

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 (<u>R01AN5548</u>).



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] <u>https://www.renesas.com/document/eln/plc-240004-end-life-eol-process-select-part-numbers?r=1503996</u>

[The product page] <u>https://www.renesas.com/products/wireless-connectivity/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module</u>



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: <u>https://youtu.be/gcJwNEQ2eLY</u> Document: <u>https://pages.awscloud.com/rs/112-TZM-766/images/EV_iot-deepdive-aws2_Sep-2020.pdf</u>
- 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
 <u>https://docs.aws.amazon.com/iot/latest/apireference/API_CreateCertificateFromCsr.html</u>
- Provisioning devices that don't have device certificates using fleet provisioning <u>https://docs.aws.amazon.com/iot/latest/developerguide/provision-wo-cert.html</u>
- How to automate onboarding of IoT devices to AWS IoT Core at scale with Fleet Provisioning
 <u>https://aws.amazon.com/blogs/iot/how-to-automate-onboarding-of-iot-devices-to-aws-iot-core-at-scale-with-fleet-provisioning/</u>



Contents

1. Terminology	5
2. Device Provisioning	6
2.1 Provisioning Methods of AWS IoT	7
2.2 Fleet Provisioning Method	8
2.3 Provisioning by Claim (Approach Using Provisioning Claim Certificates)	9
2.3.1 Overview of Provisioning by Claim (Using Provisioning Claim Certificate)	10
2.3.2 Determining a Unique Thing Name	12
3. Preparation	13
3.1 Hardware Environment	13
3.2 Software Environment	13
3.3 Tera Term Installation and Settings	14
3.4 FreeRTOS Project	15
4. Running the Fleet Provisioning Demo	16
4.1 Preparing the Running Environment	16
4.2 AWS Preparation	
4.3 AWS Settings for Fleet Provisioning	18
4.3.1 Policy Settings	18
4.3.2 Generating a Claim Certificate and Claim Key Pair	
4.3.3 Creating a Fleet Provisioning Template	27
4.4 Creating the Sample Projects	32
4.5 FreeRTOS Settings	41
4.5.1 Modifying the Configuration File	41
4.5.2 Cellular information settings	42
4.6 Building and Running the Program	
4.7 Confirming the Results of Running the Demo	53
5. Conclusion	57
6. Websites and Support Information	57
7. Appendix	58
7.1 Points to Keep in Mind when Operating Multiple Devices within the Same LAN Environment	58
Revision History	60

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



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



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

- 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.



RX Family



Figure 2.3 Workflow of Provisioning by Claim Using CreateKeysAndCertificate Method



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.



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



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

ltem	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 M nmaya	Tera Term 4.106 (Latest)		
 ♦ teraterm-4_106 •• 0433752 ♦ 	Source code is not available.		
compare v	▼ Assets 4		
	Øteraterm-4.106.exe	12.2 MB	Jul 12
		8.63 MB	Jul 12
	Source code (zip)		Jul 12
	Source code (tar.gz)		Jul 12
	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	
Fleet Provisioning	demo Application
AWS IoT Fleet Provisioning Library	
Freel	RTOS
RX Driver Package	
	65N

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: <u>iot-reference-rx</u> : 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 (<u>https://aws.amazon.com/</u>)

(1) Select Sign In to the Console \rightarrow Get Started for Free to create a new account.

	Contact Us Support - English - My Account - Sign In to the Console
AWS Marketplace Customer Enablement Events Explore	re More Q
2 Click Sign In to the Console and	l sign in.
	Contact Us Support - English - My Account - Sign In to the Console
AWS Marketplace Customer Enablement Events Explor	re More Q
(3) Select Services \rightarrow Internet of The	pings \rightarrow IoT Core to open the AWS IoT console
Serier	
Application Integration	FreeRTOS
🛗 AWS Cost Management	FreeRTOS is an IoT Operating System for Microcontrollers
BBB Blockchain	
Iusiness Applications	IOI I-CLICK Trigger AWS Lambda functions from simple devices
Compute	
🖮 Containers	IoT Analytics
③ Customer Enablement	Collect, preprocess, store, analyze and visualize data of for devices
🛱 Database	IoT Core
- X Devyloper Tools	Connect Devices to the Cloud
Fnd User Computing	IoT Device Defender
Eropt and Web & Mobile	Secure your fleet of connected IoT devices
	Int Davice Management
Game Development	Securely Manage Fleets as Small as One Device, or as Broad as Millions of Devices
W Internet of Things	
Machine Learning	IoT Events
Management &	Detect and respond to events from IoT sensors and Industrial IoT equipment
Governance	





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.

