RX Family

OTA Update in FreeRTOS by Implementing TLS Communication Using the TSIP Driver

Introduction

For IoT devices, security and real time processing are critical factors.

This application note describes how to manage keys safely and speed up the encryption/description processing in TLS communication by using the Trusted Secure IP (TSIP) module, which is security hardware built into RX Family MCUs. This application note also provides concrete implementation examples and sample code. With this information, the reader will be able to build an efficient and safe IoT system.

Operation of the application described in this application note has been verified in FreeRTOS with IoT Libraries in which the TSIP driver is installed. Note that “FreeRTOS with IoT Libraries” is FreeRTOS in which the libraries required to use IoT services on AWS are installed.

This allows the OTA demo application to run on FreeRTOS with IoT Libraries in which the TSIP driver is installed.

Note: The application described in this application note uses the iot-reference-rx demo project, which is an RX-compatible version of FreeRTOS with IoT Libraries.
Note that the application described in this application note supports v202210.01-LTS-rx-1.3.0 or a later version of iot-reference-rx.

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

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

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

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

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

Target Device

RX65N: R5F565NEHDF

Documents for Reference

The application provided by this application note demonstrates an OTA update.
For detailed procedures for OTA update, refer to the following application note:

- How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later) (R01AN7037)

For details on TLS communication using the TSIP driver and the driver's API, refer to the following application note:

- RX Family TSIP (Trusted Secure IP) Module Firmware Integration Technology (R20AN0548)

For details on firmware update, refer to the following application note:

- Renesas MCU Firmware Update Design Policy (R01AN5548)

Contents

Notes:

- AWS™ is a trademark of Amazon.com, Inc. or its affiliates. (https://aws.amazon.com/trademark-guidelines)
- FreeRTOS™ is a trademark of Amazon Web Services, Inc. (https://freertos.org/copyright.html)
- Git® is a trademark of Software Freedom Conservancy, Inc. (https://www.git-scm.com/about/trademark)
- GitHub® is a trademark of GitHub, Inc. (https://github.com/logos)
- Arm® is a trademark of Arm Limited or its subsidiaries. (https://www.arm.com/company/policies/trademarks/guidelines-trademarks)
- Mbed™ is a trademark of Arm Limited or its subsidiaries. (https://www.arm.com/company/policies/trademarks/guidelines-trademarks)
- OpenSSL™ is a trademark of OpenSSL Software Foundation. (https://www.openssl.org/policies/general/TrademarkPolicy.html)

1. Overview

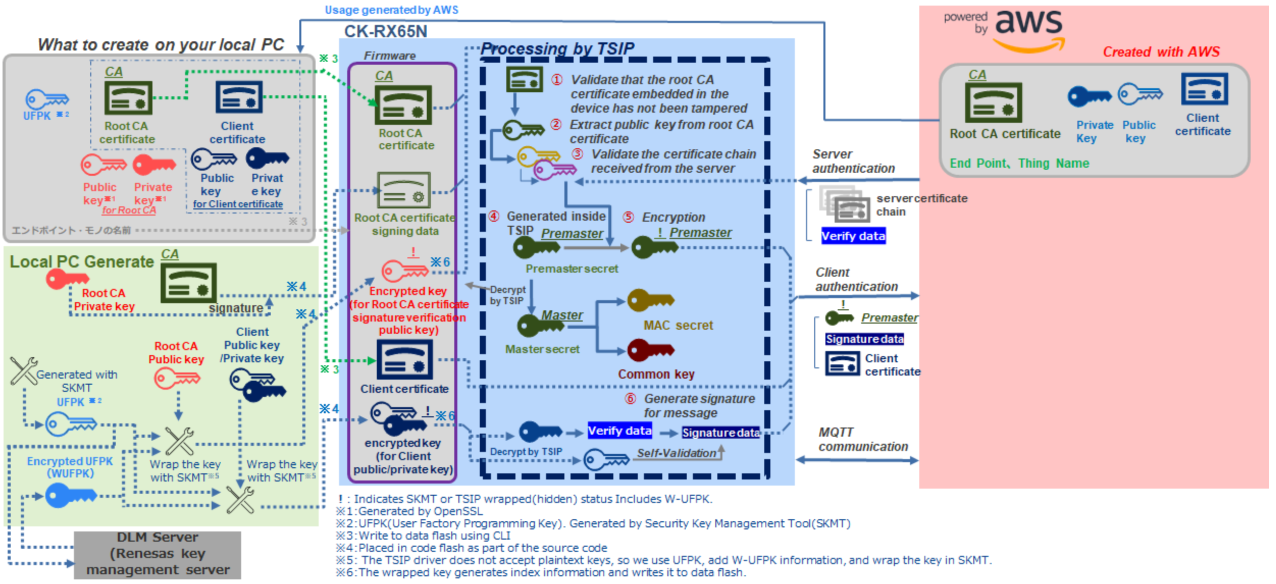
1.1 Advantages of TLS Communication Using the TSIP

The TSIP driver supports API functions related to TLS communication. Using this API offers the following two advantages:

- **Advantage 1: Key information in plaintext format is not handled in the TLS protocol processing. Therefore, the risk of leaking the customer's key information stored on the device can be reduced.**
- **Advantage 2: Hardware acceleration of cryptographic processing is faster than full software processing.**

1.2 Flow of TLS Communication Using the TSIP

The following figure shows the flow of TLS communication performed by the demo project provided by this application note.
Note that this flow applies when the RSA key exchange method is used.



! : Indicates SKMT or TSIP wrapped(hidden) status Includes W-UFPK.
※1:Generated by OpenSSL
※2:UFPK(User Factory Programming Key). Generated by Security Key Management Tool(SKMT)
※3:Write to data flash using CLI
※4:Placed in code flash as part of the source code
※5: The TSIP driver does not accept plaintext keys, so we use UFPK, add W-UFPK information, and wrap the key in SKMT.
※6:The wrapped key generates index information and writes it to data flash.

Figure 1-1 Flow of TLS Communication Using the TSIP

1.3 Cipher Suites Supported by the TSIP Driver

The TSIP driver supports the following cipher suites that are compliant with TLS 1.2:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

1.4 Definition of Terms

The following table defines the terms used in this application note.
For the sections where the listed items are used, refer to Figure 1-1, Flow of TLS Communication Using the TSIP.

Table 1-1 Terms

Term	Description
Key injection	To inject a wrapped key into a device at the factory.
User key	A plaintext crypto key used by the user. This is not used on a device. If the key exchange method is RSA or ECC, public and private keys are user keys.
Encrypted key	Key information generated by encrypting a user key by UFPK and adding a message authentication code (MAC) to it. Encrypted keys for the same user key share the same value for each device.
Wrapped Key	Data converted from an encrypted key by key injection so that it can be used for the TSIP. Because a wrapped key is wrapped by HUK, wrapped keys converted from the same encrypted key have device-specific values.
UFPK (User Factory Programming Key)	A user-configured key ring used to generate an encrypted key from a user key in key injection. This is not used on a device.
W-UFPK (Wrapped UFPK)	Key information generated by wrapping a UFPK by HRK on the Renesas DLM server. This is decrypted by HRK into a UFPK inside the TSIP.
Hardware Root Key (HRK)	A common crypto key that exists in the TSIP and Renesas secure rooms only.
Hardware Unique Key (HUK)	A device-specific crypto key that is derived in the TSIP and used for key protection.
Wrap (wrapping)	In this application note, “wrapping” refers to a process that performs UFPK-based encryption and adds a message authentication code (MAC) during generation of an encrypted key. Because the TSIP driver does not accept a plaintext user key as an input, the crypto key to be input must be wrapped.
Renesas DLM (Device Lifecycle Management) server (https://dlm.renesas.com/)	The Renesas key management server used by the Renesas Key Wrap service. This is used for UFPK-based wrapping.

1.5 Environment in Which Operation Was Verified (Hardware)

The following table shows the (hardware) environment in which operation of this demo project was verified.

Table 1-2 Environment in Which Operation Was Verified (Hardware)

Item	Description
Board used	CK-RX65N v1 (Cellular / Ethernet)*[1]
Cellular module	RYZ014A PMOD module (bundled with CK-RX65N v1)
SIM	SIM (microSIM) card supporting LTE Cat-M1*[2]
Debugger	Debugger built into the E2 Lite emulator (CK-RX65N v1)

Notes: 1. The application described in this application note uses cellular communication.

2. If you use the SIM card that is bundled with the CK-RX65N v1, activate the SIM card. For details, refer to “4.1.5 Activating SIM card” in the following application note:
SIM activation, Creating the trial account and using Dashboard with RYZ014A or Ethernet Application for AWS - Getting Started Guide (R01QS0064)

1.6 Environment in Which Operation Was Verified (Software)

The following table shows the (software) environment in which operation of this demo project was verified.

Table 1-3 Environment in Which Operation Was Verified (Software)

Item	Description
Integrated development environment	e^2 studio 2024-01
Compiler	RX compiler CC-RX V3.06.00 for e² studio
FreeRTOS	v202210.01-LTS-rx-1.3.0
Driver package (RDP)	RX Driver Package V1.42
TSIP driver	RX Family TSIP (Trusted Secure IP) Module Firmware Integration Technology Rev.1.20
Firmware update module	RX Family Firmware Update Module Firmware Integration Technology Rev.2.02
Log monitor tool	Tera Term v4.106
Python runtime environment	Python 3.11.0
Key generation tool	Win64 OpenSSL v3.2.1
PGP encryption/decryption tool	Gpg4win (Kleopatra) v4.3.1
Shell script (bash) execution environment	Cygwin version 3.4.6
Flash memory programming tool	Renesas Flash Programmer v3.14.00
Renesas Image Generator	Version 3.03 (bundled with Firmware Update Module Rev.2.02)
Key creation tool	Security Key Management Tool V.1.06

2. Preparation

Execution of the demo project described in this application note requires the software tools listed in the following table.

Table 2-1 List of Software Tools Required

Software Tool Name	Purpose
Tera Term	This tool is used to view the serial operation log of programs.
Python	This tool is used as an interpreter for running the Renesas Image Generator program.
OpenSSL	This tool is used to exchange keys for the TSIP and to generate keys for OTA update.
Gpg4win (Kleopatra)	This tool is used to perform PGP-based encryption for a UFPK that is used to wrap a key for the TSIP.
Cygwin	This tool is used to execute Bash scripts.
Renesas Image Generator	This tool is used to create a firmware image to be used for OTA update.
Security Key Management Tool	This is used to wrap a key for the TSIP.

Some of the above listed software tools are also used for AWS IoT OTA update. For the installation procedures of such software tools, refer to the corresponding sections in Chapter 2 in the following application note:
“RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037)
The following shows the relevant software tools and the sections describing the installation procedures:

- Tera Term: Section 2.1
- Python: Section 2.2
- OpenSSL: Section 2.3
- Renesas Image Generator: Section 2.4

Connection of the target board CK-RX65N v1 is described in section 2.5. When you connect the CK-RX65N v1, follow the procedure described in this section.

2.1 Installing Gpg4win (Kleopatra)

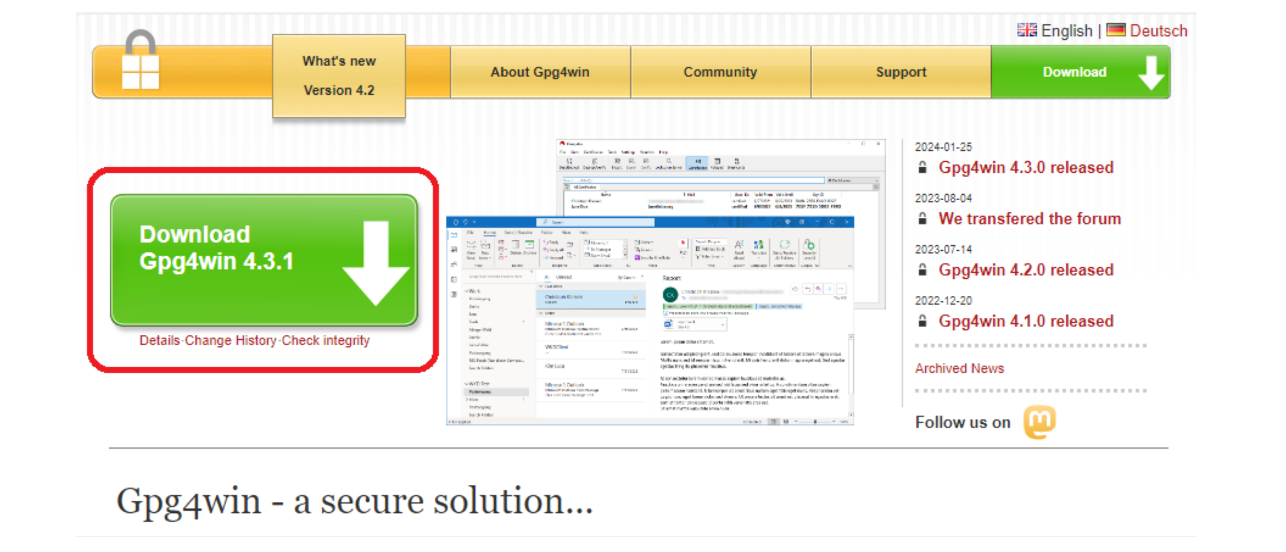
In this application note, Gpg4win (Kleopatra) is used in the procedure for encrypting/decrypting a UFPK by PGP to generate W-UFPK.

Install Gpg4win 4.3.1 by using the procedure described below.

(1) Downloading Gpg4win

Access the following website that provides GnuPG for Windows:
https://www.gpg4win.org/

Click the **Download** button shown in the following figure.



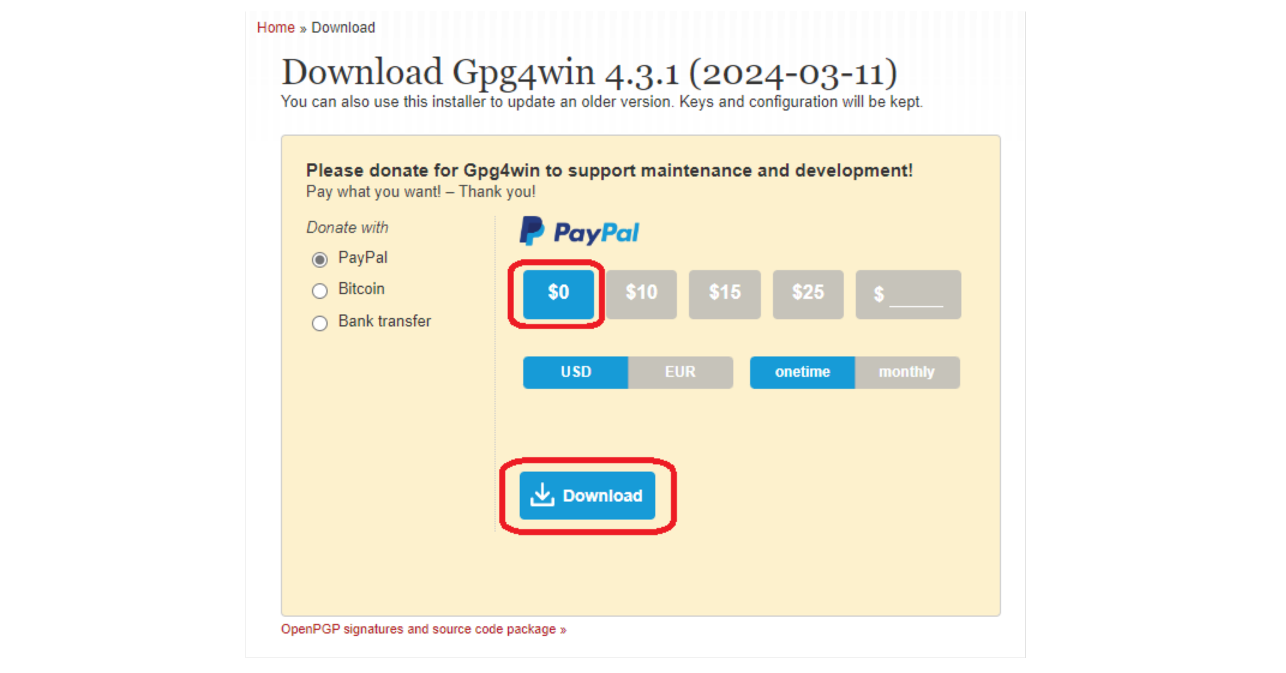
2024-01-25
Gpg4win 4.3.0 released
2023-08-04
We transfered the forum
2023-07-14
Gpg4win 4.2.0 released
2022-12-20
Gpg4win 4.1.0 released

Archived News

Gpg4win - a secure solution...

Figure 2-1 Downloading Gpg4win (1)

In the window that appears, select **$0**, and then click the **Download** button. The installation file is downloaded.



Download Gpg4win 4.3.1 (2024-03-11)

You can also use this installer to update an older version. Keys and configuration will be kept.

Please donate for Gpg4win to support maintenance and development!
Pay what you want! – Thank you!

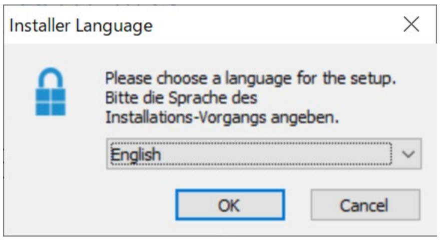
Donate with

● PayPal
○ Bitcoin
○ Bank transfer

Figure 2-2 Downloading Gpg4win (2)

(2) Executing the Gpg4win installation

Make sure that you have the installation file you obtained in section (1), and then run it as administrator. The installer starts. Click the **Next** button. When the following dialog box appears, select the language you want to use, and then click the **OK** button.



Installer Language

Please choose a language for the setup.
Bitte die Sprache des Installations-Vorgangs angeben.

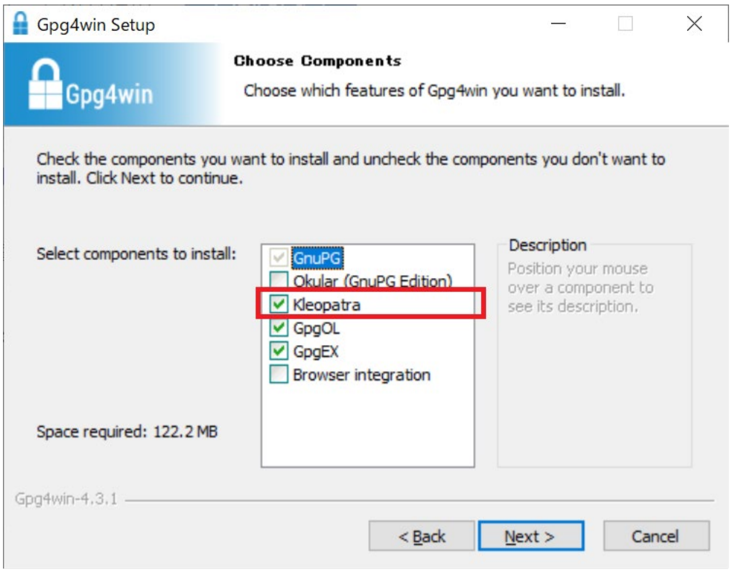
English

Figure 2-3 Installing Gpg4win (1)

(3) Selecting components

When the window for selecting the components to be installed appears as shown in the following figure, click **Next** without changing the initial settings.

In this application note, Kleopatra is used to manage key pairs. Therefore, do not clear the check box of Kleopatra.



Gpg4win Setup

Choose Components

Choose which features of Gpg4win you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ☑ GnuPG
- □ Okular (GnuPG Edition)
- ☑ Kleopatra
- ☑ GpgOL
- ☑ GpgEX
- □ Browser integration

Description
Position your mouse over a component to see its description.

Space required: 122.2 MB

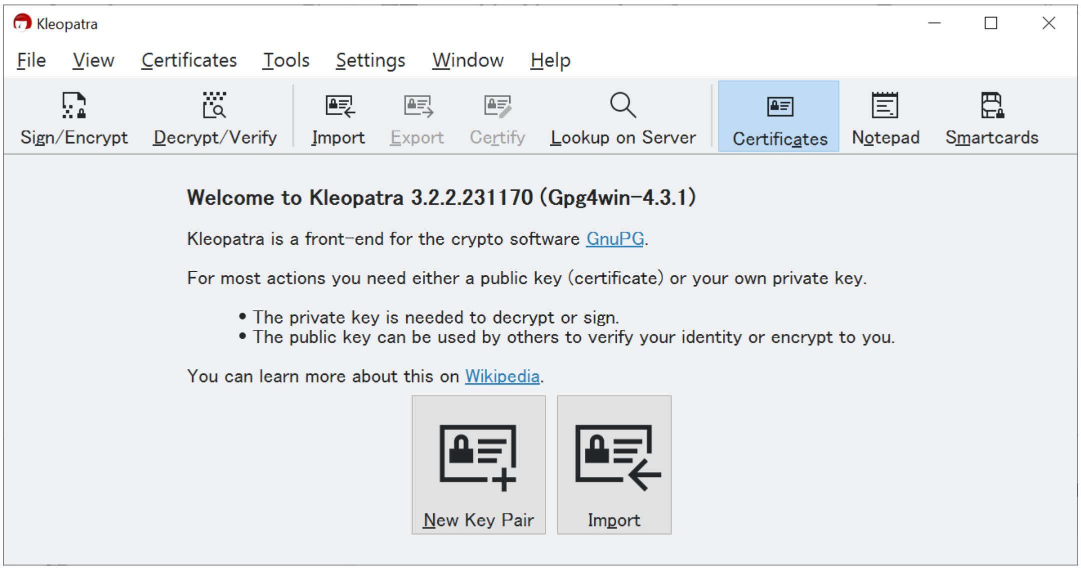
Gpg4win-4.3.1

Figure 2-4 Installing Gpg4win (2)

(4) Completing installation

In all windows that appear before installation finishes, click the **Next** button.

When installation is complete, Kleopatra starts.



Welcome to Kleopatra 3.2.2.231170 (Gpg4win-4.3.1)

Kleopatra is a front-end for the crypto software GnuPG.

For most actions you need either a public key (certificate) or your own private key.

- The private key is needed to decrypt or sign.
- The public key can be used by others to verify your identity or encrypt to you.

You can learn more about this on Wikipedia.

Figure 2-5 Window for Executing Kleopatra

2.2 Initial Setup of the Renesas Key Wrap Service and Kleopatra

The TSIP adopts a mechanism that prevents plaintext user keys from being exposed externally to protect user keys against leakage. This application note shows a method that uses the Renesas Key Wrap service to encrypt a UFPK (to generate W-UFPK).
This section describes the procedure for specifying the initial settings of the Renesas Key Wrap service. This section also describes the procedure for using Kleopatra to generate the PGP key that is used during UFPK encryption for exchanging files with the service and registering the generated key.

Use the following procedure to configure and register a key. The information created in this section is used in the procedure described in section 5.1.5, Wrapping Keys and Registering Them in the Project. For details on keys, refer also to section 5.1.5.

(1) Registering the Renesas Key Wrap service

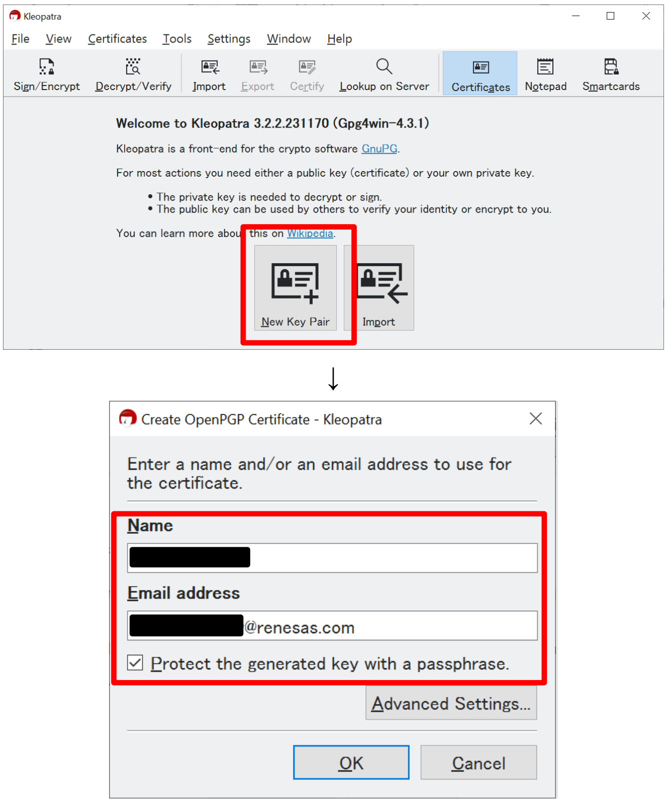
When you use the Renesas Key Wrap service for the first time, you must perform user registration and PGP key exchange. These tasks are required only the first time. Perform initial registration by logging in at the following URL:
Key Wrap Service Login (renesas.com)

For details on the Renesas Key Wrap service, refer to the following operation manual:
KeyWrap Service Operation Manual.pdf (renesas.com)

When you have completed user registration and the first-time PGP key exchange, log in to the Renesas Key Wrap service.

(2) Creating an OpenPGP key pair

Use Kleopatra to create a key pair in order to exchange public keys with the Renesas DLM server.
Start Kleopatra, and then, on the page that opens, click the **New Key Pair** button. The **Create OpenPGP Certificate** dialog box appears. You can create an OpenPGP certificate from this dialog box. Fill in the **Name** and **Email address** fields. You can optionally select the **Protect the generated key with a passphrase** check box to strengthen security by using a passphrase. Do not forget the passphrase if you set it.



Welcome to Kleopatra 3.2.2.231170 (Gpg4win-4.3.1)

Kleopatra is a front-end for the crypto software GnuPG.

For most actions you need either a public key (certificate) or your own private key.

- The private key is needed to decrypt or sign.
- The public key can be used by others to verify your identity or encrypt to you.

You can learn more about this on Wikipedia.

Enter a name and/or an email address to use for the certificate.

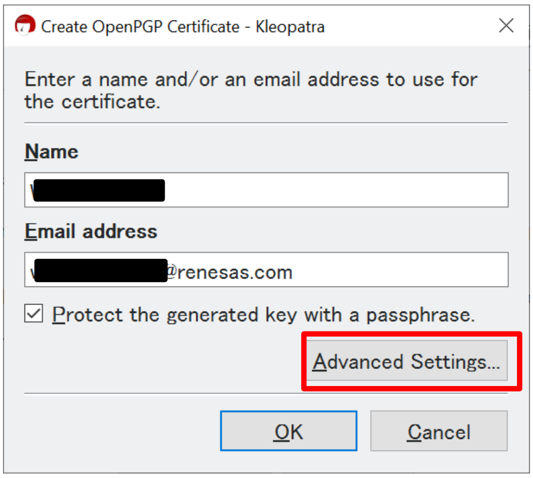
Name
[redacted]

Email address
[redacted]@renesas.com

☑ Protect the generated key with a passphrase.

Figure 2-6 Creating an OpenPGP Key Pair (1)

Then, click the **Advanced Settings** button.



Create OpenPGP Certificate - Kleopatra

Enter a name and/or an email address to use for the certificate.

Name

[redacted]

Email address

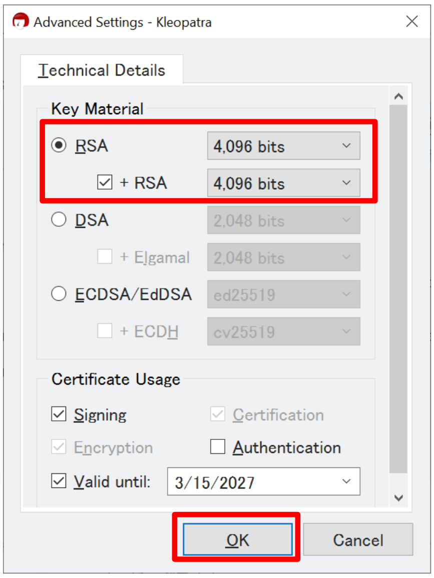
[redacted]@renesas.com

☑ Protect the generated key with a passphrase.

Figure 2-7 Creating an OpenPGP Key Pair (2)

In the **Advanced Settings** dialog box that appears, in the **Key Material** area, select the **RSA** radio button, and then select **4,096 bits**. Do not change the other setting items from their initial settings.
When you have completed the settings, click the **OK** button.

Note: Only RSA keys can be exchanged with the Renesas DLM server.



Advanced Settings - Kleopatra

Technical Details

Key Material

● RSA 4,096 bits

☑ + RSA 4,096 bits

○ DSA 2,048 bits

□ + Elgamal 2,048 bits

○ ECDSA/EdDSA ed25519

□ + ECDH cv25519

Certificate Usage

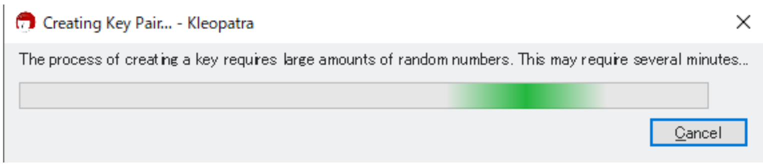
☑ Signing ☑ Certification

☑ Encryption □ Authentication

☑ Valid until: 3/15/2027

Figure 2-8 Creating an OpenPGP Key Pair - Advanced Settings

When the **Create OpenPGP Certificate** dialog box appears again, click the **OK** button. Then, the following dialog box appears, and generation of a key pair starts. Generation of a key pair takes some time.



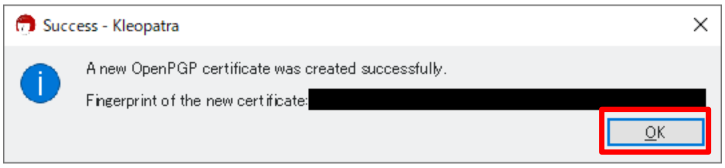
Creating Key Pair... - Kleopatra

The process of creating a key requires large amounts of random numbers. This may require several minutes...

Figure 2-9 Generating a PGP Key Pair

If you have set a passphrase, the window for entering the passphrase appears next. When this window appears, enter the passphrase.

When the following dialog box appears, a key pair has successfully been created. Click the **OK** button. The Kleopatra main window appears again.

