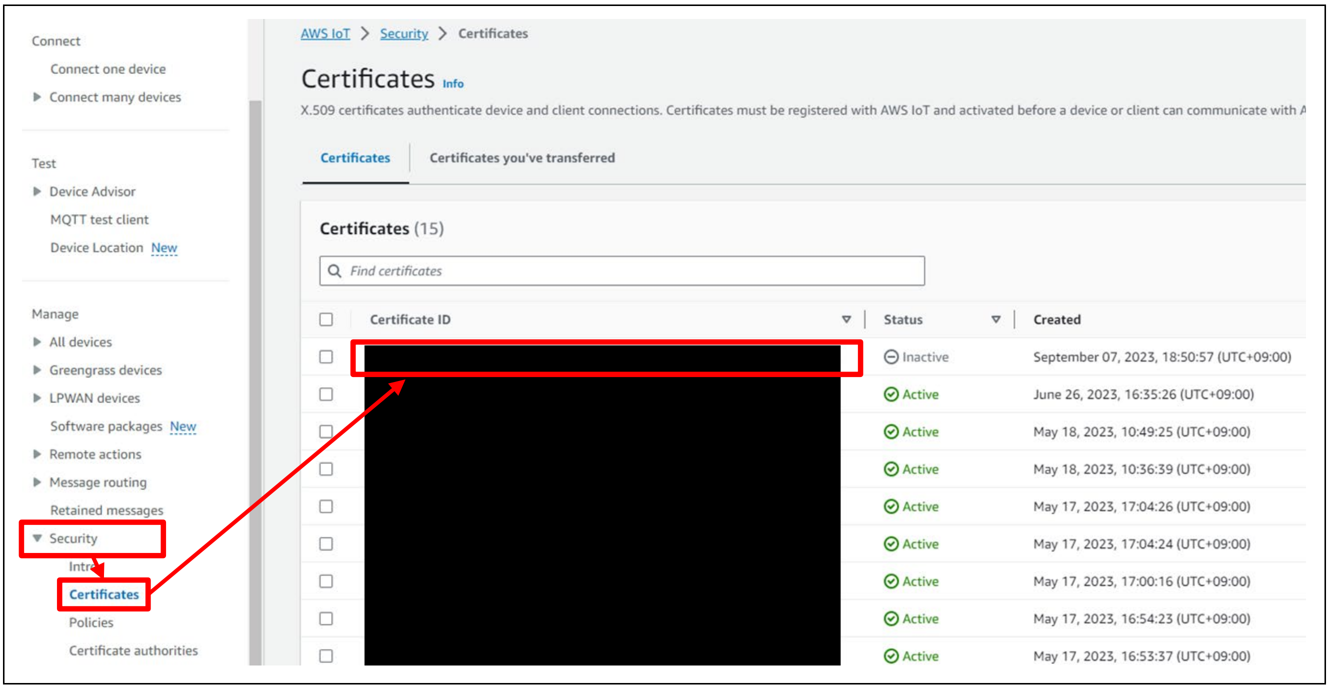
Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

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 from our developer guides.

Figure 4.10 Downloading the Certificate and Key Pair

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



Certificates Info

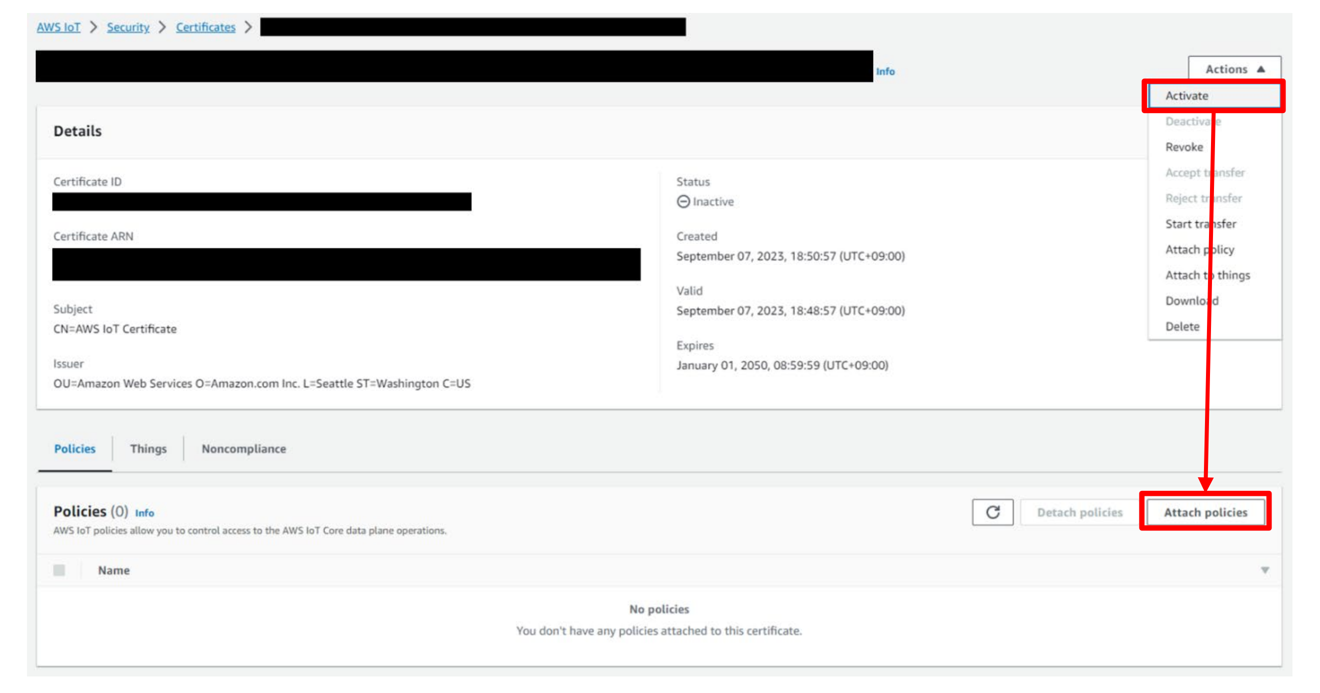
X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with A

Certificates (15)

□	Certificate ID	Status	Created
□	[redacted]	⊖ Inactive	September 07, 2023, 18:50:57 (UTC+09:00)
□	[redacted]	✓ Active	June 26, 2023, 16:35:26 (UTC+09:00)
□	[redacted]	✓ Active	May 18, 2023, 10:49:25 (UTC+09:00)
□	[redacted]	✓ Active	May 18, 2023, 10:36:39 (UTC+09:00)
□	[redacted]	✓ Active	May 17, 2023, 17:04:26 (UTC+09:00)
□	[redacted]	✓ Active	May 17, 2023, 17:04:24 (UTC+09:00)
□	[redacted]	✓ Active	May 17, 2023, 17:00:16 (UTC+09:00)
□	[redacted]	✓ Active	May 17, 2023, 16:54:23 (UTC+09:00)
□	[redacted]	✓ Active	May 17, 2023, 16:53:37 (UTC+09:00)

Figure 4.11 Certificate Settings

Click **Actions** → **Activate** to activate the certificate. Also 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
September 07, 2023, 18:50:57 (UTC+09:00)

Valid
September 07, 2023, 18:48:57 (UTC+09:00)

Expires
January 01, 2050, 08:59:59 (UTC+09:00)

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

□	Name

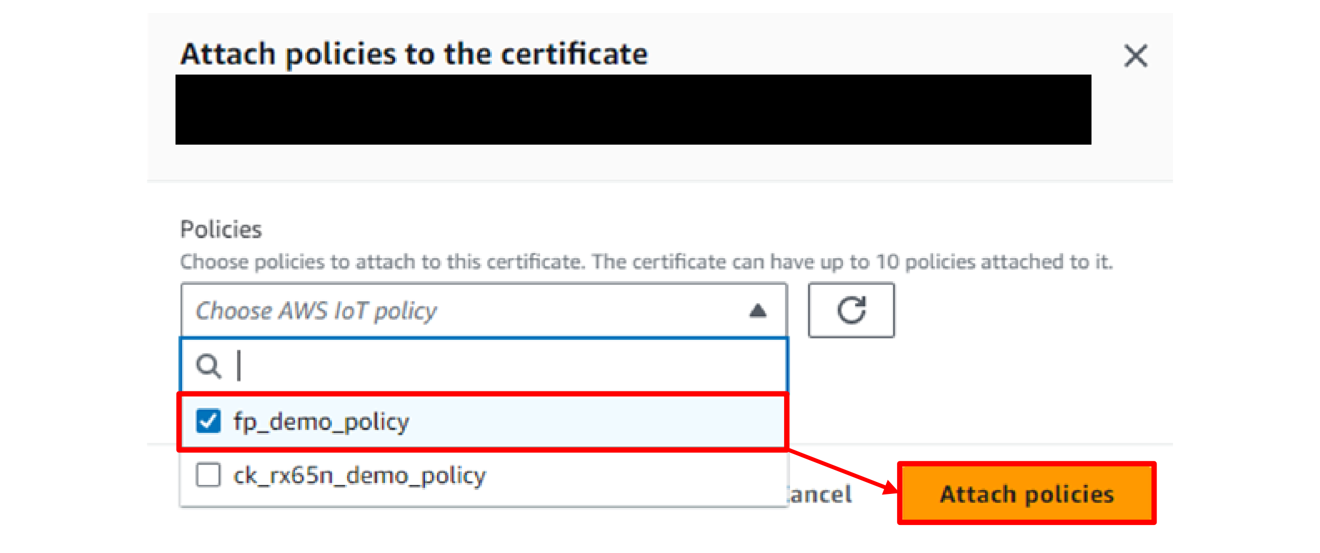
No policies
You don't have any policies attached to this certificate.

Figure 4.12 Certificate Settings: Attach Policies (1)

Clicking the **Attach policies** button opens the dialog box shown in Figure 4.13.

Select the policy to be used when fleet provisioning is run, created in 4.3.1, Policy Settings, and then click the **Attach policies** button to attach it to the certificate.

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



[redacted]

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

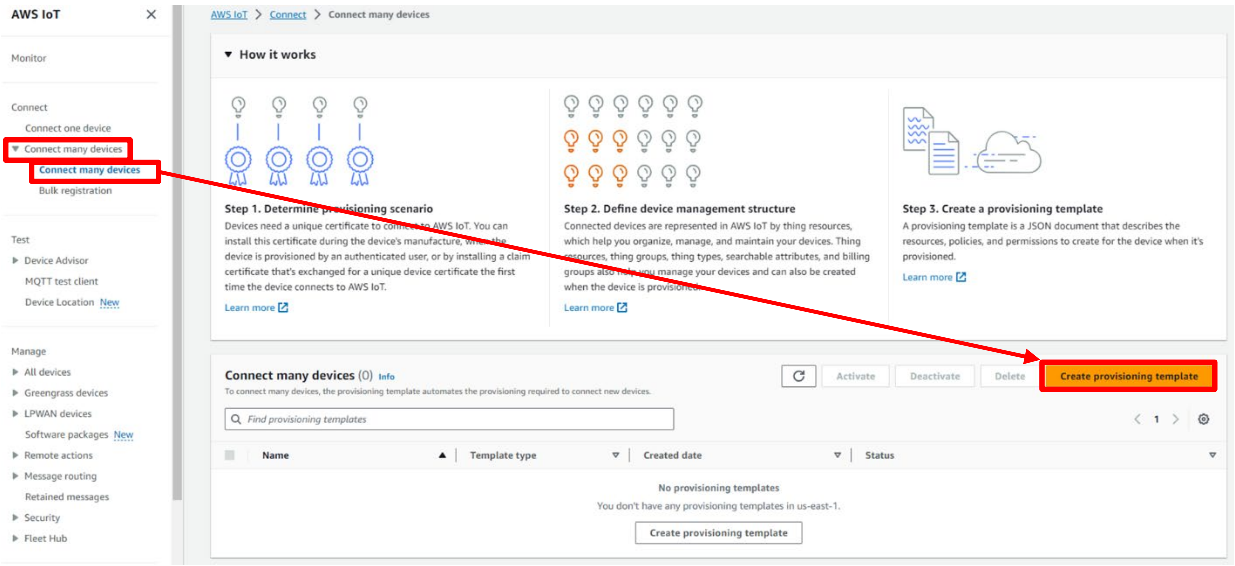
Choose AWS IoT policy ▲

☑ fp_demo_policy
□ ck_rx65n_demo_policy

Figure 4.13 Certificate Settings: Attach Policies (2)

4.3.3 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.

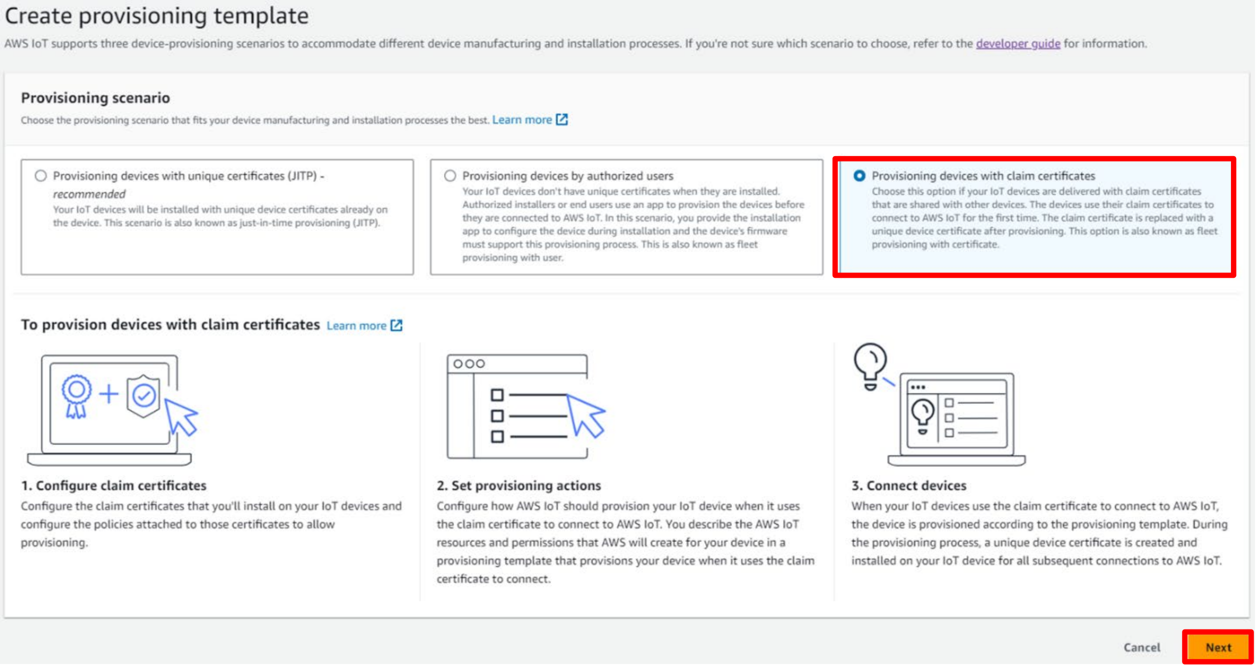
Step 3. Create a provisioning template
A provisioning template is a JSON document that describes the resources, policies, and permissions to create for the device when it's provisioned.

Connect many devices (0)
To connect many devices, the provisioning template automates the provisioning required to connect new devices.

No provisioning templates
You don't have any provisioning templates in us-east-1.

Figure 4.14 Creating a Provisioning Template (1)

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.

○ 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

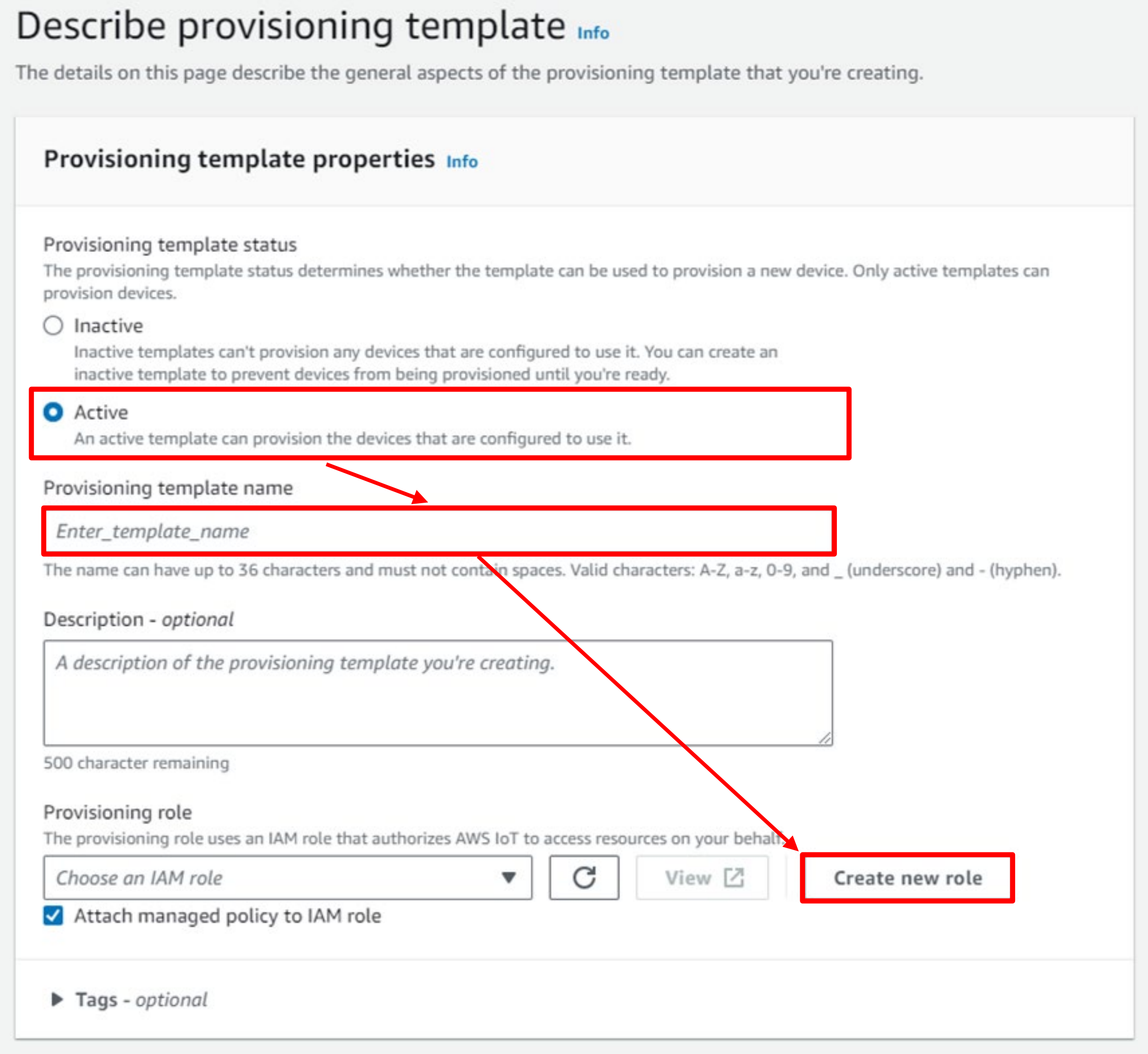
1. Configure claim certificates
Configure the claim certificates that you'll install on your IoT devices and configure the policies attached to those certificates to allow provisioning.

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 4.15 Creating a Provisioning Template (2)

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.