AWS IoT ×	AWS IoT > Security > Policies
Monitor	AWS IoT policies (1) Info AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies.
Connect Connect one device Connect many devices	C Delete Create policy Q Find policies D Policy name
Test	ck rx65n demo policy
 Device Advisor MQTT test client Device Location New 	
Manage	
All devices	
Greengrass devices	
LPWAN devices	
Software packages <u>New</u>	
Message routing	
Retained messages	
Security Intro Certificates Policies Certificate authorities	

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

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 statements Policy examples	
Policy document Info An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.	Builder J50N
Policy document Version": "2012-10-17", "Statement": [""""""""""""""""""""""""""""""""""""	
	Cancel
	Cleate

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/*"
     ]
   }
 ]
}
```





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

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

AWS IoT ×	AWS IOT > Security > Policies
Monitor	AWS IoT policies (1) Info AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies.
Connect Connect one device Connect many devices	C Delete Create policy Q Find policies Delete Policy name
Test Device Advisor MQTT test client Device Location New 	<u>ck rx65n deno policy</u>
Manage All devices Greengrass devices LPWAN devices Software packages <u>New</u> Remote actions 	
 Message routing Retained messages Security Intro Certificates Policies Certificate authorities 	

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.

PolicyName							
policy name is an alphanumeric st	ring that can also contain period	d (.), comma (,), hyphen(-), underscore	(_), plus sign (+), equal sign	(=), and at sign (@) characters, but no space	5.		
Tags - optional							
olicy statements Policy	r examples						
olicy document Info		\mathbf{A}					Builder JSON
olicy document Info AWS IoT policy contains one or m	ore policy statements. Each pol	licy statument contains actions, resour	rces, and an effect that gran	ts or denies the actions by the resources.			Builder JSON
Dicy document Info AWS IoT policy contains one or n	ore policy statements. Each pol	licy statument contains actions, resour	rces, and an effect that gran	ts or denies the actions by the resources.		_	Builder JSON
Olicy document Info AWS IoT policy contains one or n licy effect Illow	nore policy statements. Each pol	licy statement contains actions, resour Policy action iot.Connect	rces, and an effect that gran	ts or denies the actions by the resources. Policy resource *	Re	move	Builder JSON
Dicy document info AWS IOT policy contains one or n licx effect llow	ore policy statements. Each pol	licy statution contains actions, resour Policy action lot:Connect iot:Publish	rces, and an effect that gran	ts or denies the actions by the resources. Policy resource *	Re	move	Builder JSON
Dlicy document info AWS IoT policy contains one or n licy effect Illow	vore policy statements. Each pol	licy staturent contains actions, resour Policy action fot:Connect fot:Publish	rces, and an effect that gran	ts or denies the actions by the resources. Policy resource *	Re	move	Builder JSON
Dicy document info AWS IoT policy contains one or in licy effect Illow	vore policy statements. Each pol	licy staturent contains actions, resour Policy action iot.Connect iot.Publish iot.Subscribe	vces, and an effect that gran	ts or denies the actions by the resources. Policy resource *	Re Re	move move	Builder JSON
olicy document info AWS IoT policy contains one or n olicy effect Allow Allow	vore policy statements. Each pol	licy staturent contains actions, resour Pelicy action iot:Connect iot:Publish iot:Subscribe iot:Receive	cces, and an effect that gran	ts or denies the actions by the resources.	Re Re Re	move move move	Builder JSON

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 \rightarrow Certificates and then click Add certificate \rightarrow Create certificate.

Device Location New	AWS IoT > Security > Certificates	
Manage All devices Greengrass devices LPWAN devices	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 AWS IoT. Certificates Certificates you've transferred	
Software packages <u>New</u> Remote actions Message routing Retained messages 	Certificates (14) C Actions Actions	Add certificate Create certificate Register certificates
▼ Security	Certificate ID V Status V Created	•
Intro Certificates	Q Active June 26, 2023, 16:35:26 (UTC+09:00)	
Policies	C Active May 18, 2023, 10:49:25 (UTC+09:00)	
Certificate authorities	□ Active May 18, 2023, 10:36:39 (UTC+09:00)	
Role aliases Authorizers	O Active May 17, 2023, 17:04:26 (UTC+09:00)	

Figure 4.8 Creating a Certificate



Click Auto-generate new certificate (recommended) \rightarrow Create.



Figure 4.9 Creating a Certificate Automatically

Download the newly created certificate (1) and key pair (2)(3), then click the **Continue** button.

Download certificates and key Device certificate Image: Contribution of the certificate and key files on our device so that it can connet securely to AVS to You can download the certificate now, or later, but the key files can only be downloaded of now Device certificate Image: Contribution of 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. Image: Contribution of the key files Image: Contribution of the key file Image: Contribu	Download certificates and keys	×
Certificate downloadd Figure 10 this certificate and car'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 Private key file	Download certificates and keys Download and install the certificate and key files to your device so that it can connect securely to A IoT. You can download the certificate now, or later, but the key files can only be downloaded now. Device certificate	AWS