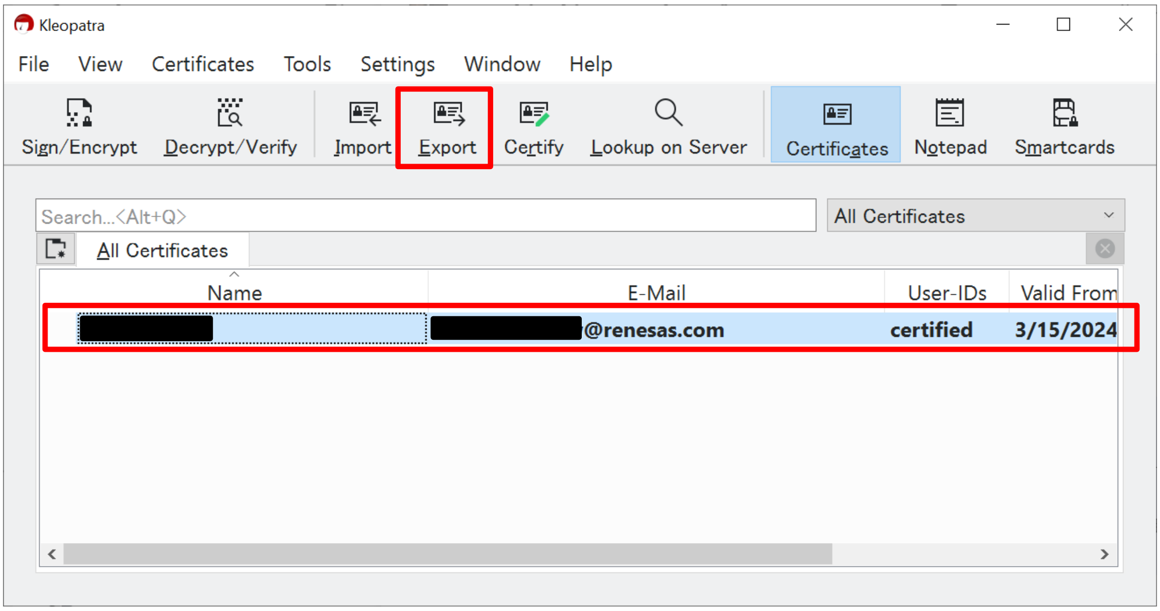


A new OpenPGP certificate was created successfully.

Fingerprint of the new certificate: [redacted]

Figure 2-10 Dialog Box Appearing When a Key Pair Was Created Successfully

The information about created key pairs is registered in the Kleopatra window. Select a key pair, and then click the **Export** button to output the OpenPGP public key. The key pair is saved in a file with the extension ".asc".

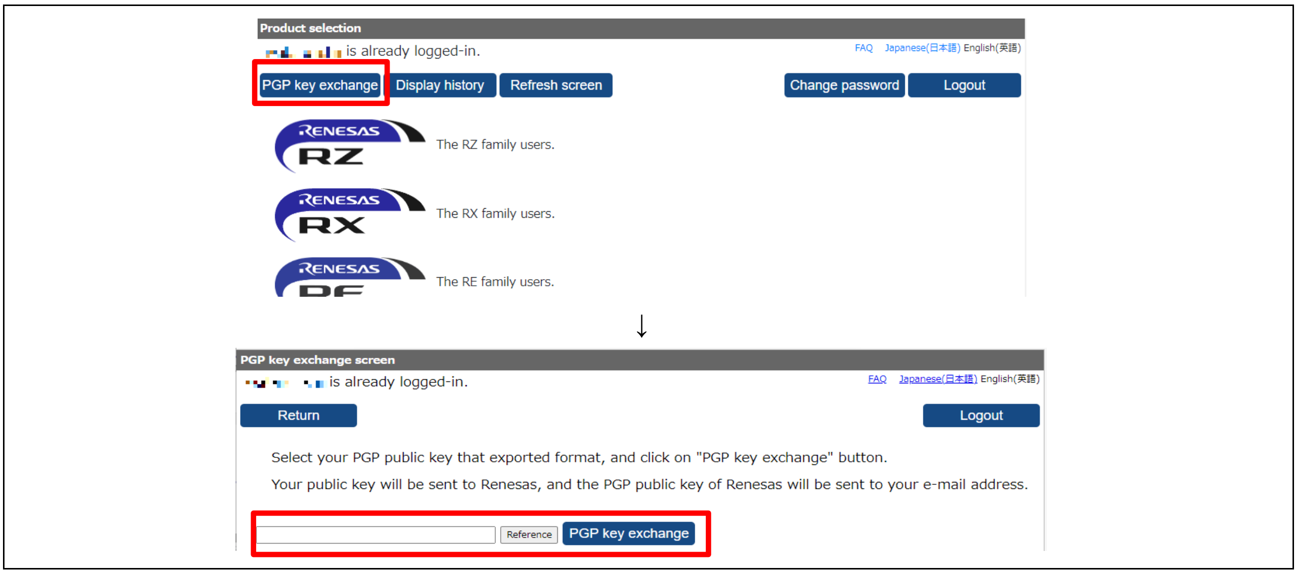


Name	E-Mail	User-IDs	Valid From
[redacted]	[redacted]@renesas.com	certified	3/15/2024

Figure 2-11 Registered OpenPGP Key Pair

(3) Exchanging PGP public key with the Renesas DLM server

Now you have an OpenPGP public key that was output in section (2). Exchange it for Renesas' PGP public key by using the Renesas DLM server. Access the Renesas Key Wrap Service website that you registered in section (1), and then click **PGP key exchange** to register the OpenPGP public key that you created.
When registration is completed successfully, a Renesas PGP public key (**keywrap-pub.key**) is sent to your email address. When you receive it, save it in any folder of your choice.



[redacted] is already logged-in.

The RZ family users.

The RX family users.

The RE family users.

[redacted] is already logged-in.

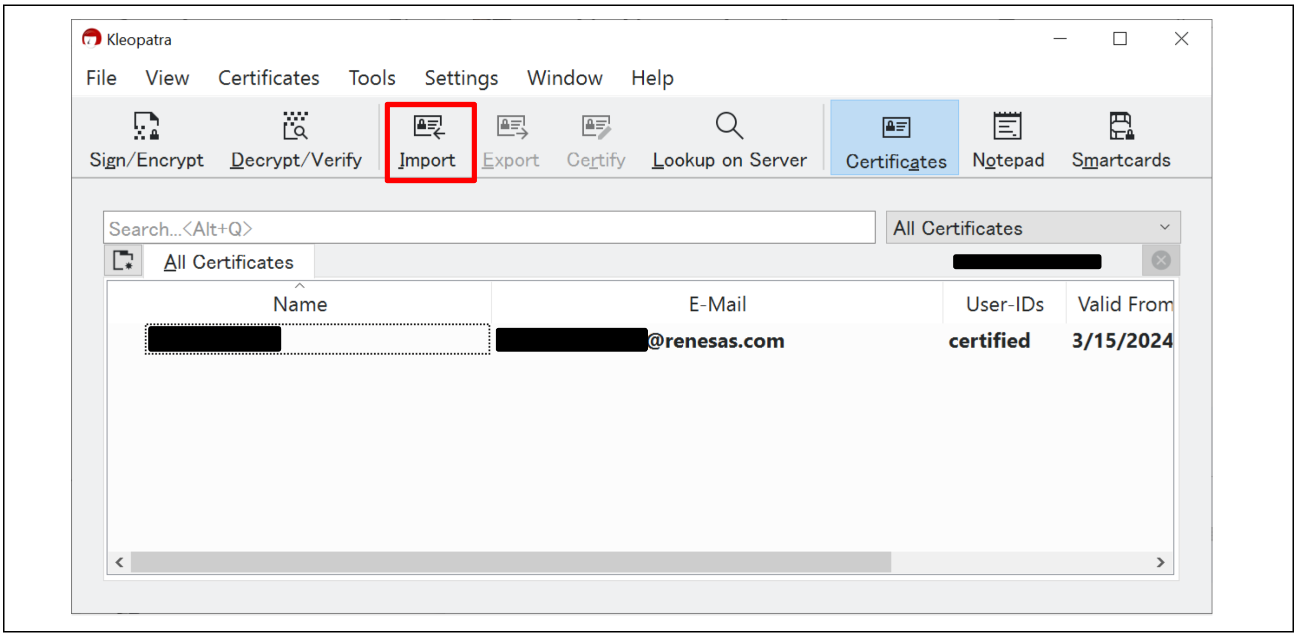
Select your PGP public key that exported format, and click on "PGP key exchange" button.

Your public key will be sent to Renesas, and the PGP public key of Renesas will be sent to your e-mail address.

Figure 2-12 Exchanging PGP public Key with the Renesas DLM Server

(4) Registering the Renesas OpenPGP public key

The Renesas PGP public key is used to decrypt the key encrypted by PGP on the Renesas DLM server.
In Kleopatra, register the Renesas PGP public key (**keywrap-pub.key**) that you received by email.
In the Kleopatra window, click **Import**.

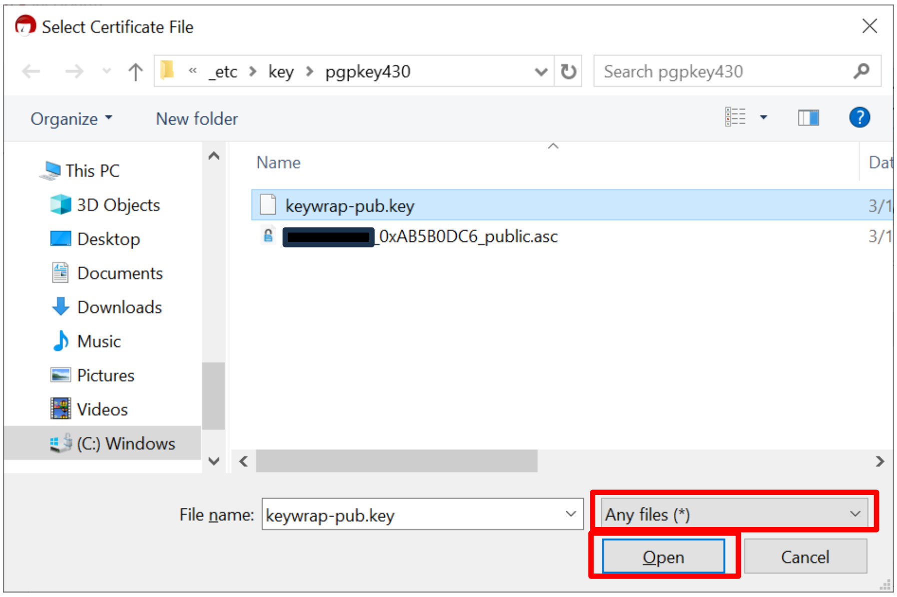


[redacted]

Name	E-Mail	User-IDs	Valid From
[redacted]	[redacted]@renesas.com	certified	3/15/2024

Figure 2-13 Registering the Renesas OpenPGP Public Key

When the **Select Certificate File** dialog box appears, select **Any files (*)** as the file extension, specify **keywrap-pub.key**, which is the PGP public key sent from Renesas, and then click the **Open** button.



keywrap-pub.key 3/1

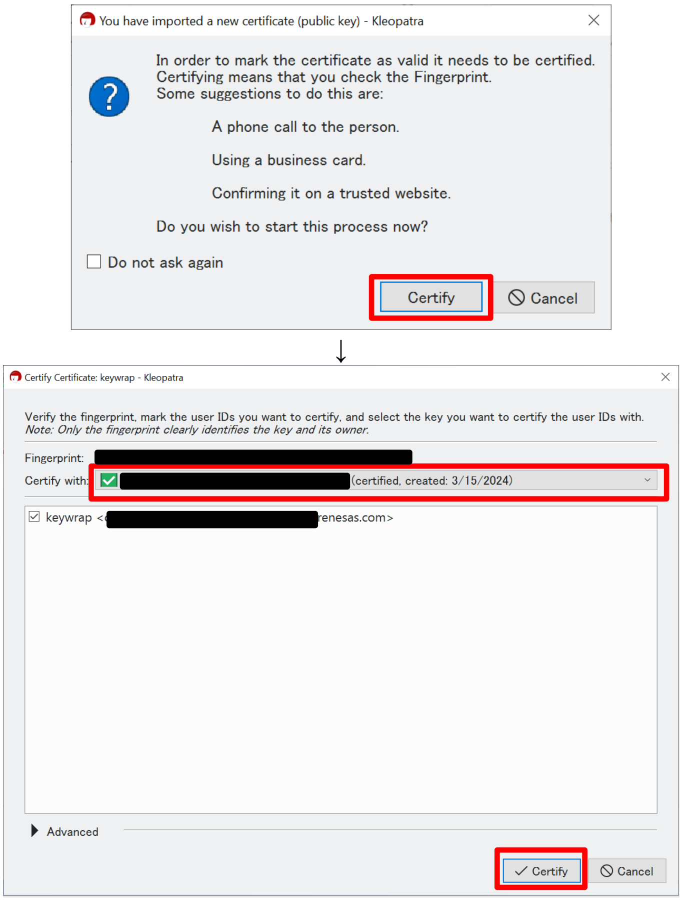
[redacted]_0xAB5B0DC6_public.asc 3/1

File name: keywrap-pub.key

Any files (*)

Figure 2-14 Specifying the OpenPGP Public Key Sent from Renesas

If the **You have imported a new certificate (public key)** dialog box shown in the following figure appears, click the **Certify** button. In the **Certify Certification** dialog box that appears, confirm that the certificate that you registered yourself is selected, and then click the **Certify** button.
If you have registered a passphrase in section (2), enter the passphrase.



You have imported a new certificate (public key) - Kleopatra

In order to mark the certificate as valid it needs to be certified.
Certifying means that you check the Fingerprint.
Some suggestions to do this are:

A phone call to the person.

Using a business card.

Confirming it on a trusted website.

Do you wish to start this process now?

□ Do not ask again

Certify Certificate: keywrap - Kleopatra

Verify the fingerprint, mark the user IDs you want to certify, and select the key you want to certify the user IDs with.
Note: Only the fingerprint clearly identifies the key and its owner.

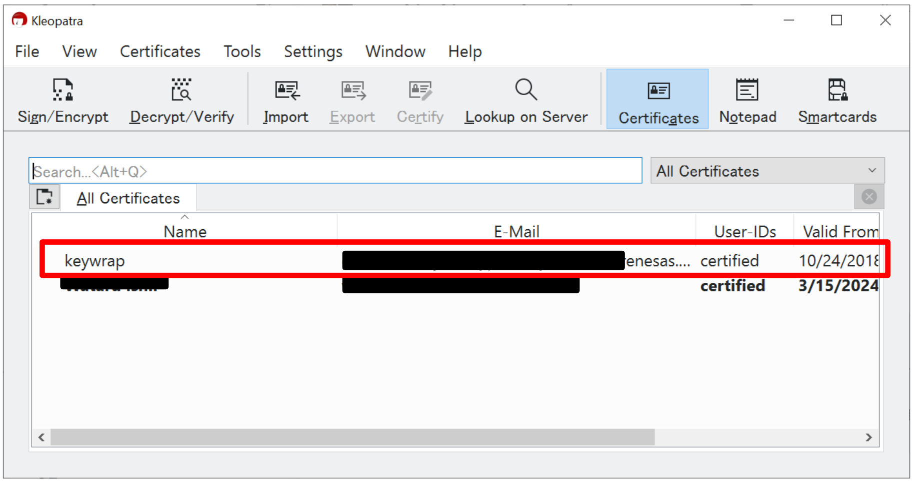
Fingerprint: [redacted]

Certify with: ☑ [redacted] (certified, created: 3/15/2024)

☑ keywrap <[redacted]renesas.com>

Figure 2-15 Importing the OpenPGP Public Key

When the **Certification successful** message box appears, close it by clicking the **OK** button. Confirm that an entry named “keywrap” has been added. If “certified” is displayed in the **User-IDs** column, import is completed.



Name	E-Mail	User-IDs	Valid From
keywrap	[redacted]enesas....	certified	10/24/201
[redacted]	[redacted]	**certified**	**3/15/2024**

Figure 2-16 Registered OpenPGP Public Key

2.3 Installing Cygwin

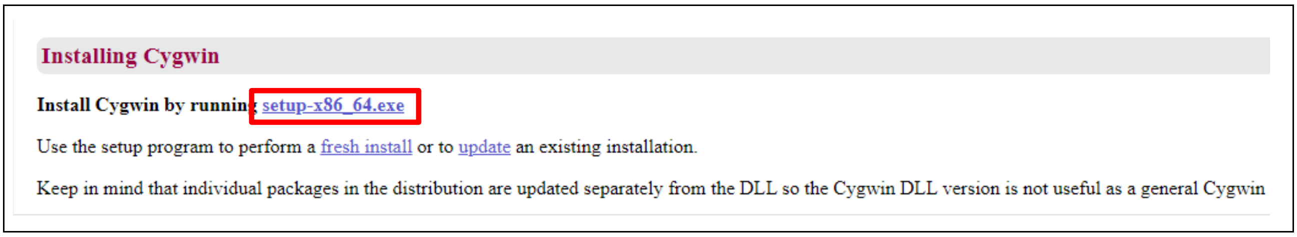
The application described in this application note uses Bash scripts when registering certificates and keys in a project (source code). Cygwin is used as the script execution environment.

Use the following procedure to install Cygwin.

(1) Access the Cygwin download website.

[Cygwin download website]

(2) In the download website, click the following link to download the installer.



Installing Cygwin

Install Cygwin by running setup-x86_64.exe

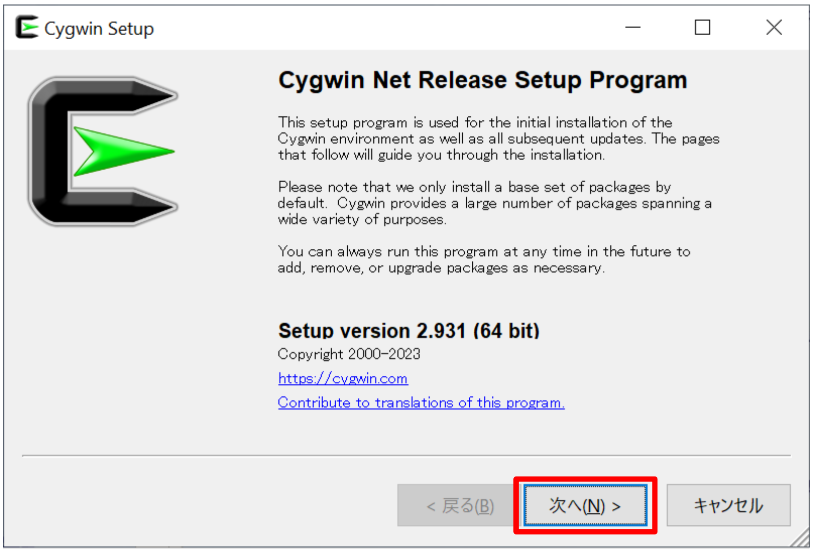
Use the setup program to perform a fresh install or to update an existing installation.

Keep in mind that individual packages in the distribution are updated separately from the DLL so the Cygwin DLL version is not useful as a general Cygwin

Figure 2-17 Downloading Cygwin

(3) Start the installer, and install Cygwin as instructed by the installer.

During installation, accept all initial settings in principle except when selecting packages. If necessary packages are missing, obtain them.



Cygwin Net Release Setup Program

This setup program is used for the initial installation of the Cygwin environment as well as all subsequent updates. The pages that follow will guide you through the installation.

Please note that we only install a base set of packages by default. Cygwin provides a large number of packages spanning a wide variety of purposes.

You can always run this program at any time in the future to add, remove, or upgrade packages as necessary.

Setup version 2.931 (64 bit)

Copyright 2000-2023

https://cygwin.com

Contribute to translations of this program.

Figure 2-18 Installing Cygwin

(4) In the **Start** menu, click the **Cygwin65 Terminal** icon, and then confirm that the Cygwin terminal screen is displayed.

2.4 Installing Security Key Management Tool

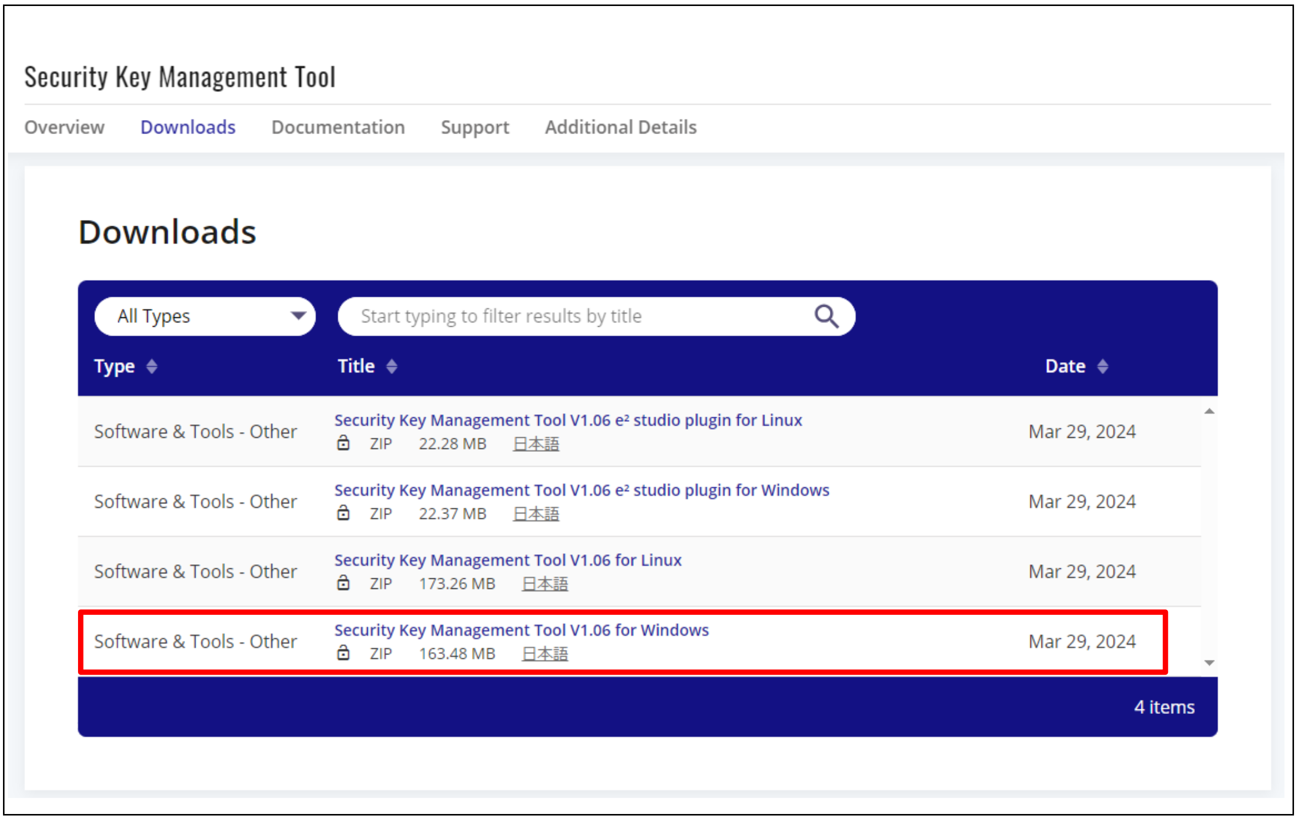
The application described in this application note uses Security Key Management Tool to convert key information so that it can be used with the TSIP.

Use the following procedure to install Security Key Management Tool.

(1) Download Security Key Management Tool

Access the Security Key Management Tool download website, and download the latest version of Security Key management Tool for Windows.

In the following example, version 1.06 is the latest version of Security Key Management Tool.



Type	Title	Date
Software & Tools - Other	Security Key Management Tool V1.06 e² studio plugin for Linux 🔒 ZIP 22.28 MB 日本語	Mar 29, 2024
Software & Tools - Other	Security Key Management Tool V1.06 e² studio plugin for Windows 🔒 ZIP 22.37 MB 日本語	Mar 29, 2024
Software & Tools - Other	Security Key Management Tool V1.06 for Linux 🔒 ZIP 173.26 MB 日本語	Mar 29, 2024
Software & Tools - Other	Security Key Management Tool V1.06 for Windows 🔒 ZIP 163.48 MB 日本語	Mar 29, 2024

4 items

Figure 2-19 Downloading Security Key Management Tool

(2) When the download is completed, start the installer, and install Security Key Management Tool as instructed by the installer.

(3) After installation is completed, confirm that Security Key Management Tool can be started from the **Start** menu.

3. AWS Setup

To perform demonstration of OTA update as in this application note, you must have an account for connecting to AWS. This account is a root user or an IAM user authorized to access AWS IoT and the FreeRTOS cloud service.

For details on the AWS setup procedure, refer to the following application note: "RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)" (R01AN7037). The information on the following AWS webpage will also be helpful:

- "Setting up your AWS account and permissions"
  https://docs.aws.amazon.com/freertos/latest/userguide/freertos-prereqs.html

For the demo application described in this application note to be able to communicate with AWS, the source code must be modified. For details on modifying the source code, refer to Chapter 4, Preparing for the Demo Project and subsequent sections.

3.1 Settings That Must Be Specified from the AWS Console

Log in to AWS Console, and then specify the initial settings and other necessary settings. To do so, perform the procedure described in Chapter 3 in the following application note: "RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)" (R01AN7037).

The following table lists the AWS settings that must be specified to set up the source code of the project in this application note.

Table 3-1 List of Settings Required for AWS

Name in AWS	Description	Remarks
Things*1	Register the name of a device (thing) to be connected to AWS.	Record the name of the thing that you set. Refer to: 3.3.2(3)
Endpoint*1	Register the connection destination (URL) in AWS.	Record the endpoint name that you set. Refer to: 3.3.3(1)
Device certificate	A client certificate used for connection to AWS. In this application note, this item is called a "client certificate".	Download it from AWS and save it.*2 Refer to: 3.3.2(6)
Public key file	A public key used for connection to AWS. In this application note, this item is called a "client certificate public key".	Download it from AWS and save it.*2 Refer to: 3.3.2(6)
Private key file	A private key used for connection to AWS. In this application note, this item is called a "client certificate private key".	Download it from AWS and save it.*2 Refer to: 3.3.2(6)
Root CA certificate*3	A root CA certificate used for connection to AWS.	Download it from AWS and save it.

Notes: 1. The thing and endpoint names must be registered in the project that is executed later. Record the registered names.

2. Note that you can download the public and private keys for the client certificate only when registering devices in AWS.

3. The procedure for downloading a root CA certificate is not covered in the following application note: "RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)" (R01AN7037). The download procedure is described in section 5.1.2 in this application note.

4. Preparing for the Demo Project

This chapter describes how to create the project used for demonstration.

The CK-RX65N v1 board described in this application note is the cellular version. The RYZ014A board is bundled with the CK-RX65N v1 board. Connect the RYZ014A board to the PMOD1 pin of the CK-RX65N v1 so that the CK-RX65N v1 board can connect to a mobile network.

For details on connection, refer to section 2.5 in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037).

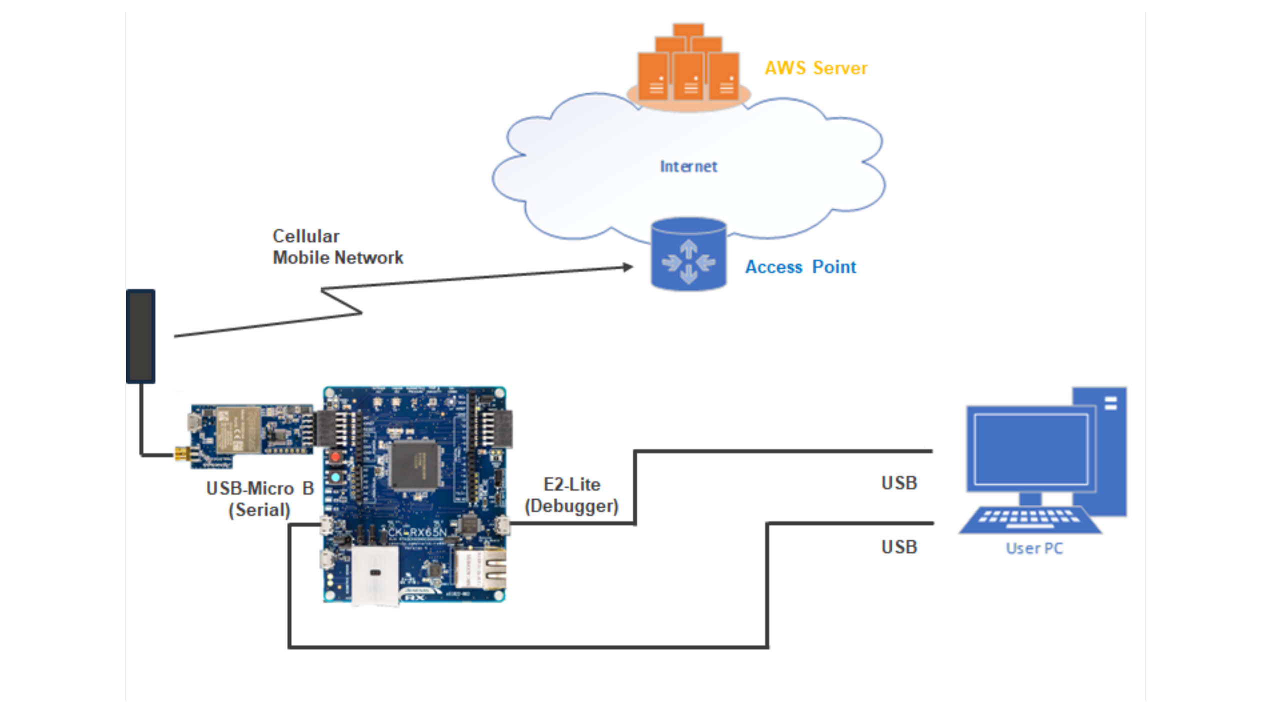


Figure 4-1 Overview of Connections in the Demo Project

The Demo project is based on the FreeRTOS project. FreeRTOS provides IoT Libraries, which contain source code necessary for IoT devices. Mbed TLS in these libraries is used as an open-source cryptographic library.