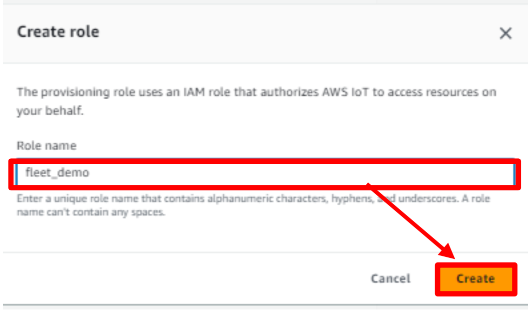
Choose an IAM role ▼

☑ Attach managed policy to IAM role

Figure 4.16 Creating a Provisioning Template (3)

Here **Role name** is set to **fleet_demo**, but you can create any role name you wish.

Next, click the **Create** button.



Create role

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

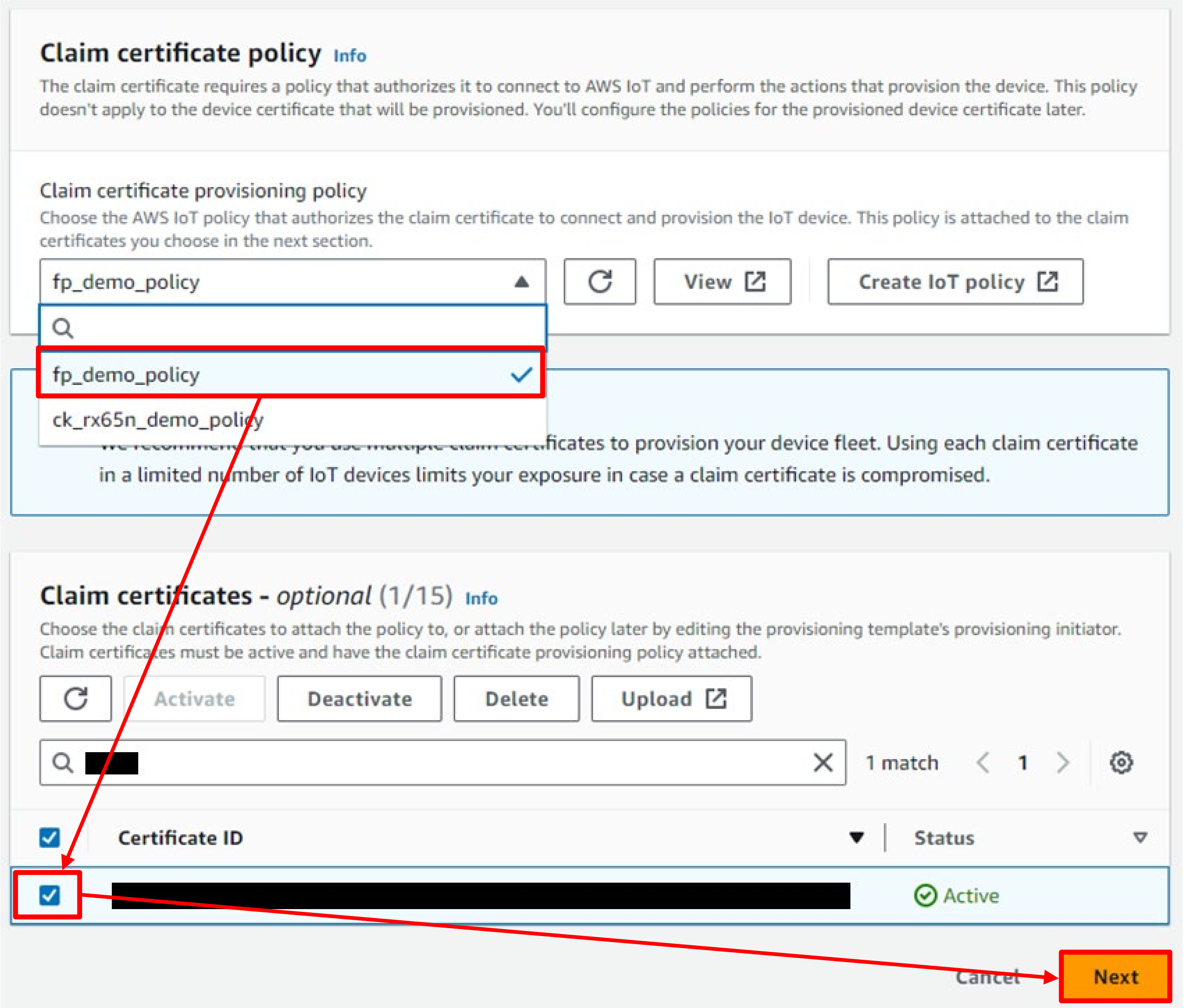
Role name

fleet_demo

Enter a unique role name that contains alphanumeric characters, hyphens, and underscores. A role name can't contain any spaces.

Figure 4.17 Creating a New Role

For **Claim certificate policy**, select the policy to be used when fleet provisioning is run, created in 4.3.1, for **Claim certificate**, select the certificate created in 4.3.2, 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.

fp_demo_policy

- fp_demo_policy ✓
- ck_rx65n_demo_policy

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

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

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

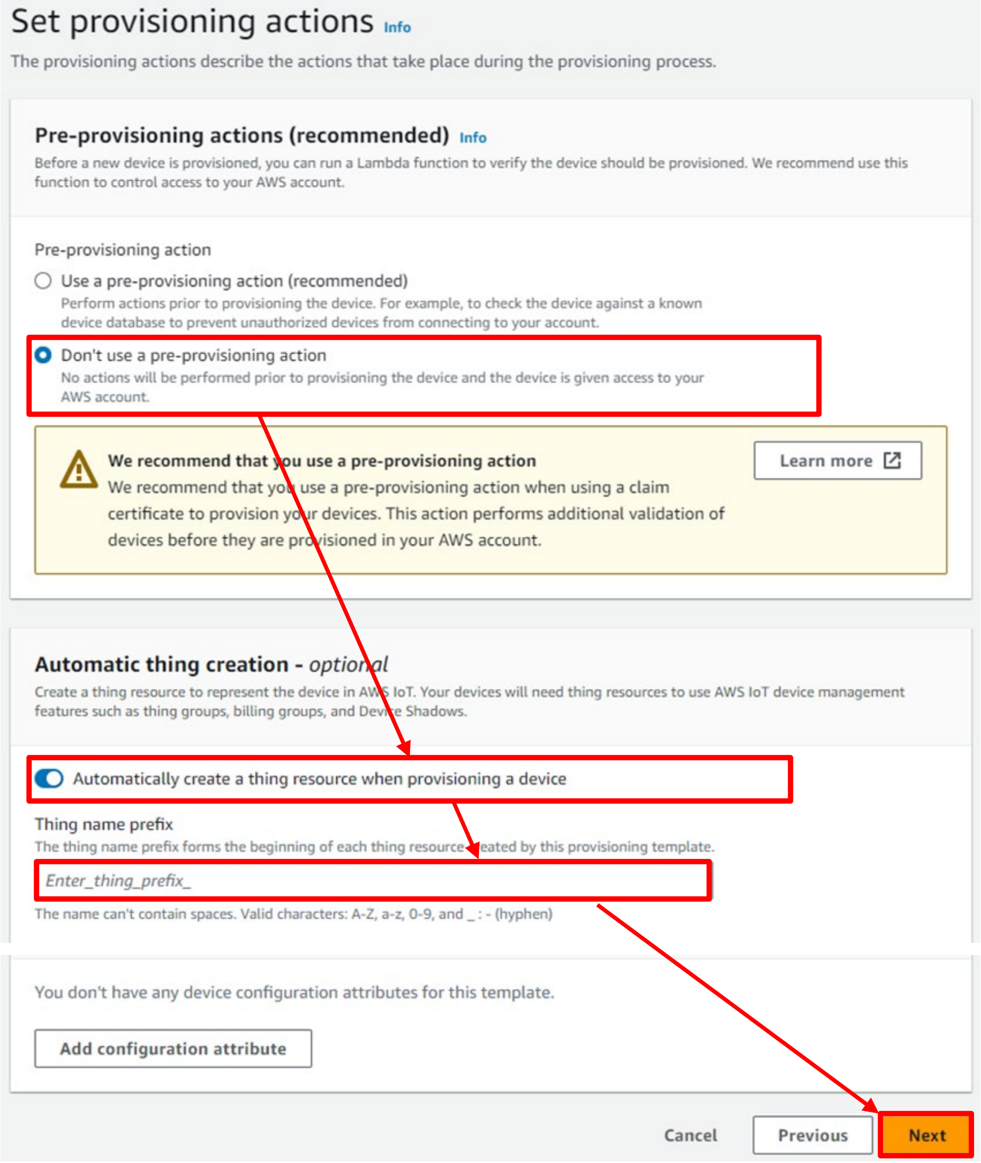
Search: [redacted] — 1 match

☑	Certificate ID	Status
☑	[redacted]	Active

Figure 4.18 Creating a Provisioning Template (4)

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.
https://docs.aws.amazon.com/iot/latest/developerguide/provision-wo-cert.html
"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.

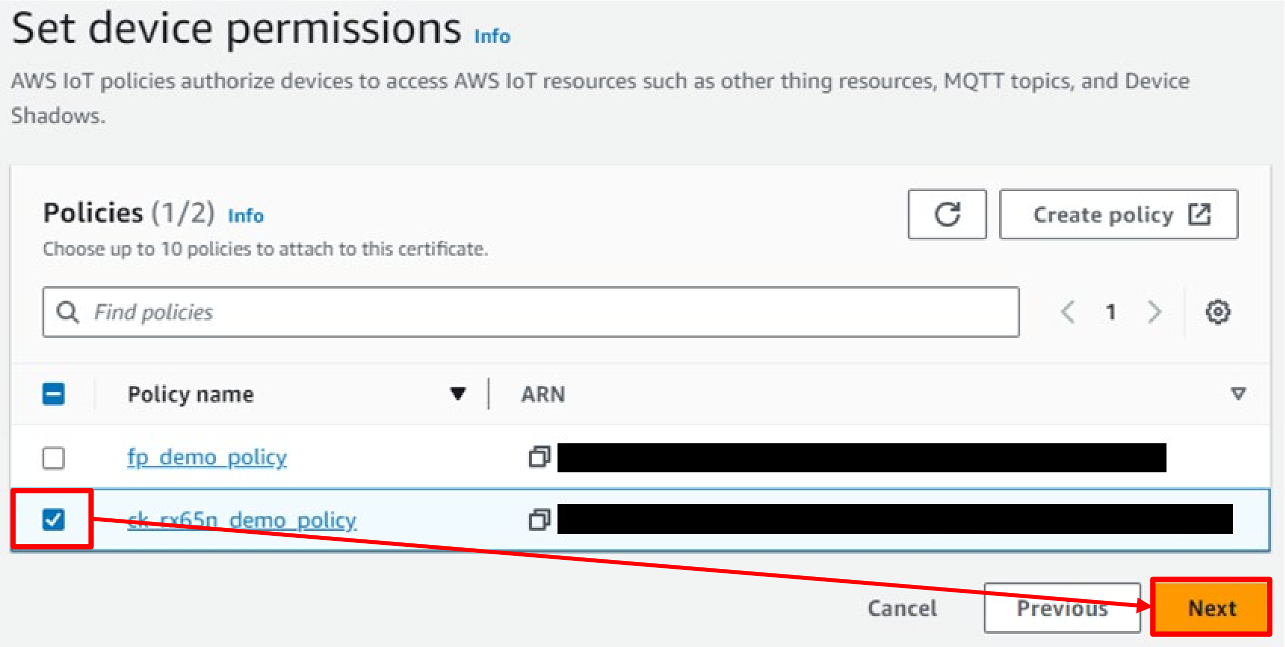
*Enter_thing_prefix_*

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

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

Figure 4.19 Creating a Provisioning Template (5)

For **Set device permissions**, check the box next to the policy attached to newly created things, which was created in 4.3.1, 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.

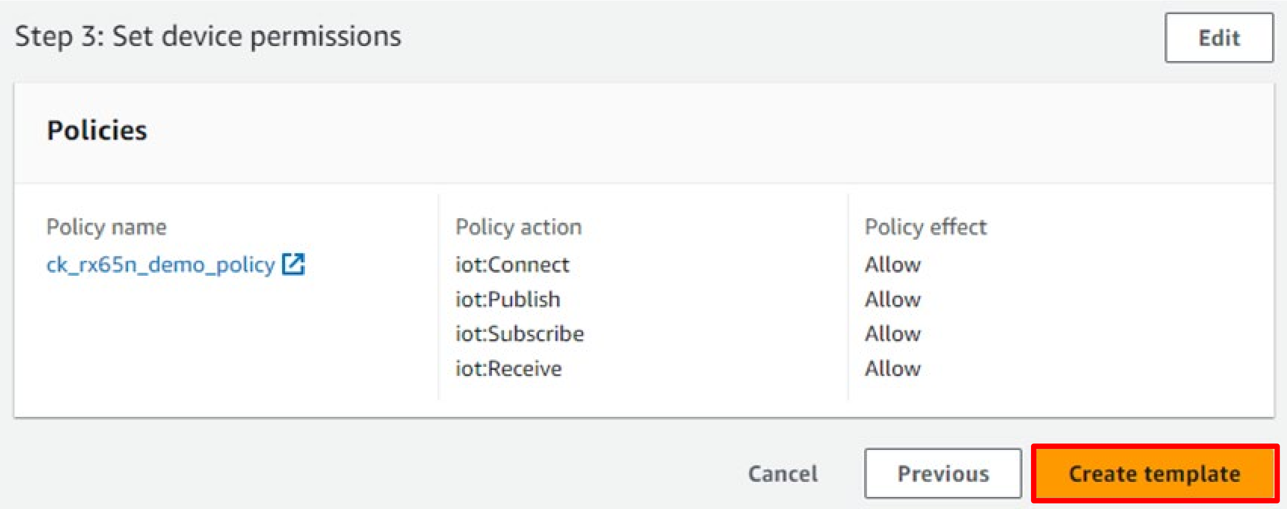
Policies (1/2) Info

Choose up to 10 policies to attach to this certificate.

	Policy name	ARN
□	fp_demo_policy	[redacted]
☑	ck_rx65n_demo_policy	[redacted]

Figure 4.20 Creating a Provisioning Template (6)

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_demo_policy	iot:Connect	Allow
	iot:Publish	Allow
	iot:Subscribe	Allow
	iot:Receive	Allow

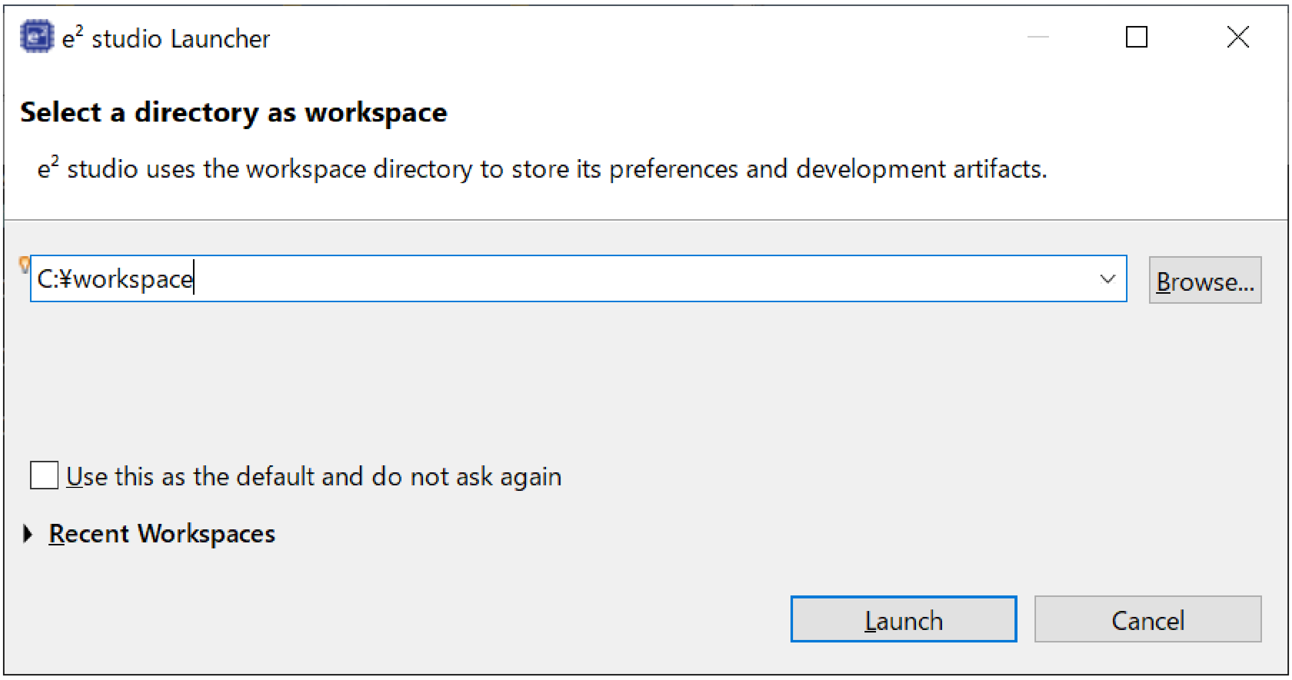
Figure 4.21 Creating a Provisioning Template (7)

4.4 Creating the Sample Projects

Follow the procedure below to create a sample project to perform provisioning for IoT devices using Amazon Web Services, as described in 4.5.

If you wish to use the import function to run the demo project, refer to the instructions in Getting Started Guide.

(1) Create a workspace in e^2 studio.
Launch e^2 studio and create a new workspace.
Keep the names of the workspace and the project files as short as possible. If the full path to the files at the lowest level of the directory structure exceeds 256 bytes, an error will occur when you build the project.
Errors may also occur if the file path contains Japanese characters, so make sure the name you enter contains only alphanumeric characters.
Example: Creating a workspace in location **C:\workspace**



e² studio Launcher

Select a directory as workspace

e² studio uses the workspace directory to store its preferences and development artifacts.

C:¥workspace

Figure 4.22 Dialog Box for Creating a Workspace

After launching e^2 studio, from the **File** menu select **New** > **Renesas C/C++ Project** > **Renesas RX** to display the **New C/C++ Project** dialog box.