▲ This is the only time you can download the key files for this certificate. Public key file ② ● Download Private key file ③ ● Download Private key file ③ ● Download Ownload the root CA certificates ③ ● Download Ownload the root CA certificates 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 ● ● Download 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. ● <th>Certificate downlo 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.</th> <td>baded</td>	Certificate downlo 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.	baded
Image: Several control of the severa	This is the only time you can download the key files for this certificate. Public key file Download Download	d
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.	Private key file Image: Constraint of the second secon	ed d ed
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.	Root CA certificates Download the root CA certificate file that corresponds to the type of data endpoint and cipher suit you're using. You can also download the root CA certificates later. Amazon trust services endpoint DSA 2040 bit loss: Appane Root CA 1	te d
root CA certificates. These root CA certificates and others are available from our developer guides.	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 additio	d
	root CA certificates. These root CA certificates and others are available from our developer guides.	

Figure 4.10 Downloading the Certificate and Key Pair



RX Family

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

Connect Connect one device Connect many devices	AWS IOT > Security > Certificates Certificates Info X.509 certificates authenticate device and client connections. Certificates in	nust be registered with AWS IoT and activate	d before a device or client can communicate with A
Test	Certificates Certificates you've transferred		
Device Advisor			
MQTT test client	Certificates (15)		
Device Location New	Q. Find certificates		
Manage			
Manage	Certificate ID	⊽ Status ⊽	Created
Greengrass devices		⊖ Inactive	September 07, 2023, 18:50:57 (UTC+09:00)
 LPWAN devices 		⊘ Active	June 26, 2023, 16:35:26 (UTC+09:00)
Software packages New		⊘ Active	May 18, 2023, 10:49:25 (UTC+09:00)
Remote actions		⊘ Active	May 18, 2023, 10:36:39 (UTC+09:00)
Retained messages		⊘ Active	May 17, 2023, 17:04:26 (UTC+09:00)
▼ Security		⊘ Active	May 17, 2023, 17:04:24 (UTC+09:00)
Intre		⊘ Active	May 17, 2023, 17:00:16 (UTC+09:00)
Policies		⊘ Active	May 17, 2023, 16:54:23 (UTC+09:00)
Certificate authorities		⊘ Active	May 17, 2023, 16:53:37 (UTC+09:00)

Figure 4.11 Certificate Settings



Click Actions \rightarrow Activate to activate the certificate. Also click the Attach policies button.

		Actions
		Activate
etails ertificate ID ertificate ARN	Status ⊖ Inactive Created September 07, 2023, 18:50:57 (UTC+09:00)	Revoke Accept transfer Reject transfer Start transfer Attach pplicy Attach to things
ubject N=AWS IoT Certificate suer U=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US olicies Things Noncompliance	Valid September 07, 2023, 18:48:57 (UTC+09:00) Expires January 01, 2050, 08:59:59 (UTC+09:00)	Downlo d Delete
olicies (0) Info WS IoT policies allow you to control access to the AWS IoT Core data plane operations.	CDe	tach policies Attach policies

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.

Policies Choose policies to attach to this certificate.	The certificate can have up to 10 policies attached to it.
Choose AWS IoT policy	▲ C
Q	
🗖 fa dama polizy	
Tp_demo_policy	

Figure 4.13 Certificate Settings: Attach Policies (2)



4.3.3 Creating a Fleet Provisioning Template

Select Connect many devices \rightarrow Connect many devices, then click the Create provisioning template button.

AWS IoT ×	AWS IoT > Connect > Connect many devices		
Monitor	▼ How it works		
Connect Connect one device ▼ Connect many devices Connect many devices Buik registration		Image: Constraint of the constraint	
Test ▶ Device Advisor MQTT test client Device Location New	Step 1. Determine providening scenario Devices need a unique certificate to commune AutyS IoT. You can install this certificate during the device's manufacture, avoider 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. Learn more [2]	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 sciences, thing groups, thing types, searchable attributes, and billing groups abom the our manage your devices and can also be created when the device is provisible. Learn more [2]	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. Learn more
Manage All devices Greengrass devices LPWAN devices	Connect many devices (0) Info To connect many devices, the provisioning template automates the provisioning requi	red to connect new devices.	Deactivate Delete Create provisioning template
Software packages <u>New</u> Remote actions Message routing 	Name Template type	▼ Created date ▼ State	us. V
Retained messages Security Fleet Hub 		No provisioning templates You don't have any provisioning templates in us-east-1. Create provisioning template	

Figure 4.14 Creating a Provisioning Template (1)

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

Create provisioning template

Provisioning scenario	overse the best Learn more 12	
and second provide and second or the gran second method and instantice pr		
 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 scamerio, 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 certificate is connect to AWS IoT for their titum. The claim certificate is replaced with unique device certificate after provisioning. This option is also known as flee provisioning with certificate.