In this Demo project, the processing in the Mbed TLS library is partially replaced by TSIP driver's API functions related to TLS communication. The following figure shows the software structure of the Demo project.

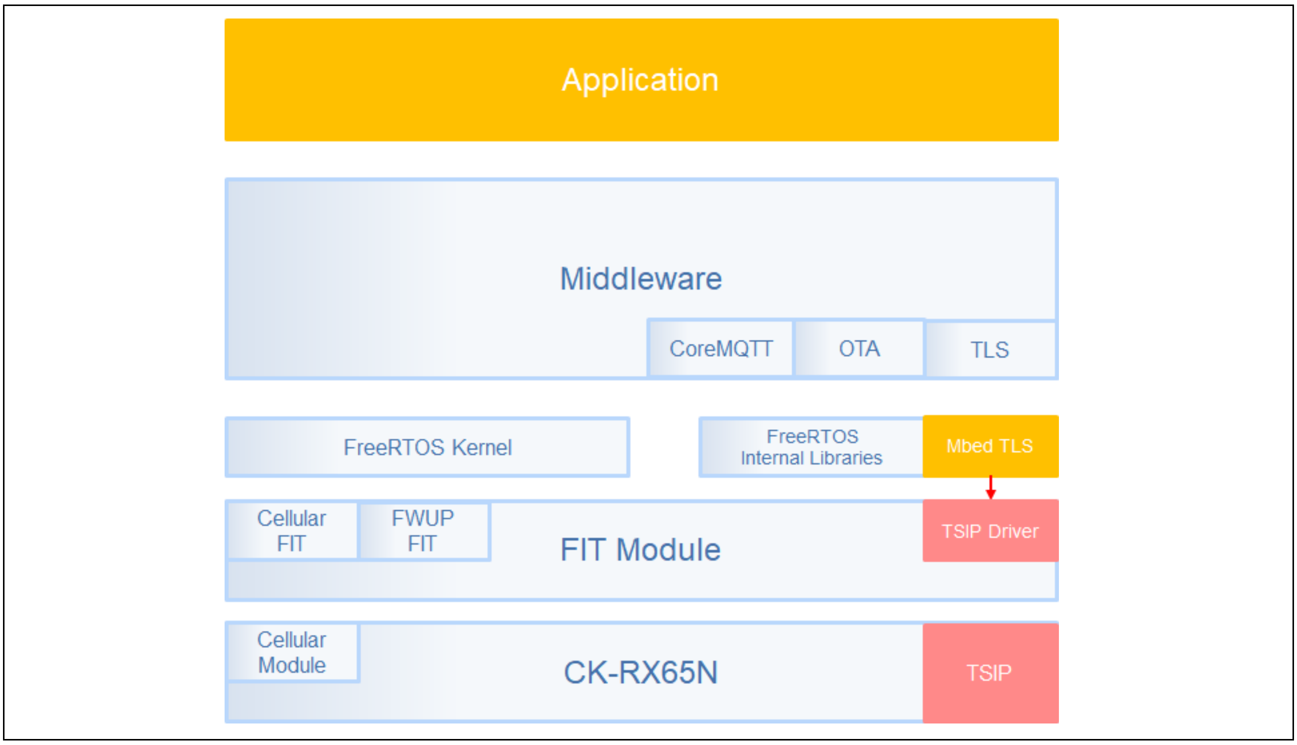


Figure 4-2   Software Structure of the Demo Project

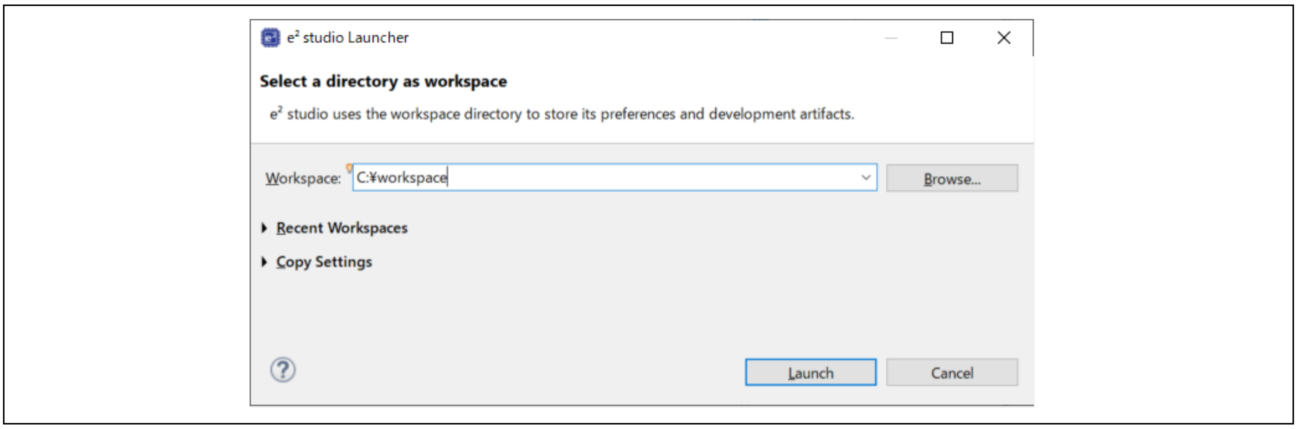
The Demo project uses version 1.3.0 of the FreeRTOS project for the RX Family MCU provided by the following GitHub repository.

https://github.com/renesas/iot-reference-rx

4.1 Creating a Workspace

Start e^2 studio and create a new workspace.
Make sure that the length of the path name (including the folder name) of the workspace does not exceed 35 characters. This is due to restrictions in e^2 studio. Specifying a path name longer than 35 characters causes an error when building a project. Also make sure that the path name consists of only ASCII characters.

The following figure shows an example of creating “C:\workspace” as a new workspace.



Select a directory as workspace

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

Workspace: C:¥workspace

Figure 4-3 Dialog Box for Creating a Workspace

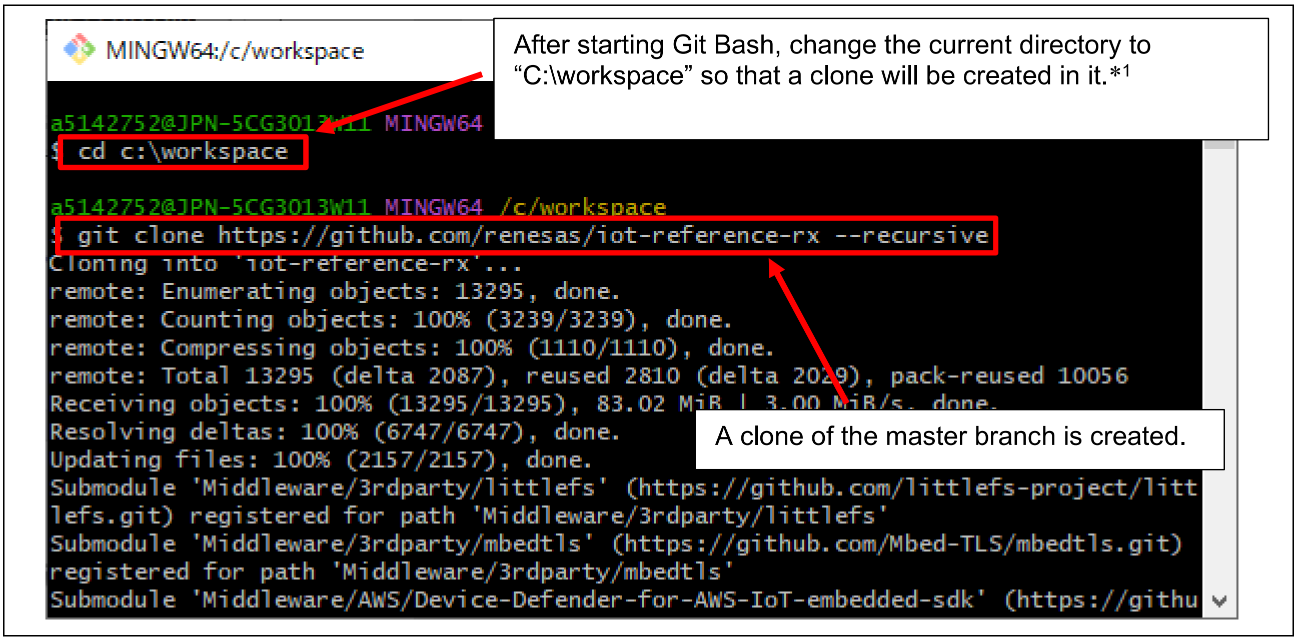
4.2 Downloading the Demo Project

(1) Cloning the demo project

Clone the demo project from GitHub (iot-reference-rx: FreeRTOS reference repository). This document describes the cloning procedure when Git for Windows is used.

Start GitBush, and then execute the following command:

```
cd c:\workspace
git clone https://github.com/renesas/iot-reference-rx --recursive
```



After starting Git Bash, change the current directory to “C:\workspace” so that a clone will be created in it.*1

```
a5142752@JPN-5CG3013W11 MINGW64
$ cd c:\workspace

a5142752@JPN-5CG3013W11 MINGW64 /c/workspace
$ git clone https://github.com/renesas/iot-reference-rx --recursive
Cloning into 'iot-reference-rx'...
remote: Enumerating objects: 13295, done.
remote: Counting objects: 100% (3239/3239), done.
remote: Compressing objects: 100% (1110/1110), done.
remote: Total 13295 (delta 2087), reused 2810 (delta 2029), pack-reused 10056
Receiving objects: 100% (13295/13295), 83.02 MiB | 3.00 MiB/s, done.
Resolving deltas: 100% (6747/6747), done.
Updating files: 100% (2157/2157), done.
Submodule 'Middleware/3rdparty/littlefs' (https://github.com/littlefs-project/litt
lefs.git) registered for path 'Middleware/3rdparty/littlefs'
Submodule 'Middleware/3rdparty/mbedtls' (https://github.com/Mbed-TLS/mbedtls.git)
registered for path 'Middleware/3rdparty/mbedtls'
Submodule 'Middleware/AWS/Device-Defender-for-AWS-IoT-embedded-sdk' (https://githu
```

A clone of the master branch is created.

Figure 4-4 Cloning the Demo Project

Note: 1. Make sure that the length of the path name (including the folder name) of the cloning destination does not exceed 35 characters. This is due to restrictions in e^2 studio. Specifying a path name longer than 35 characters causes an error when building a project.
In the above example, a clone is created in “C:\workspace”.

(2) Folder structure

The demo project downloaded from GitHub has the folder structure shown below.
The shown structure covers only important folders.

The items indicated in red font are either files and folders that have been added to the standard project or files and folders that have been modified in the standard project for use with the TSIP driver. To check the differences from the standard project in details, use a Diff tool.

```
iot-reference-rx
|--Common
|  |--common_api/r_common_api_tsip.c/h
|--Demos
|  |--key_flash_wr_with_tsip
|--IDT_config
|--Middleware
|  |--mbedtls_config/aws_mbedtls_config_with_tsip.h
|  |--mbedtls_with_TSIP
|  |--network_transport/using_mbedtls_pkcs11_with_tsip
|--Project
|  |--aws_ether_tsip_ck_rx65n
|  |--aws_ryz014a_tsip_ck_rx65n
|  |  |--e2studio_ccrx
|  |  |  |--src
|  |  |     |--application_code/main.c
|  |  |     |--frtos_startup/freertos_start.c
|  |  |     |--userdata_tsip
|  |  |--flash_project
|  |  |--key_crt_sig_generator
|--boot_loader_ck_rx65n
|  |--e2studio_ccrx
|  |  |--src
|--Test
|--Tools
|--Getting_Started_Guide.md
|--README.md
```

Figure 4-5 Folder Structure of the Demo Project

Major changes made to the standard project (so that the project can be used with the TSIP driver) are as follows:

- The processing of FreeRTOS and Mbed TLS was partially replaced by API functions for the TSIP driver. Also, code necessary for this replacement was added.
- Exclusive access control was added to prevent an access contention for the TSIP driver in a multi-task environment.
- The files and folders in which to save certificates and their signatures were added.
- Processing to write added key data to data flash memory was added.
- A folder was added to store the settings files for user keys and certificates used for connection to the TSIP driver.

The downloaded folders store not only the source code of the demo project, but also tools for configuring the demo project. The following table outlines these folders.

Table 4-1 Content of the demo Project

Folder name	Description
Common Demos IDT_config Middleware Test Tool	These folders store common code used in each project and modules such as libraries. Links to these folders are created in each project as needed.
aws_ryz014a_tsip_ck_rx65n	This folder stores the cellular connection version of project that is compatible with the TSIP used in this application note. This folder is imported by using the procedure described below.
aws_ether_tsip_ck_rx65n	This folder stores the Ethernet connection version of project that is compatible with the TSIP. This project can be used using the same procedure as the cellular version of project.
boot_loader_ck_rx65n	This folder stores the bootloader project for the CK-RX65N v1 board used in this application note. This folder is imported by using the procedure described below.
flash_project	This folder stores the project files for Renesas Flash Programmer that is used to write executable files to the CK-RX65N when performing an OTA update.
key_crt_sig_generator	This folder stores the tools for generating keys and certificates used for encryption. This folder also stores the work folders for those tools.

4.3 Importing a Project

After you have downloaded a project, import it by using the following procedure.

(1) Start e^2 studio.

(2) From the **File** menu, select **Import** to open the **Import** dialog box.

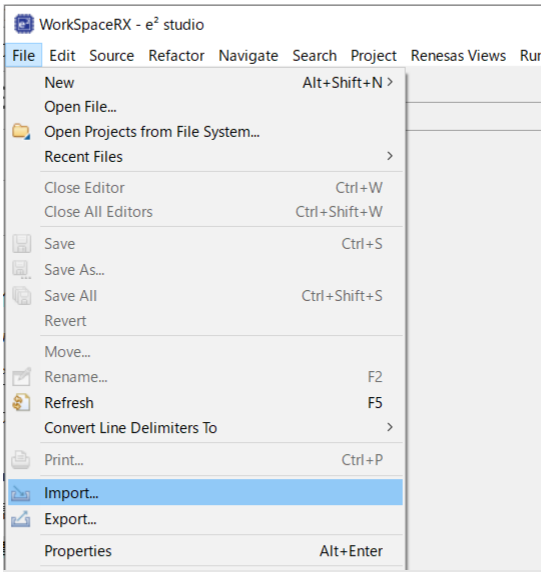


Figure 4-6 Opening the Import Dialog Box

(3) Select **Existing Projects into Workspace**, and then click the **Next** button.

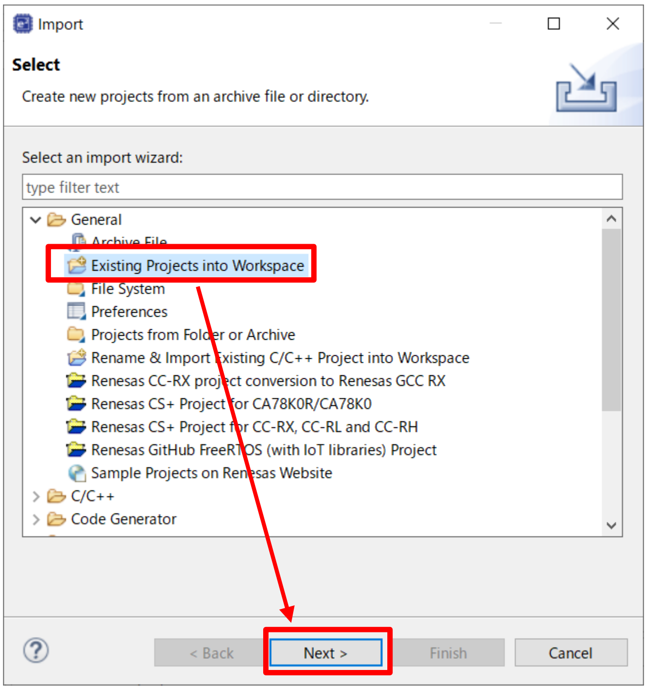
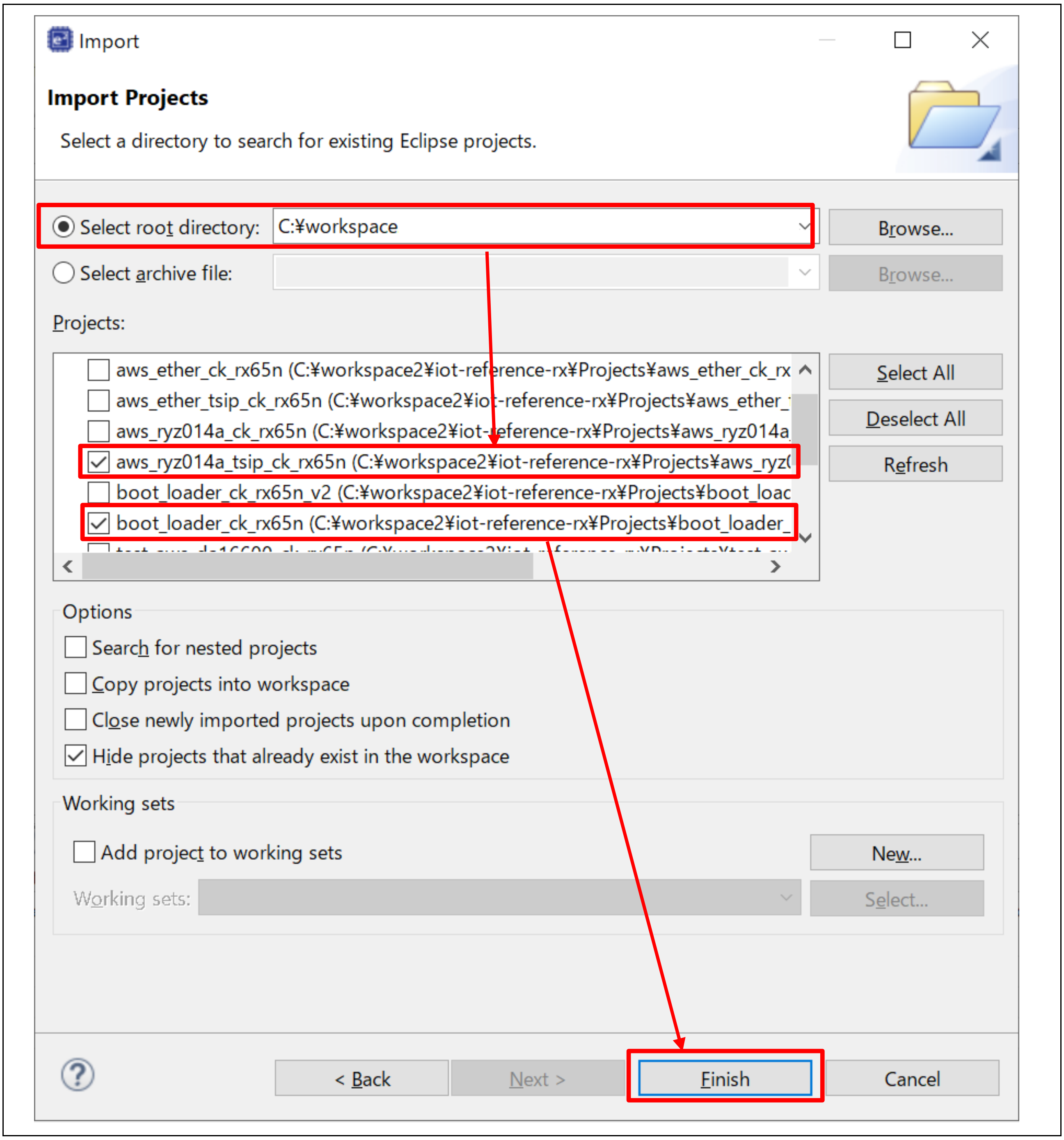

Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Archive File
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Project into Workspace
Renesas CC-RX project conversion to Renesas GCC RX
Renesas CS+ Project for CA78K0R/CA78K0
Renesas CS+ Project for CC-RX, CC-RL and CC-RH
Renesas GitHub FreeRTOS (with IoT libraries) Project
Sample Projects on Renesas Website
C/C++
Code Generator
< Back
Next >
Finish
Cancel

Figure 4-7 Importing an Existing Project into a Workspace

(4) Select the **Select root directory** radio button, select the folder in which you created a clone in section 4.2(1), select the two projects as shown in the following figure, and then click the **Finish** button.

- aws_ryz014a_tsip_ck_rx65
- boot_loader_ck_rx65n



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:¥workspace [Browse...]

○ Select archive file: [Browse...]

Projects:

- □ aws_ether_ck_rx65n (C:¥workspace2¥iot-reference-rx¥Projects¥aws_ether_ck_rx
- □ aws_ether_tsip_ck_rx65n (C:¥workspace2¥iot-reference-rx¥Projects¥aws_ether_
- □ aws_ryz014a_ck_rx65n (C:¥workspace2¥iot-reference-rx¥Projects¥aws_ryz014a
- ☑ aws_ryz014a_tsip_ck_rx65n (C:¥workspace2¥iot-reference-rx¥Projects¥aws_ryz(
- □ boot_loader_ck_rx65n_v2 (C:¥workspace2¥iot-reference-rx¥Projects¥boot_loac
- ☑ boot_loader_ck_rx65n (C:¥workspace2¥iot-reference-rx¥Projects¥boot_loader_

Options

- □ Search for nested projects
- □ Copy projects into workspace
- □ Close newly imported projects upon completion
- ☑ Hide projects that already exist in the workspace

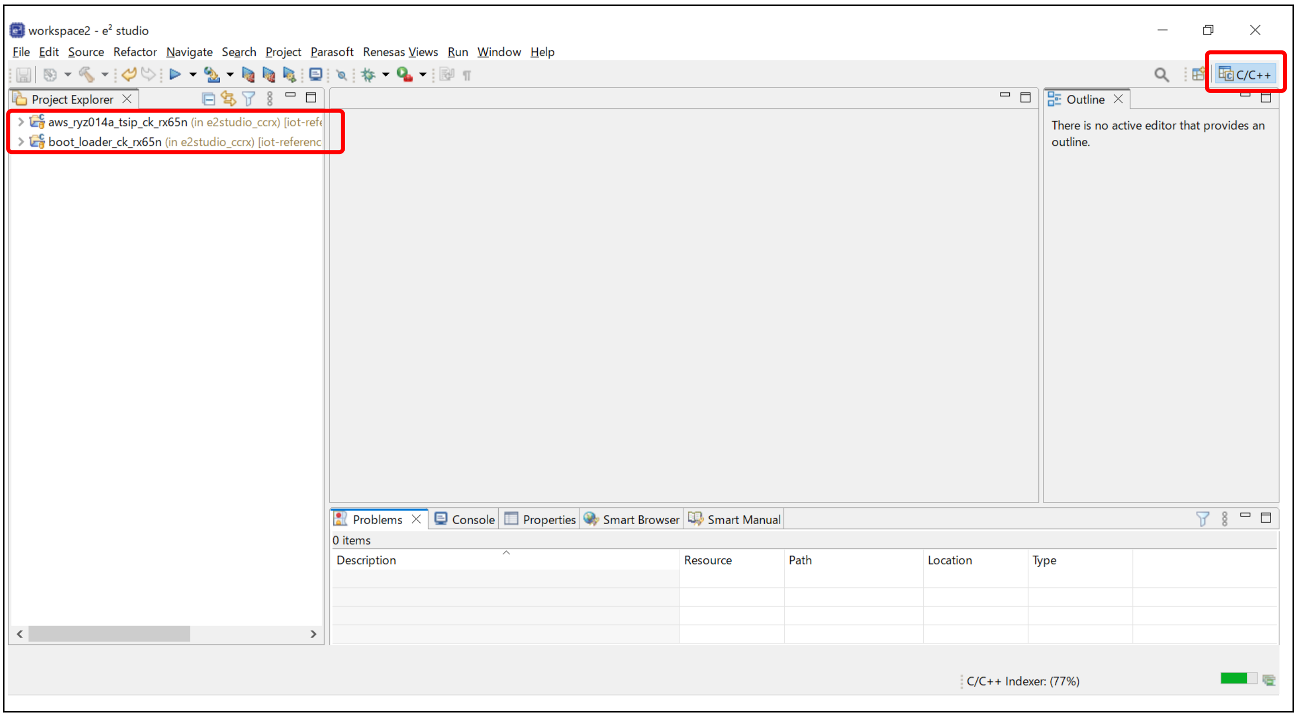
Working sets

- □ Add project to working sets

Working sets:

Figure 4-8 Importing the Projects of the Bootloader and Application

(5) When the projects are imported successfully, “aws_ryz014a_tsip_ck_rx65” and “boot_loader_ck_rx65n”, which are imported projects, are added to the Project Explorer view as shown below.
If the Project Explorer view is not displayed, click the **C/C++** perspective at the top right of the window, and then select **Window** > **Show View** > **Project Explorer**.


workspace2 - e² studio
File Edit Source Refactor Navigate Search Project Parasoft Renesas Views Run Window Help
C/C++
Project Explorer
aws_ryz014a_tsip_ck_rx65n (in e2studio_ccrx) [iot-refe
boot_loader_ck_rx65n (in e2studio_ccrx) [iot-referenc
Outline
Problems
Console
Properties
Smart Browser
Smart Manual
C/C++ Indexer: (77%)

There is no active editor that provides an outline.

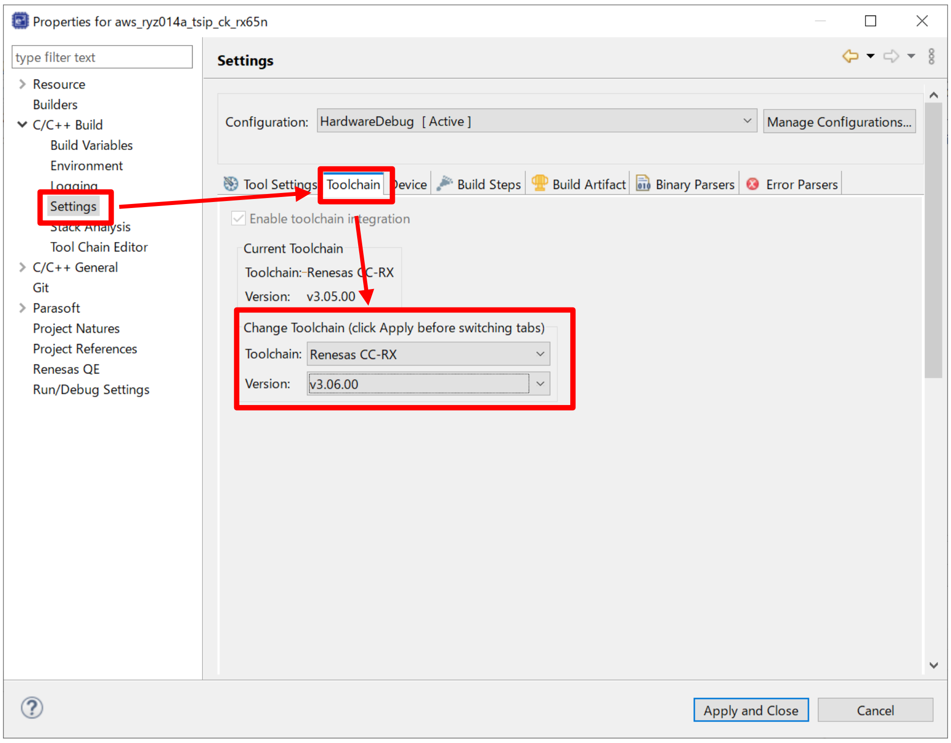
0 items

Description	Resource	Path	Location	Type

Figure 4-9 Window after the Projects Are Imported

(6) Checking the project environment settings

For the imported two projects, confirm that “Renesas CC-RX” is set as the toolchain. To do so, in the menu, select **Project** > **Properties** > **C/C++Build** > **Settings**, and then click the **Toolchain** tab.



Configuration: HardwareDebug [Active]

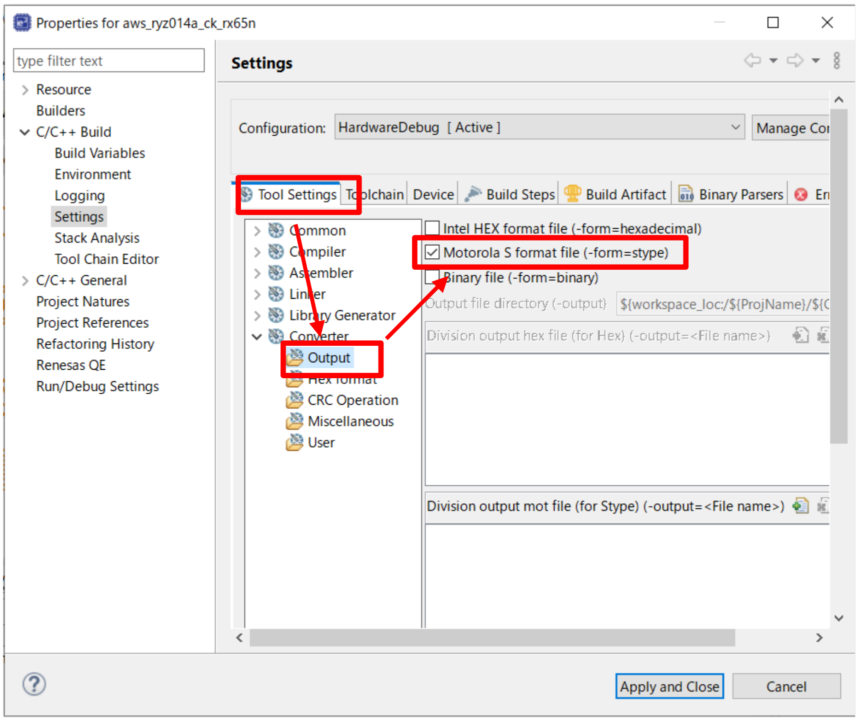
☑ Enable toolchain integration

Current Toolchain
Toolchain: Renesas CC-RX
Version: v3.05.00

Change Toolchain (click Apply before switching tabs)
Toolchain: Renesas CC-RX
Version: v3.06.00

Figure 4-10 Checking the Toolchain

Next, click the **Tool Settings** tab, select **Converter** > **Output**, and then confirm that the **Motorola S format file** check box is selected.