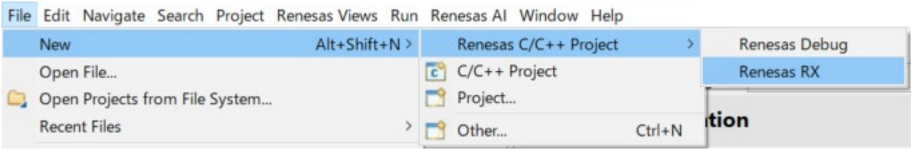
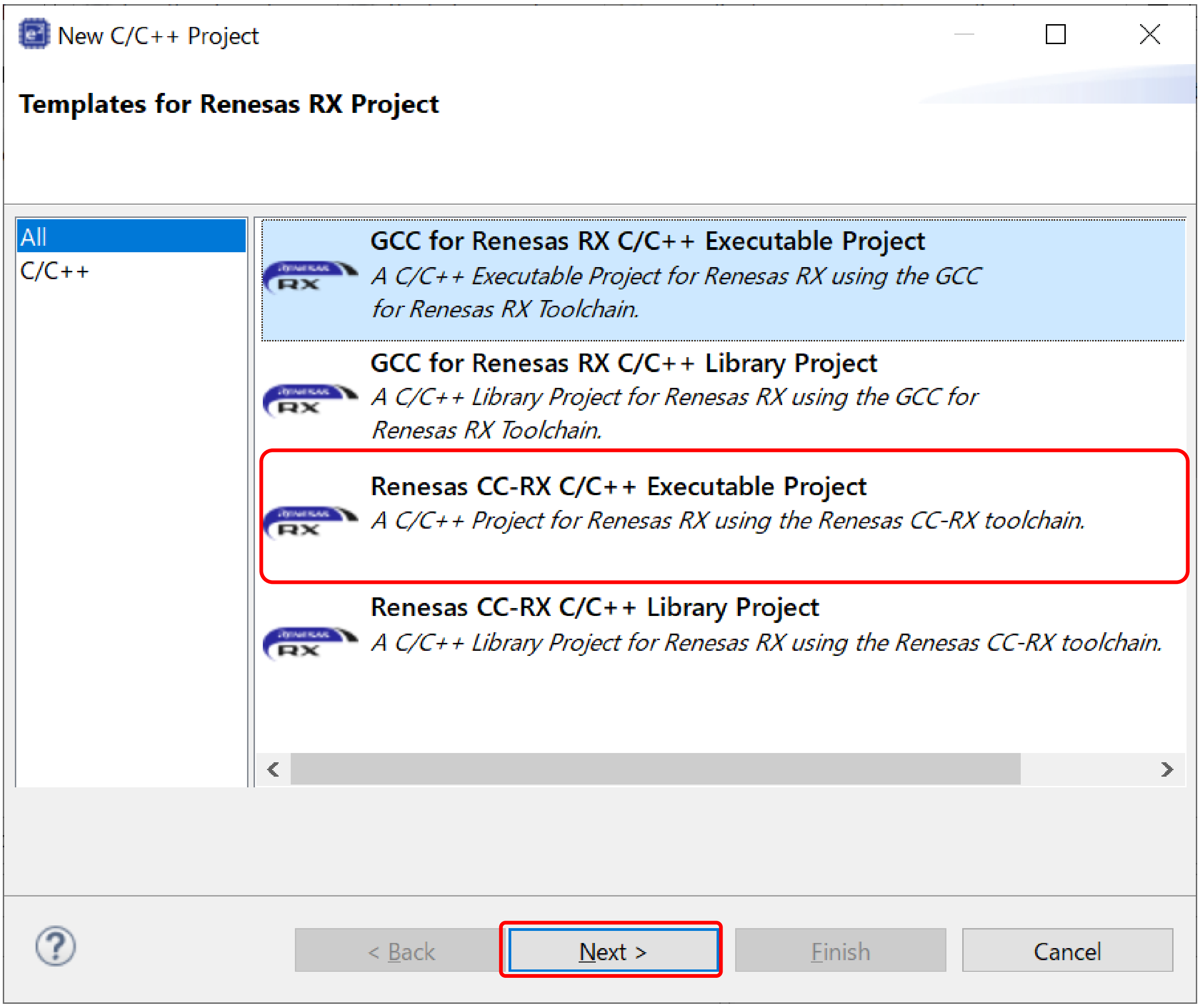


Figure 4.23 Creating a New Project from the File Menu

In the **New C/C++ Project** dialog box, select the type of project to be created. In this case, select **All**, then select **Renesas CC-RX C/C++ Executable Project** and click the **Next** button.
Selecting the project type opens the **New Renesas CC-RX Executable Project** dialog box.
To use GCC, select GCC for Renesas RX C/C++ Executable Project as the project type.



New C/C++ Project

Templates for Renesas RX Project

GCC for Renesas RX C/C++ Executable Project
A C/C++ Executable Project for Renesas RX using the GCC for Renesas RX Toolchain.

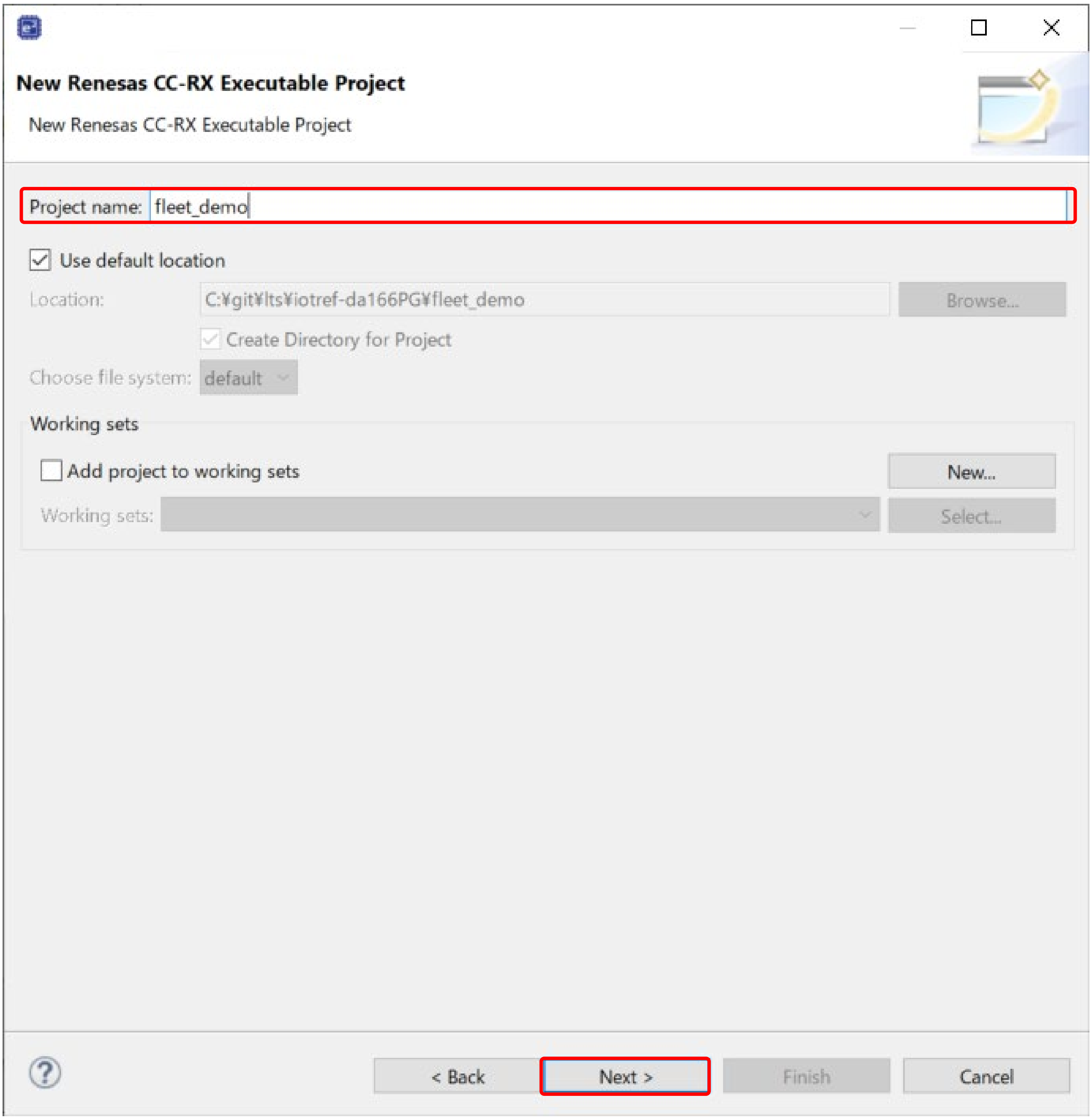
GCC for Renesas RX C/C++ Library Project
A C/C++ Library Project for Renesas RX using the GCC for Renesas RX Toolchain.

Renesas CC-RX C/C++ Executable Project
A C/C++ Project for Renesas RX using the Renesas CC-RX toolchain.

Renesas CC-RX C/C++ Library Project
A C/C++ Library Project for Renesas RX using the Renesas CC-RX toolchain.

Figure 4.24 Dialog Box for Selecting the Project Type

(2) Create a sample project.
Here you will specify the project name. Enter **fleet_demo** as the project name and click the **Next** button. The **Select toolchain, device & debug settings** dialog box opens.



New Renesas CC-RX Executable Project

New Renesas CC-RX Executable Project

Project name: fleet_demo

☑ Use default location

Location: C:¥git¥lts¥iotref-da166PG¥fleet_demo

☑ Create Directory for Project

Choose file system: default

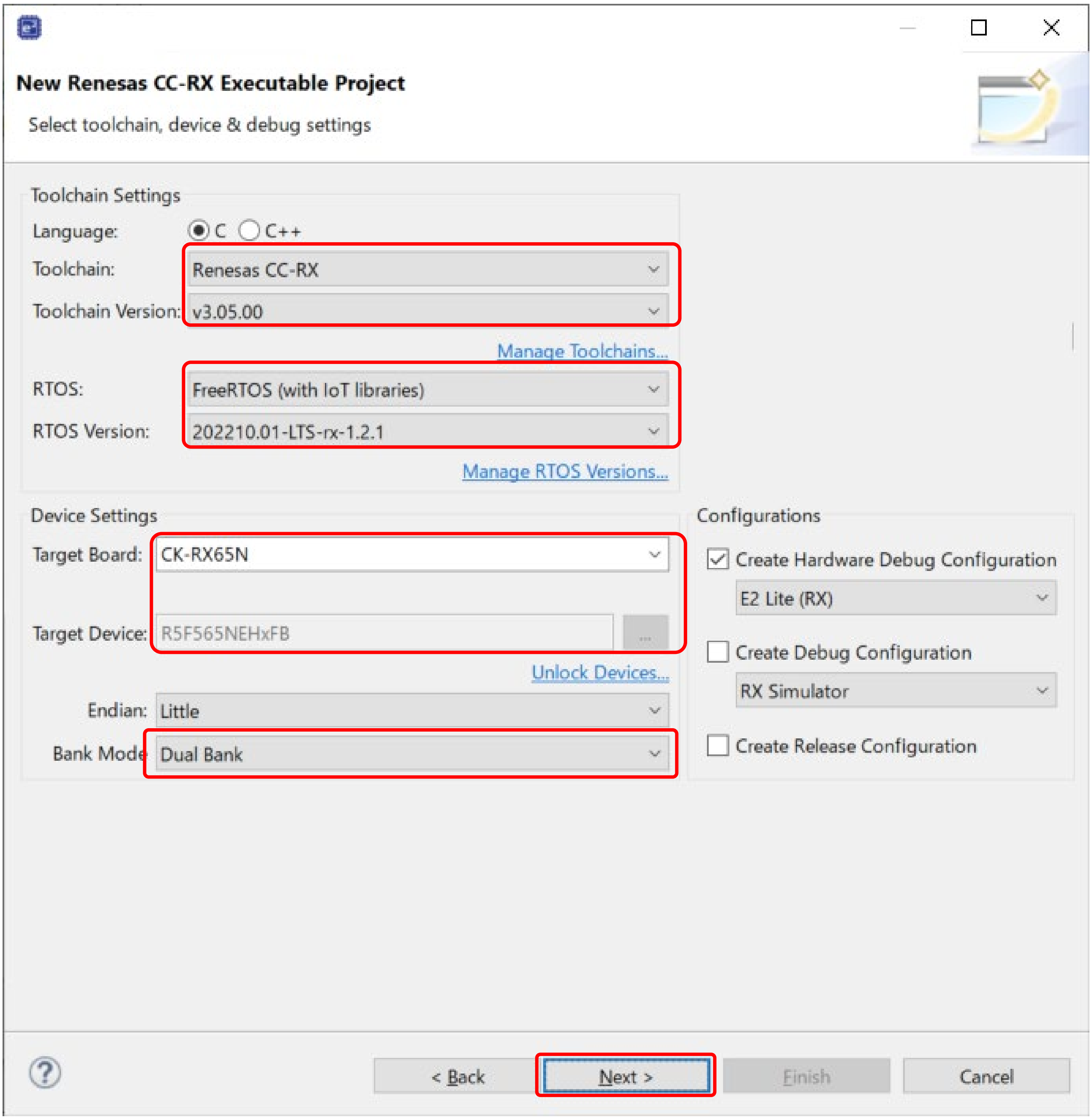
Working sets

□ Add project to working sets

Working sets:

Figure 4.25 Dialog Box for Specifying the Project Name

Configure the toolchain, device, and debug settings to use for the project.
The setting for **Toolchain** is pre-selected based on the type of project. To change the toolchain version, select the version of your choice from the drop-down list next to **Toolchain Version**.
For RTOS, select **Free RTOS (with IoT libraries)**, and for **RTOS Version** select **202210.01-LTS-rx-1.2.1**. If you are running e^2 studio for the first time or if the desired version does not appear in the list, click **Manage RTOS Versions...** to display the **RTOS Module Download** dialog box, check the box next to the desired version, and click the **Download** button to download it.
For **Target Board**, select **CK-RX65N**. (The setting for **Target Device** is selected automatically.)
For **Bank Mode**, select **Dual Bank**.
After all the settings have been configured, click the **Next** button.



New Renesas CC-RX Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: ● C ○ C++

Toolchain: Renesas CC-RX

Toolchain Version: v3.05.00

RTOS: FreeRTOS (with IoT libraries)

RTOS Version: 202210.01-LTS-rx-1.2.1

Device Settings

Target Board: CK-RX65N

Target Device: R5F565NEHxFB

Endian: Little

Bank Mode: Dual Bank

Configurations

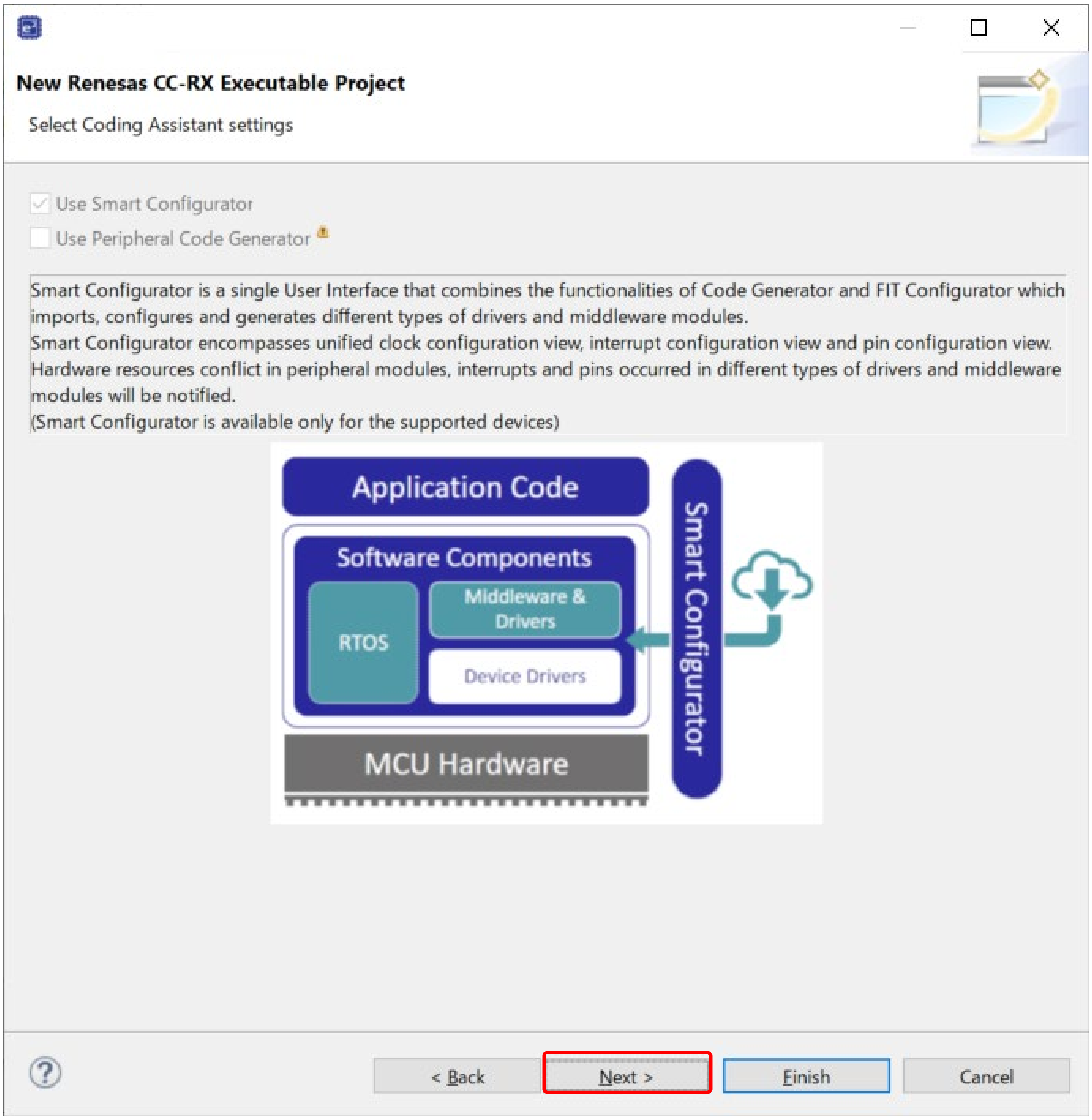
☑ Create Hardware Debug Configuration: E2 Lite (RX)

□ Create Debug Configuration: RX Simulator

□ Create Release Configuration

Figure 4.26 Dialog Box for Specifying Toolchain, Device, and Debug Settings

When the **Select Coding Assistant settings** dialog box appears, click the **Next** button without changing any settings.



New Renesas CC-RX Executable Project

Select Coding Assistant settings

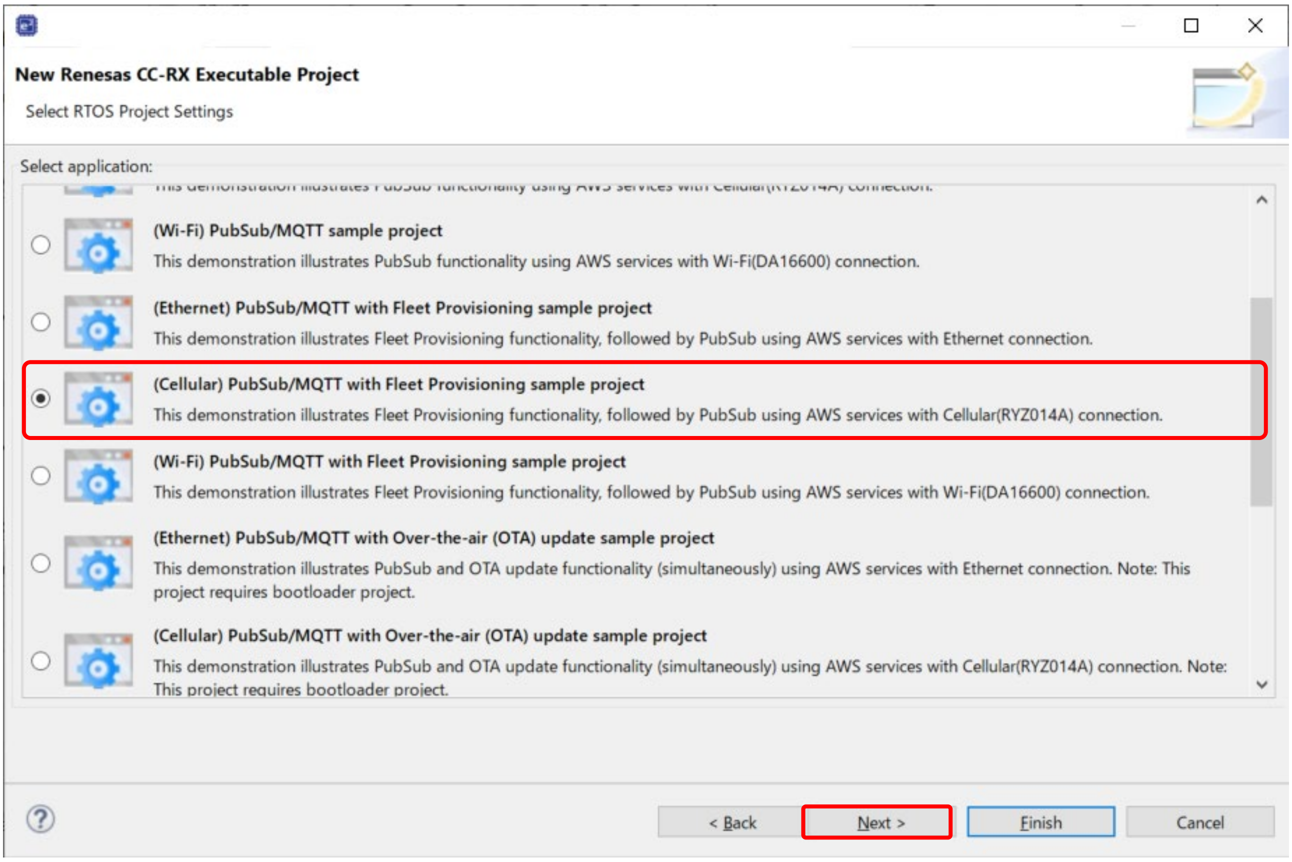
☑ Use Smart Configurator

□ Use Peripheral Code Generator

Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configures and generates different types of drivers and middleware modules.
Smart Configurator encompasses unified clock configuration view, interrupt configuration view and pin configuration view.
Hardware resources conflict in peripheral modules, interrupts and pins occurred in different types of drivers and middleware modules will be notified.
(Smart Configurator is available only for the supported devices)

Figure 4.27 Dialog Box for Selecting the Coding Assistant Tool

A list of sample projects is displayed in the **Select RTOS Project Settings** dialog box.
Use the scroll bar to scroll down the list, select **(Cellular) PubSub/MQTT with Fleet Provisioning sample project**, and click the **Next** button.