o provision devices with claim certificates Learn more 2 Configure Claim certificates onfigure the claim certificates that you'll install on your IoT devices and onfigure the policies attached to those certificates to allow rovisioning.	OOO Image: Cool of the second se	Open set of the set o

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.

details on this page describe the general aspects of the provisioning template that you're creating. Provisioning template properties Info Provisioning template status determines whether the template can be used to provision a new device. Only active templates can create an inactive templates to prevent devices from being provisioned until you're ready. I nactive emplates can't provision any devices that are configured to use it. You can create an inactive template can provision the devices that are configured to use it. Provisioning template aname Enter_template_name Inter_template_name Inter_template and must not combin spaces. Valid characters: A-Z, a-Z, 0-9, and _ (underscore) and - (hyphen). Description - optional A description of the provisioning template you're creating. io0 character remaining Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behavior. C View C Create new role A ttach managed policy to IAM role Tags - optional	escribe p	provisioning template Info
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 rovision devices. Inactive Implates can't provision any devices that are configured to use it. You can create an inactive template can provision the devices from being provisioned until you're ready. Active An active template can provision the devices that are configured to use it. Active template can provision the devices that are configured to use it. Active template can provision the devices that are configured to use it. Active template can provision the devices that are configured to use it. Active template can provision the devices that are configured to use it. Active template name Enter_template_name Inter an have up to 36 characters and must not comb spaces. Valid characters: A-Z, a-z, 0-9, and _ (underscore) and - (hyphen). Description - optional A description of the provisioning template you're creating. Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behave. Choose an IAM role Attach managed policy to IAM role Tags - optional	e details on this p	age describe the general aspects of the provisioning template that you're creating.
Provisioning template status The provisioning template status determines whether the template can be used to provision a new device. Only active templates can arrovision devices. Inactive Inactive templates can't provision any devices that are configured to use it. You can create an inactive template can provision the devices that are configured to use it. You can create an inactive template can provision the devices that are configured to use it. 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 control spaces. Valid characters: A-Z, a-z, 0-9, and _ (underscore) and - (hyphen). Description - optional A description of the provisioning template you're creating. Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalts. Choose an IAM role A tatch managed policy to IAM role Tags - optional	Provisioning	template properties Info
 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. Solo character remaining Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalts. Choose an IAM role C View C Create new role Tags - optional 	Provisioning tem The provisioning tem provision devices. O Inactive Inactive templa inactive templa	plate status nplate status determines whether the template can be used to provision a new device. Only active templates can tes can't provision any devices that are configured to use it. You can create an te to prevent devices from being provisioned until you're ready.
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. Goo character remaining Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalts Choose an IAM role A ttach managed policy to IAM role Tags - optional	 Active An active temple 	ate can provision the devices that are configured to use it.
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. Solo character remaining Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalter. C View C Create new role Attach managed policy to IAM role Tags - optional	Provisioning tem	plate 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. A description of the provisioning template you're creating. O create remaining Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalts. Choose an IAM role Attach managed policy to IAM role Tags - optional	Enter_template	_name
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 behalts Choose an IAM role C View C Create new role Attach managed policy to IAM role	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).
A description of the provisioning template you're creating.	Description - opt	ional
Soo character remaining Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalts Choose an IAM role ✓ C View C Create new role ✓ Attach managed policy to IAM role ✓ Tags - optional	A description of	the provisioning template you're creating.
Provisioning role The provisioning role uses an IAM role that authorizes AWS IoT to access resources on your behalts Choose an IAM role Attach managed policy to IAM role Tags - optional	500 character rema	ining
Choose an IAM role Create new role Attach managed policy to IAM role Tags - optional	Provisioning role	e uses an IAM role that authorizes AWS IoT to access resources on your behalt
 Attach managed policy to IAM role Tags - optional 	Choose an IAM	role C View C Create new role
▶ Tags - optional	Attach manage	jed policy to IAM role