Configuration: HardwareDebug [Active]

□ Intel HEX format file (-form=hexadecimal)
☑ Motorola S format file (-form=stype)
□ Binary file (-form=binary)
Output file directory (-output) ${workspace_loc:/${ProjName}/${C
Division output hex file (for Hex) (-output=<File name>)
Division output mot file (for Stype) (-output=<File name>)

Figure 4-11 Checking the Output Format

5. Creating Keys and Certificates

For the project described in this document, you must create multiple keys and certificates in order to establish a TLS connection by using the TSIP and execute an OTA update.

After you have created the data of keys and certificates, register the data in the project that you imported into e^2 studio in Chapter 4, Preparing for the Demo Project. The procedure for creating the data of keys and certificates is described below.

5.1 Preparing the Keys and Certificates for the TSIP

To perform TLS communication by using the TSIP in this sample program, you must register the information about the keys and certificates listed in Table 5-1.
This section describes how to obtain these keys and certificates. This section also describes the procedure for converting them for use with the TSIP and the procedure for registering them in the sample application project.
Generate these keys and certificates and register them in the sample application project as instructed in this section.
The table below provides brief explanations on the necessary items and how to obtain them. For the flows of creating the keys and certificates listed in Table 5-1, refer to Figure 5-1, Flows of Creating Keys and Certificates for Use with the TSIP.

Table 5-1 Keys and Certificates Used in the Sample Application Project and How to Obtain Them

Key/certificate	**How to obtain the item**	**How to install the item**
Root CA certificate	The user downloads this item from AWS.	Use CLI.*1, *2
Root CA certificate signature data	The user creates this item by using OpenSSL or a similar tool.*6	Place the file in the appropriate folder in the project.
Root CA certificate public key	The user creates this item by using OpenSSL or a similar tool.*6 The user then wraps the created data.*4 The wrapped data is used as the root CA certificate signature verification public key.	Place the file in the appropriate folder in the project.
Client certificate	The user downloads this item from AWS when registering the device.*5	Use CLI.*1, *3
Client certificate public key	The user downloads this item from AWS when registering the device.*5 The user then wraps the downloaded data.*4	Place the file in the appropriate folder in the project.
Client certificate private key	The user downloads this item from AWS when registering the device.*5 The user then wraps the downloaded data.*4	Place the file in the appropriate folder in the project.

Notes: 1. “CLI” here is the name of the command line interface provided by this project. For details, refer to section 6.1.5.
2. For details on how to install the item from CLI, refer to section 6.1.5(3).
3. For details on how to install the item from CLI, refer to section 6.1.5(2).
4. For the wrapping procedure, refer to section 5.1.5 and subsequent sections.
5. For the download procedure, refer to section 3.1.
6. For details on how to create this item by using OpenSSL, refer to section 5.1.4.

5.1.1 Flows of Creating Certificates and Keys

This section describes the flows of the tasks for obtaining certificates and keys to be performed in this application note. These flows are excerpts from the flows shown in Figure 1-1, Flow of TLS Communication Using the TSIP.

The following figure shows the flows of creating the certificates and keys that are listed in Table 5-1 and used in the procedures described in this application note.

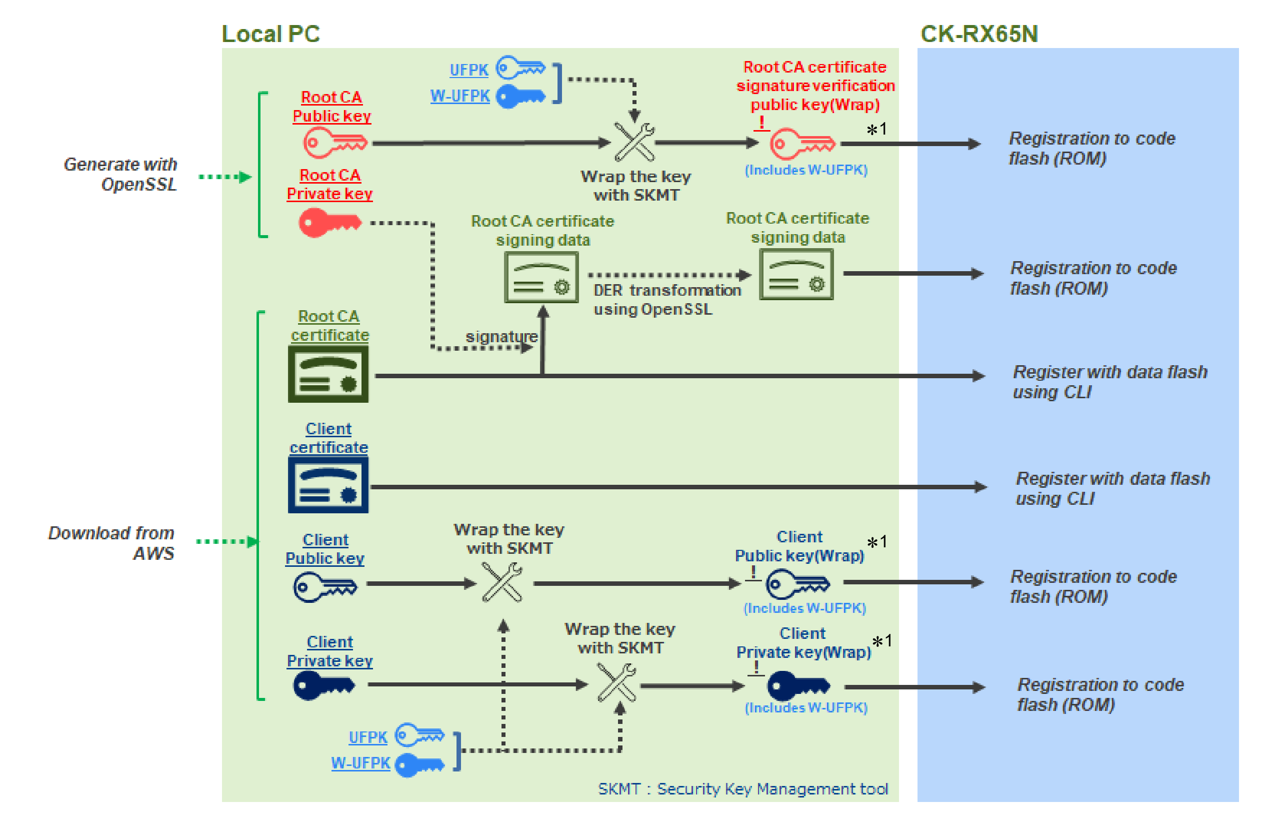


Figure 5-1 Flows of Creating Keys and Certificates for Use with the TSIP

Note: 1. The root CA certificate signature verification public key, client certificate public key, or client certificate private key is equivalent to an encrypted key in Table 1-1. However, each of these items is created as a single file consisting of key information and W-UFPK information, and then registered in the code flash memory.

The following figure shows the flow of creating a UFPK and W-UFPK shown in Figure 5-1.

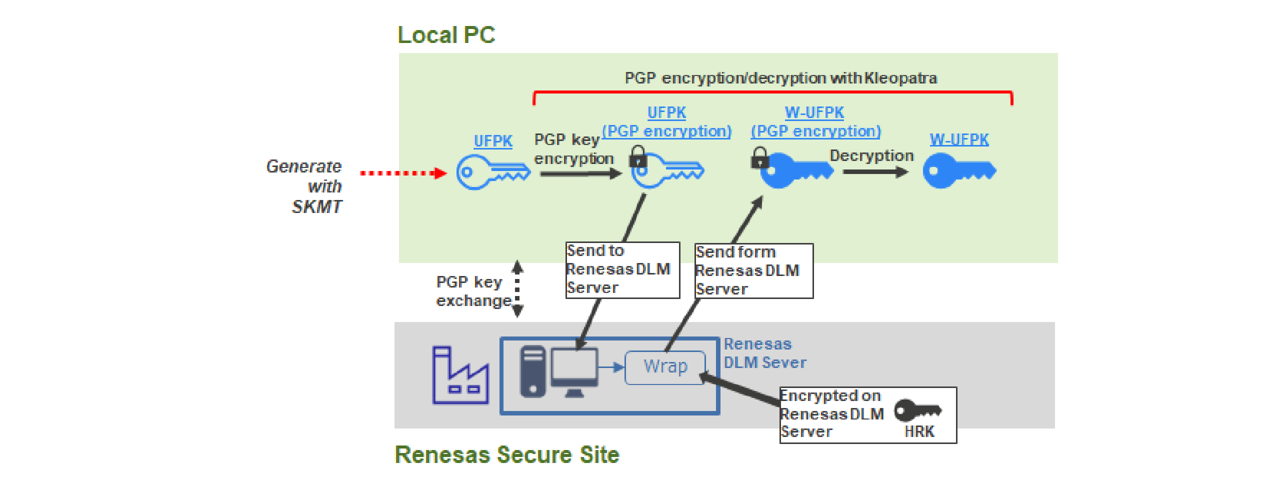


Figure 5-2 Flow of Creating a UFPK and W-UFPK

The procedures for obtaining or creating the necessary certificates and keys are described below.

5.1.2 Obtaining a Root CA Certificate

This section describes how to obtain a root CA certificate. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

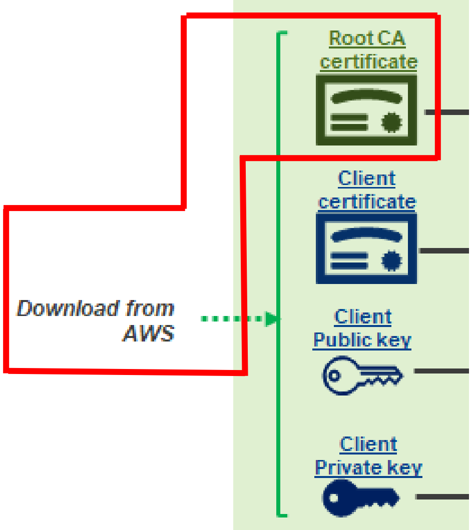
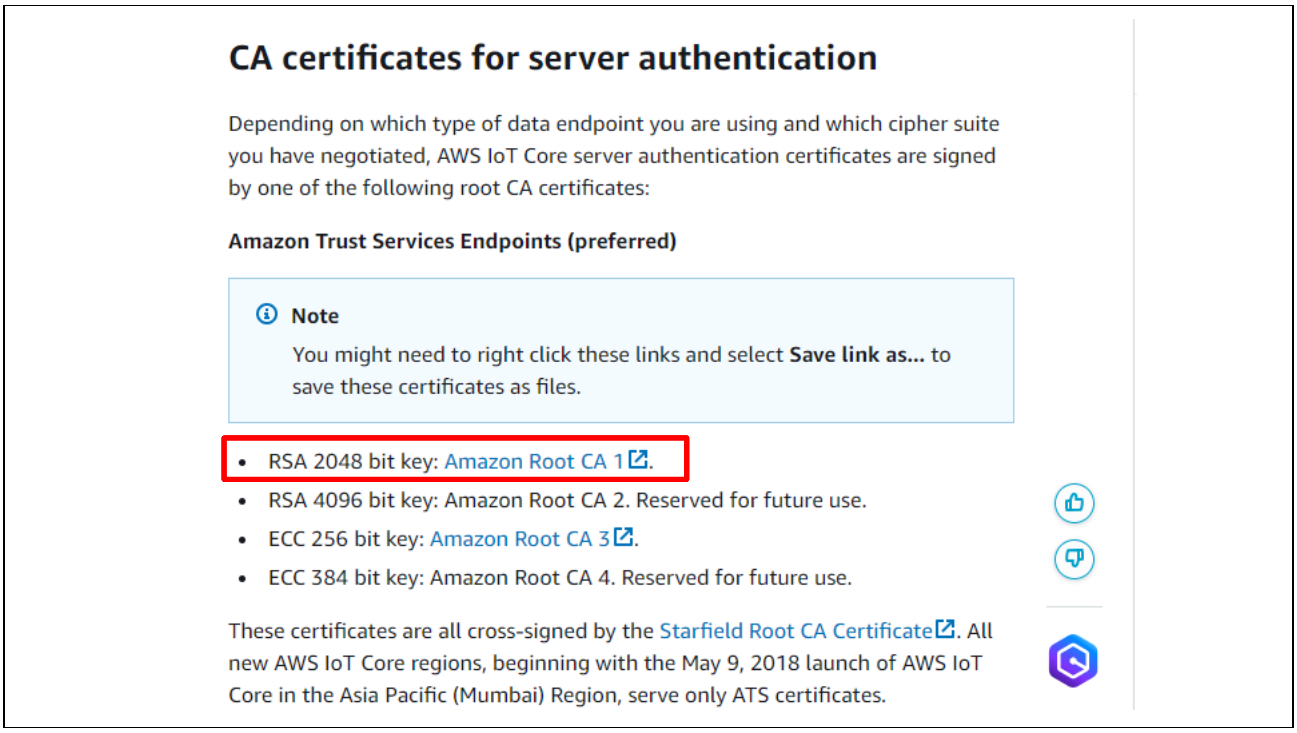


Figure 5-3 Obtaining a Root CA Certificate

Obtain the root CA certificate used for connection to AWS. Download the root CA certificate from the following URL:

https://docs.aws.amazon.com/iot/latest/developerguide/server-authentication.html#server-authentication-certs

Because an RSA certificate is used in this project, download “Amazon Root CA 1”. If the web browser you are using is Edge, you can download it by right-clicking the link and selecting **Save link as**.



CA certificates for server authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT Core server authentication certificates are signed by one of the following root CA certificates:

Amazon Trust Services Endpoints (preferred)

> **Note**
>
> You might need to right click these links and select **Save link as...** to save these certificates as files.

- RSA 2048 bit key: Amazon Root CA 1.
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: Amazon Root CA 3.
- ECC 384 bit key: Amazon Root CA 4. Reserved for future use.

These certificates are all cross-signed by the Starfield Root CA Certificate. All new AWS IoT Core regions, beginning with the May 9, 2018 launch of AWS IoT Core in the Asia Pacific (Mumbai) Region, serve only ATS certificates.

Figure 5-4 Downloading the Root CA Certificate

Confirm that the following file is downloaded:

- AmazonRootCA1.pem

After downloading the certificate, place it in the location indicated in red font in the following figure. Note that the **key_crt_sig_generator** folder exists in the **aws_ryz014a_tsip_ck_rx65n** folder in the project.

```
key_crt_sig_generator
|-- ca
|    |-- AmazonRootCA1.pem
|-- ca-sign-keypair-rsa2048
|-- client-rsa2048
|-- output
|-- 1_rsa2048_convertCrt.sh
|-- convertCrt.sh
```

5.1.3 Obtaining a Key Pair and Client Certificate for RSA

This section describes how to obtain a key pair (client certificate public and private keys) and certificate for RSA. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

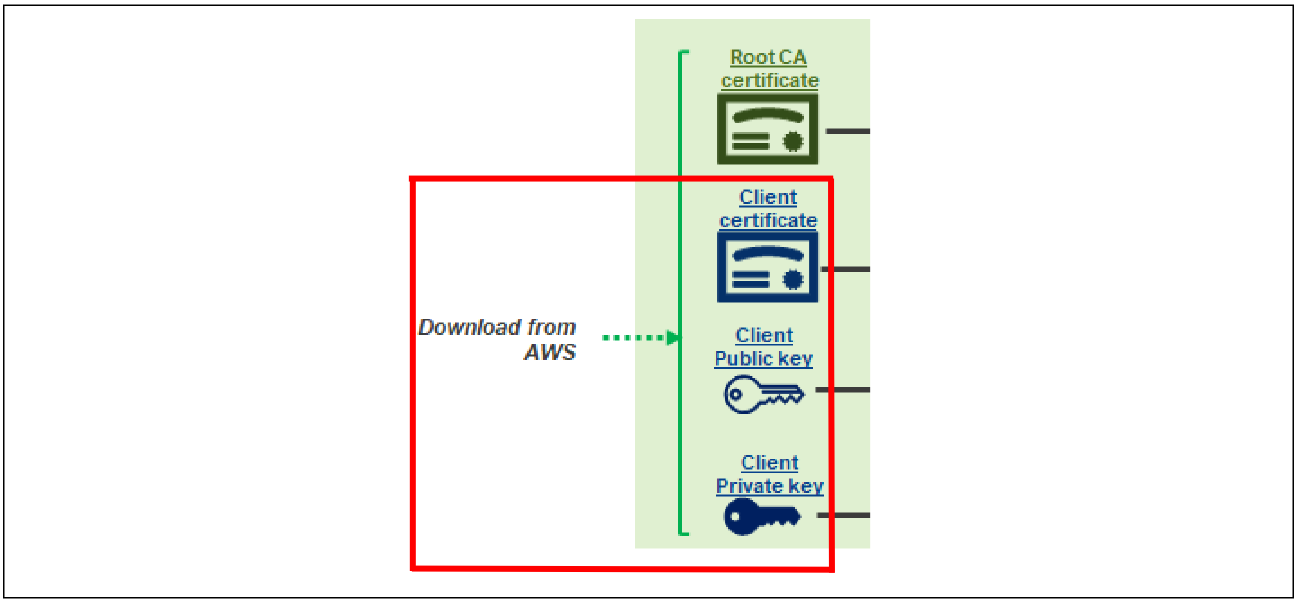
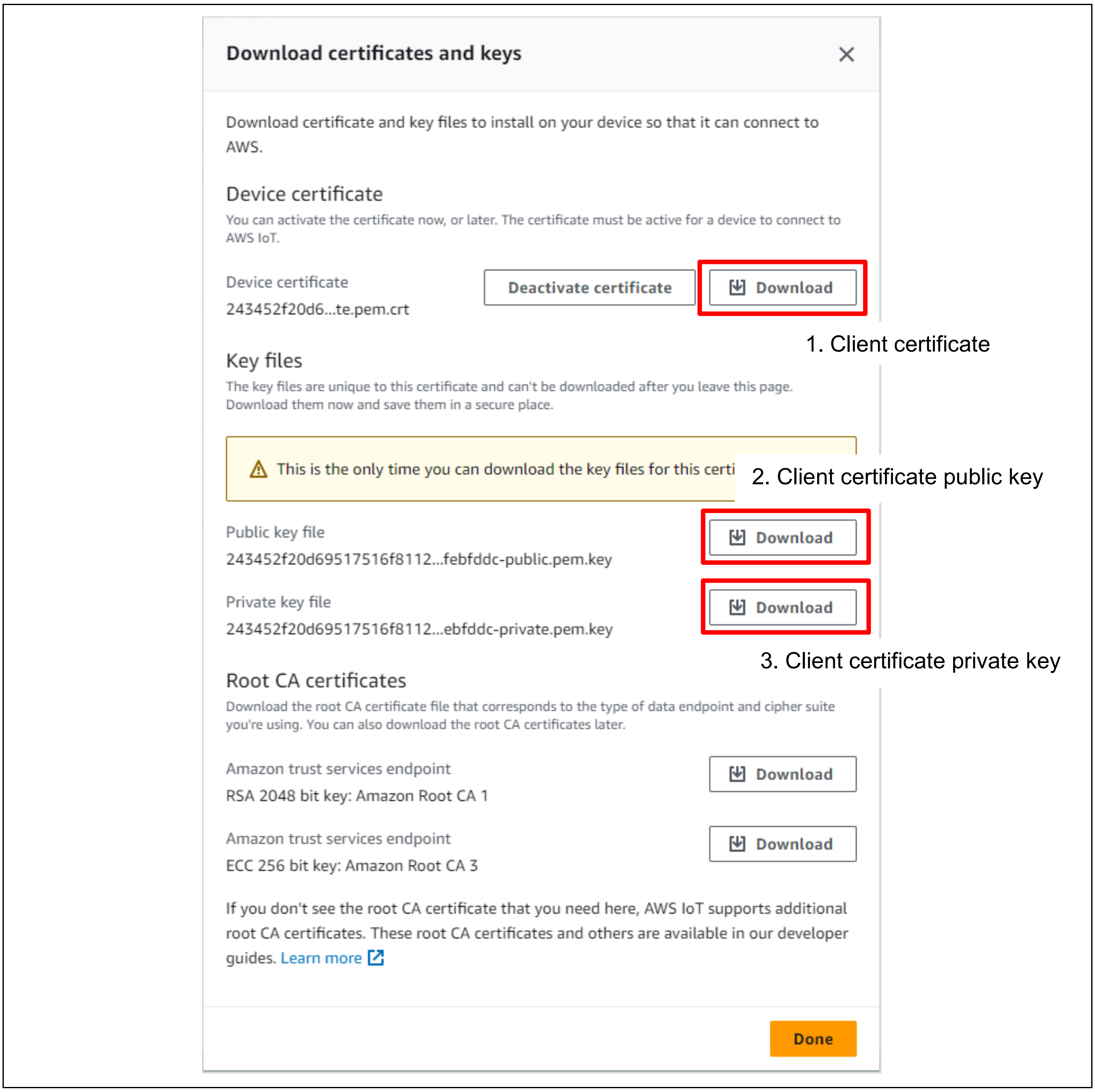


Figure 5-5 Obtaining a Client Certificate/Client Certificate Public and Private Keys

The client certificate and the client certificate public and private keys are automatically generated on the AWS server. Register a device (thing) at the AWS IoT Core website. You can download the key pair and client certificate when registering the thing.

Download these items when you perform the procedures described in section 3.1, Signing in to the AWS Console to section 3.3, Registering your device in AWS in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037).



Download certificates and keys

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

Device certificate

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

Device certificate
243452f20d6...te.pem.crt

Key files

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

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

Public key file
243452f20d69517516f8112...febfddc-public.pem.key

Private key file
243452f20d69517516f8112...ebfddc-private.pem.key

Root CA certificates

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

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

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

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

Figure 5-6 Downloading Keys and a Certificate

You can download three files described in the following table.

Table 5-2 List of Files to Be Downloaded

No.	Name	File name	Name in this application note
1	Device certificate	xxx-certificate.pem.crt	Client certificate
2	Public key file	xxx-public.pem.key	Client certificate public key
3	Private key file	xxx-private.pem.key	Client certificate private key

Notes: 1. You can download key pair files only when creating a thing from AWS Console.
2. In the above, xxx is an arbitrary character string.

After downloading the certificate and keys, place them in the locations indicated in red font in the following figure under the **key_crt_sig_generator** folder in the project.

```
key_crt_sig_generator
|-- ca
|    |-- AmazonRootCA1.pem
|-- ca-sign-keypair-rsa2048
|-- client-rsa2048
|    |-- xxx-certificate.pem.crt
|    |-- xxx-public.pem.key
|    |-- xxx-private.pem.key
|-- output
|-- 1_rsa2048_convertCrt.sh
|-- convertCrt.sh
```

5.1.4 Generating a Signature of the Root CA Certificate

This section describes how to perform registration in the project (source code) by using the downloaded root CA certificate. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

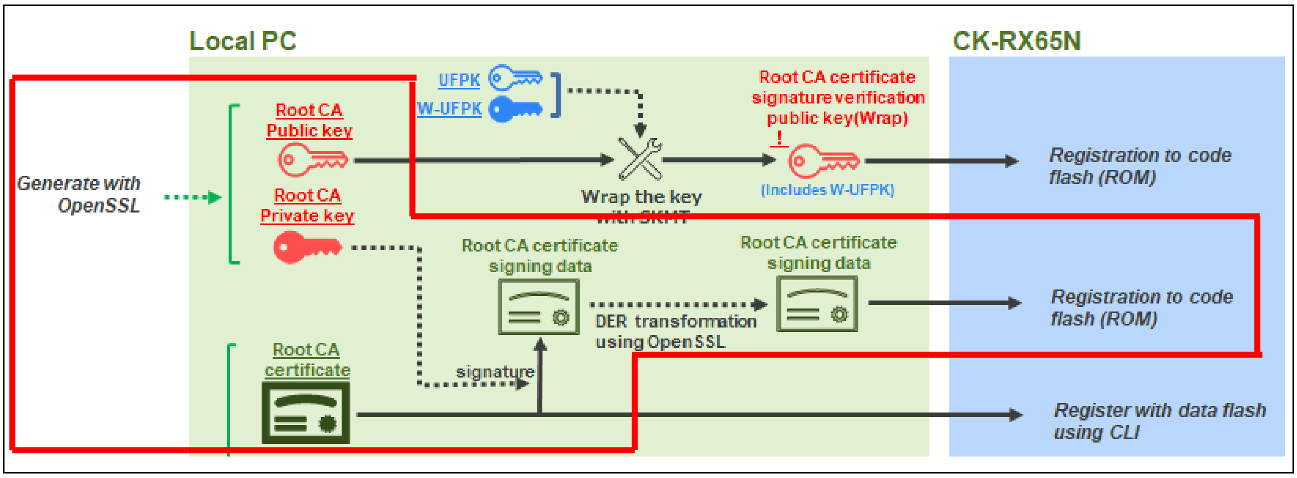

Local PC
CK-RX65N
Generate with OpenSSL
Root CA Public key
Root CA Private key
UFPK
W-UFPK
Wrap the key with SKMT
Root CA certificate signature verification public key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
Root CA certificate signing data
DER transformation using OpenSSL
Root CA certificate signing data
Registration to code flash (ROM)
Root CA certificate
signature
Register with data flash using CLI

Figure 5-7 Creating a Signature of the Root CA Certificate

Convert the certificate by using the procedure described later.
When performing this conversion, execute the script file provided in the **key_crt_sig_generator** project folder. Note that the script file contains a Bash shell script.
This application note shows an example in which Cygwin is used to provide a Unix-like interface. Prepare an environment in which Cygwin can operate by referring to section 2.3.

Before executing the script, place the key files in the appropriate folders by using the procedures described in section 5.1.2 and section 5.1.3.

[Note] When executing the script, be sure to refer to section 6.1.5(6) and delete the data flash when executing the program.
Also, please perform the series of operations from section 5.1.4 to section 5.1.5 consecutively.
Partial steps can produce inconsistent data.

(1) Converting the Root CA Certificate for RSA

Execute the script provided by this project. In this script, the root CA certificate for RSA is converted into the DER format.
Next, a signature of the root CA certificate and a key pair used for signature verification (root CA certificate public and private keys) in RSA-2048 format are generated. Then, the generated root CA certificate private key is used to generate the root CA certificate signature data.
The signature data after conversion has been converted in the array format in C language so that it can be registered in the source code of the project.

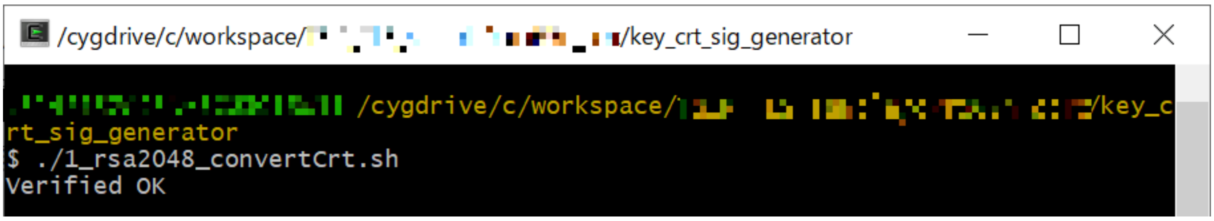
Activate Cygwin, and then move to the **key_crt_sig_generator** folder in the project by entering the following command:

```
 cd /cygdrive/c/workspace/key_crt_sig_generator
```

Note: The above command line applies when the script file is placed in the **workspace/key_crt_sig_generator** folder.

Execute the script by entering the following command:

```
./1_rsa2048_convertCrt.sh
```



/cygdrive/c/workspace/[redacted]/key_crt_sig_generator

```
[redacted] /cygdrive/c/workspace/[redacted]/key_crt_sig_generator
$ ./1_rsa2048_convertCrt.sh
Verified OK
```

Figure 5-8 Executing the Script

(2) Registering the converted files in the project

After executing the script, six files (indicated in blue or red font in the following figure) are generated in the **key_crt_sig_generator/output** folder.

```
key_crt_sig_generator
|-- ca
|-- ca-sign-keypair-rsa2048
|    |-- rsa2048-private.pem
|    |-- rsa2048-public.pem
|-- client-rsa2048
|-- output
|    |-- AmazonRootCA1_crt.der
|    |-- AmazonRootCA1_crt_array.txt
|    |-- AmazonRootCA1_sig.sig
|    |-- AmazonRootCA1_sig_array.txt
|--1_rsa2048_convertCrt.sh
|--convertCrt.sh
```

The six generated files are described in the following table.

Table 5-3 Description of the Converted Files

Name	File name
Root CA certificate private key (in PEM format)	rsa2048-private.pem
Root CA certificate public key (in PEM format)	rsa2048-public.pem
Root CA certificate data (in DER format)	AmazonRootCA1_crt.der
Root CA certificate data (coded as an array of type uint8_t in C language)	AmazonRootCA1_crt_array.txt
Root CA certificate signature data signed by the private key	AmazonRootCA1_sig.sig
Root CA certificate signature data signed by the private key (coded as an array of type uint8_t in C language)	AmazonRootCA1_sig_array.txt

Of the above output files, **AmazonRootCA1_sig_array.txt** is the root CA certificate signature data signed by the private key.
This file contains generated binary data coded as an array of type uint8_t in C language. When **AmazonRootCA1_sig_array.txt** is generated, copy it to the following folder in the project, overwriting the file with the same name:

```
\iot-reference-rx\
Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\src\userdata_tsip
```

Also, the root CA certificate public key file (**rsa2048-public.pem**) and the root CA certificate private key file (**rsa2048-private.pem**) are generated in the **key_crt_sig_generator/ca-sign-keypair-rsa2048** folder. The public key file will later be used when the root CA certificate signature verification public key is generated in section 5.1.5.

5.1.5 Wrapping Keys and Registering Them in the Project

Wrap the root CA certificate public key, client certificate public key, and client certificate private key that you generated, and then register them in the project (source code).
These three keys are user keys.

In this section, you use the following keys: the root CA certificate public key generated in section 5.1.4; and the client certificate public and private keys downloaded from AWS in section 5.1.3.