New Renesas CC-RX Executable Project

Select RTOS Project Settings

Select application:

This demonstration illustrates PubSub functionality using AWS services with Cellular(RYZ014A) connection.

○ **(Wi-Fi) PubSub/MQTT sample project**
This demonstration illustrates PubSub functionality using AWS services with Wi-Fi(DA16600) connection.

○ **(Ethernet) PubSub/MQTT with Fleet Provisioning sample project**
This demonstration illustrates Fleet Provisioning functionality, followed by PubSub using AWS services with Ethernet connection.

● **(Cellular) PubSub/MQTT with Fleet Provisioning sample project**
This demonstration illustrates Fleet Provisioning functionality, followed by PubSub using AWS services with Cellular(RYZ014A) connection.

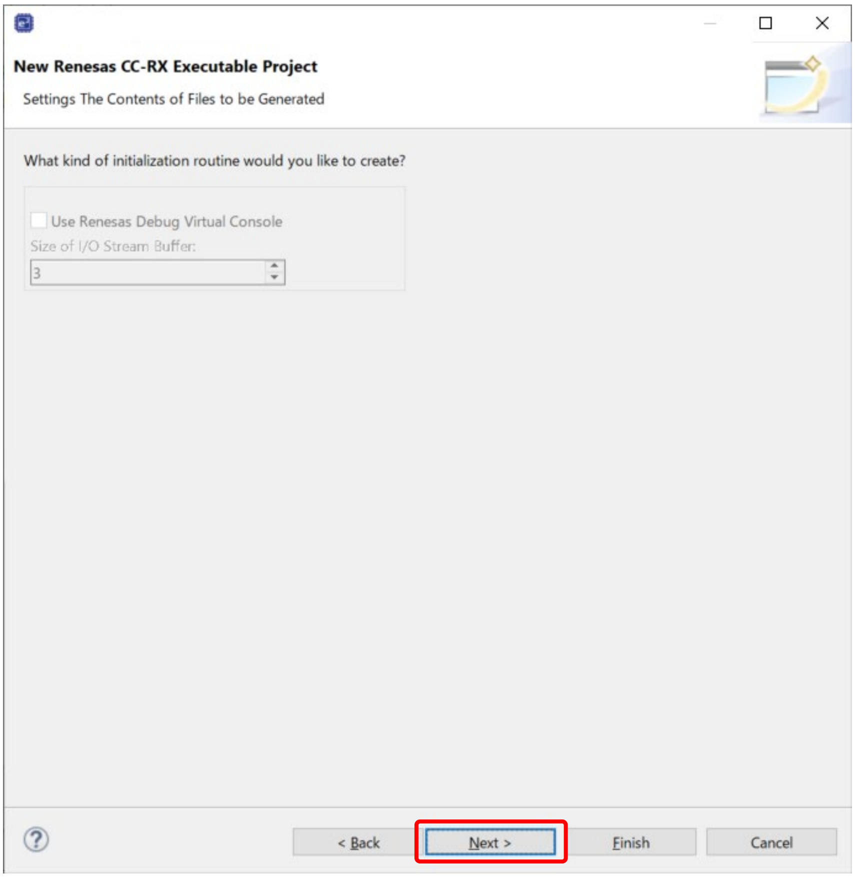
○ **(Wi-Fi) PubSub/MQTT with Fleet Provisioning sample project**
This demonstration illustrates Fleet Provisioning functionality, followed by PubSub using AWS services with Wi-Fi(DA16600) connection.

○ **(Ethernet) PubSub/MQTT with Over-the-air (OTA) update sample project**
This demonstration illustrates PubSub and OTA update functionality (simultaneously) using AWS services with Ethernet connection. Note: This project requires bootloader project.

○ **(Cellular) PubSub/MQTT with Over-the-air (OTA) update sample project**
This demonstration illustrates PubSub and OTA update functionality (simultaneously) using AWS services with Cellular(RYZ014A) connection. Note: This project requires bootloader project.

Figure 4.28 Dialog Box for Selecting RTOS Project Settings

When the **Settings The Contents of Files to be Generated** dialog box appears, click the **Next** button without changing any settings.



New Renesas CC-RX Executable Project

Settings The Contents of Files to be Generated

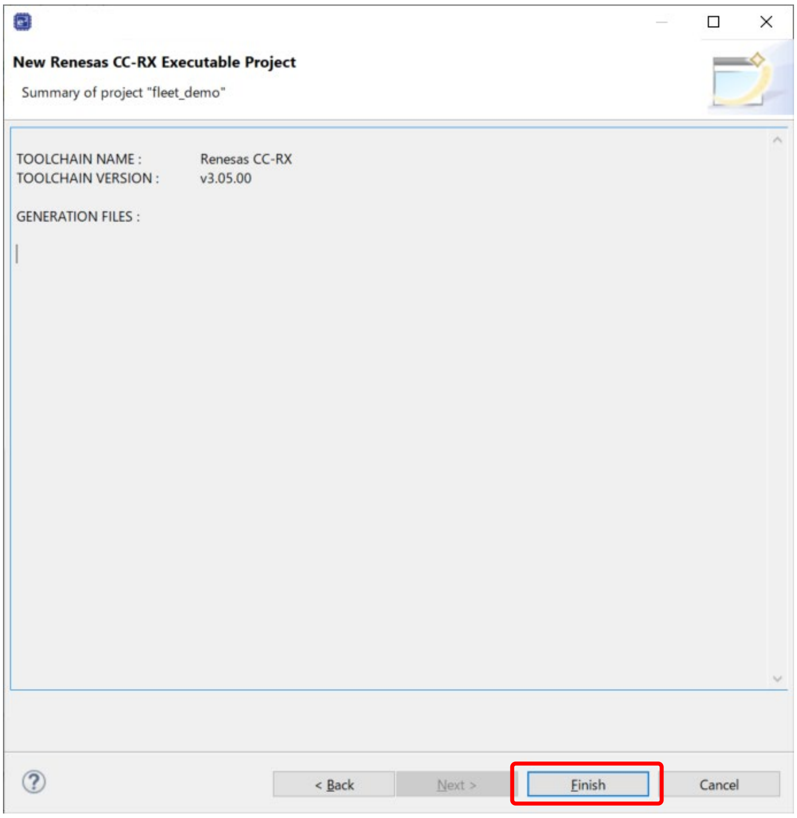
What kind of initialization routine would you like to create?

□ Use Renesas Debug Virtual Console

Size of I/O Stream Buffer: 3

Figure 4.29 Dialog Box for Specifying Details of Files to Be Created

A dialog box appears indicating that the project has been created. If everything appears to be in order, click the **Finish** button.



New Renesas CC-RX Executable Project

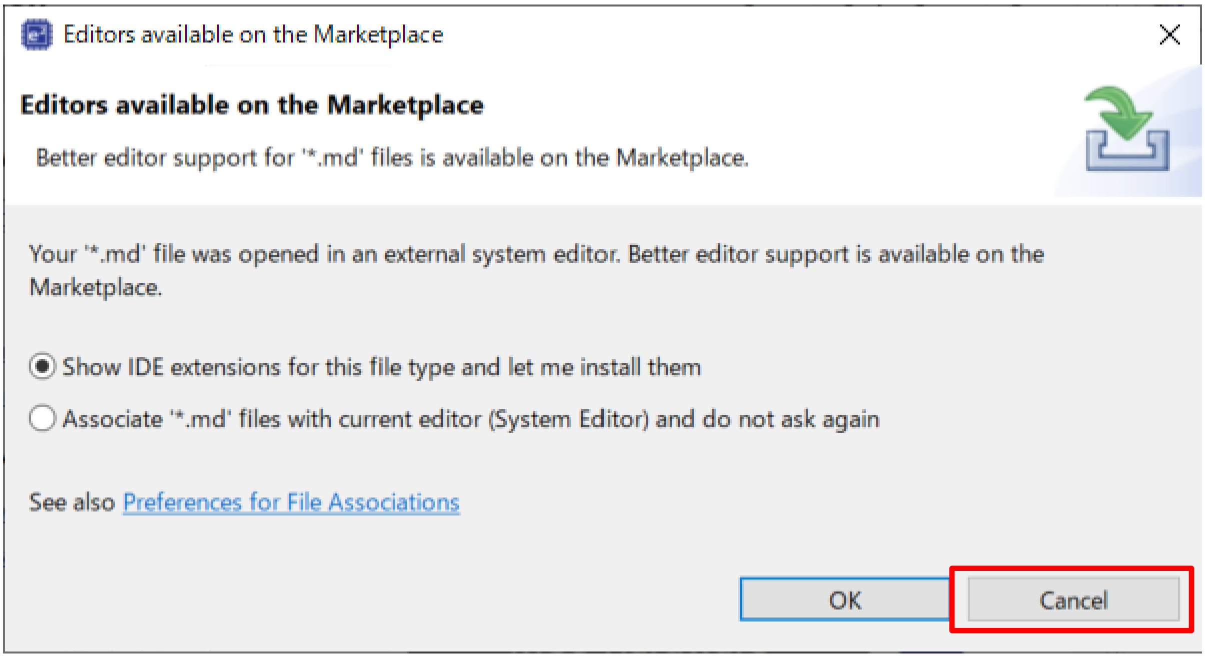
Summary of project "fleet_demo"

TOOLCHAIN NAME : Renesas CC-RX
TOOLCHAIN VERSION : v3.05.00

GENERATION FILES :

Figure 4.30 Dialog Box Indicating that Project Creation is Complete

When the **Editors available on the Marketplace** dialog box appears, click the **Cancel** button to close it.



Editors available on the Marketplace

Editors available on the Marketplace

Better editor support for '*.md' files is available on the Marketplace.

Your '*.md' file was opened in an external system editor. Better editor support is available on the Marketplace.

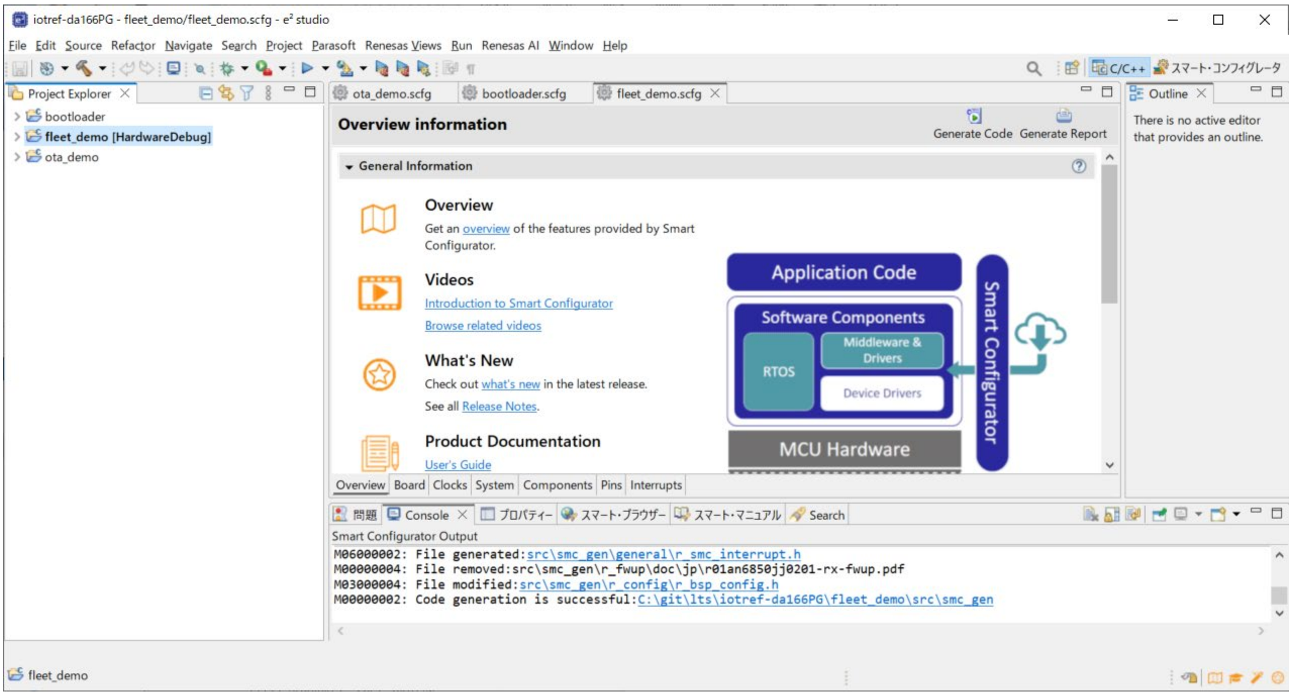
● Show IDE extensions for this file type and let me install them

○ Associate '*.md' files with current editor (System Editor) and do not ask again

See also Preferences for File Associations

Figure 4.31 Dialog Box for Displaying Editors Available on the Marketplace

This completes the procedure for creating the project in e^2 studio.
If Project Explorer is not visible, click the button for the **C/C++** perspective in the upper right corner of the window, then select **Window** > **Show View** > **Project Explorer**.



```
M06000002: File generated:src\smc_gen\general\r_smc_interrupt.h
M00000004: File removed:src\smc_gen\r_fwup\doc\jp\r01an6850jj0201-rx-fwup.pdf
M03000004: File modified:src\smc_gen\r_config\r_bsp_config.h
M00000002: Code generation is successful:C:\git\lts\iotref-da166PG\fleet_demo\src\smc_gen
```

Figure 4.32 Window Displayed after Creation of Fleet Provisioning Sample Project

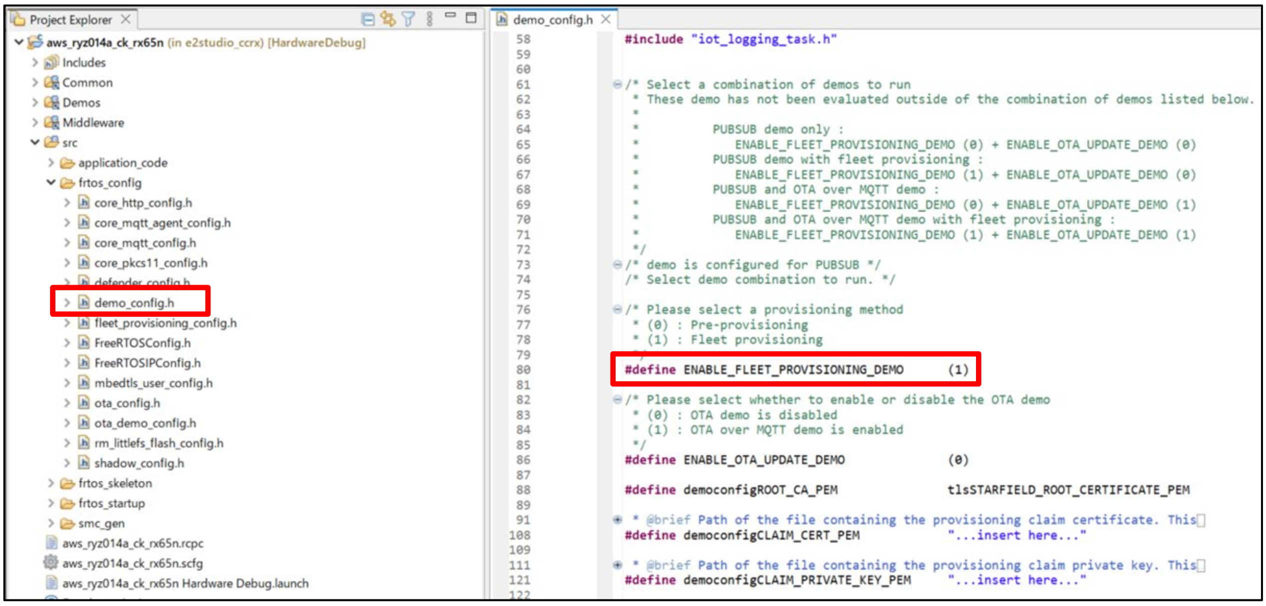
In the discussion that follows, replace the project name **aws_ryz014a_ck_rx65n** with **fleet_demo** as appropriate.

4.5 FreeRTOS Settings

You will need to make a modification to the program in order to run the demo.