► Tags - optional		
	▶ Tags - option	al

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 or your behalf.	л
Role name	
fleet_demo	
Enter a unique role name that contains alphanumeric characters, hyphens, and underscores. A role name can't contain any spaces.	
Cancel	

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.

doesn't apply to the device certificate	hat will be provisioned. You'll o	onfigure the policies for	the provisioned device ce	rtificate later.
Claim certificate provisioning pol Choose the AWS IoT policy that author certificates you choose in the next sect	icy izes the claim certificate to con ion.	nect and provision the lo	oT device. This policy is att	ached to the claim
fp_demo_policy		C View Z	Create IoT po	olicy 🖸
Q				
fp_demo_policy	~			
ck_rx65n_demo_policy in a limited number of IoT	devices limits your exposu	es to provision your o re in case a claim cert	device fleet. Using each tificate is compromised	n claim certificate I.
ck_rx65n_demo_policy in a limited number of IoT Claim certificates - option Choose the claim certificates to attach Claim certificates must be active and h	ac matteries commercial devices limits your exposur nal (1/15) Info the policy to, or attach the pol ave the claim certificate provisi	es to provision your of re in case a claim cert icy later by editing the p oning policy attached.	device fleet. Using each tificate is compromised	n claim certificate I. visioning initiator.
ck_rx65n_demo_policy in a limited number of IoT Claim certificates - option Choose the claim certificates to attach Claim certificates must be active and h C Activate Dea	ac matupe come condicat devices limits your exposur and (1/15) Info the policy to, or attach the pol ave the claim certificate provisi ctivate Delete	es to provision your of re in case a claim cert icy later by editing the p oning policy attached.	device fleet. Using each tificate is compromised	n claim certificate I. visioning initiator.
ck_rx65n_demo_policy in a limited number of IoT Claim certificates - option Choose the clain certificates to attach Claim certificates must be active and h C Activate Dea	activate Contract of the policy to, or attach	es to provision your of re in case a claim cert icy later by editing the p oning policy attached. Upload	device fleet. Using each tificate is compromised rovisioning template's pro	visioning initiator.
ck_rx65n_demo_policy in a limited number of IoT Claim certificates - option Choose the claim certificates to attach Claim certificates must be active and h C Activate Dea C Certificate ID	ac matupe come condicat devices limits your exposur and (1/15) Info the policy to, or attach the pol ave the claim certificate provisi ctivate Delete	es to provision your of re in case a claim cert icy later by editing the p oning policy attached. Upload	device fleet. Using each tificate is compromised rovisioning template's pro X 1 match V Status	visioning initiator.

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"

Pre-pr	ovisioning actions (recommended) Info
efore a r unction t	ew device is provisioned, you can run a Lambda function to verify the device should be provisioned. We recommend use this o control access to your AWS account.
re-prov	isioning action
O Use a Perfo device	pre-provisioning action (recommended) m actions prior to provisioning the device. For example, to check the device against a known database to prevent unauthorized devices from connecting to your account.
Don' No ac AWS a	use a pre-provisioning action ions will be performed prior to provisioning the device and the device is given access to your ccount.
≙	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
	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account.
	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account.
Autom Create a t features s	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. atic thing creation - optional ning resource to represent the device in AWS IoT. Your devices will need thing resources to use AWS IoT device management uch as thing groups, billing groups, and Device Shadows.
Autom Create a t eatures s	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. atic thing creation - <i>optional</i> ning resource to represent the device in AWs IoT. Your devices will need thing resources to use AWS IoT device management uch as thing groups, billing groups, and Device Shadows. omatically create a thing resource when provisioning a device
Autom Create a t eatures s Aut Ching na The thing	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. atic thing creation - <i>optional</i> ing resource to represent the device in AWS IoT. Your devices will need thing resources to use AWS IoT device management uch as thing groups, billing groups, and Device Shadows. omatically create a thing resource when provisioning a device me prefix name prefix forms the beginning of each thing resource geated by this provisioning template.
Autom Treate a t Treatures s Aut Thing na The thing Enter_t	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. atic thing creation - optional ing resource to represent the device in AWE IoT. Your devices will need thing resources to use AWS IoT device management uch as thing groups, billing groups, and Device Shadows. omatically create a thing resource when provisioning a device me prefix name prefix forms the beginning of each thing resource greated by this provisioning template. hing_prefix_