5.1.5.1 Overview of Wrapping Keys

Because the TSIP driver does not accept a plaintext user key as an input, the user key must be wrapped in a format that can be accepted by the TSIP driver.

Keys used for TLS communication are generally provided in PEM format in the same way as certificates. For a key to be used for the TSIP driver, extract the data of the key from the PEM-formatted key file. Then, wrap the key by using the Renesas Key Wrap service (Renesas DLM server) and Security Key Management Tool. Note that to exchange key data by using the Renesas Key Wrap service, the data must be encrypted by OpenPGP.

An overview of the procedure is as follows. The detailed steps are given later.

1. Use Security Key Management Tool to create a plaintext UFPK.
2. Use Kleopatra to encrypt the UFPK by PGP (so that it can be handled by Key Wrap).
3. Send the PGP-encrypted UFPK to Renesas by using the Renesas Key Wrap service.
4. Re-encrypt the PGP-encrypted UFPK on the Renesas DLM server (a hardware root key (HRK) is used internally in Renesas).
5. Receive the Renesas-encrypted UFPK (UFPK encrypted by PGP for transmission).
6. Use Kleopatra to decrypt the PGP encryption and obtain an encrypted UFPK (W-UFPK).

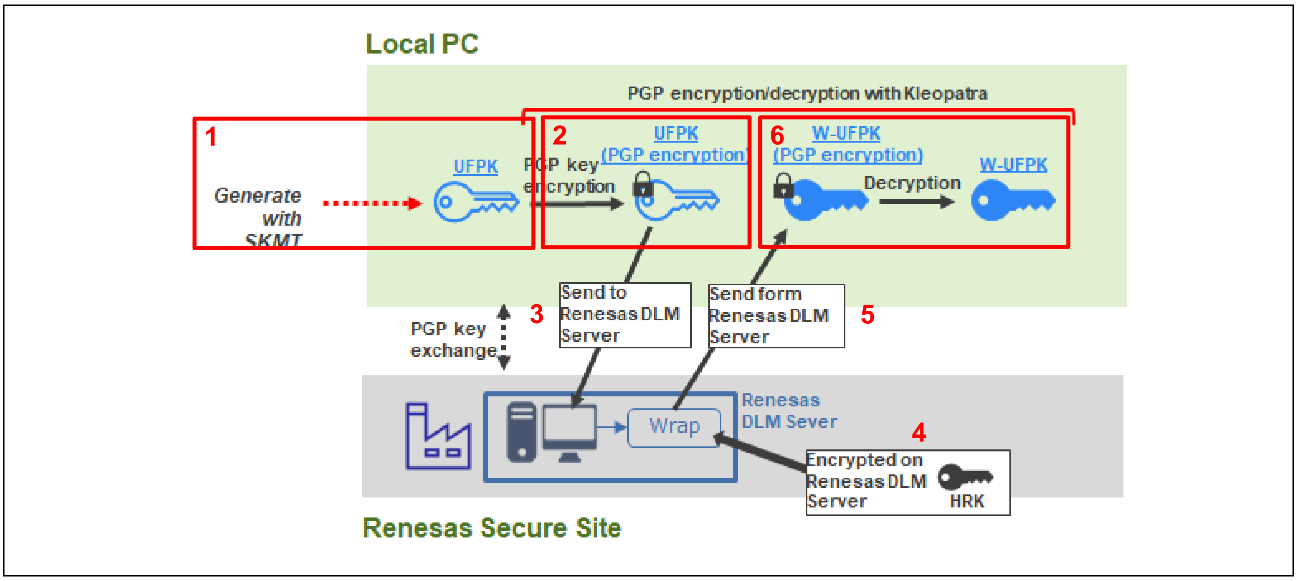


Figure 5-9 Procedure for Generating a UFPK/W-UFPK

7. Use Security Key Management Tool to wrap the user keys (root CA certificate public key, client certificate public key, and client certificate private key) by using the UFPK (generate an encrypted key). Combine the encrypted key with a W-UFPK into a key that will be used after wrapping.
8. Register the wrapped encrypted key files in the source code.

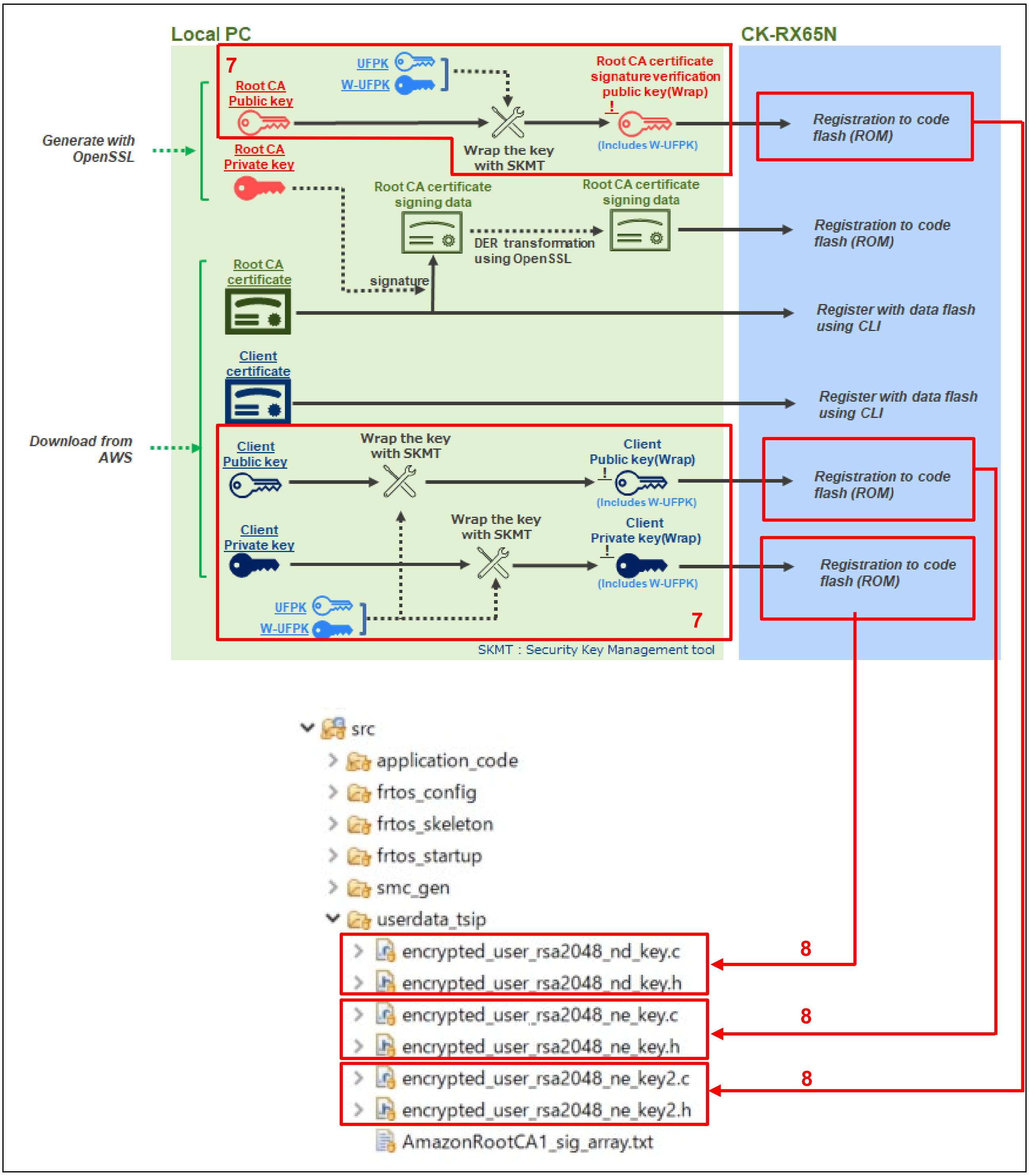

Local PC
CK-RX65N
7
Root CA Public key
UFPK
W-UFPK
Root CA certificate signature verification public key(Wrap)
(Includes W-UFPK)
Wrap the key with SKMT
Registration to code flash (ROM)
Generate with OpenSSL
Root CA Private key
Root CA certificate signing data
Root CA certificate signing data
DER transformation using OpenSSL
Registration to code flash (ROM)
signature
Root CA certificate
Register with data flash using CLI
Client certificate
Register with data flash using CLI
Download from AWS
Client Public key
Wrap the key with SKMT
Client Public key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
Client Private key
Wrap the key with SKMT
Client Private key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
UFPK
W-UFPK
7
SKMT : Security Key Management tool
src
application_code
frtos_config
frtos_skeleton
frtos_startup
smc_gen
userdata_tsip
encrypted_user_rsa2048_nd_key.c
encrypted_user_rsa2048_nd_key.h
encrypted_user_rsa2048_ne_key.c
encrypted_user_rsa2048_ne_key.h
encrypted_user_rsa2048_ne_key2.c
encrypted_user_rsa2048_ne_key2.h
AmazonRootCA1_sig_array.txt
8
8
8

Figure 5-10 Procedure for Wrapping Key Information

Three encrypted keys (root CA certificate signature verification public key, client certificate public key, and client certificate private key) are generated as wrapped keys, which are encrypted chip-specific keys, inside the TSIP by using an HRK and HUK.
The wrapped keys are decrypted by the processing inside the TSIP and the decrypted keys are used for connection to AWS.

5.1.5.2 Generating a UFPK and W-UFPK

This section describes how to generate a UFPK (user factory programming key) and W-UFPK (a UFPK wrapped by the Renesas Key Wrapping service), which are used to wrap keys. The task performed in this section corresponds to the flows shown in Figure 5-2.

Generate your own UFPK, and then upload it to the DLM server to generate a W-UFPK. A UFPK is a key used to wrap a public key for verifying a signature.
You can use Security Key Management Tool to create a UFPK in a format that can be accepted by the DLM server.*1

Once you create a UFPK and W-UFPK, you do not need to create them again.

Note: 1. This section describes the procedure for generating a UFPK that is to be used for key wrapping and an encrypted UFPK (W-UFPK).
The key information generated by this procedure is sample information. Therefore, it cannot be used for actual operation.
To use keys for mass production, you must generate your own keys. Details on how to do so are described in another application note provided by Renesas.
If you have already used Renesas MCUs or you plan to adopt Renesas MCUs, Renesas provides that application note. If you want to reference it, contact the relevant Renesas sales office.
https://www.renesas.com/contact/

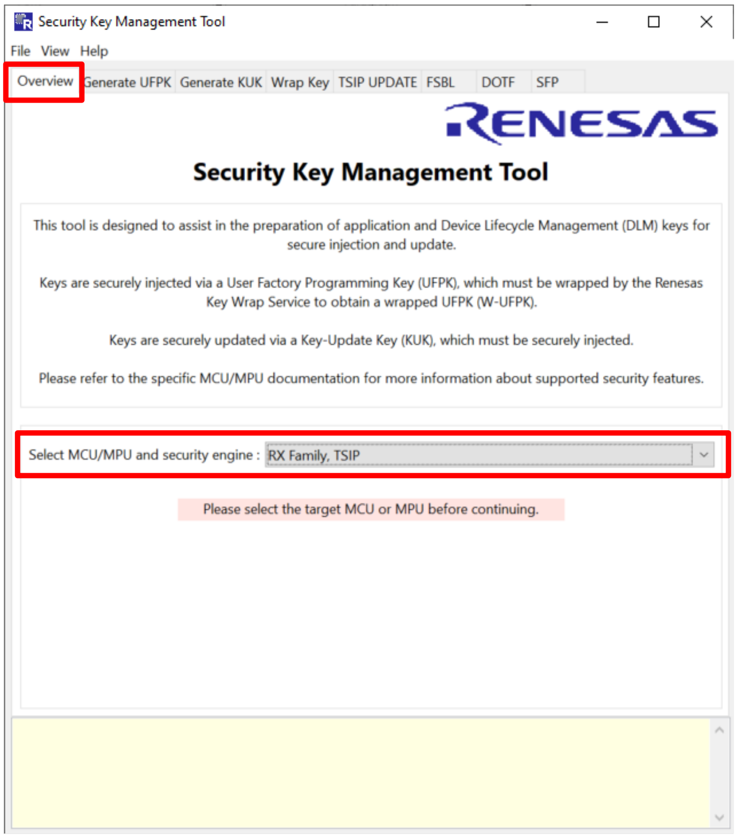
(1) Preparing tools

The following tools are used to wrap a key. Make sure that these tools are ready for use by performing the relevant setup procedure for each tool described in this document.

• Gpg4win (Kleopatra):	Refer to section 2.1.
• Renesas Key Wrap service:	Refer to section 2.2.
• Security Management Tool:	Refer to section 2.4.

(2) Setting up Security Key Management Tool (SKMT)

Start Security Key Management Tool that you installed, click the **Overview** tab, and then, from the **Select MCU/MPU and security engine** drop-down list, select **RX Family, TSIP**.



Security Key Management Tool

This tool is designed to assist in the preparation of application and Device Lifecycle Management (DLM) keys for secure injection and update.

Keys are securely injected via a User Factory Programming Key (UFPK), which must be wrapped by the Renesas Key Wrap Service to obtain a wrapped UFPK (W-UFPK).

Keys are securely updated via a Key-Update Key (KUK), which must be securely injected.

Please refer to the specific MCU/MPU documentation for more information about supported security features.

Select MCU/MPU and security engine : RX Family, TSIP

Please select the target MCU or MPU before continuing.

Figure 5-11 Selecting the MCU and Security Engine

(3) Creating a plaintext UFPK

This section describes how to create a UFPK. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-2.

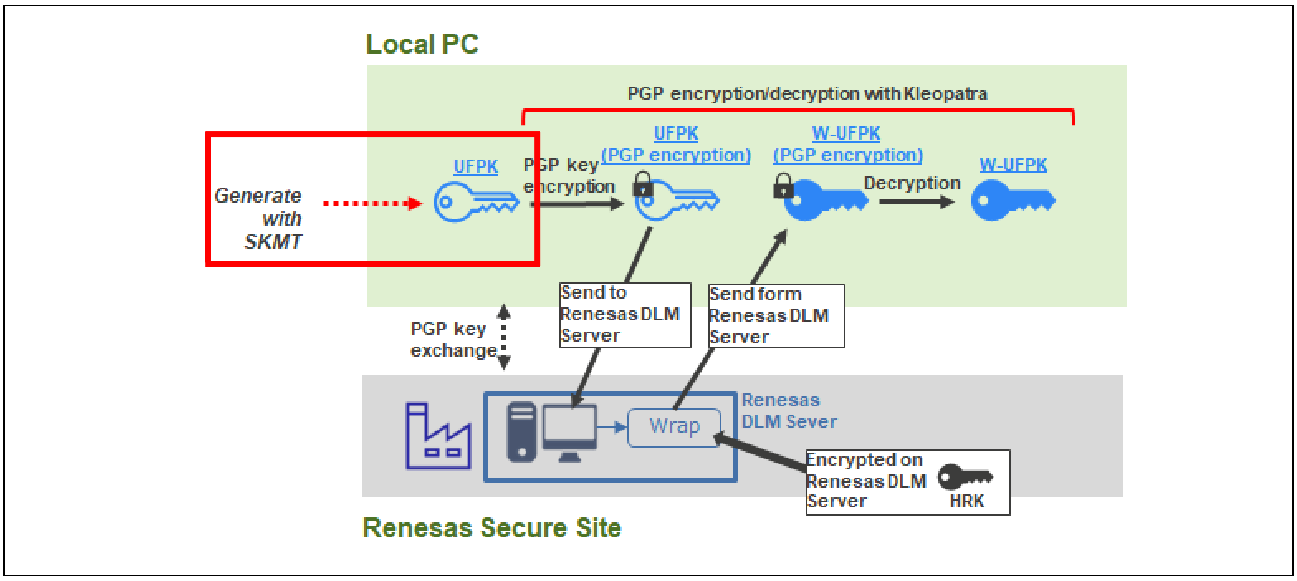
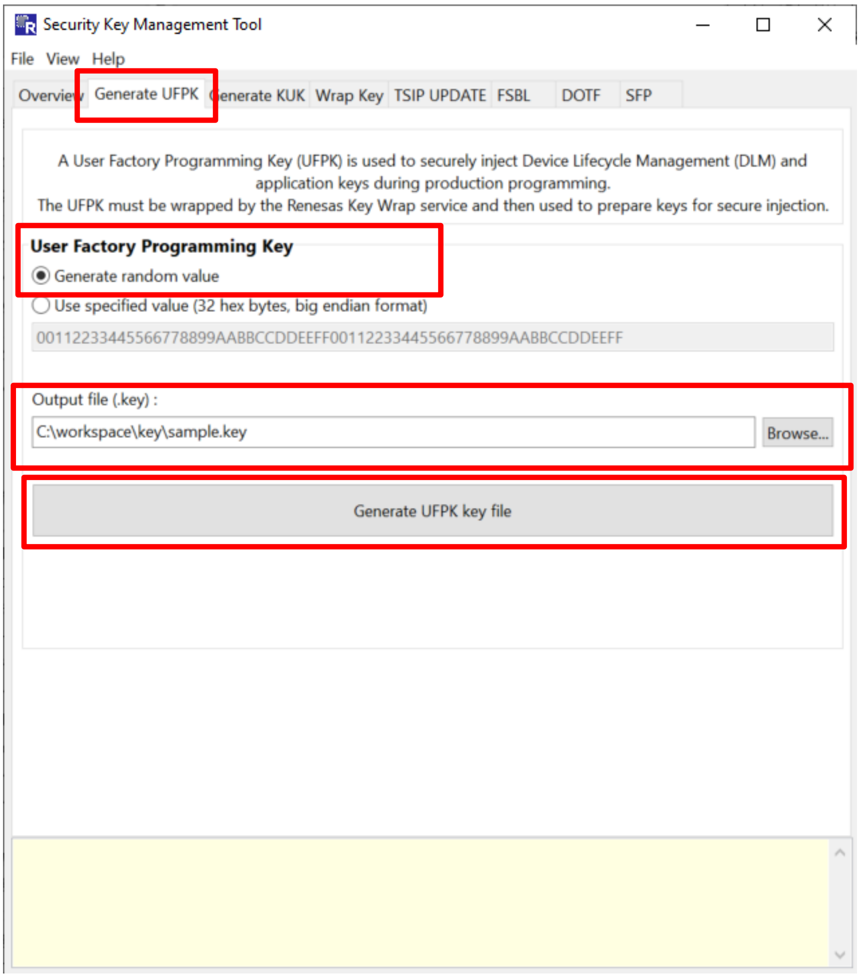


Figure 5-12 Generating a Plaintext UFPK

In the Security Key Management Tool application, click the **Generate UFPK** tab, and then specify the following settings:

- **User Factory Programming Key** Select **Generate random value**.
- **Output file** Set the file to which the generated UFPK is to be output. You can specify a file in any folder of your choice. In this example, the file is named “sample.key”.

When you have completed the settings, click the **Generate UFPK key file** button. A UFPK is output to the specified folder.



A User Factory Programming Key (UFPK) is used to securely inject Device Lifecycle Management (DLM) and application keys during production programming.
The UFPK must be wrapped by the Renesas Key Wrap service and then used to prepare keys for secure injection.

User Factory Programming Key

● Generate random value

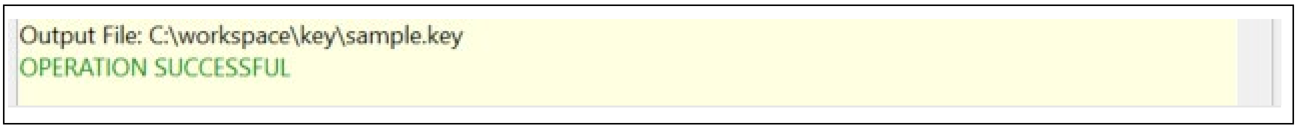
○ Use specified value (32 hex bytes, big endian format)

00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF

Output file (.key) : C:\workspace\key\sample.key

Figure 5-13 Generating a UFPK by Using the Security Key Management Tool

When the following message is displayed at the bottom of the window, the file has been successfully generated. Confirm that the **sample.key** file has been output to the specified folder.



```
Output File: C:\workspace\key\sample.key
OPERATION SUCCESSFUL
```

Figure 5-14 Output Result Message

(4) Encrypting the UFPK by PGP

This section describes how to encrypt the **sample.key** UFPK file (created in section 5.1.5.2(3)) by using an OpenPGP key pair created with Kleopatra and a PGP public key that you received from Renesas in return for your own key. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-2.

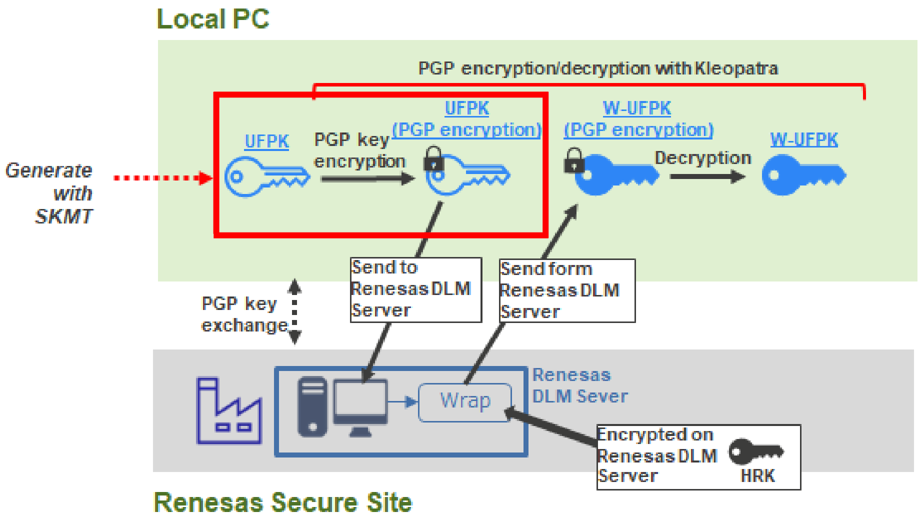
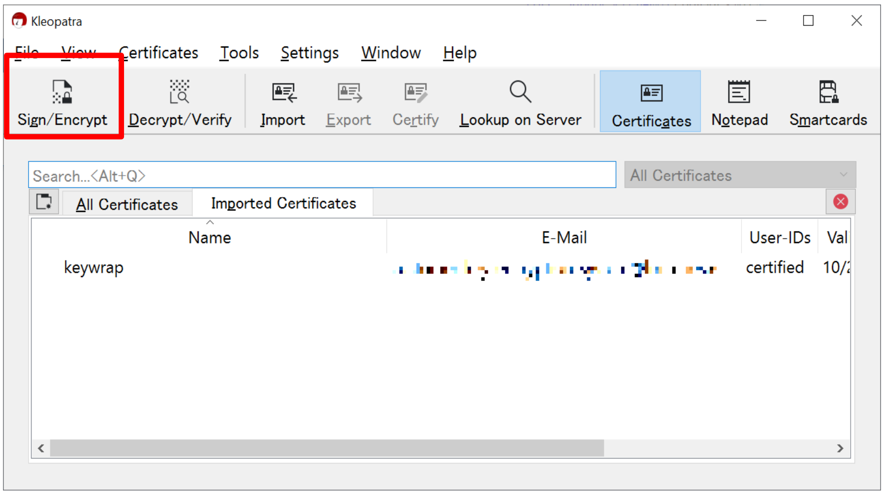


Figure 5-15 PGP Encryption of a UFPK

In the Kleopatra window, click the **Sign/Encrypt** button. When the dialog box for selecting a file appears, specify **sample.key**.



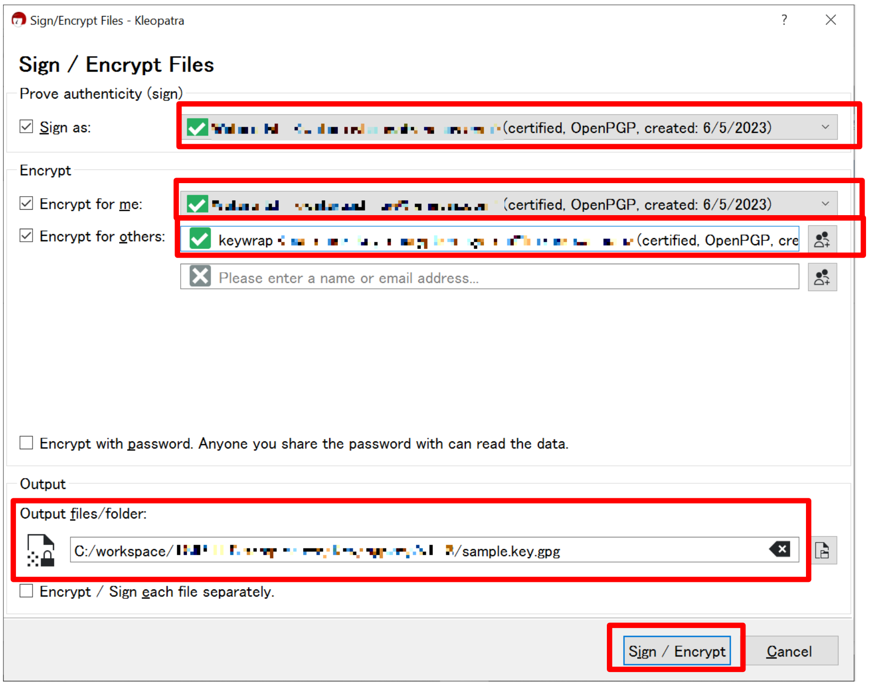
Name	E-Mail	User-IDs	Val
keywrap	[redacted]	certified	10/

Figure 5-16 Encrypting the “sample.key” file

The **Sign/Encrypt Files** dialog box for the selected file appears. This dialog box asks you whether to perform signature and encryption. In this dialog box, specify the key pair and key that you created as follows:

- **Sign as**: Specify your own OpenPGP public key that you created in section 2.2(2).
- **Encrypt for me**: Specify your own OpenPGP public key that you created in section 2.2(2).
- **Encrypt for others**: Specify the Renesas PGP public key (keywrap) that you registered in section 2.2(4).

Set the output destination folder in **Output files/folder**, and then click the **Sign/Encrypt** button. A file named **sample.key.gpg** is created in the specified folder. This file is a UFPK encrypted by PGP (W-UFPK).


Sign/Encrypt Files - Kleopatra

Sign / Encrypt Files

Prove authenticity (sign)

☑ Sign as: [redacted] (certified, OpenPGP, created: 6/5/2023)

Encrypt

☑ Encrypt for me: [redacted] (certified, OpenPGP, created: 6/5/2023)

☑ Encrypt for others: keywrap [redacted] (certified, OpenPGP, cre

Please enter a name or email address...

☐ Encrypt with password. Anyone you share the password with can read the data.

Output

Output files/folder:

C:/workspace/[redacted]/sample.key.gpg

☐ Encrypt / Sign each file separately.

Figure 5-17 Encrypting the "sample.key" File by PGP

(5) Encrypting the PGP-encrypted UFPK on the DLM server

This section describes how to encrypt the PGP-encrypted UFPK by using the Renesas Key Wrap service. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-2. Within the DLM server, a hardware root key (HRK) is used for encryption.

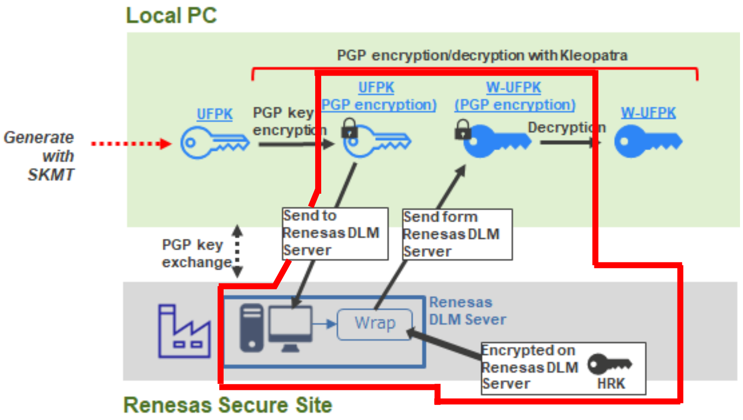
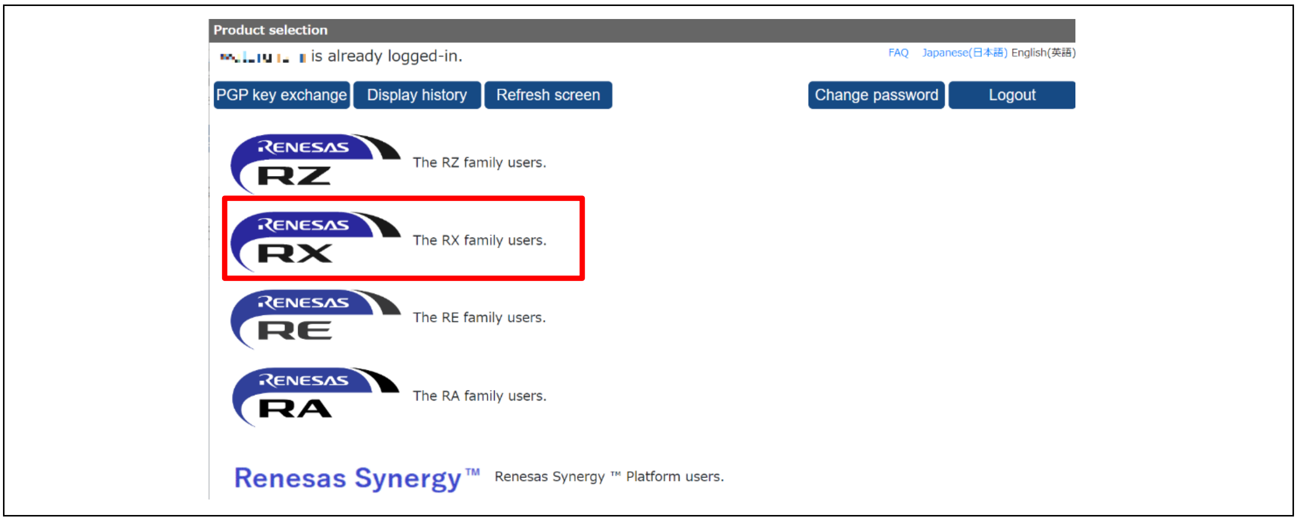


Figure 5-18 Uploading the UFPK for Encryption

Log in to the Renesas Key Wrap service website, and then click **RENESAS RX** on the top page.