4.5.1 Modifying the Configuration File

From the **Project Explorer** panel in e^2 studio, open aws_ryz014a_ck_rx65n/src/frtos_config/demo_config.h and change the value of **ENABLE_FLEET_PROVISIONING_DEMO** to **1**.



```
#include "iot_logging_task.h"


/* Select a combination of demos to run
 * These demo has not been evaluated outside of the combination of demos listed below.
 *
 *          PUBSUB demo only :
 *             ENABLE_FLEET_PROVISIONING_DEMO (0) + ENABLE_OTA_UPDATE_DEMO (0)
 *          PUBSUB demo with fleet provisioning :
 *             ENABLE_FLEET_PROVISIONING_DEMO (1) + ENABLE_OTA_UPDATE_DEMO (0)
 *          PUBSUB and OTA over MQTT demo :
 *             ENABLE_FLEET_PROVISIONING_DEMO (0) + ENABLE_OTA_UPDATE_DEMO (1)
 *          PUBSUB and OTA over MQTT demo with fleet provisioning :
 *             ENABLE_FLEET_PROVISIONING_DEMO (1) + ENABLE_OTA_UPDATE_DEMO (1)
 */
/* demo is configured for PUBSUB */
/* Select demo combination to run. */

/* 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)

#define democonfigROOT_CA_PEM               tlsSTARFIELD_ROOT_CERTIFICATE_PEM

 * @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..."
```

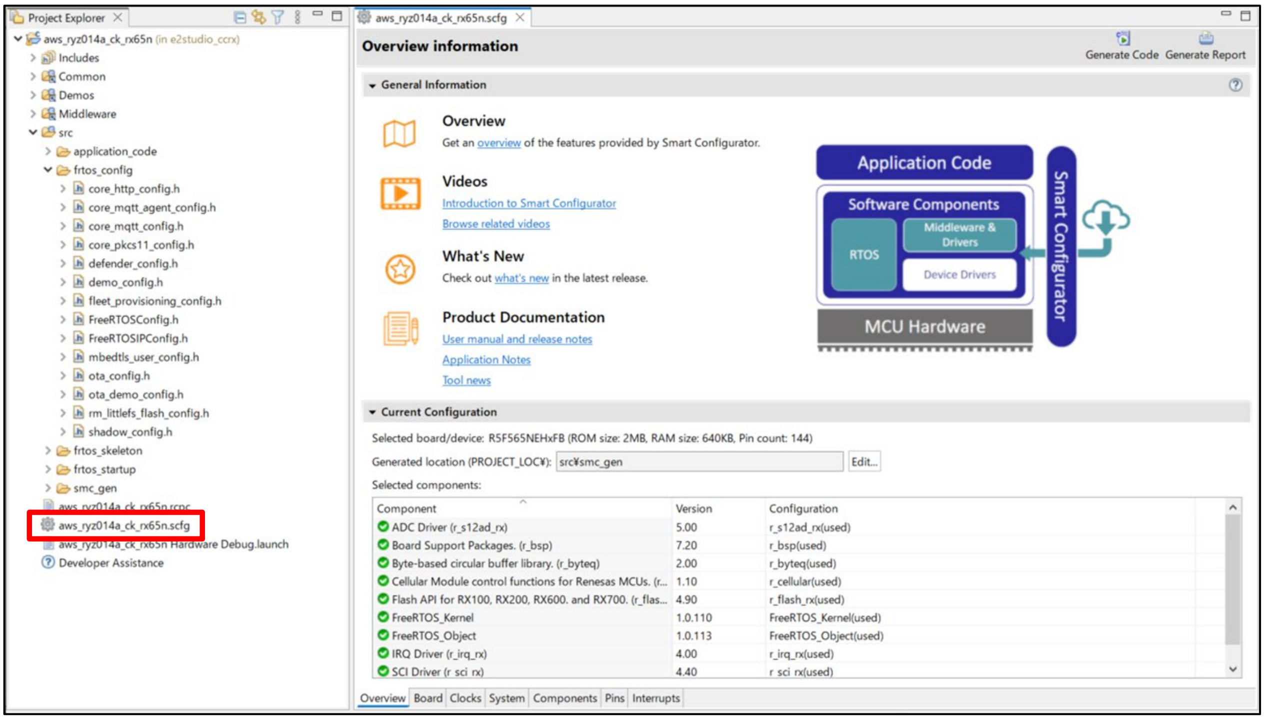
Figure 4.33 Location of Modification in demo_config.h

4.5.2 Cellular information settings

From the **Project Explorer** panel in e2 studio, open aws_ryz014a_ck_rx65n/aws_ryz014a_ck_rx65n.scfg and launch the Smart Configurator. (Figure 4.34)

Select the **Components** tab in the Smart Configurator and select **Middleware** → **Generic** → **r_cellular** from the Components. Set each item of **Access point name**, **Access point login ID**, **Access point password** and **SIM card PIN code** according to the SIM card you are using. If there is no content to enter, leave it blank. (Figure 4.35)

After entering the Cellular information, click the **Generate Code** button to apply the settings to the program.



Get an overview of the features provided by Smart Configurator.

Check out what's new in the latest release.

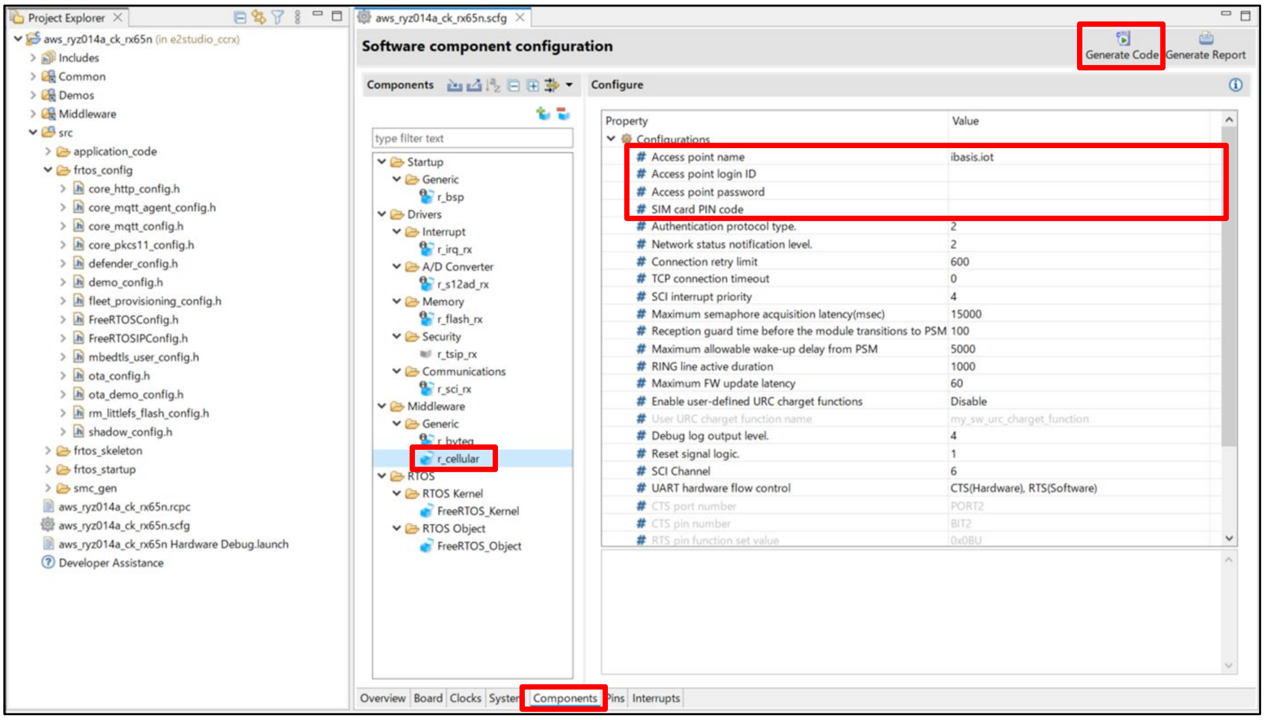
Selected board/device: R5F565NEHxFB (ROM size: 2MB, RAM size: 640KB, Pin count: 144)

Generated location (PROJECT_LOC¥): src¥smc_gen Edit...

Selected components:

Component	Version	Configuration
ADC Driver (r_s12ad_rx)	5.00	r_s12ad_rx(used)
Board Support Packages. (r_bsp)	7.20	r_bsp(used)
Byte-based circular buffer library. (r_byteq)	2.00	r_byteq(used)
Cellular Module control functions for Renesas MCUs. (r...	1.10	r_cellular(used)
Flash API for RX100, RX200, RX600. and RX700. (r_flas...	4.90	r_flash_rx(used)
FreeRTOS_Kernel	1.0.110	FreeRTOS_Kernel(used)
FreeRTOS_Object	1.0.113	FreeRTOS_Object(used)
IRQ Driver (r_irq_rx)	4.00	r_irq_rx(used)
SCI Driver (r_sci_rx)	4.40	r_sci_rx(used)

Figure 4.34 Launch the Smart Configurator



Property	Value
Configurations	
Access point name	ibasis.iot
Access point login ID	
Access point password	
SIM card PIN code	
Authentication protocol type.	2
Network status notification level.	2
Connection retry limit	600
TCP connection timeout	0
SCI interrupt priority	4
Maximum semaphore acquisition latency(msec)	15000
Reception guard time before the module transitions to PSM	100
Maximum allowable wake-up delay from PSM	5000
RING line active duration	1000
Maximum FW update latency	60
Enable user-defined URC charget functions	Disable
User URC charget function name	my_sw_urc_charget_function
Debug log output level.	4
Reset signal logic.	1
SCI Channel	6
UART hardware flow control	CTS(Hardware), RTS(Software)
CTS port number	PORT2
CTS pin number	BIT2
RTS pin function set value	0x0BU

Figure 4.35 Entering Cellular information

Note: About the setting of Wi-Fi network with DA16600 module, please refer to the GitHub「Settings of Wi-Fi network (Only using Wi-Fi)」. And regarding setting of country code and GMT timezone, please refer to the Settings of Country code and GMT timezone (Only using Wi-Fi) as needed.

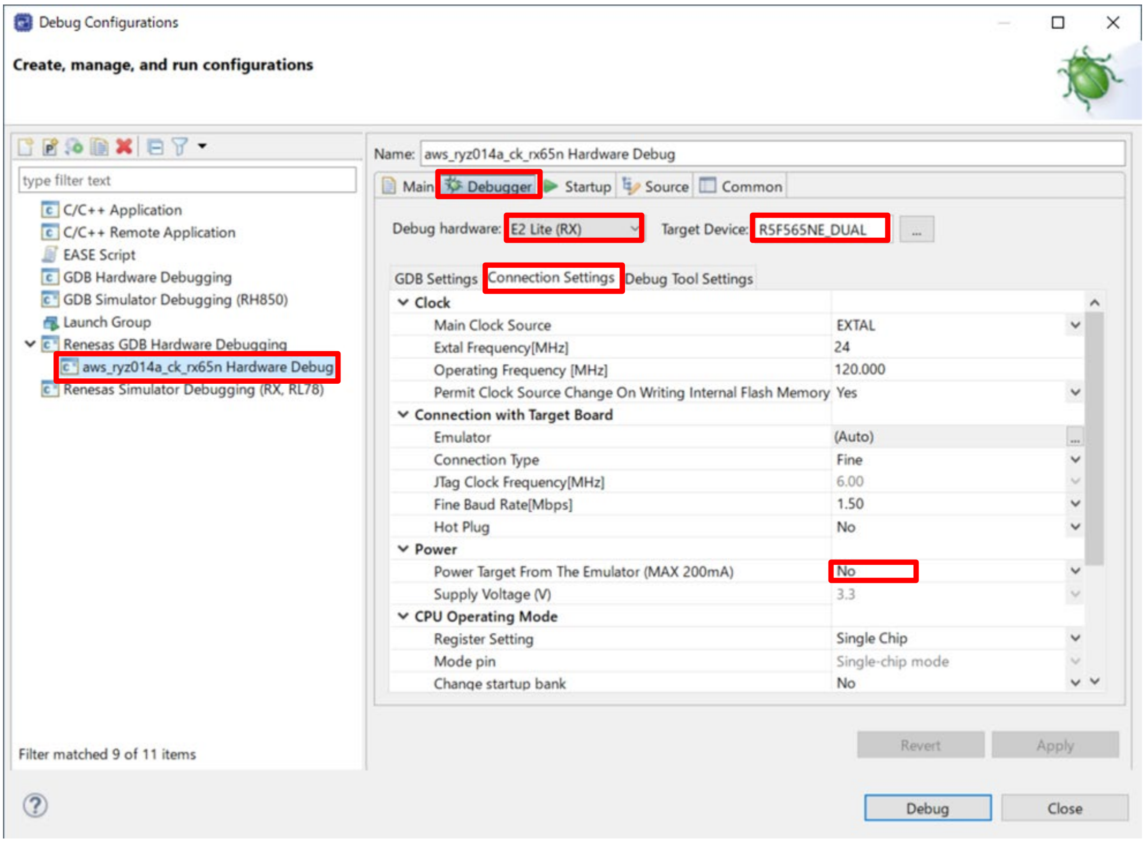
4.6 Building and Running the Program

Build the project, program it to the device, and run the demo.

First, on the **Project Explorer** panel, right-click aws_ryz014a_ck_rx65n and select **Build Project** to build the project.

Next, select **Run** → **Debug Configurations…** from the e^2 studio menu to open the Debug Configurations window. In the list at the left of the Debug Configurations window, select **Renesas GDB Hardware Debugging** → **aws_ryz014a_ck_rx65n Hardware Debug**. Then select the **Debugger** tab followed by the **Connection Settings** tab (indicated by arrows in Figure 4.36).

Check to make sure that the settings of the items enclosed by red frames in Figure 4.34 match those shown, then click the **Debug** button to download to the device the executable data produced by building the project.



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- EASE Script
- GDB Hardware Debugging
- GDB Simulator Debugging (RH850)
- Launch Group
- Renesas GDB Hardware Debugging
  - aws_ryz014a_ck_rx65n Hardware Debug
- Renesas Simulator Debugging (RX, RL78)

Filter matched 9 of 11 items

Name: aws_ryz014a_ck_rx65n Hardware Debug

Main | Debugger | Startup | Source | Common

Debug hardware: E2 Lite (RX)   Target Device: R5F565NE_DUAL

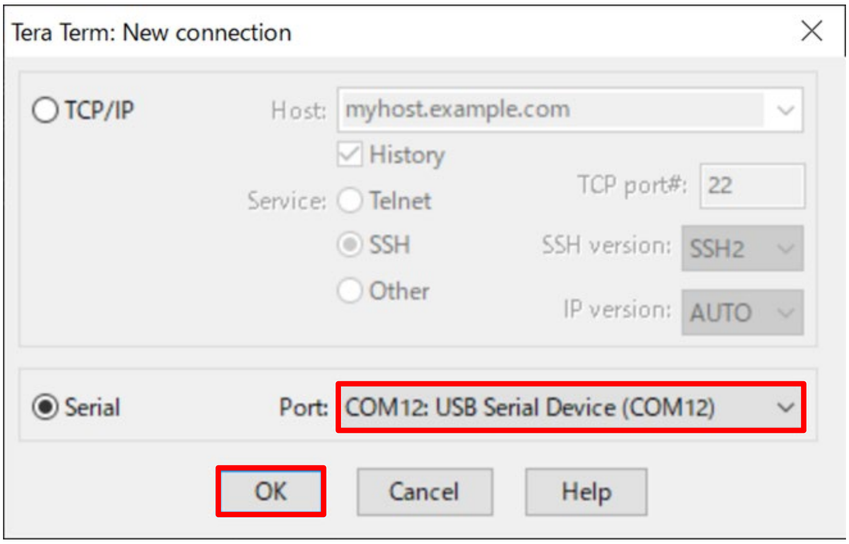
GDB Settings | Connection Settings | Debug Tool Settings

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

Figure 4.36 Debug Configurations

Launch Tera Term in order to enter the claim certificate, claim private key, endpoint, and provisioning template name.

After Tera Term starts, select **Serial** and **USB Serial Device**, then click the **OK** button.



Tera Term: New connection

○ TCP/IP Host: myhost.example.com

☑ History

Service: ○ Telnet TCP port#: 22

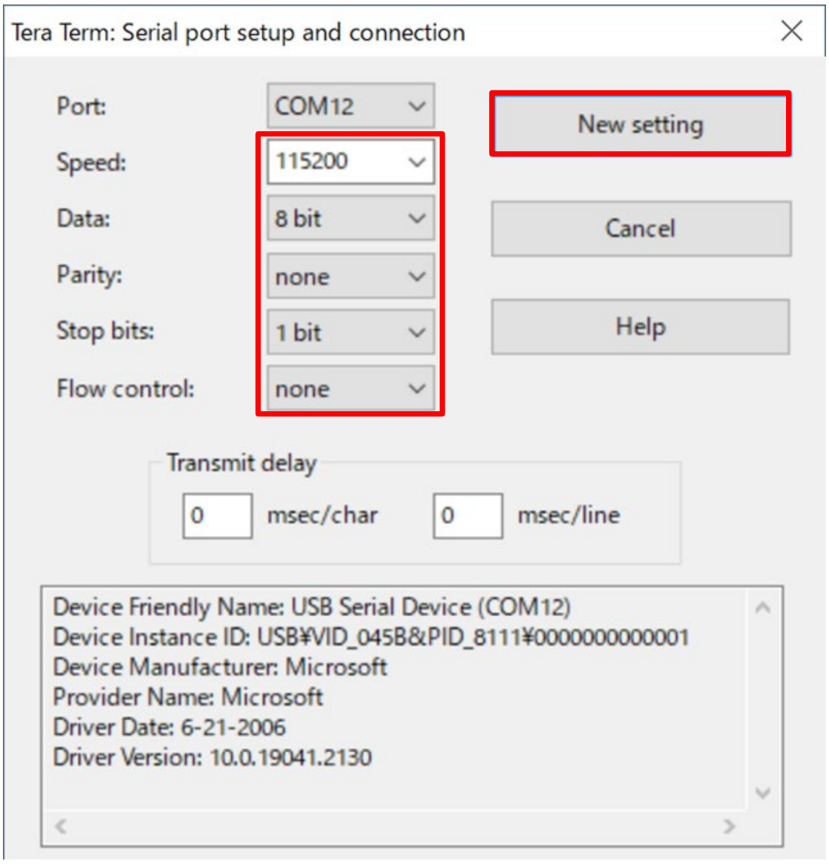
● SSH SSH version: SSH2

○ Other IP version: AUTO

● Serial Port: COM12: USB Serial Device (COM12)

Figure 4.37 Initial Window when Tera Term Starts

Select **Setup** → **Serial port…** from the menu, configure the serial port setting items enclosed by red frames as shown, and then click the **New setting** button.



Tera Term: Serial port setup and connection

Port: COM12

Speed: 115200

Data: 8 bit

Parity: none

Stop bits: 1 bit

Flow control: none

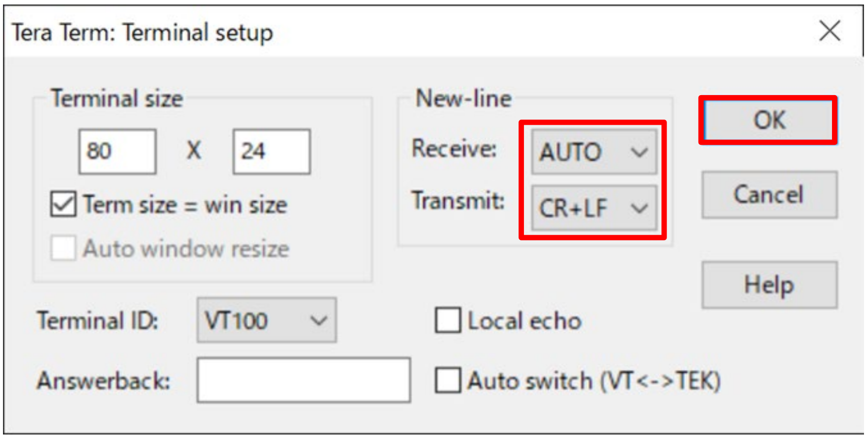
Transmit delay

0 msec/char 0 msec/line

Device Friendly Name: USB Serial Device (COM12)
Device Instance ID: USB¥VID_045B&PID_8111¥000000000001
Device Manufacturer: Microsoft
Provider Name: Microsoft
Driver Date: 6-21-2006
Driver Version: 10.0.19041.2130

Figure 4.38 Serial Port Setup

Select **Setup** → **Terminal…** from the menu, set **Receive:** to **AUTO** and **Transmit:** to **CR+LF** as shown in the red frames, and then click the **OK** button.