Autom Treate a t eatures s Autom Thing na The thing Enter_1 The name	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. atic thing creation - <i>optional</i> ming resource to represent the device in AWS IoT. Your devices will need thing resources to use AWS IoT device management ach as thing groups, billing groups, and Device Shadows. omatically create a thing resource when provisioning a device me prefix name prefix forms the beginning of each thing resource teated by this provisioning template. <i>hing_prefix_</i> can't contain spaces. Valid characters: A-Z, a-z, 0-9, and _: - (hyphen)
Autom Create a t eatures s Aut Thing na The thing Enter_1 The name	certificate to provision your devices. This action performs additional validation of devices before they are provisioned in your AWS account. atic thing creation - optional hing resource to represent the device in AWS IoT. Your devices will need thing resources to use AWS IoT device management ich as thing groups, billing groups, and Device Shadows. omatically create a thing resource when provisioning a device me prefix name prefix forms the beginning of each thing resource geated by this provisioning template. hing_prefix_ can't contain spaces. Valid characters: A-Z, a-z, 0-9, and _:- (hyphen) 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.

C Create po	licy 🛽
< 1	> @
ARN	7
D .	
Ø	
Cancel Previous	

Figure 4.20 Creating a Provisioning Template (6)

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

Policies		
Policy name	Policy action	Policy effect
k_rx65n_demo_policy 🛽	iot:Connect	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 <u>Getting Started</u> <u>Guide</u>.

(1) Create a workspace in e^2 studio.

Launch e² 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

i e² studio Launcher			×
Select a directory as workspace			
e ² studio uses the workspace directory to store its preferences and developr	nent artifacts.		
C:¥workspace		✓ Browner Stress	owse
Use this as the default and do not ask again			
• <u>R</u> ecent Workspaces			
Lau	unch	Cancel	

Figure 4.22 Dialog Box for Creating a Workspace

After launching e² studio, from the **File** menu select **New** > **Renesas** C/C++ **Project** > **Renesas** RX to display the **New** C/C++ **Project** dialog box.

n Renesas Al Window Help	
Renesas C/C++ Project >	Renesas Debug
C/C++ Project	Renesas RX
Project	
📑 Other Ctrl+N	tion
> >	un Renesas Al Window Help Renesas C/C++ Project C C/C++ Project Project Other Ctrl+N

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.



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.

8							×
New Renesas CC-I	RX Executable Pro K Executable Project	ject					2
Project name: fleet	_demo						
Use default loca	tion						
Location:	C:¥git¥lts¥iotref-da	166PG¥fleet_demo			E	Browse	
	Create Directory	for Project					
Choose file system:	default 🗠						
Working sets							
Add project to	working sets			[1	New	
Working sets:				~	S	elect	
?		< Back	Next >	Finish		Cancel	

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 e² 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.

0		– 🗆 X
New Renesas	CC-RX Executable Project	
Toolchain Settin Language: Toolchain: Toolchain Versio RTOS: RTOS Version:	e C C++ Renesas CC-RX v3.05.00 Manage Toolchains FreeRTOS (with IoT libraries) 202210.01-LTS-rx-1.2.1 v	
Device Settings Target Board: Target Device:	CK-RX65N ~ R5F565NEHxFB	Configurations Create Hardware Debug Configuration E2 Lite (RX) Create Debug Configuration
Endian: Bank Mode	Little ~ Dual Bank ~	RX Simulator ~
?	< <u>B</u> ack <u>N</u> ext >	Einish Cancel

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.



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.

6					- [o x
New Renesa	is CC-RX Executable Project					-
Select RTOS F	Project Settings					4
Select applicat			CONTRECTOR			
pursuit	(Wi-Ei) PubSub/MOTT cample project	(^
0	This demonstration illustrates PubSub functionality using AWS services with Wi-Fi(D)A16600) cc	onnection.			
	(Ethernet) PubSub/MQTT with Fleet Provisioning sample project					1.1
	This demonstration illustrates Fleet Provisioning functionality, followed by PubSub	using AWS s	services with Eth	ernet connection.		
•	(Cellular) PubSub/MQTT with Fleet Provisioning sample project					
	This demonstration illustrates Fleet Provisioning functionality, followed by PubSub a	using AWS s	services with Ce	llular(RYZ014A) connec	tion.	
0 👩	(Wi-Fi) PubSub/MQTT with Fleet Provisioning sample project		convicor with Wi	-Ei/DA16600) connectio	0.0	
_	(Ethernet) BubSub (MOTT with Over the siz (OTA) undete comple project	using Aws :	services with wi		011.	
0 👩	This demonstration illustrates PubSub and OTA update functionality (simultaneous)	y) using AW	/S services with	Ethernet connection. N	ote: This	5
	project requires bootloader project.					
	(Cellular) PubSub/MQTT with Over-the-air (OTA) update sample project					201001
	This demonstration illustrates PubSub and OTA update functionality (simultaneous). This project requires bootloader project.	y) using AW	/S services with	Cellular(RYZ014A) conr	nection.	Note:
?	< <u>B</u> ack		<u>N</u> ext >	Einish	С	ancel

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.

							¥č		×
lew Renesas CC-RX Executal Settings The Contents of Files to	ble Projec	t ed						F	*
								-	
What kind of initialization routine	e would you	like to create	?						
Use Renesas Debug Virtual C	Console								
Size of I/O Stream Buffer:									
3	÷								
2		< Back	[Next >		Finich		Cancel	
		Dack		Mext	<u> </u>	Lunsti		Cancel	

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.

8				-	- [⊐ ×
New Renesas CC-RX Exe	ecutable Proj	ect				-
Summary of project "fleet	_demo"					1
TOOLCHAIN NAME : TOOLCHAIN VERSION : GENERATION FILES :	Renesas CC- v3.05.00	RX				~
						~
?		< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	C	ancel

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 Marketplace.	support is available on the