Product selection

[redacted] is already logged-in.

The RZ family users.

The RX family users.

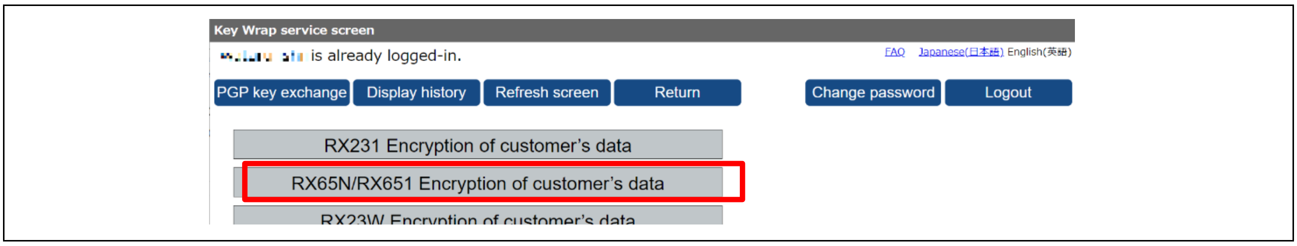
The RE family users.

The RA family users.

Renesas Synergy ™ Platform users.

Figure 5-19 Selecting the Type (Family) of the Encryption-Target MCU

When the following device selection screen appears, click **RX65N/RX651 Encryption of customer's data**.



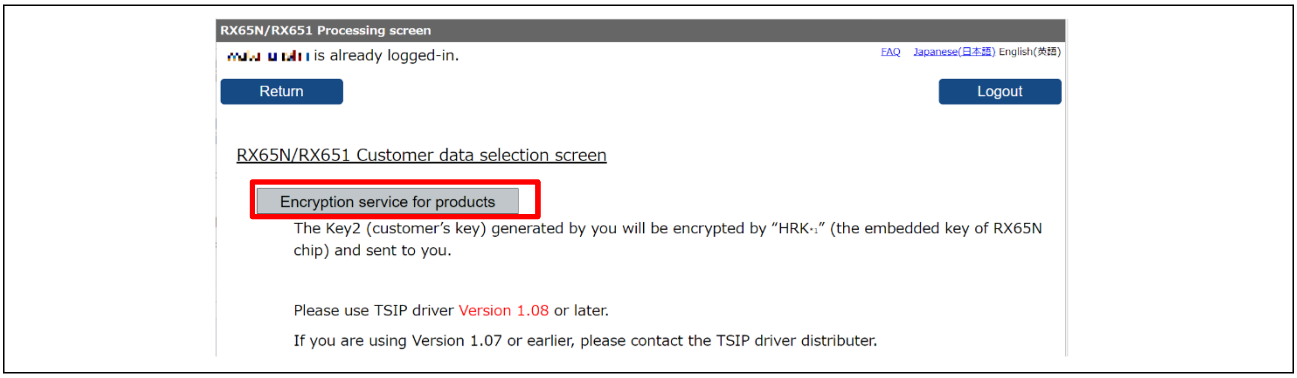
Key Wrap service screen

[redacted] is already logged-in.

FAQ
Japanese(日本語) English(英語)
PGP key exchange
Display history
Refresh screen
Return
Change password
Logout
RX231 Encryption of customer's data
RX65N/RX651 Encryption of customer's data
RX23W Encryption of customer's data

Figure 5-20 Selecting the Device Group

Click the **Encryption service for products** link.



RX65N/RX651 Processing screen

[redacted] is already logged-in.

FAQ
Japanese(日本語) English(英語)
Return
Logout
RX65N/RX651 Customer data selection screen
Encryption service for products

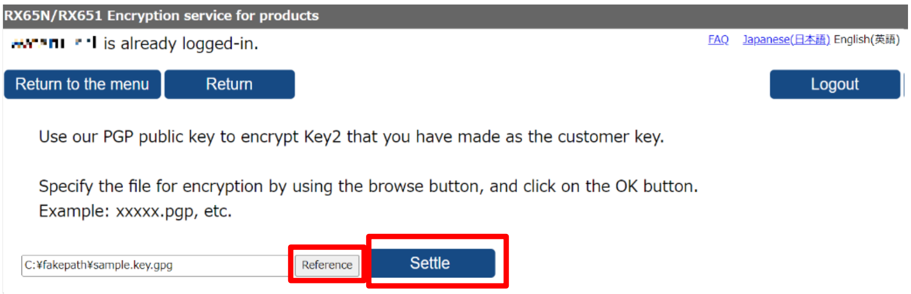
The Key2 (customer's key) generated by you will be encrypted by "HRK-1" (the embedded key of RX65N chip) and sent to you.

Please use TSIP driver Version 1.08 or later.

If you are using Version 1.07 or earlier, please contact the TSIP driver distributer.

Figure 5-21 Selecting the Encryption Service

When the following upload screen appears, click the **Reference** button. Then, specify the **sample.key.pgp** file that you created in section 5.1.5.2(4), and click the **Settle** button.



RX65N/RX651 Encryption service for products

[redacted] is already logged-in.

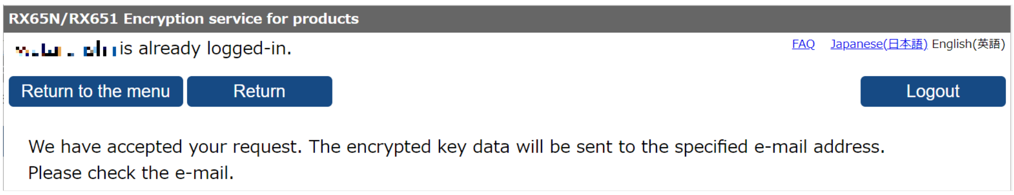
Use our PGP public key to encrypt Key2 that you have made as the customer key.

Specify the file for encryption by using the browse button, and click on the OK button.
Example: xxxxx.pgp, etc.

C:¥fakepath¥sample.key.gpg

Figure 5-22 Uploading a UFPK

When an upload finishes, the following screen appears, and encryption starts on the Renesas DLM server. When encryption finishes, the **sample.key_enc.key.pgp** file is sent from Renesas to you by email. Save the file in any folder of your choice.



RX65N/RX651 Encryption service for products

[redacted] is already logged-in.

We have accepted your request. The encrypted key data will be sent to the specified e-mail address.
Please check the e-mail.

Figure 5-23 Screen Appearing When an Upload to the DLM Server Finishes

(6) Decrypting “sample.key_enc.key.pgp” by OpenPGP

The **sample.key_enc.key.pgp** file sent from Renesas is a PGP-encrypted W-UFPK. Decrypt it by using your own OpenPGP key to create a normal W-UFPK (encrypted UFPK). The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-2.

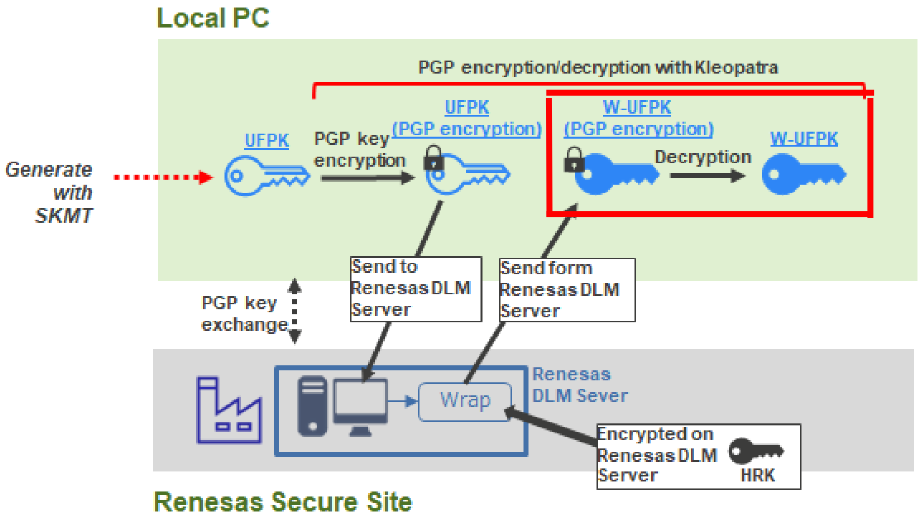
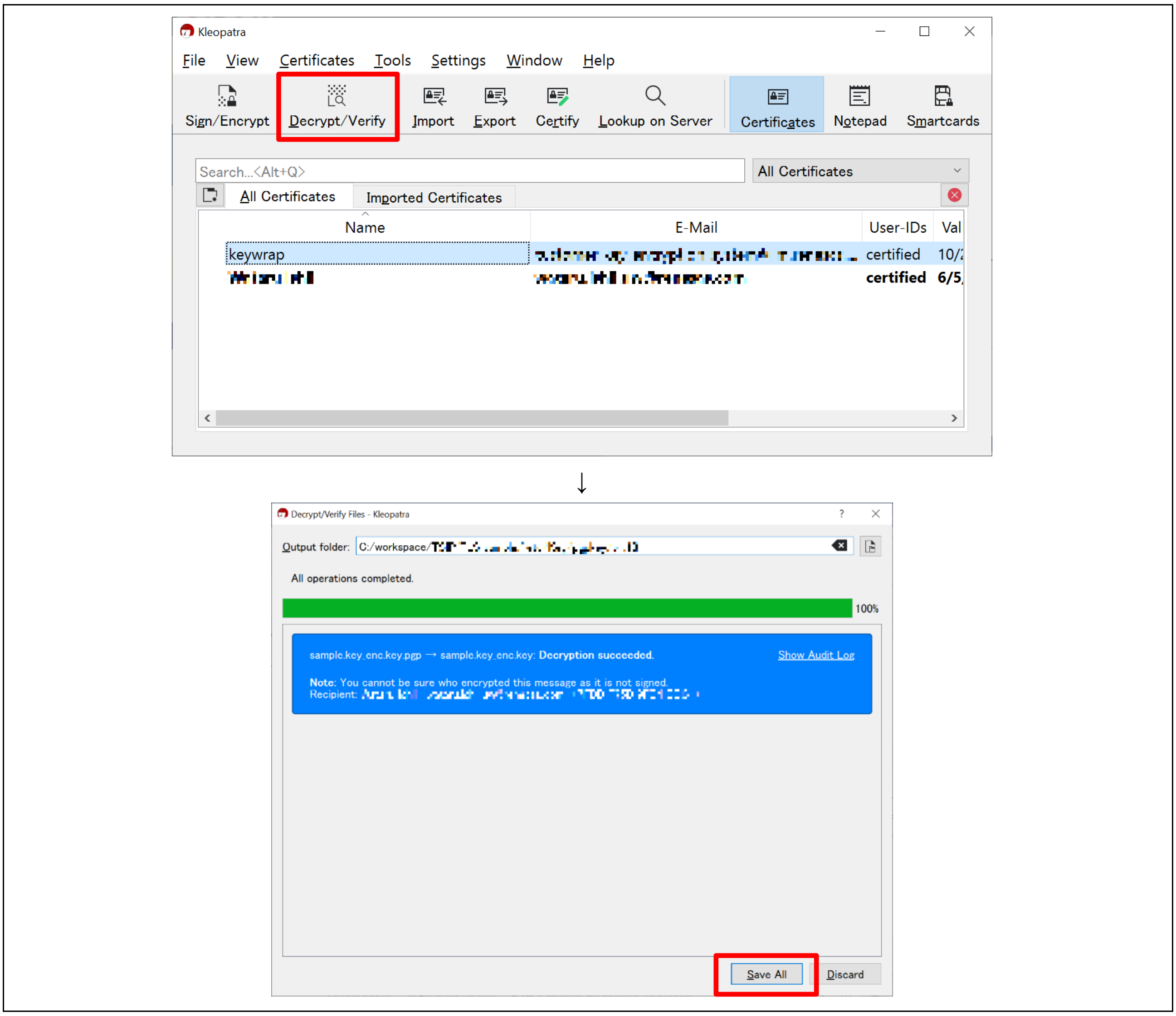


Figure 5-24 Creating a W-UFPK

In Kleopatra, click the **Decrypt/Verify** button, and then, in the file selection screen, select **sample.key_enc.key.pgp**. Decryption starts.
When decryption finishes and a message to that effect appears, click the **Save All** button to save the decrypted key. The decrypted file (**sample.key_enc.key**) is output to the folder that stores the file before decryption (**sample.key_enc.key.pgp**).
Encryption of **sample.key** is now complete. The **sample.key_enc.key** file is a W-UFPK.



Name	E-Mail	User-IDs	Val
keywrap	[redacted]	certified	10/:
[redacted]	[redacted]	certified	6/5,

Output folder: C:/workspace/[redacted]

All operations completed.

100%

sample.key_enc.key.pgp → sample.key_enc.key: **Decryption succeeded.** Show Audit Log

Note: You cannot be sure who encrypted this message as it is not signed.
Recipient: [redacted]

Figure 5-25 Creating a W-UFPK by PGP Decryption

5.1.5.3 Wrapping Key Data

This section describes how to convert the data of the root CA certificate public key (root CA certificate signature verification public key), client certificate public key, and client certificate private key into data to be registered in the project. This conversion requires **sample.key** (as a UFPK) and **sample.key_enc.key** (as a W-UFPK) that you created in section 5.1.5.2, Generating a UFPK and W-UFPK.
The conversion wraps the three keys by using the UFPK, and then combines them with the W-UFPK.

(1) Key data used

From the root CA certificate public key you generated (in section 5.1.4, Generating a Signature of the Root CA Certificate) and the PEM-formatted client certificate public and private keys you obtained from AWS (in section 5.1.3, Obtaining a Key Pair and Client Certificate for RSA), extract the root CA certificate public key, client certificate public key, and client certificate private key. The files of these three keys are as follows:

- Root CA certificate public key file (PEM)
  /key_crt_sig_generator /ca-sign-keypair-rsa2048 /rsa2048-public.pem
- Client certificate public key file (PEM)
  /key_crt_sig_generator /client-rsa2048 /*xxxx*-public.pem.key
  Note: In the above, xxxx is an arbitrary character string.
- Client certificate private key file (PEM)
  /key_crt_sig_generator /client-rsa2048 /*xxxx*-private.pem.key
  Note: In the above, xxxx is an arbitrary character string.

Note: Wrap the above three keys by using the following two keys, which were created in section 5.1.5.2:

- UFPK (sample.key)
- W-UFPK (sample.key_enc.key)

The keys used in this section are enclosed in red frames in the following figure.
The same UFPK and W-UFPK are used to wrap the root CA certificate signature verification public key, client certificate public key, and client certificate private key.

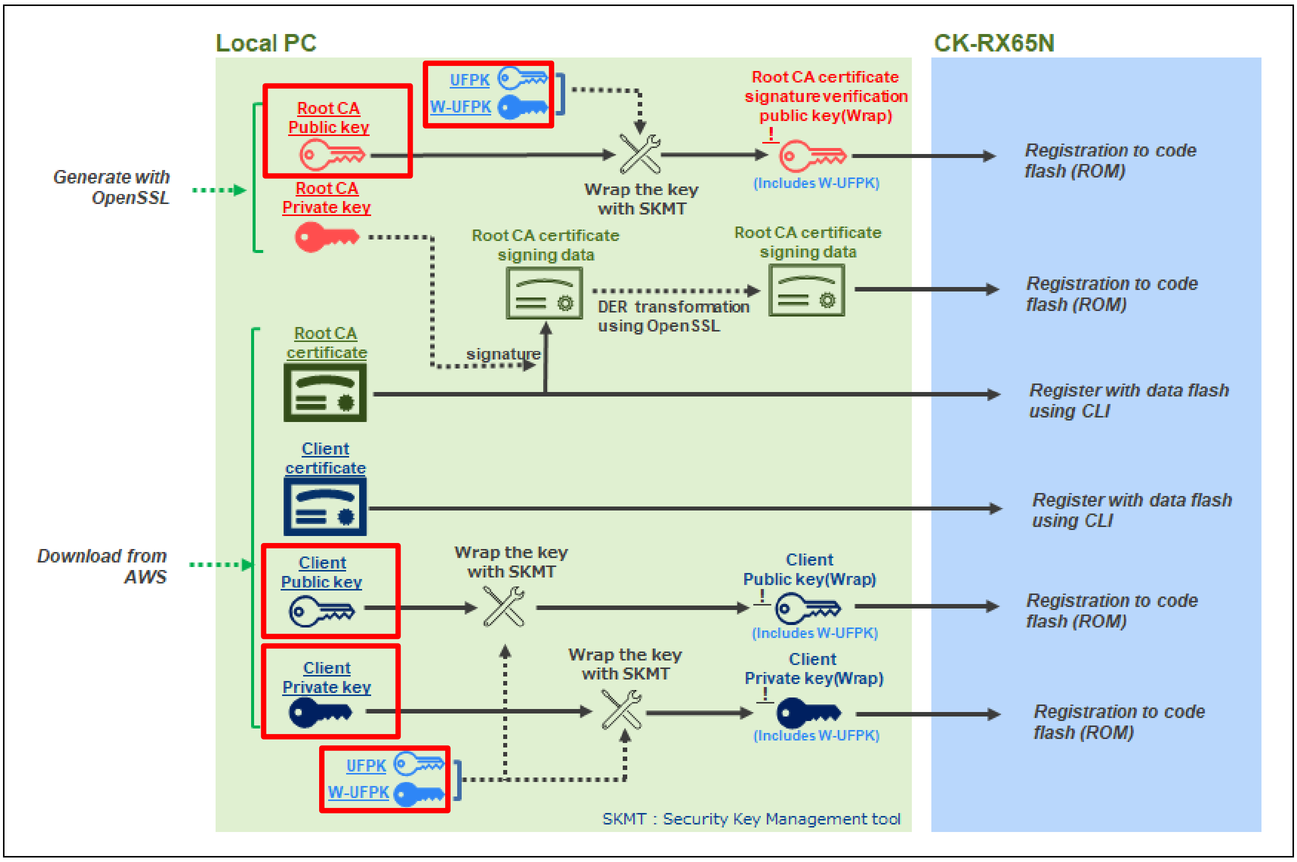

Local PC
CK-RX65N
Generate with OpenSSL
Root CA Public key
Root CA Private key
UFPK
W-UFPK
Wrap the key with SKMT
Root CA certificate signature verification public key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
Root CA certificate signing data
Root CA certificate signing data
DER transformation using OpenSSL
Registration to code flash (ROM)
signature
Root CA certificate
Register with data flash using CLI
Client certificate
Register with data flash using CLI
Download from AWS
Client Public key
Wrap the key with SKMT
Client Public key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
Client Private key
Wrap the key with SKMT
Client Private key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
UFPK
W-UFPK
SKMT : Security Key Management tool

Figure 5-26 Key Data Used in This Section

Enter the file names of these keys in Security Key Management Tool to generate wrapped key files.
The key files generated here are keys (encrypted keys) wrapped by using a UFPK and combined with a W-UFPK. The generated key files are source code that can be installed in the project.
The W-UFPK is used to remove the wraps by the processing inside the TSIP.

The above key files are specific to the device (thing) created on AWS. You must re-wrap the key files each time you connect to a different device.

The following shows the procedure for wrapping key files.

(2) Generating a root CA certificate signature verification public key

Generate a root CA certificate signature verification public key. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

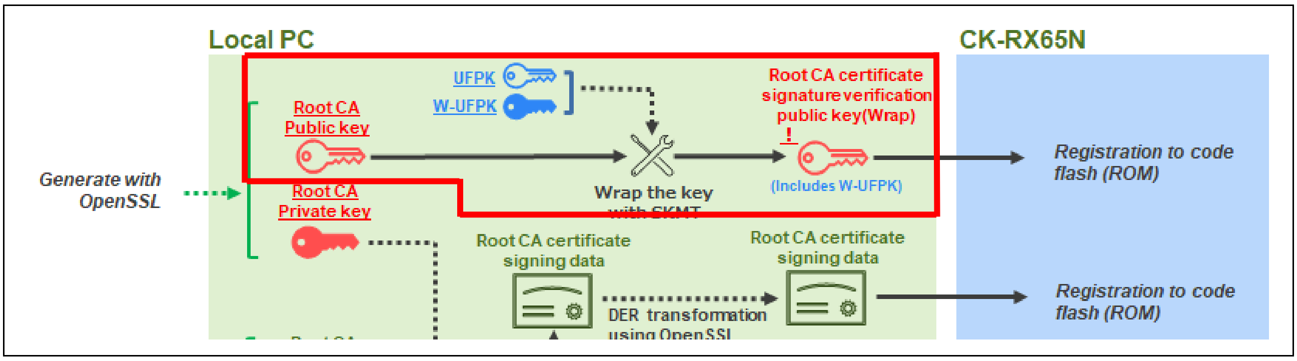
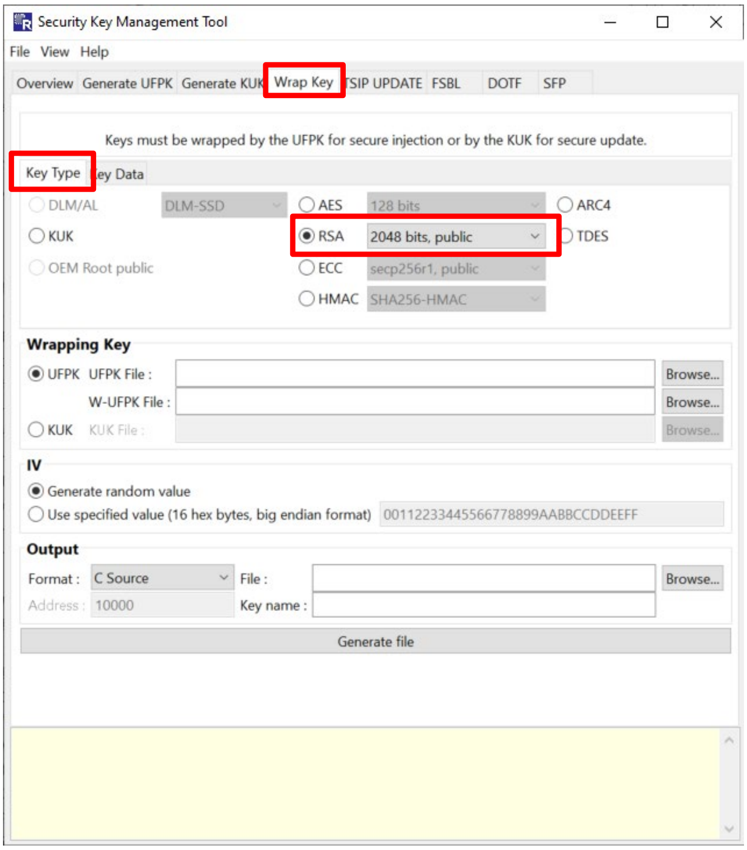


Figure 5-27 Generating a Root CA Certificate Signature Verification Public Key

Create a root CA certificate signature verification public key to be installed in the project, by wrapping the following root CA certificate public key file:

/key_crt_sig_generator /ca-sign-keypair-rsa2048 /rsa2048-public.pem

1. In the Security Key Management Tool application, open the **Wrap key** tab. Then, in the **Key Type** tab, select the **RSA** radio button, and select “2048bits, public” from the drop-down list.



Security Key Management Tool

Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

○ DLM/AL DLM-SSD
○ KUK
○ OEM Root public
○ AES 128 bits
● RSA 2048 bits, public
○ ECC secp256r1, public
○ HMAC SHA256-HMAC
○ ARC4
○ TDES

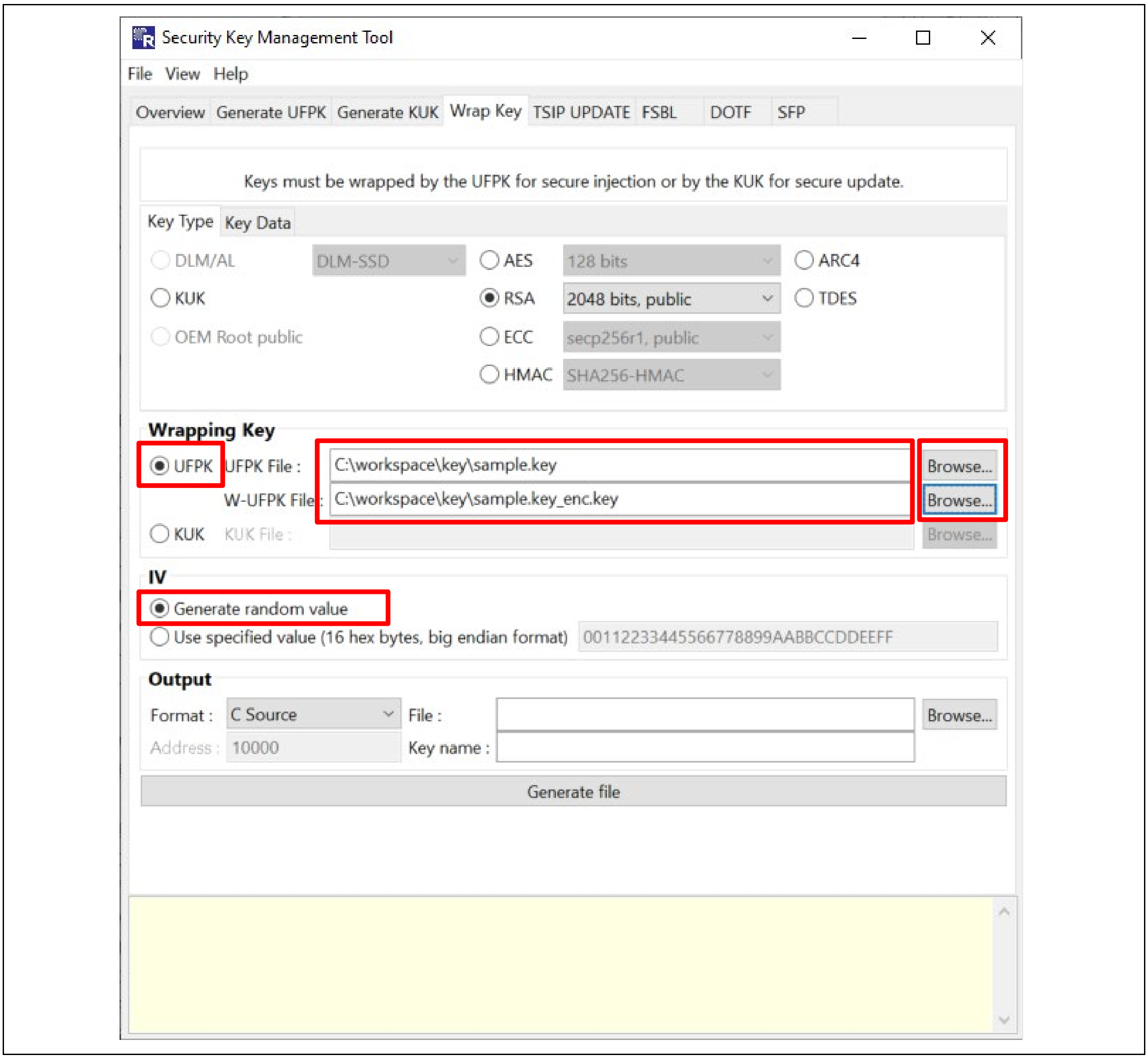
Wrapping Key
● UFPK UFPK File :
W-UFPK File :
○ KUK KUK File :

IV
● Generate random value
○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output
Format : C Source
File :
Address : 10000
Key name :

Figure 5-28 Wrapping a Public key by Using Security Key Management Tool

2. Register the UFPK and W-UFPK. In the **Wrapping Key** area, select the **UFPK** radio button, and then perform the following operations:
   You have a UFPK named **sample.key** and a W-UFPK named **sample.key_enc.key** created in section 5.1.5.2. Specify the UFPK in the **UFPK File** field and the W-UFPK in the **W-UFPK File** field by clicking the **Browse** button for each field.
   In the **IV** area, select the **Generate random value** radio button.



Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

Wrapping Key

● UFPK UFPK File : C:\workspace\key\sample.key

W-UFPK File : C:\workspace\key\sample.key_enc.key

IV

● Generate random value

○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

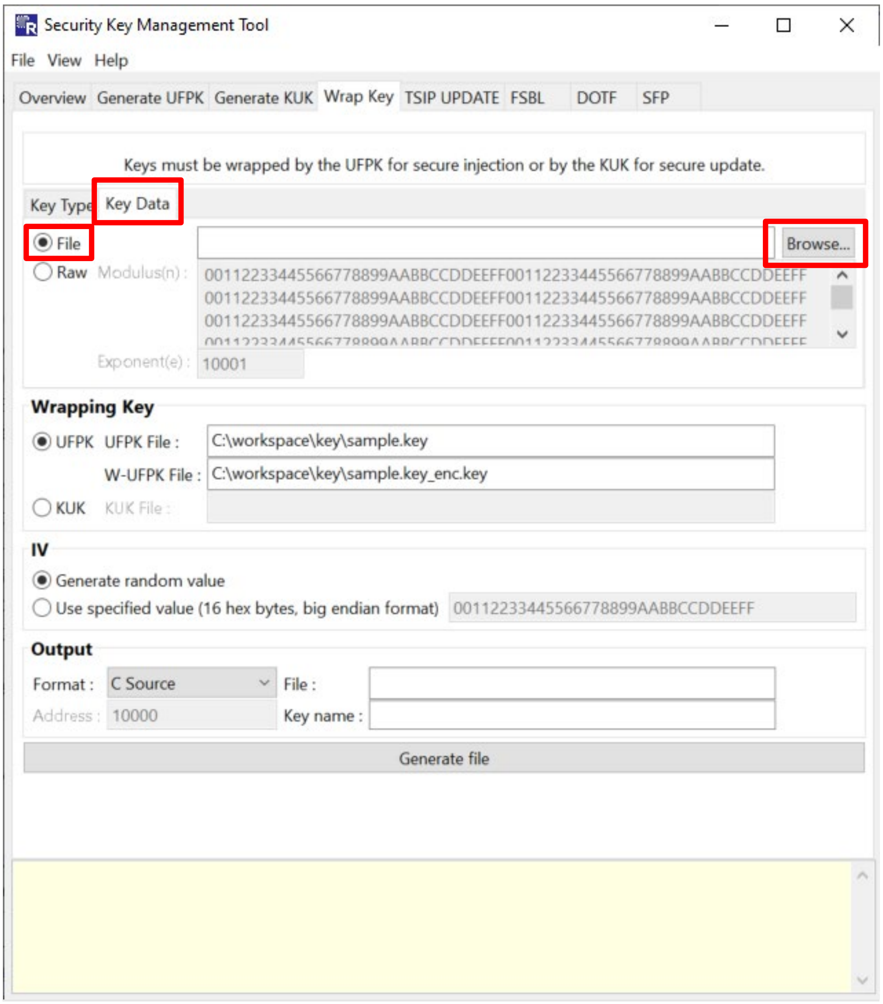
Output

Format : C Source File :

Address : 10000 Key name :

Figure 5-29 Specifying the UFPK and W-UFPK

3. Click the **Key Data** tab.
   Select the **File** radio button, and then click the **Browse** button.



Security Key Management Tool
File View Help
Overview Generate UFPK Generate KUK Wrap Key TSIP UPDATE FSBL DOTF SFP
Key Type Key Data
Browse...
Generate file

Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

● File
○ Raw Modulus(n) : 00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
Exponent(e) : 10001

Wrapping Key

● UFPK UFPK File : C:\workspace\key\sample.key
W-UFPK File : C:\workspace\key\sample.key_enc.key
○ KUK KUK File :

IV

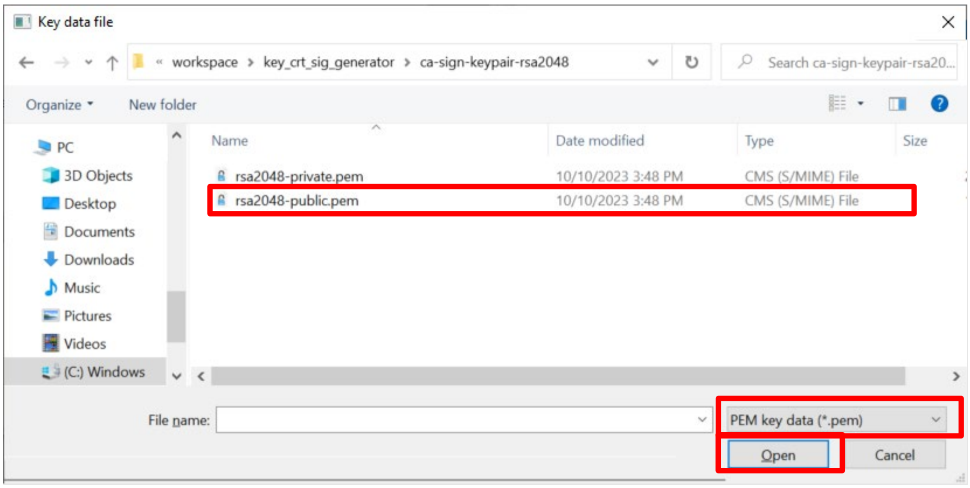
● Generate random value
○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output

Format : C Source File :
Address : 10000 Key name :

Figure 5-30 Selecting the Root CA Certificate Public Key

When the dialog box for selecting the key data file opens, select **PEM key data (*.pem)** as the file type. This allows you to find the root CA certificate public key file (in PEM format) easily. Select the following file, and then click the **Open** button:
key_crt_sig_generator/ca-sign-keypair-rsa2048/rsa2048-public.pem

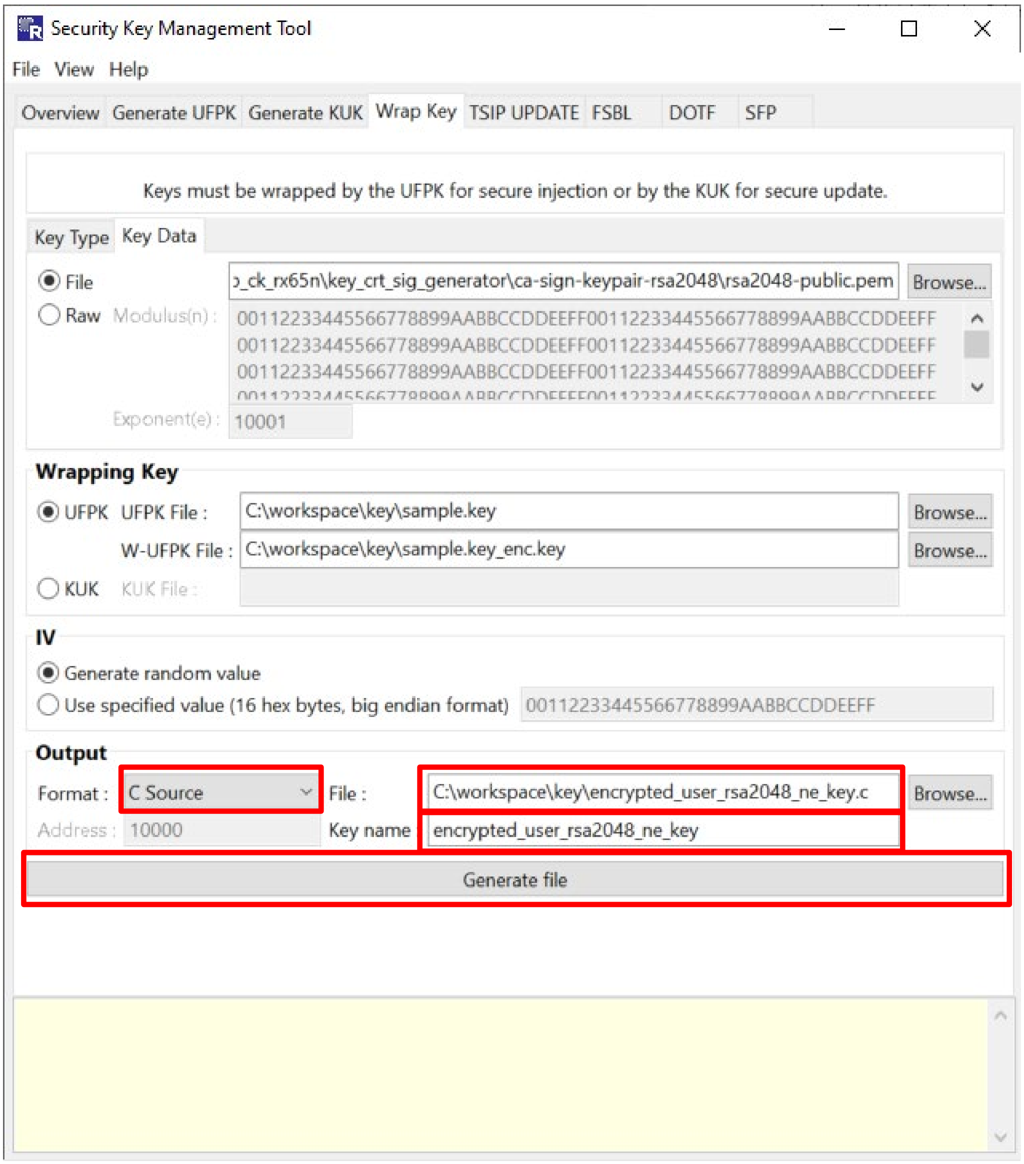


Name	Date modified	Type	Size
rsa2048-private.pem	10/10/2023 3:48 PM	CMS (S/MIME) File	
rsa2048-public.pem	10/10/2023 3:48 PM	CMS (S/MIME) File	

File name:
PEM key data (*.pem)

Figure 5-31 File Selection Dialog Box

4. In the **Output** area, from the **Format** drop-down list, select **C Source**.
   In the **File** text box, select any folder of your choice, and then enter the following file name:
   encrypted_user_rsa2048_ne_key.c
   In the **Key name** text box, enter the following string:
   encrypted_user_rsa2048_ne_key
   When entry is complete, click the **Generate file** button to generate the root CA certificate signature verification public key.
   Note: Always enter the indicated strings without change because they are strings that are hard-coded in the source code.


Security Key Management Tool
File View Help
Overview Generate UFPK Generate KUK Wrap Key TSIP UPDATE FSBL DOTF SFP
Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

● File ɔ_ck_rx65n\key_crt_sig_generator\ca-sign-keypair-rsa2048\rsa2048-public.pem Browse...
○ Raw Modulus(n) : 00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
Exponent(e) : 10001

Wrapping Key
● UFPK UFPK File : C:\workspace\key\sample.key Browse...
W-UFPK File : C:\workspace\key\sample.key_enc.key Browse...
○ KUK KUK File :

IV
● Generate random value
○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output
Format : C Source File : C:\workspace\key\encrypted_user_rsa2048_ne_key.c Browse...
Address : 10000 Key name encrypted_user_rsa2048_ne_key

Figure 5-32 Generating the Data of the Root CA Certificate Signature Verification Public Key

When the following message is displayed at the bottom of the window, the file has been successfully generated. Confirm that the following file has been output to the specified folder:
encrypted_user_rsa2048_ne_key.c/h

The root CA certificate public key generated here will be used as the root CA certificate signature verification public key in the project.

OPERATION SUCCESSFUL

Figure 5-33 Output Result Message

(3) Generating a client certificate public key

This section describes how to generate a client certificate public key. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

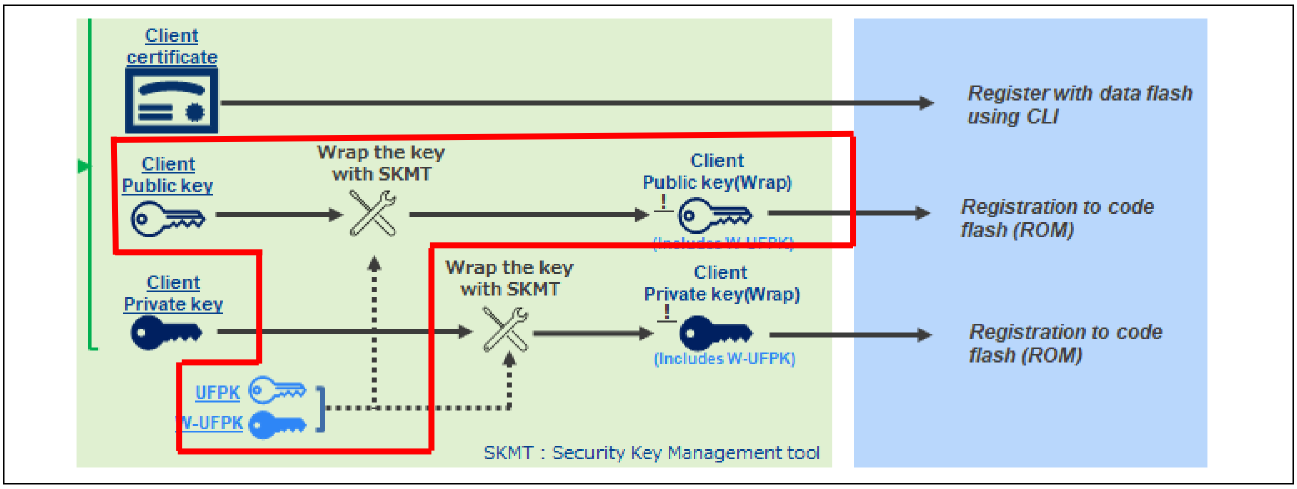


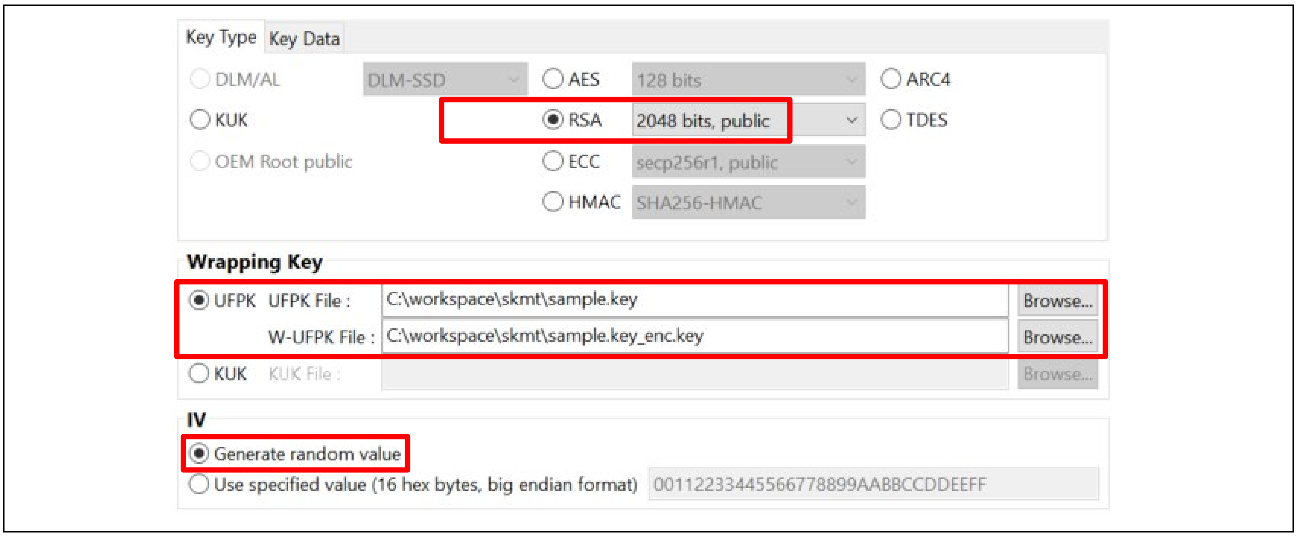
Figure 5-34 Generating a Client Certificate Public Key

Wrap the following key file to create a client certificate public key to be installed in the project:

/key_crt_sig_generator / client-rsa2048 / *xxxx*-public.pem.key

Note: This is a file that was downloaded from AWS. The *xxxx* portion is an arbitrary character string.

1. Generate a client certificate public key.
   First, rename ***xxxx*-public.pem.key** to ***xxxx*-public.pem**.
   Then, open the **Key Type** tab, and make sure that the **RSA** radio button and **2048bits, public** are selected in the same way as in (2). For the **Wrapping Key** and **IV** areas, specify the same settings that were specified in (2).



Key Type | Key Data

○ DLM/AL DLM-SSD ○ AES 128 bits ○ ARC4
○ KUK ● RSA 2048 bits, public ○ TDES
○ OEM Root public ○ ECC secp256r1, public
○ HMAC SHA256-HMAC

Wrapping Key

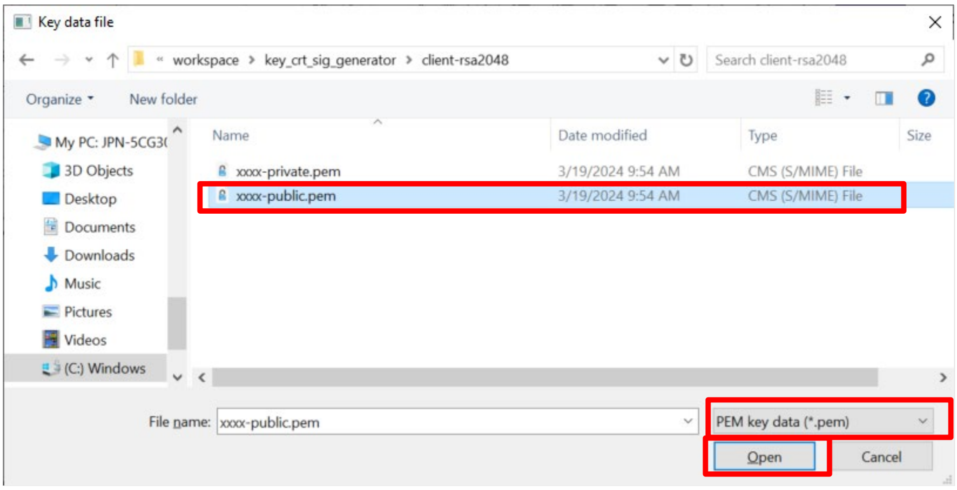
● UFPK UFPK File : C:\workspace\skmt\sample.key Browse...
W-UFPK File : C:\workspace\skmt\sample.key_enc.key Browse...
○ KUK KUK File : Browse...

IV

● Generate random value
○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Figure 5-35 Confirming the Settings in the "Key Type" Tab

2. Open the **Key Data** tab again.
   Click the **Key Data** tab.
   Select the **File** radio button, and then click the **Browse** button.
   When the dialog box for selecting the key data file opens, select **PEM key data (*.pem)** as the file type. This allows you to find the client certificate public key file (in PEM format) easily. Select the following file, and then click the **Open** button:
   key_crt_sig_generator / client-rsa2048 / *xxxx*-public.pem



Key data file

« workspace › key_crt_sig_generator › client-rsa2048

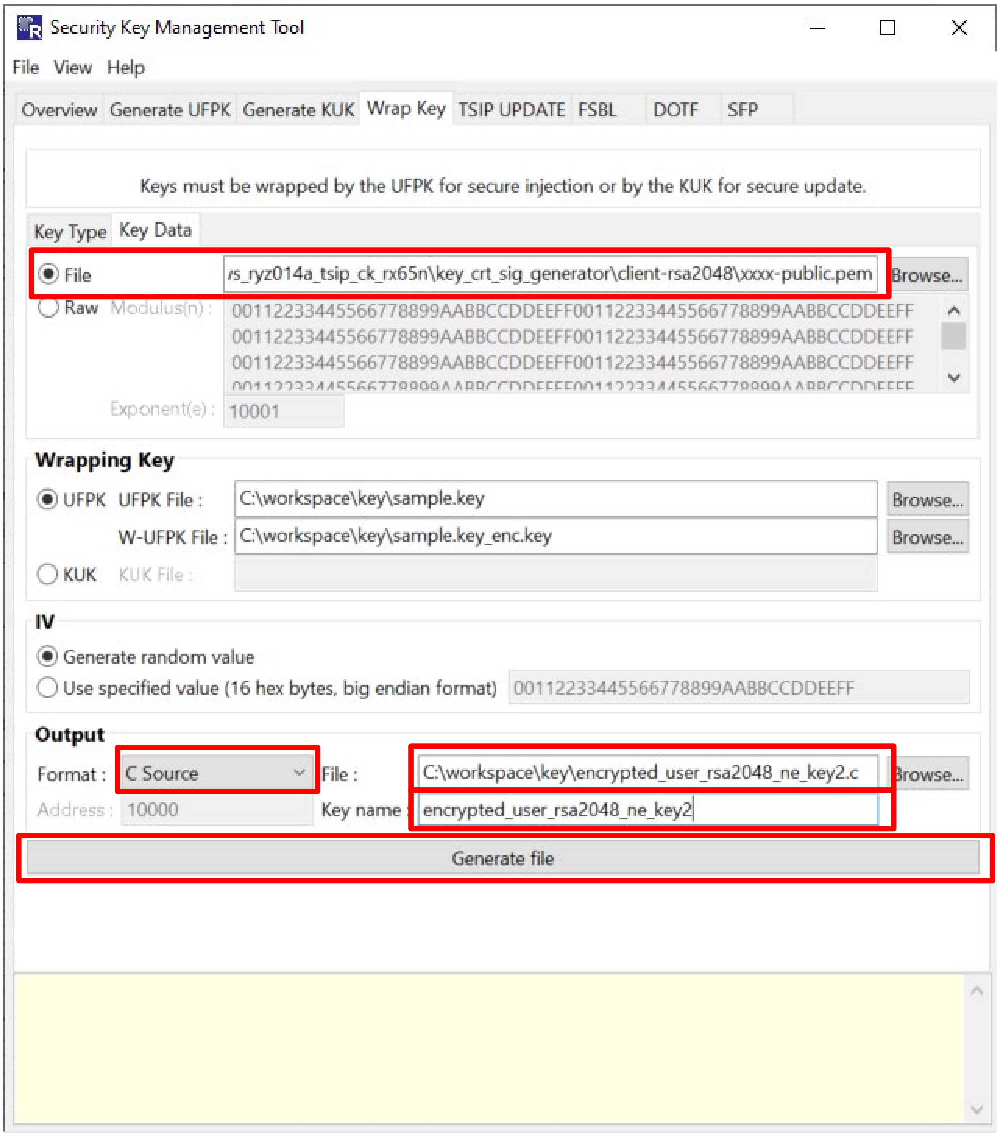
Name	Date modified	Type	Size
xxxx-private.pem	3/19/2024 9:54 AM	CMS (S/MIME) File	
xxxx-public.pem	3/19/2024 9:54 AM	CMS (S/MIME) File	

File name: xxxx-public.pem

PEM key data (*.pem)

Figure 5-36 File Selection Dialog Box

3. In the **Output** area, from the **Format** drop-down list, select **C Source**.
   In the **File** text box, select any folder of your choice, and then enter the following file name:
   encrypted_user_rsa2048_ne_key2.c
   In the **Key name** text box, enter the following string:
   encrypted_user_rsa2048_ne_key2
   When entry is complete, click the **Generate file** button to generate client certificate public key data.
   Note: Always enter the indicated strings without change because they are strings that are hard-coded in the source code.



Security Key Management Tool
File View Help
Overview Generate UFPK Generate KUK Wrap Key TSIP UPDATE FSBL DOTF SFP

Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

● File /s_ryz014a_tsip_ck_rx65n\key_crt_sig_generator\client-rsa2048\xxxx-public.pem Browse...

○ Raw Modulus(n) : 00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF

Exponent(e) : 10001

Wrapping Key

● UFPK UFPK File : C:\workspace\key\sample.key Browse...

W-UFPK File : C:\workspace\key\sample.key_enc.key Browse...

○ KUK KUK File :

IV

● Generate random value

○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

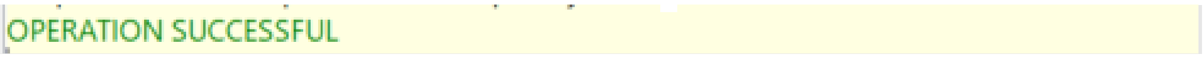
Output

Format : C Source File : C:\workspace\key\encrypted_user_rsa2048_ne_key2.c Browse...

Address : 10000 Key name : encrypted_user_rsa2048_ne_key2

Figure 5-37 Generating the Data of a Client Certificate Public Key

When the following message is displayed at the bottom of the window, the file has been successfully generated. Confirm that the following file has been output to the specified folder:
encrypted_user_rsa2048_ne_key2.c/h



OPERATION SUCCESSFUL

Figure 5-38 Output Result Message

(4) Generating a client certificate private key

This section describes how to generate a client certificate private key. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

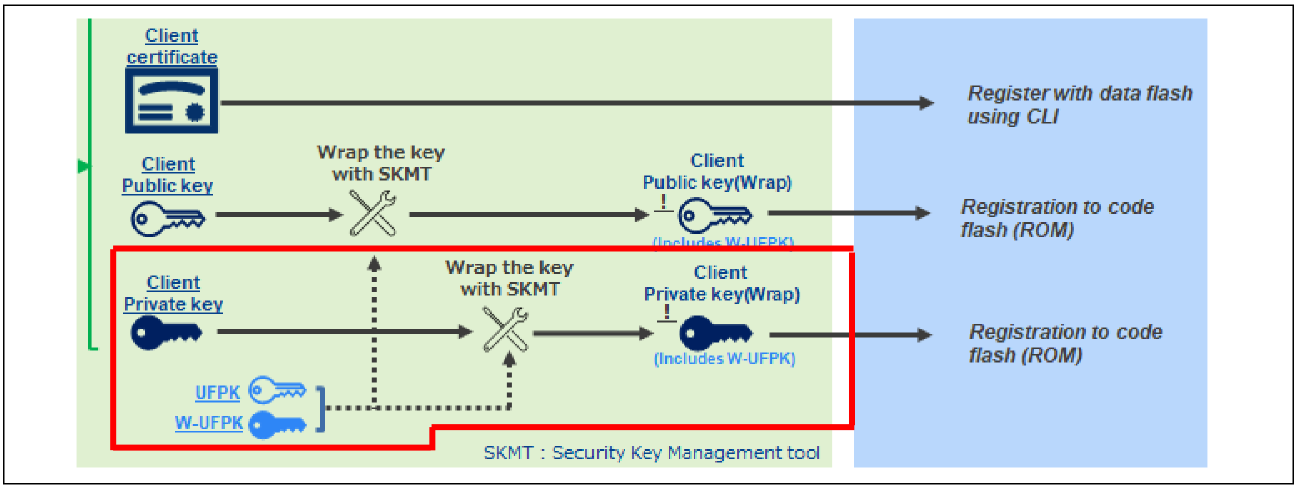


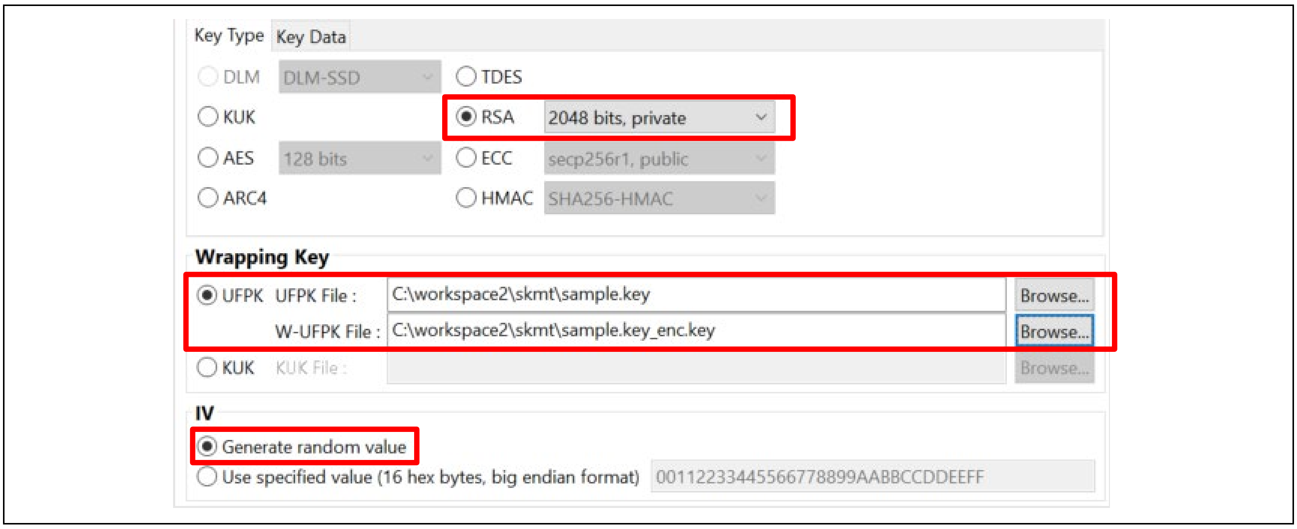
Figure 5-39 Generating a Client Certificate Private Key

Wrap the following key file to create a client certificate private key to be installed in the project:

/key_crt_sig_generator / client-rsa2048 / *xxxx*-private.pem.key

Note: This is a file that was downloaded from AWS. The *xxxx* portion is an arbitrary character string.

1. Generate a client certificate private key.
   First, rename ***xxxx*-private.pem.key** to ***xxxx*-private.pem**.
   Open the **Key Type** tab, and then select the **RSA** radio button and **2048 bits, private**.
   For the **Wrapping Key** and **IV** areas, specify the same settings that were specified in (2).