Tera Term: Terminal setup

Terminal size

80 X 24

☑ Term size = win size

☐ Auto window resize

New-line

Receive: AUTO

Transmit: CR+LF

Terminal ID: VT100

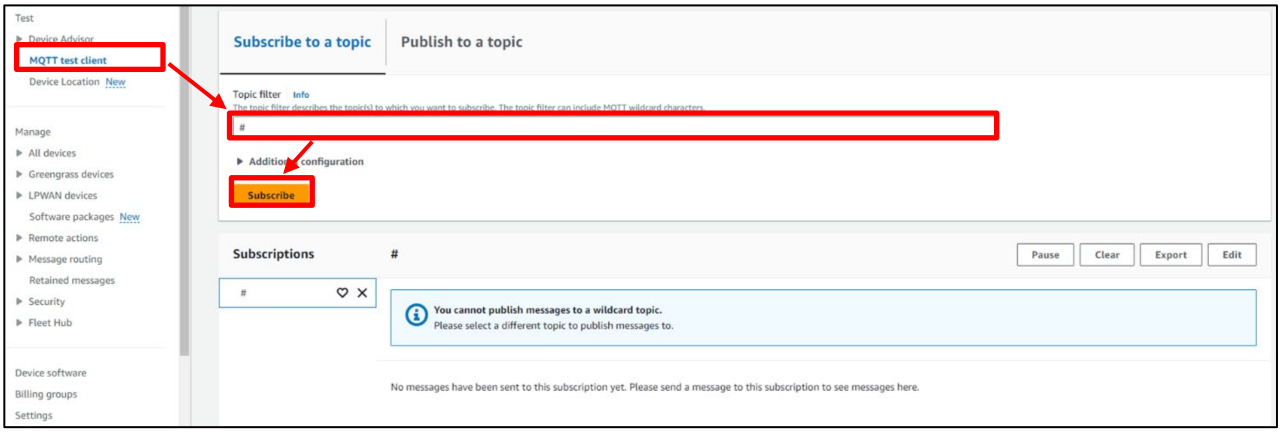
Answerback:

☐ Local echo

☐ Auto switch (VT<->TEK)

Figure 4.39 Terminal Setup

From the AWS IoT console, select **MQTT test client**, enter **#** under **Topic filter**, and click the **Subscribe** button.



Topic filter Info

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

#

Subscriptions #

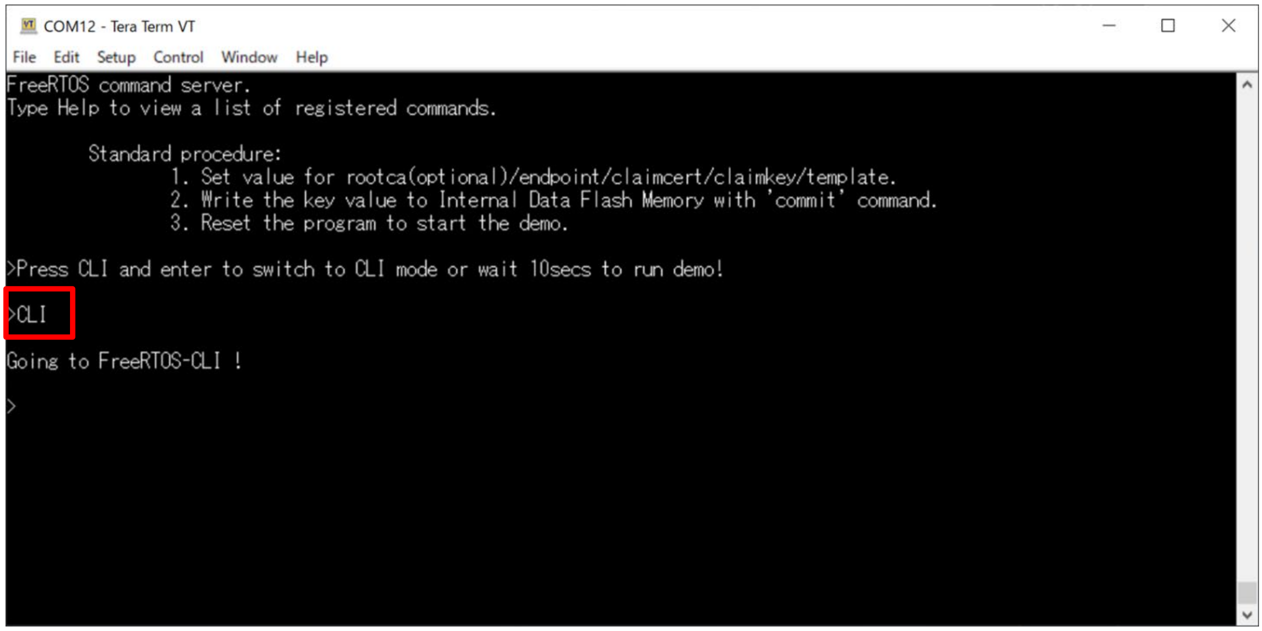
#

You cannot publish messages to a wildcard topic.
Please select a different topic to publish messages to.

No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

Figure 4.40 MQTT Test Client Settings

In e^2 studio, press **Resume** (F8) to display the text output shown below in Tera Term. Within 10 seconds, type **CLI** in Tera Term and press the Enter key.

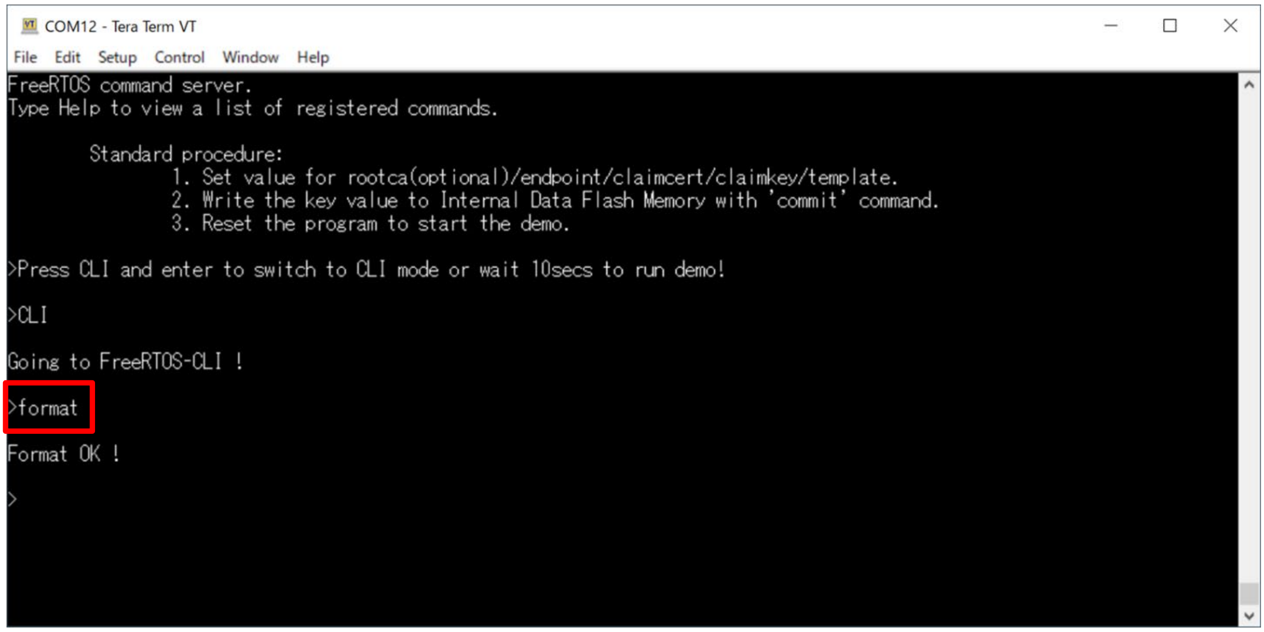


```
FreeRTOS command server.
Type Help to view a list of registered commands.

       Standard procedure:
              1. Set value for rootca(optional)/endpoint/claimcert/claimkey/template.
              2. Write the key value to Internal Data Flash Memory with 'commit' command.
              3. Reset the program to start the demo.

>Press CLI and enter to switch to CLI mode or wait 10secs to run demo!

>CLI

Going to FreeRTOS-CLI !

>
```

Figure 4.41 Entering Information Using CLI (1)

It is possible that information may have been stored already if the demo was run previously, so type **format** in Tera Term and press the Enter key.

This causes all stored information to be erased.

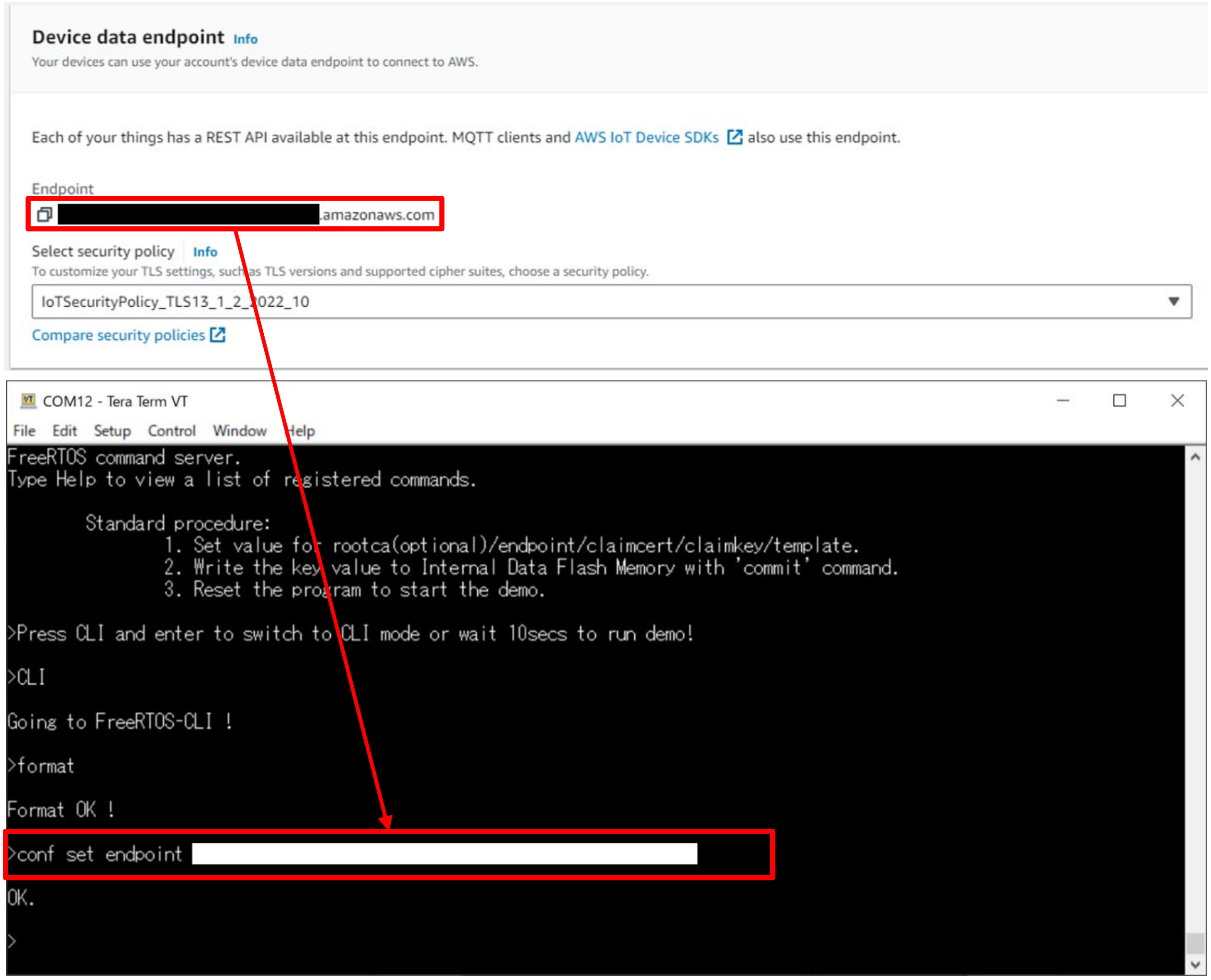


```
FreeRTOS command server.
Type Help to view a list of registered commands.

       Standard procedure:
              1. Set value for rootca(optional)/endpoint/claimcert/claimkey/template.
              2. Write the key value to Internal Data Flash Memory with 'commit' command.
              3. Reset the program to start the demo.

>Press CLI and enter to switch to CLI mode or wait 10secs to run demo!

>CLI

Going to FreeRTOS-CLI !

>format

Format OK !

>
```

Figure 4.42 Entering Information Using CLI (2)

To enter the endpoint, type **conf set endpoint <endpoint>** in Tera Term and press the Enter key.

For **<endpoint>**, enter the value in the format **xxxxxxxxxx.amazonaws.com** that is displayed for **Endpoint** when you select **Settings** → **Device data endpoint** on the AWS IoT console.



Device data endpoint Info

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

[redacted].amazonaws.com

Select security policy Info

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

IoTSecurityPolicy_TLS13_1_2_2022_10

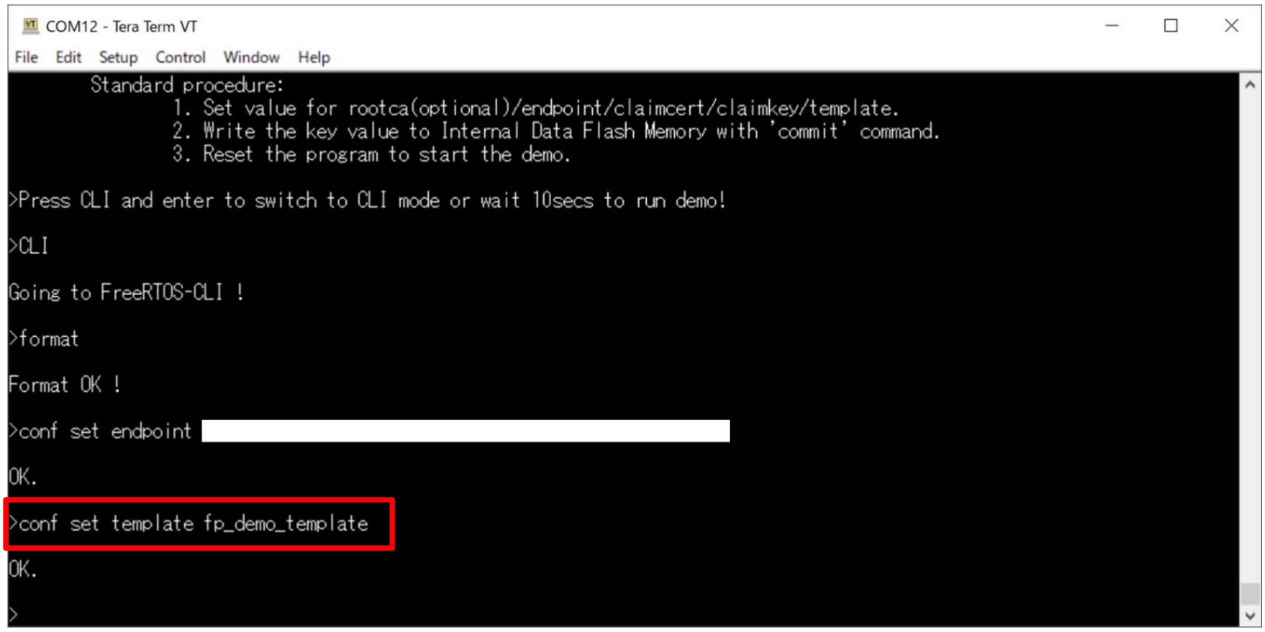
Compare security policies

```
FreeRTOS command server.
Type Help to view a list of registered commands.

        Standard procedure:
                1. Set value for rootca(optional)/endpoint/claimcert/claimkey/template.
                2. Write the key value to Internal Data Flash Memory with 'commit' command.
                3. Reset the program to start the demo.

>Press CLI and enter to switch to CLI mode or wait 10secs to run demo!

>CLI

Going to FreeRTOS-CLI !

>format

Format OK !

>conf set endpoint [redacted]

OK.

>
```

Figure 4.43 Entering Information Using CLI (3)

To enter the provisioning template name, type **conf set template <template_name>** in Tera Term and press the Enter key.

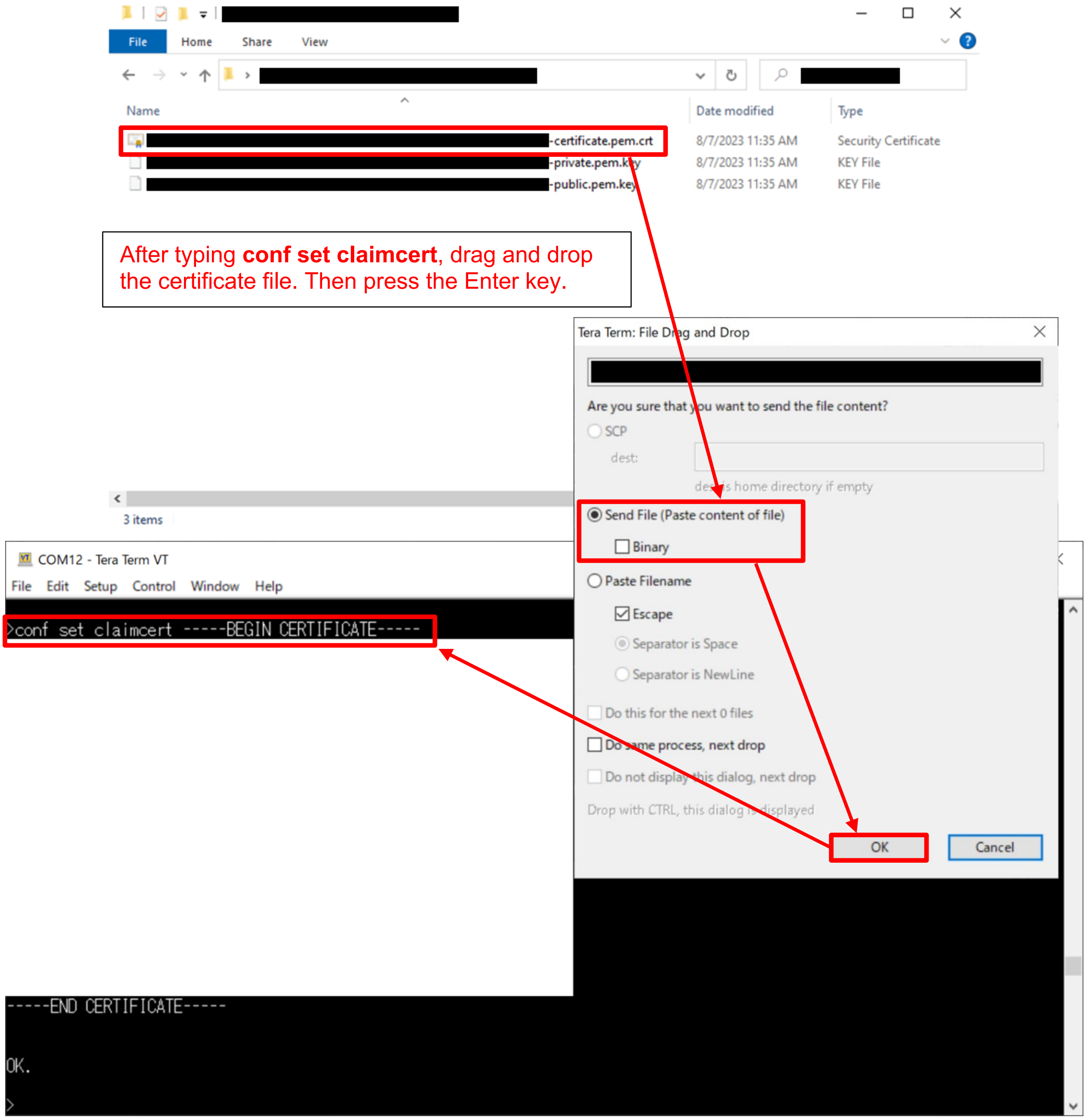
For **<template_name>**, enter the name of the provisioning template created in 4.3.3.



```
       Standard procedure:
              1. Set value for rootca(optional)/endpoint/claimcert/claimkey/template.
              2. Write the key value to Internal Data Flash Memory with 'commit' command.
              3. Reset the program to start the demo.

>Press CLI and enter to switch to CLI mode or wait 10secs to run demo!

>CLI

Going to FreeRTOS-CLI !

>format

Format OK !

>conf set endpoint [redacted]

OK.

>conf set template fp_demo_template

OK.

>
```

Figure 4.44 Entering Information Using CLI (4)

To enter the provisioning claim certificate, type **conf set claimcert** in Tera Term. Next, drag and drop the provisioning claim certificate file (**xxxx-certificate.pem.crt**) created in 4.3.2 onto the Tera Term window (**Send File**). Finally, press the Enter key in Tera Term.