\odot Show IDE extensions for this file type and let me install them	
\bigcirc Associate '*.md' files with current editor (System Editor) and do no	ot ask again
See also Preferences for File Associations	
	OK Cancel

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**.



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 e² studio, open aws_ryz014a_ck_rx65n/src/frtos_config/demo_config.h and change the value of **ENABLE_FLEET_PROVISIONING_DEMO** to **1**.



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** \rightarrow **Generic** \rightarrow **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.



Figure 4.34 Launch the Smart Configurator



Project Explorer X	∰ aws_ryz014a_ck_rx65n.scfg ×		_	- 0
✓	Software component configura	tion		6
> 🔊 Includes	Software component comgut		G	enerate Code Generate Report
> 🚱 Common	Company A state of the	Configure		0
> 🤀 Demos	Components 🔛 🖂 🖓 🕒 🕀 🖛 👻	Configure		U
> 🚱 Middleware	10 T	Property	Value	<u>^</u>
✓ 28 src	type filter text	× @ Configurations		
> 🧁 application_code	type inter text	# Access point name	ibasis int	
✓ ⇐ frtos_config	V 🗁 Startup	# Access point login ID	100515.101	
> Core http config.h	✓ 🧁 Generic	# Access point nassword		
> in core matt agent config.h	r_bsp	# SIM card PIN code		
> h core matt config.h	V 🗁 Drivers	# Authentication protocol type.	2	
> Core pkcs11 config.h	Interrupt	# Network status notification level.	2	
> De defender config h		# Connection retry limit	600	
> a demo confia h	A/D Converter	# TCP connection timeout	0	
Elect provisioning config h	Memoni	# SCI interrupt priority	4	
FreePTOS Config h	• Memory	# Maximum semaphore acquisition latency(msec)	15000	
> In FreeRTOS/DCapfie h		# Reception guard time before the module transitions to PSM	100	
> In FreekTOSIPConlig.n	Bill r tsin ry	# Maximum allowable wake-up delay from PSM	5000	
> im mbedtis_user_config.n	× Communications	# RING line active duration	1000	
> in ota_config.h	t sci rx	# Maximum FW update latency	60	
> In ota_demo_config.h	V Com Middleware	# Enable user-defined URC charget functions	Disable	
> Im rm_littlefs_flash_config.h	V 🗁 Generic	# User URC charget function name	my_sw_urc_charget_function	
> 🖪 shadow_config.h	9 r byteg	# Debug log output level.	4	
> 🧁 frtos_skeleton	💣 r cellular	# Reset signal logic.	1	
> 🗁 frtos_startup	V 🗁 RTOS	# SCI Channel	6	
> 🗁 smc_gen	✓ → RTOS Kernel	# UART hardware flow control	CTS(Hardware), RTS(Software)	F
aws_ryz014a_ck_rx65n.rcpc	FreeRTOS_Kernel	# CTS port number		
@ aws_ryz014a_ck_rx65n.scfg	✓ → RTOS Object	# CTS pin number		
aws_ryz014a_ck_rx65n Hardware Debug.launch	FreeRTOS_Object	# RTS pin function set value	0x0BU	\$
⑦ Developer Assistance				~
				×
	Overview Board Clocks System Compone	nts Pins Interrupts		

Figure 4.35 Entering Cellular information

Note: About the setting of Wi-Fi network with DA16600 module, please refer to the <u>GitHub [Settings of Wi-Fi</u> <u>network (Only using Wi-Fi)]</u>. And regarding setting of country code and GMT timezone, please refer to the <u>Settings of Country code and GMT timezone (Only using Wi-Fi)</u> 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** \rightarrow **Debug Configurations...** from the e² studio menu to open the Debug Configurations window. In the list at the left of the Debug Configurations window, select **Renesas GDB Hardware Debugging** \rightarrow **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.

	Name. aws_tyzo14a_ck_txo51111ardware bebug		
pe filter text	Main Startup V Startup V Source Common		
C/C++ Application C/C++ Remote Application EASE Script GDB Hardware Debugging	Debug hardware: E2 Lite (RX) Target Device: R5F565NE	DUAL	
GDB Simulator Debugging (RH850)	Clock		
Launch Group	Main Clock Source	FXTAL	~
Renesas GDB Hardware Debugging	Extal Frequency[MHz]	24	
aws_ryz014a_ck_rx65n Hardware Debug	Operating Frequency [MHz]	120.000	
Renesas Simulator Debugging (RX, RL78)	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	4
	✓ CPU Operating Mode		
	Register Setting	Single Chip	~
	Mode pin	Single-chip mode	~

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.

O TCP/IP	Host:	myhost.examp	ole.com		~
	Service:	History Telnet	TCP port#	22	
		● SSH	SSH version:	SSH2	
		Other	IP version:	AUTO	
Serial	Port:	COM12: USB S	Serial Device (COM	12)	~

Figure 4.37 Initial Window when Tera Term Starts

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

ra Term: Serial port	setup and co	nnection		\times
Port:	COM12	~	New setting	٦
Speed:	115200	~	,	_
Data:	8 bit	~	Cancel	
Parity:	none	~		
Stop bits:	1 bit	~	Help	
Flow control:	none	~		
Device Friendly N Device Instance II Device Manufact Provider Name: N Driver Date: 6-21	msec/chai msec/chai ame: USB Ser D: USB¥VID_0 urer: Microsoft 2006	r 0 ial Device ((45B&PID_81 ft	msec/line COM12) 111¥000000000001	~
Driver Version: 10	.0.19041.2130			

Figure 4.38 Serial Port Setup



Select Setup \rightarrow 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	New-line Receive: AUTO ~ Transmit: CR+LF ~	OK Cancel
Terminal ID: VT100 ~ Answerback:	Local echo	Help >TEK)

Figure 4.39 Terminal Setup

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

Test Device Advisor MQTT test client	Subscribe to a topic	Publish to a topic	
Device Location New	Topic filter Info The topic filter describes the topic(s) to	which you work to solveribe. The tools filter can include MOTT wildowid sharesters.	
Manage	#		
All devices	 Additional configuration 		
Greengrass devices	Addition		
LPWAN devices	Subscribe		
Software packages New	a second second second		
Remote actions			
Message routing	Subscriptions	#	Pause Clear Export Edit
Retained messages			
Security	# VX	You cannot publish messages to a wildcard topic.	