[redacted]

[redacted]

[redacted]

Name	Date modified	Type
[redacted]-certificate.pem.crt	8/7/2023 11:35 AM	Security Certificate
[redacted]-private.pem.key	8/7/2023 11:35 AM	KEY File
[redacted]-public.pem.key	8/7/2023 11:35 AM	KEY File

3 items

After typing **conf set claimcert**, drag and drop the certificate file. Then press the Enter key.

Tera Term: File Drag and Drop

[redacted]

Are you sure that you want to send the file content?

○ SCP

dest:

dest is home directory if empty

● Send File (Paste content of file)

□ Binary

○ Paste Filename

☑ Escape

○ Separator is Space

○ Separator is NewLine

□ Do this for the next 0 files

□ Do same process, next drop

□ Do not display this dialog, next drop

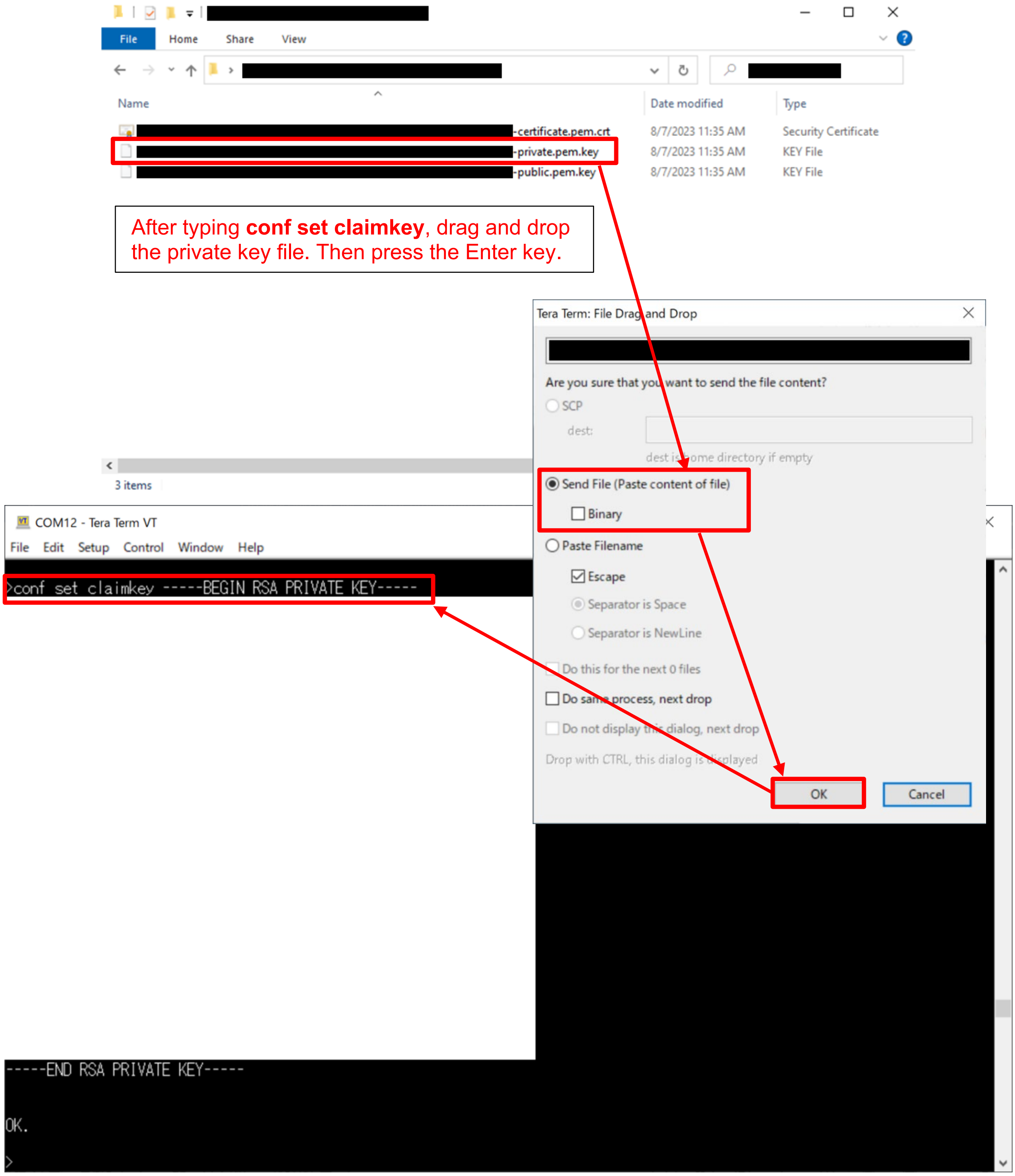
Drop with CTRL, this dialog is displayed

COM12 - Tera Term VT

```
>conf set claimcert -----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

OK.
>
```

Figure 4.45 Entering Information Using CLI (5)

To enter the provisioning claim private key, type **conf set claimkey** in Tera Term. Next, drag and drop the provisioning claim private key file (**xxxx-private.pem.key**) created in 4.3.2 onto the Tera Term window (**Send File**). Finally, press the Enter key in Tera Term.



Name	Date modified	Type
[redacted]-certificate.pem.crt	8/7/2023 11:35 AM	Security Certificate
[redacted]-private.pem.key	8/7/2023 11:35 AM	KEY File
[redacted]-public.pem.key	8/7/2023 11:35 AM	KEY File

After typing **conf set claimkey**, drag and drop the private key file. Then press the Enter key.

[redacted]

Are you sure that you want to send the file content?

○ SCP

dest:

dest is home directory if empty

● Send File (Paste content of file)

□ Binary

○ Paste Filename

☑ Escape

○ Separator is Space

○ Separator is NewLine

□ Do this for the next 0 files

□ Do same process, next drop

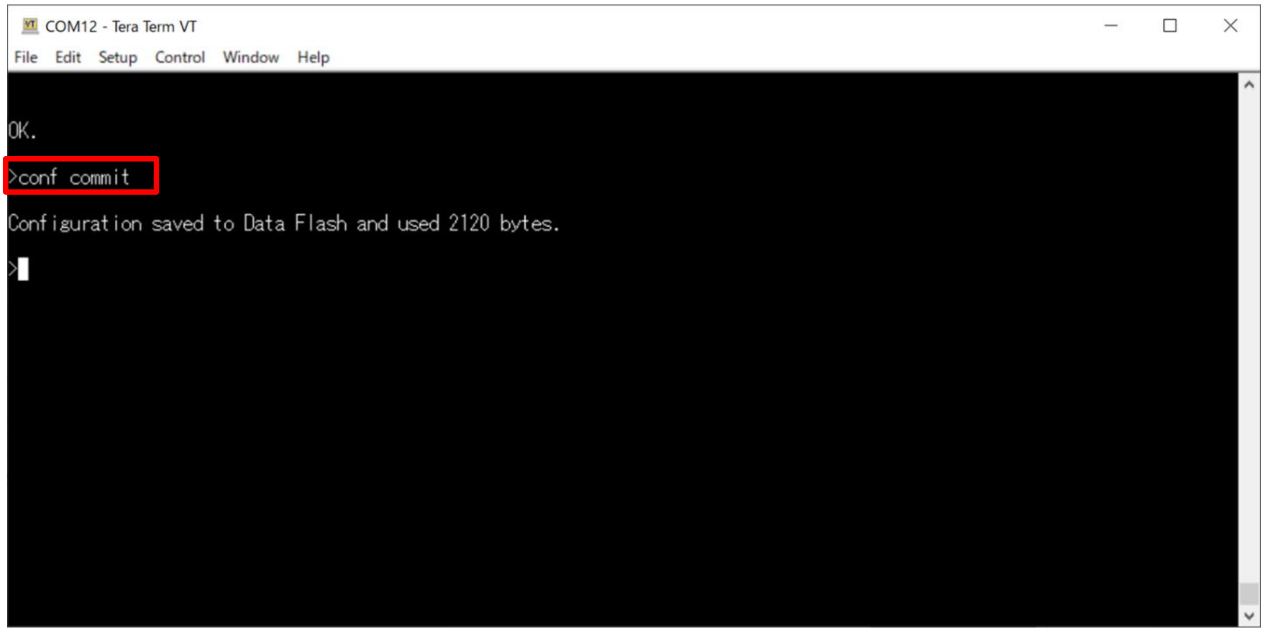
□ Do not display this dialog, next drop

Drop with CTRL, this dialog is displayed

```
>conf set claimkey -----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----

OK.
>
```

Figure 4.46 Entering Information Using CLI (6)

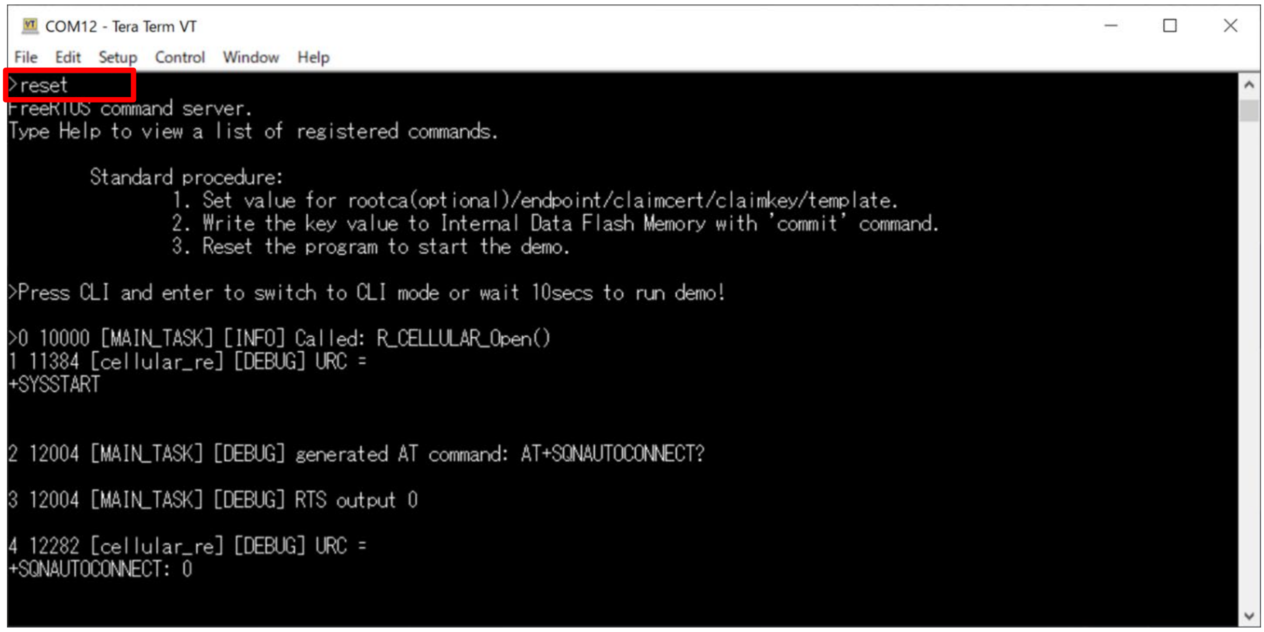
To store the information entered up to this point in the data flash memory, type **conf commit** in Tera Term and press the Enter key.



```
OK.

>conf commit

Configuration saved to Data Flash and used 2120 bytes.

>
```

Figure 4.47 Entering Information Using CLI (7)

To start the demo, type **reset** in Tera Term and press the Enter key. If nothing is entered in Tera Term for 10 seconds after the reset, the demo starts.



```
>reset
FreeRTOS command server.
Type Help to view a list of registered commands.

        Standard procedure:
                1. Set value for rootca(optional)/endpoint/claimcert/claimkey/template.
                2. Write the key value to Internal Data Flash Memory with 'commit' command.
                3. Reset the program to start the demo.

>Press CLI and enter to switch to CLI mode or wait 10secs to run demo!

>0 10000 [MAIN_TASK] [INFO] Called: R_CELLULAR_Open()
1 11384 [cellular_re] [DEBUG] URC =
+SYSSTART


2 12004 [MAIN_TASK] [DEBUG] generated AT command: AT+SQNAUTOCONNECT?

3 12004 [MAIN_TASK] [DEBUG] RTS output 0

4 12282 [cellular_re] [DEBUG] URC =
+SQNAUTOCONNECT: 0
```

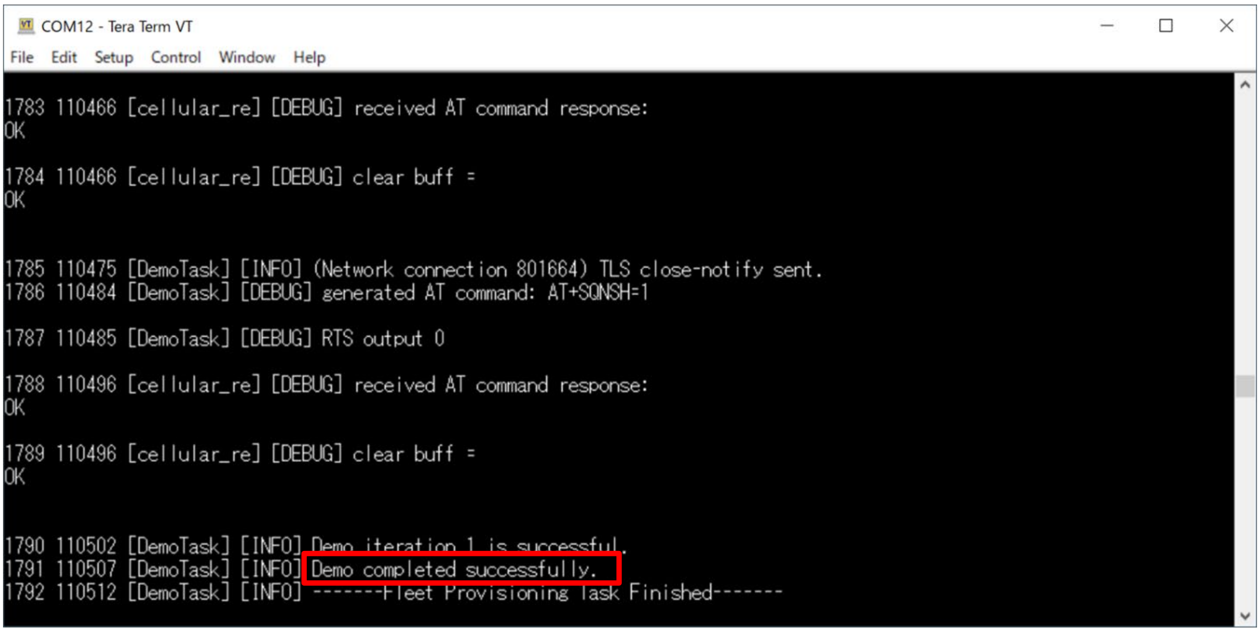
Figure 4.48 Entering Information Using CLI (8)

4.7 Confirming the Results of Running the Demo

Figure 4.49 shows a log file produced by running the fleet provisioning demo.

(The log is displayed in Tera Term.)

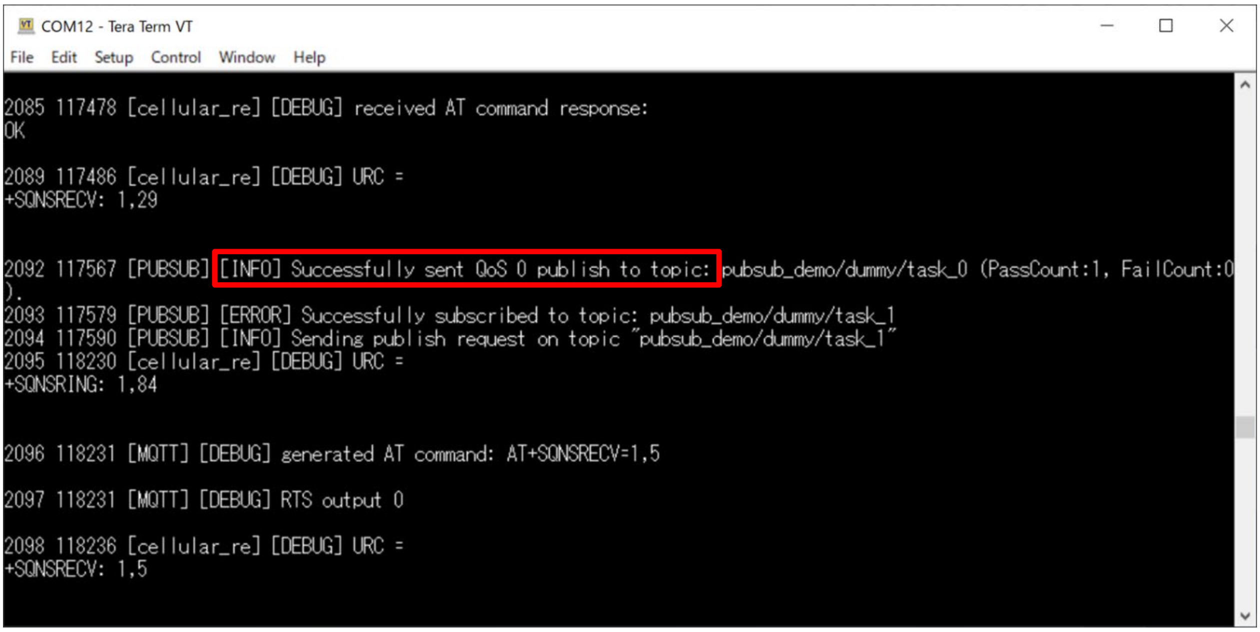
If the text string “Demo completed successfully.” appears at the end of the log, the fleet provisioning demo completed successfully. Successful completion of the demo means that a new thing has been registered on AWS IoT Core and an individual device certificate assigned to it.



```
1783 110466 [cellular_re] [DEBUG] received AT command response:
OK

1784 110466 [cellular_re] [DEBUG] clear buff =
OK


1785 110475 [DemoTask] [INFO] (Network connection 801664) TLS close-notify sent.
1786 110484 [DemoTask] [DEBUG] generated AT command: AT+SQNSH=1

1787 110485 [DemoTask] [DEBUG] RTS output 0

1788 110496 [cellular_re] [DEBUG] received AT command response:
OK

1789 110496 [cellular_re] [DEBUG] clear buff =
OK


1790 110502 [DemoTask] [INFO] Demo iteration 1 is successful.
1791 110507 [DemoTask] [INFO] Demo completed successfully.
1792 110512 [DemoTask] [INFO] -------Fleet Provisioning Task Finished-------
```

Figure 4.49 Log Produced when Fleet Provisioning Demo Completes Successfully

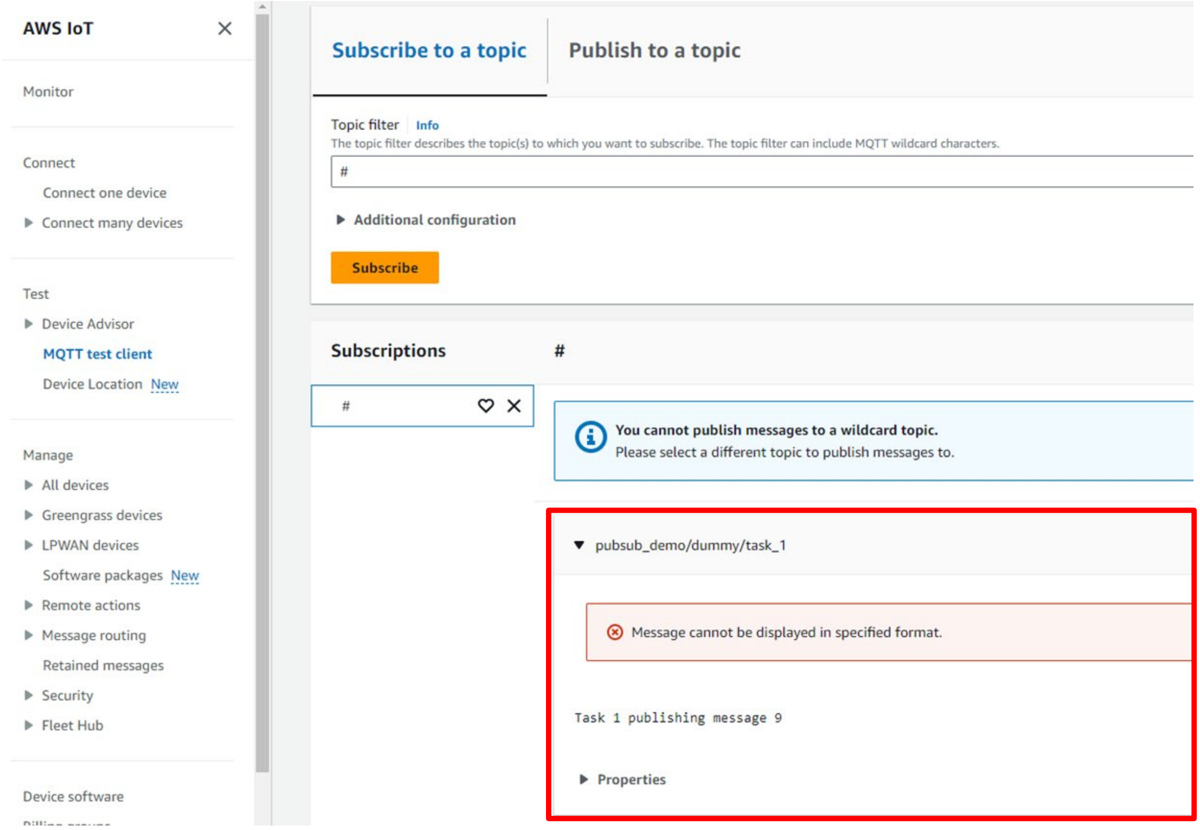
After running the fleet provisioning demo, you can use the individual device certificate and private key obtained from AWS to run the PubSub demo. Check to confirm that the text string “Successfully sent QoS 0 publish to topic:” appears in the log as shown in Figure 4.50.



```
2085 117478 [cellular_re] [DEBUG] received AT command response:
OK

2089 117486 [cellular_re] [DEBUG] URC =
+SQNSRECV: 1,29


2092 117567 [PUBSUB] [INFO] Successfully sent QoS 0 publish to topic: pubsub_demo/dummy/task_0 (PassCount:1, FailCount:0
).
2093 117579 [PUBSUB] [ERROR] Successfully subscribed to topic: pubsub_demo/dummy/task_1
2094 117590 [PUBSUB] [INFO] Sending publish request on topic "pubsub_demo/dummy/task_1"
2095 118230 [cellular_re] [DEBUG] URC =
+SQNSRING: 1,84


2096 118231 [MQTT] [DEBUG] generated AT command: AT+SQNSRECV=1,5

2097 118231 [MQTT] [DEBUG] RTS output 0

2098 118236 [cellular_re] [DEBUG] URC =
+SQNSRECV: 1,5
```

Figure 4.50 Log Produced when PubSub Demo Completes Successfully

You can also check MQTT messages sent to AWS from CK-RX65N by selecting **MQTT test client** from the AWS IoT console.



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

Subscriptions #

You cannot publish messages to a wildcard topic.
Please select a different topic to publish messages to.

pubsub_demo/dummy/task_1

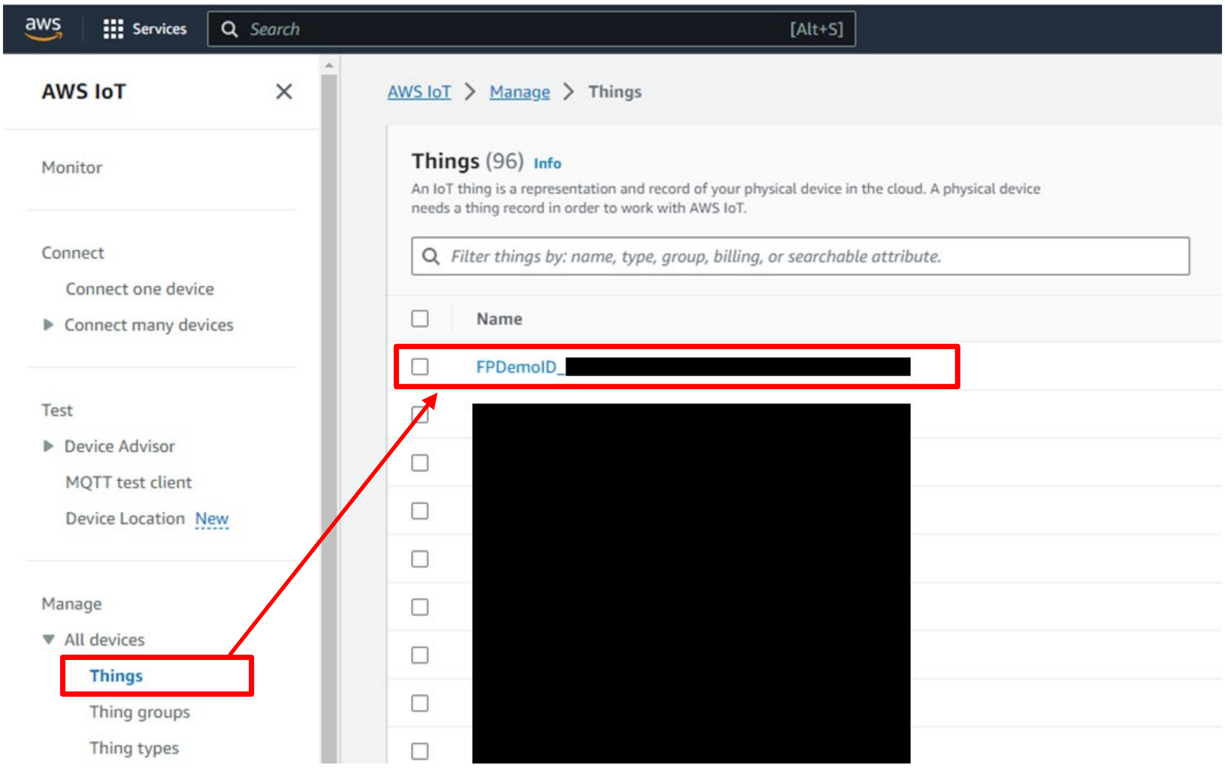
Message cannot be displayed in specified format.

```
Task 1 publishing message 9
```

Figure 4.51 MQTT Test Client after Successful Completion of PubSub Demo

You can check on the thing registered by the fleet provisioning demo from the AWS IoT console.

Under **All devices**, select **Things**. The thing (shown as "FPDemoID_xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxxx" in Figure 4.52) will have been registered if the demo completed successfully.



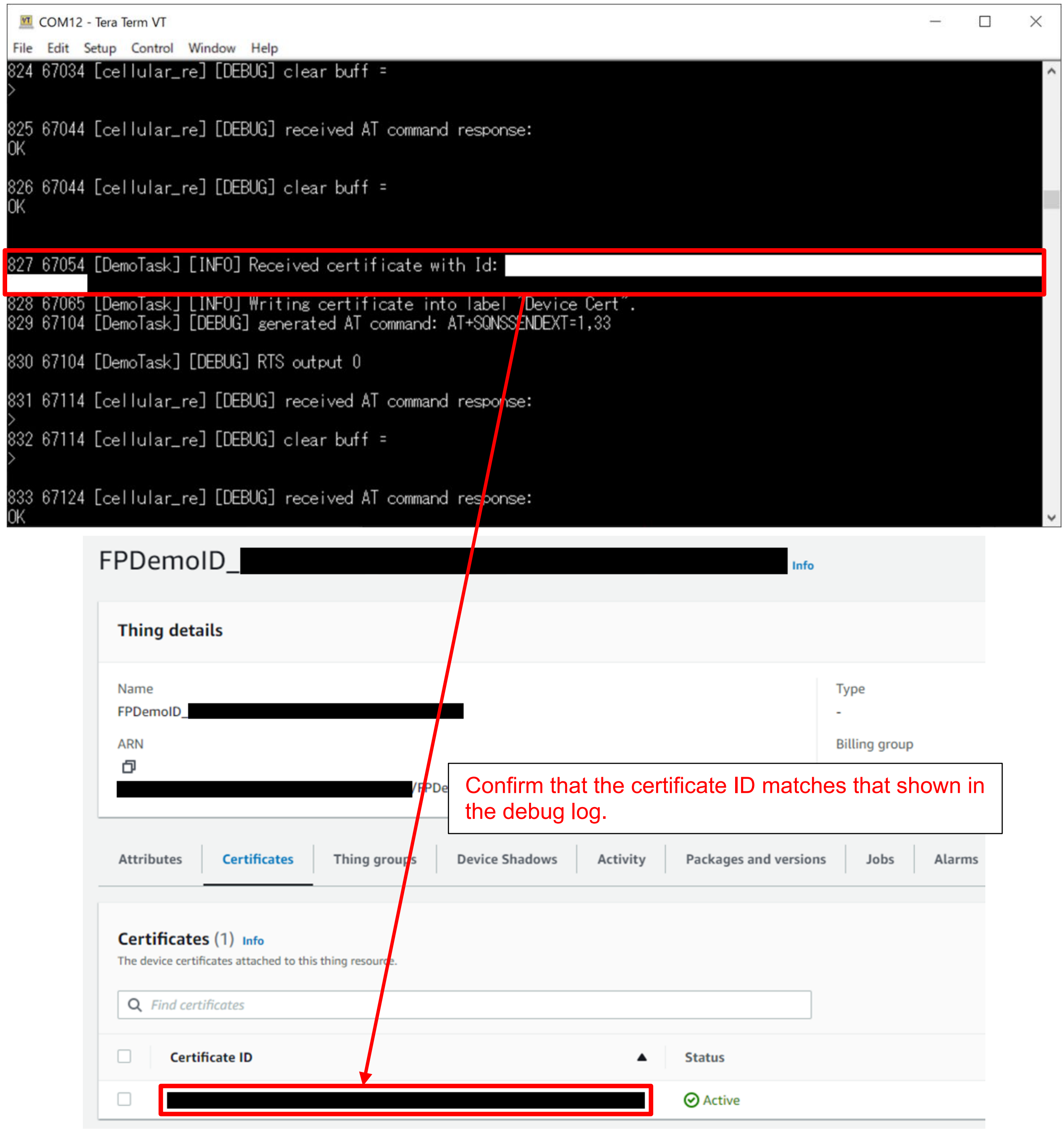
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

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

□	Name
□	FPDemoID_[redacted]
□	[redacted]
□	
□	
□	
□	
□	
□	
□	

Figure 4.52 Confirming the Results of Running the Demo (1)

By checking the registered things, you can confirm that the individual device certificate generated and assigned by fleet provisioning (**Certificate ID** in Figure 4.53) has been attached and activated.



```
824 67034 [cellular_re] [DEBUG] clear buff =
>

825 67044 [cellular_re] [DEBUG] received AT command response:
OK

826 67044 [cellular_re] [DEBUG] clear buff =
OK

827 67054 [DemoTask] [INFO] Received certificate with Id: [redacted]
828 67065 [DemoTask] [INFO] Writing certificate into label "Device Cert".
829 67104 [DemoTask] [DEBUG] generated AT command: AT+SQNSSENDEXT=1,33

830 67104 [DemoTask] [DEBUG] RTS output 0

831 67114 [cellular_re] [DEBUG] received AT command response:
>
832 67114 [cellular_re] [DEBUG] clear buff =
>

833 67124 [cellular_re] [DEBUG] received AT command response:
OK
```

FPDemoID_[redacted] Info

Thing details

Name
FPDemoID_[redacted]

ARN
[redacted]/FPDe

Type
-

Billing group

Certificates (1) Info
The device certificates attached to this thing resource.

Find certificates

□	Certificate ID ▲	Status
□	[redacted]	✓ Active

Confirm that the certificate ID matches that shown in the debug log.

Figure 4.53 Confirming the Results of Running the Demo (2)

5. Conclusion

As mentioned earlier, there are multiple provisioning methods, and there are also various ways to enhance security. Nowadays, it is essential to select and deploy an appropriate provisioning method that matches the actual application in the target market, the scale of the system (number of devices), and the required level of security.

However, it is not a simple matter to maintain, manage, and operate a secure manufacturing facility in-house in order to implement provisioning functionality. This is why the fleet provisioning method had gained so much attention as an approach to the device provisioning process, and this is probably why market demand for this method is growing rapidly.

The provisioning method described in this document is only one example, so it will not satisfy the requirements of all users. Nevertheless, we think the information presented here will help deepen the reader’s understanding of the advantages and disadvantages of deployment. It is our hope that this document will help users build convenient and practical production lines.

6. Websites and Support Information

AWS re:Post : https://repost.aws

Renesas FreeRTOS GitHub : https://github.com/renesas/iot-reference-rx

7. Appendix

7.1 Points to Keep in Mind when Operating Multiple Devices within the Same LAN Environment

Addresses assigned from the header ID of Renesas Electronics Corporation are used as the MAC addresses contained in the sample code.

When using the sample program to operate multiple devices within the same LAN environment, it is necessary to change the MAC addresses to avoid duplication.

The sample program may not operate properly if the same MAC addresses are duplicated among multiple devices.

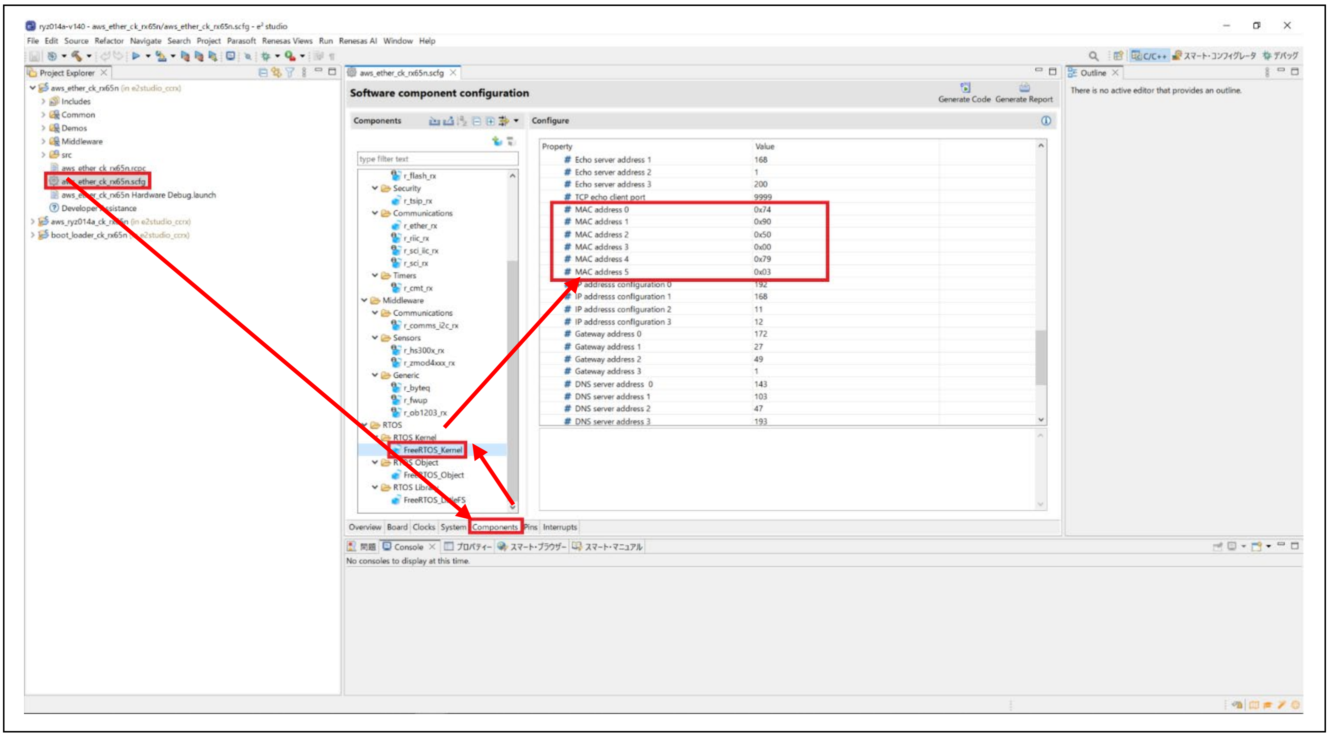
The procedure for changing the MAC addresses is described below.

Open **aws_ether_ck_rx65n.scfg** in Smart Configurator, and select the **Components** tag.

In the tree, select **RTOS** → **RTOS Kernel** → **FreeRTOS_Kernel**, then under **Property** set **Value** for **MAC address 0** to **MAC address 5** to hexadecimal values of your choice.

Enter values in the format **0xXX** (where XX represents a hexadecimal value of your choice).

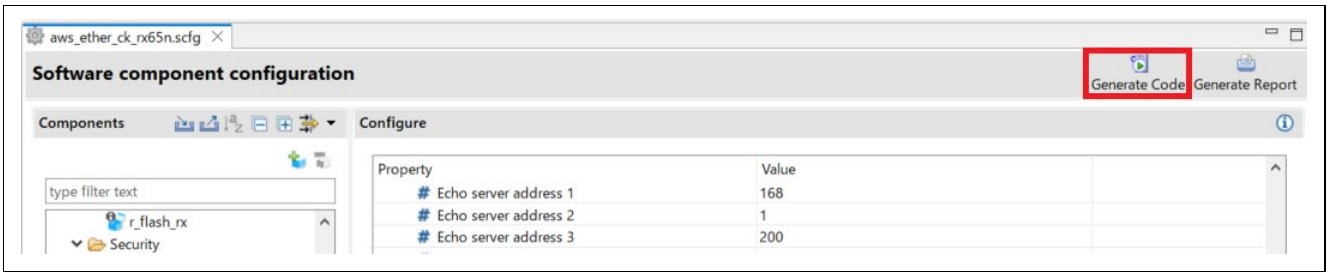
When commercializing a product for sale, make sure to apply to the IEEE to obtain the MAC addresses eventually used.



Property	Value
Echo server address 1	168
Echo server address 2	1
Echo server address 3	200
TCP echo client port	9999
MAC address 0	0x74
MAC address 1	0x90
MAC address 2	0x50
MAC address 3	0x00
MAC address 4	0x79
MAC address 5	0x03
IP addresss configuration 0	192
IP addresss configuration 1	168
IP addresss configuration 2	11
IP addresss configuration 3	12
Gateway address 0	172
Gateway address 1	27
Gateway address 2	49
Gateway address 3	1
DNS server address 0	143
DNS server address 1	103
DNS server address 2	47
DNS server address 3	193

Figure 7.1 MAC Address Settings

After making the above changes, click the **Generate Code** button in the upper right corner of the window to apply the changes made in Smart Configurator to the code.



Property	Value
Echo server address 1	168
Echo server address 2	1
Echo server address 3	200

Figure 7.2 Generating Code

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 30, 2023	—	First edition issued
1.10	Jun. 14, 2024	—	Updates to 3.2 Software Environment Updates to 3.3 Tera Term Installation and Settings Partial revisions to 4.3.1 Policy Settings Addition of 4.4 Creating the Sample Project
			Remainder intentionally left blank.

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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