▶ Fleet Hub		Please select a different topic to publish messages to.	
Device software			
Billing groups		No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.	
Settings			

Figure 4.40 MQTT Test Client Settings



In e² 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.



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.



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 **xxxxxxx.amazonaws.com** that is displayed for **Endpoint** when you select **Settings** \rightarrow **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 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_1022_10		•
Compare security policies 🔀		
COM12 - Tera Term VT File Edit Setup Control Window Help	-	×
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!		^
Going to FreeRIOS-ULI !		
>format		
Format OK !		
>conf set endpoint		
ок.		
\rightarrow		~

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.

COM12 - Tera Term VT	-	×
File Edit Setup Control Window Help		
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!		
>CL1		
Going to FreeRTOS-CLI !		
>format		
Format OK !		
>conf set endpoint		
ОК.		
>conf set template fp_demo_template		
ОК.		
\rightarrow		~

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.



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.

		- 🗆 ×
File Home Share View		~ 2
	V O Date modified	True
-cer	ificate.pem.crt 8/7/2023 11:35 AM	Security Certificate
-priv	ate.pem.key 8/7/2023 11:35 AM	KEY File
	o/ 1/2025 11:55 AM	KET FILE
After typing conf set claimkey , drag and o the private key file. Then press the Enter k	lrop ey.	
	Tera Term: File Drag and Drop	×
	Are you sure that you want to send the f	ile content?
	O SCP	
	dest is nome directory	r if empty
3 items	Send File (Paste content of file)	
0 COM12 - Tera Term VT	Binary	×
File Edit Setup Control Window Help	O Paste Filename	
≻conf set claimkeyBEGIN RSA PRIVATE KEY	Escape	î
	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 texplayed	
		OK Cancel
END RSA PRIVATE KEY		
ОК.		
\rangle		~

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.



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.



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.



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.



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.

AWS IoT ×	Subscribe to a topic Publish to a topic	
Monitor		
Connect	Topic filter Info The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.	
Connect	#	
Connect one device	Additional configuration	
Connect many devices	Additional computation	
	Subscribe	
lest		
Device Advisor	Coloridation	
MQTT test client	Subscriptions #	
Device Location New	+	
	You cannot publish messages to a wildcard topic.	
Manage	Please select a different topic to publish messages to.	
All devices		
Greengrass devices		
LPWAN devices	pubsub_demo/dummy/task_1	
Software packages New		
Remote actions		
Message routing	Message cannot be displayed in specified format.	
Retained messages		
Security		
Fleet Hub	Task 1 publishing message 9	
Device software	Properties	
Dilling success		

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

aws Services	Q Search	[Alt+S]
AWS IoT	×	AWS loT > Manage > Things
Monitor		Things (96) Info An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.
Connect Connect one device	2	Q Filter things by: name, type, group, billing, or searchable attribute.
Connect many devi	ces	Name FRDamalD
Test		
Device Advisor MOTT test client		
Device Location N	ew	
Manage		
All devices	<u> </u>	
Things Thing groups		
Thing types		

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.

COM1	2 - Tera Term VT	-		X
File Edit	Setup Control Window Help			_
824 6703 >	4 Lcellular_rej LDEBUGj clear butt =			1
825 6704 OK	4 [cellular_re] [DEBUG] received AT command response:			
826 6704 OK	4 [cellular_re] [DEBUG] clear buff =			ŀ
827 6705	4 [DemoTask] [INF0] Received certificate with Id:			ב
828 6706 829 6710	5 [DemoTask] [INF0] Writing certificate into label [Device Cert". 4 [DemoTask] [DEBUG] generated AT command: AT+SONSS <mark>2</mark> NDEXT=1,33			
830 6710	4 [DemoTask] [DEBUG] RTS output 0			
831 6711	4 [cellular_re] [DEBUG] received AT command response:			
/ 832 6711 >	4 [cellular_re] [DEBUG] clear buff =			
833 6712	4 [cellular_re] [DEBUG] received AT command response:			
UN	FPDemoID			
	Thing details			
	Name Type FPDemoID			
	ARN Billing group			
	Confirm that the certificate ID matches that sh the debug log.	iown i	n	
	Attributes Certificates Thing groups Device Shadows Activity Packages and versions Jobs	Alarm	IS	
	Certificates (1) Info The device certificates attached to this thing resource.			
	Q. Find certificates			
	Certificate ID Status			
	□ O Active			

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 $RTOS \rightarrow RTOS \text{ Kernel} \rightarrow FreeRTOS_Kernel$, 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.



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.

aws_ether_ck_rx65n.scfg ×			
oftware component configuration	n		Generate Code Generate Repo
Components 🛛 🖄 📩 📮 🕀 🖶 莽 🔻	Configure		(
🐮 😳	Property	Value	^
type filter text	# Echo server address 1	168	
r_flash_rx	# Echo server address 2	1	
	# Calco annual data 2	200	

Figure 7.2 Generating Code



Revision History

		Description	
Rev.	Date	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 is supplied until the 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 systemevaluation test for the given product.

Notice

- 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.
- 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.
- 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.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 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/.