● RSA 2048 bits, private

Wrapping Key

● UFPK UFPK File : C:\workspace2\skmt\sample.key

W-UFPK File : C:\workspace2\skmt\sample.key_enc.key

○ KUK KUK File :

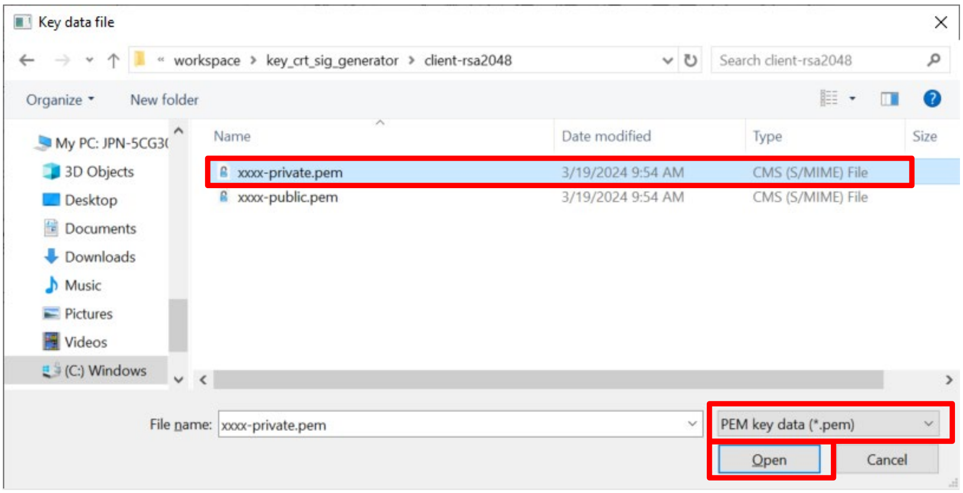
IV

● Generate random value

○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Figure 5-40 Confirming the Settings in the “Key Type” Tab

2. Open the **Key Data** tab again.
   Click the **Key Data** tab.
   Select the **File** radio button, and then click the **Browse** button.
   When the dialog box for selecting the key data file opens, select **PEM key data (*.pem)** as the file type. This allows you to find the client certificate private key file (in PEM format) easily. Select the following file, and then click the **Open** button:
   /key_crt_sig_generator/client-rsa2048/**xxxx*****-private.pem***



« workspace > key_crt_sig_generator > client-rsa2048

Search client-rsa2048

Name	Date modified	Type	Size
xxxx-private.pem	3/19/2024 9:54 AM	CMS (S/MIME) File	
xxxx-public.pem	3/19/2024 9:54 AM	CMS (S/MIME) File	

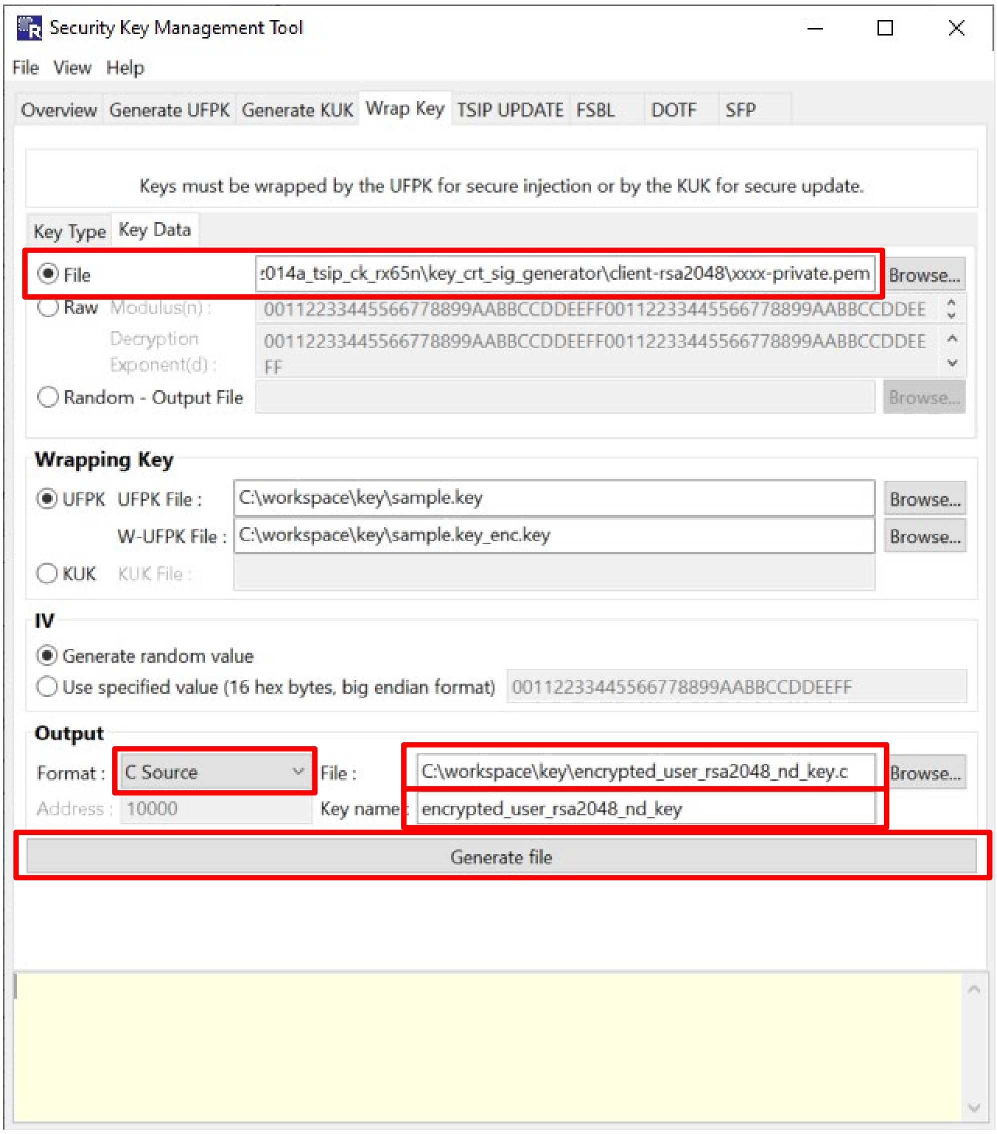
File name: xxxx-private.pem

PEM key data (*.pem)

Figure 5-41 File Selection Dialog Box

3. In the **Output** area, from the **Format** drop-down list, select **C Source**.
   In the **File** text box, select any folder of your choice, and then enter the following file name:
   encrypted_user_rsa2048_nd_key.c
   In the **Key name** text box, enter the following string:
   encrypted_user_rsa2048_nd_key
   When entry is complete, click the **Generate file** button to generate client certificate private key data.

   Note: Always enter the indicated strings without change because they are strings that are hard-coded in the source code.



Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

● File :014a_tsip_ck_rx65n\key_crt_sig_generator\client-rsa2048\xxxx-private.pem
○ Raw Modulus(n) : 00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEE
Decryption Exponent(d) : 00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEE FF
○ Random - Output File

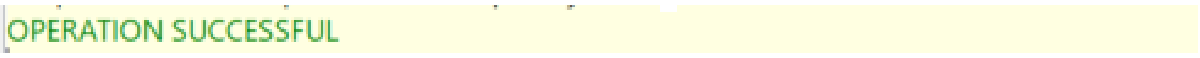
Wrapping Key
● UFPK UFPK File : C:\workspace\key\sample.key
W-UFPK File : C:\workspace\key\sample.key_enc.key
○ KUK KUK File :

IV
● Generate random value
○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output
Format : C Source
File : C:\workspace\key\encrypted_user_rsa2048_nd_key.c
Address : 10000
Key name : encrypted_user_rsa2048_nd_key

Figure 5-42 Generating the Data of a Client Certificate Private Key

When the following message is displayed at the bottom of the window, the file has been successfully generated. Confirm that the following file has been output to the specified folder:
encrypted_user_rsa2048_nd_key.c/h



```
OPERATION SUCCESSFUL
```

Figure 5-43 Output Result Message

Use the procedures shown in (2) to (4) to register the following wrapped key files in the source code. The task performed in this section corresponds to the range enclosed in the red frame of the flow shown in Figure 5-1.

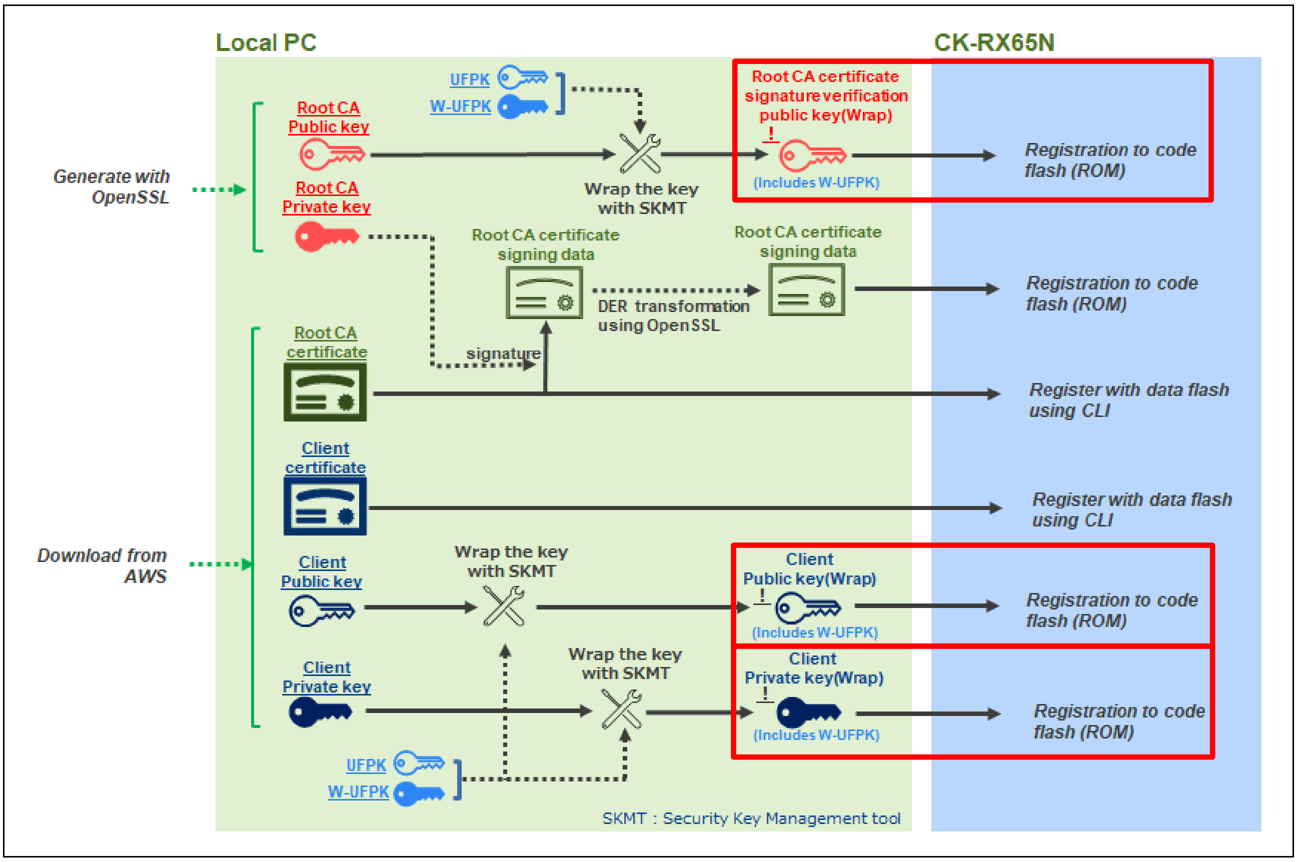

Local PC
CK-RX65N
Generate with OpenSSL
Root CA Public key
Root CA Private key
UFPK
W-UFPK
Wrap the key with SKMT
Root CA certificate signature verification public key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
Root CA certificate signing data
DER transformation using OpenSSL
Root CA certificate signing data
Registration to code flash (ROM)
signature
Root CA certificate
Register with data flash using CLI
Client certificate
Register with data flash using CLI
Download from AWS
Client Public key
Wrap the key with SKMT
Client Public key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
Client Private key
Wrap the key with SKMT
Client Private key(Wrap)
(Includes W-UFPK)
Registration to code flash (ROM)
UFPK
W-UFPK
SKMT : Security Key Management tool

Figure 5-44 Registering Generated Key Data in the Source Code

The following table lists the files of generated keys.

Table 5-4 List of Wrapped Key Files

Name	File name
Root CA certificate signature verification public key	- encrypted_user_rsa2048_ne_key.c - encrypted_user_rsa2048_ne_key.h
Client certificate public key	- encrypted_user_rsa2048_ne_key2.c - encrypted_user_rsa2048_ne_key2.h
Client certificate private key	- encrypted_user_rsa2048_nd_key.c - encrypted_user_rsa2048_nd_key.h

Copy the above six files to the following user data folder in the project, overwriting the existing files with the same names:

```
\iot-reference-
rx\Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\src\userdata_tsip
```

5.2 Generating a Key Pair and Certificates for an OTA Update

In an OTA firmware update, certificates and a key pair are used to verify whether the firmware used has not been subject to tampering.

Generate ECDSA certificates and keys by performing the procedure described in section 4.1, Generating Key pairs and certificates in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037).

The following ECDSA certificates, public key, and private key are generated. These items will be used in the settings when building a project and creating an OTA job for AWS.

• ECDSA public key:	secp256r1.publickey
• ECDSA private key:	secp256r1.privatekey
• ECDSA certificate (key pair certificate):	secp256r1.crt
• ECDSA certificate chain:	ca.crt

6. Building a Project

You created projects in Chapter 4, Preparing for the Demo Project, and certificates and keys in Chapter 5, Creating Keys and Certificates. Next, you use these items to create a project that demonstrates OTA update using the TSIP.

The following two kinds of firmware are required to demonstrate OTA update. In this chapter, you create the two kinds of firmware from the project.

- Initial firmware
- Update firmware

6.1 Building and Executing the Initial Version of Firmware

Build the initial version of firmware.

6.1.1 Importing Projects

Import the following two projects into e^2 studio and specify the initial settings by referring to Chapter 4, Preparing for the Demo Project:

- Bootloader: boot_loader_ck_rx65n
- Demo application (cellular version): aws_ryz014a_tsip_ck_rx65n

Also, prepare the following seven certificate and key files that were created in Chapter 5, Creating Keys and Certificates, and store them in the user data folder:

1.	**AmazonRootCA1_sig_array.txt**:	Root CA certificate signature data
2.	**encrypted_user_rsa2048_ne_key.c**:	Root CA certificate signature verification public key (source file)
3.	**encrypted_user_rsa2048_ne_key.h**:	Same as above (header file)
4.	**encrypted_user_rsa2048_ne_key2.c**:	Client certificate public key (source file)
5.	**encrypted_user_rsa2048_ne_key2.h**:	Same as above (header file)
6.	**encrypted_user_rsa2048_nd_key.c**:	Client certificate private key (source file)
7.	**encrypted_user_rsa2048_nd_key.h**:	Same as above (header file)

User data folder:

```
\iot-reference-
rx\Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\src\userdata_tsip
```

The data of these certificates and keys is installed in the program when the demo application is built.

The key files 2 through 7 are converted into index data that is to be used to inject keys during program execution. These files are written to the data flash memory only when the program is executed for the first time.

Note: 1. When the program is executed again, these key files are no longer written to the data flash memory and the key index information saved in the data flash memory is fetched for use from the data flash memory.
If the data flash memory is cleared, the key index information is rewritten when the program is executed again.

Note: 2. If you changed the key file, be sure to refer to section 6.1.5(6) and delete the data flash when executing the program.

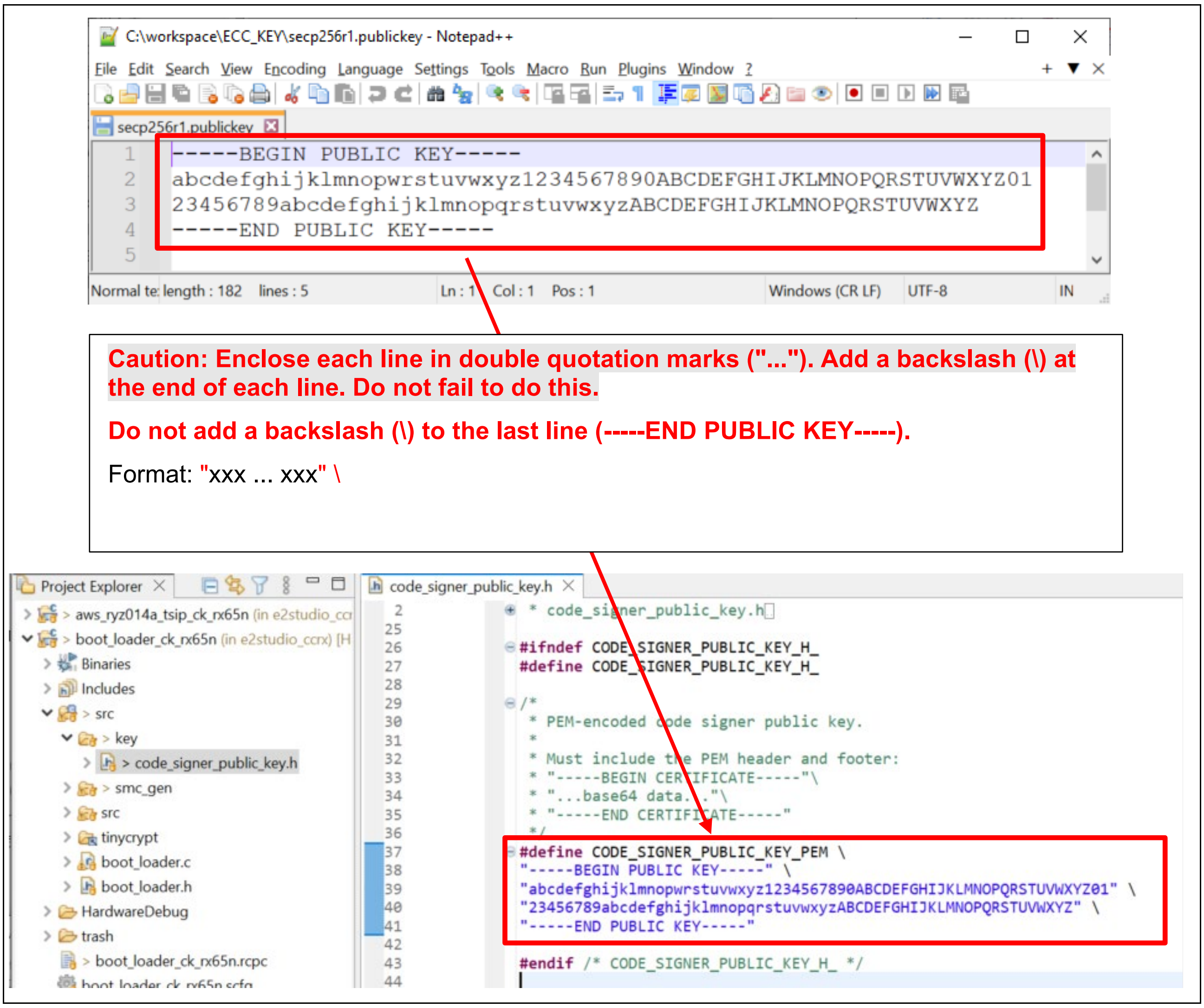
6.1.2 Setting Up and Building the Projects

Specify the settings for executing OTA update in the imported projects, and then build the projects.

(1) Setting a public key in the Bootloader

Prepare the **secp256r1.publickey** file (an ECDSA public key created in section 5.2). Open the file with a text editor, copy the content, and then paste it under the CODE_SIGNER_PUBLIC_KEY_PEM entry in the following file of the bootloader:

```
boot_loader_ck_rx65n\src¥key¥code_signer_public_key.h
```



C:\workspace\ECC_KEY\secp256r1.publickey - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
secp256r1.publickey
Normal te: length : 182 lines : 5
Ln : 1 Col : 1 Pos : 1
Windows (CR LF) UTF-8 IN

```
-----BEGIN PUBLIC KEY-----
abcdefghijklmnopwrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ01
23456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
-----END PUBLIC KEY-----
```

Caution: Enclose each line in double quotation marks ("..."). Add a backslash (\) at the end of each line. Do not fail to do this.

Do not add a backslash (\) to the last line (-----END PUBLIC KEY-----).

Format: "xxx ... xxx" \

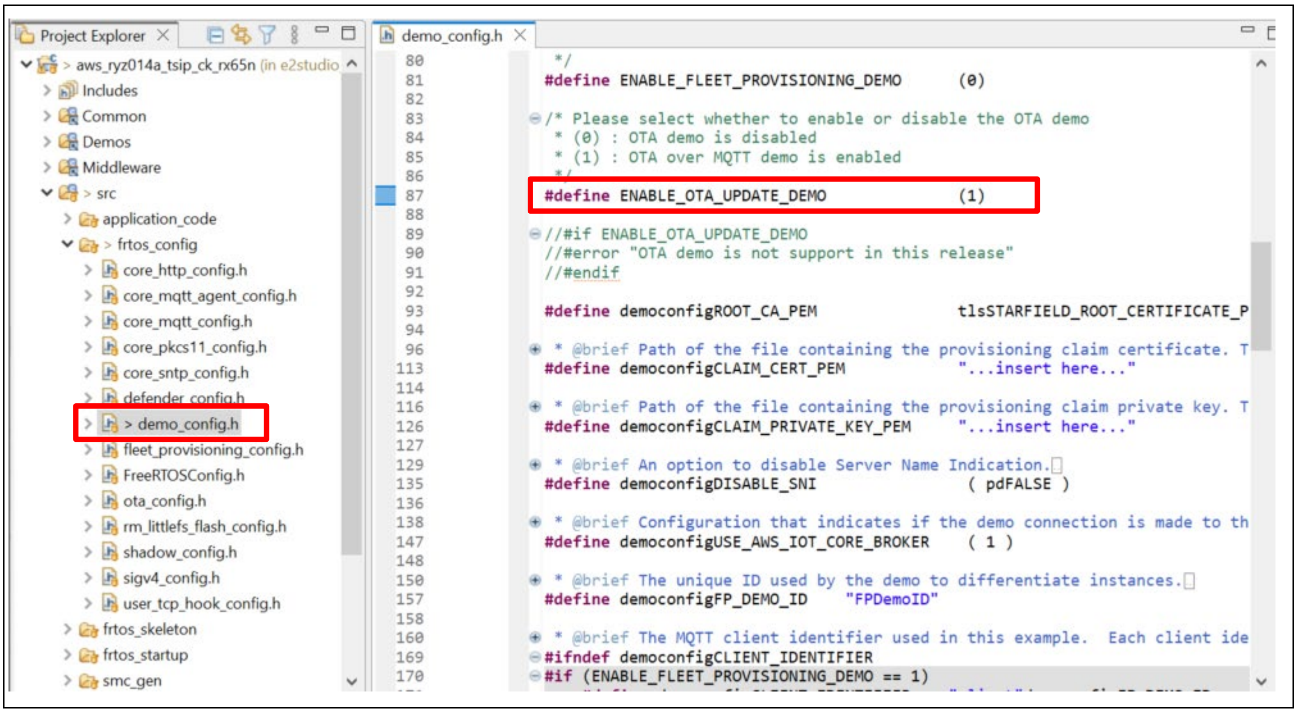
Project Explorer
> aws_ryz014a_tsip_ck_rx65n (in e2studio_ccı
> boot_loader_ck_rx65n (in e2studio_ccrx) [H
Binaries
Includes
> src
> key
> code_signer_public_key.h
> smc_gen
src
tinycrypt
boot_loader.c
boot_loader.h
HardwareDebug
trash
> boot_loader_ck_rx65n.rcpc
boot_loader_ck_rx65n.scfg
code_signer_public_key.h

```
 * code_signer_public_key.h

#ifndef CODE_SIGNER_PUBLIC_KEY_H_
#define CODE_SIGNER_PUBLIC_KEY_H_

/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"\
 * "...base64 data..."\
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM \
"-----BEGIN PUBLIC KEY-----" \
"abcdefghijklmnopwrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ01" \
"23456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ" \
"-----END PUBLIC KEY-----"

#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

Figure 6-1 Setting a Public Key in the Bootloader

(2) Enabling the OTA update demonstration definition of the demo application

In the **aws_ryz014a_tsip_ck_rx65n\src\frtos_confit\demo_config.h** file, set the ENABLE_OTA_UPDATE_DEMO definition to 1 (enabled). (By default, this definition is set to 0.)



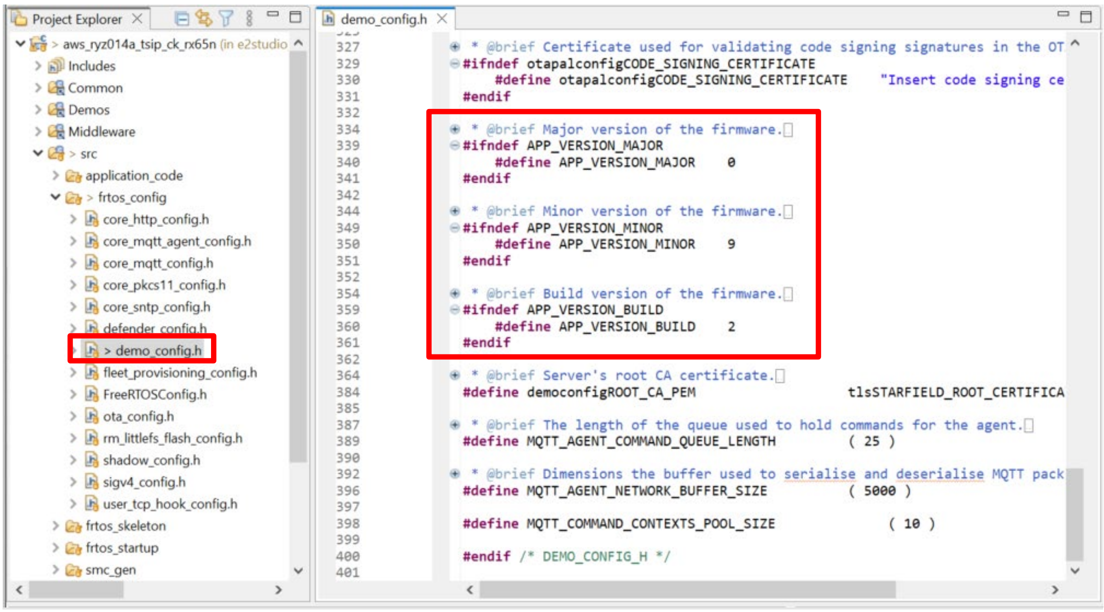
```
 */
#define ENABLE_FLEET_PROVISIONING_DEMO      (0)

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

//#if ENABLE_OTA_UPDATE_DEMO
//#error "OTA demo is not support in this release"
//#endif

#define democonfigROOT_CA_PEM               tlsSTARFIELD_ROOT_CERTIFICATE_P

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

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

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

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

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

 * @brief The MQTT client identifier used in this example.  Each client ide
#ifndef democonfigCLIENT_IDENTIFIER
#if (ENABLE_FLEET_PROVISIONING_DEMO == 1)
```

Figure 6-2 OTA Update Demonstration Definition

(3) Confirming that the initial version of the Demo application is 0.92

Confirm that the version definitions in the **aws_ryz014a_tsip _ck_rx65n\src\frtos_config\ demo_config.h** file are as follows:

APP_VERSION_MAJOR	0
APP_VERSION_MINOR	9
APP_VERSION_BUILD	2

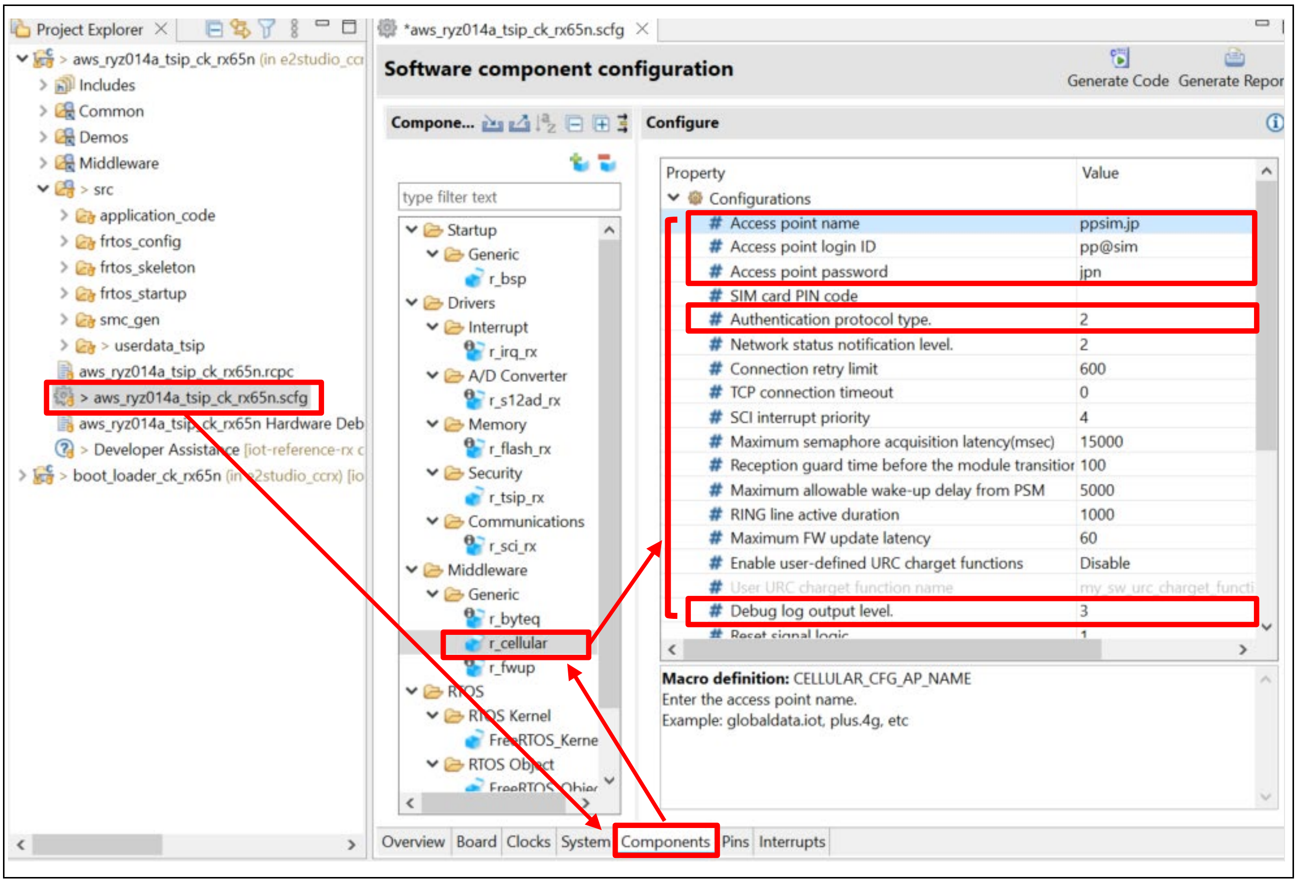


```
 * @brief Certificate used for validating code signing signatures in the OT
#ifndef otapalconfigCODE_SIGNING_CERTIFICATE
    #define otapalconfigCODE_SIGNING_CERTIFICATE    "Insert code signing ce
#endif

 * @brief Major version of the firmware.
#ifndef APP_VERSION_MAJOR
    #define APP_VERSION_MAJOR    0
#endif

 * @brief Minor version of the firmware.
#ifndef APP_VERSION_MINOR
    #define APP_VERSION_MINOR    9
#endif

 * @brief Build version of the firmware.
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    2
#endif

 * @brief Server's root CA certificate.
#define democonfigROOT_CA_PEM                   tlsSTARFIELD_ROOT_CERTIFICA

 * @brief The length of the queue used to hold commands for the agent.
#define MQTT_AGENT_COMMAND_QUEUE_LENGTH         ( 25 )

 * @brief Dimensions the buffer used to serialise and deserialise MQTT pack
#define MQTT_AGENT_NETWORK_BUFFER_SIZE          ( 5000 )

#define MQTT_COMMAND_CONTEXTS_POOL_SIZE              ( 10 )

#endif /* DEMO_CONFIG_H */
```

Figure 6-3 Version Settings

(4) Setting up the “r_cellular” module (FIT module that controls the RYZ014A cellular module)

Open Smart Configurator (aws_ryz014a_tsip_ck_rx65n.scfg), and then select the **Components** tab.
In the “r_cellular” module, set the **Access point name**, **Access point login ID**, **Access point password**, and **Authentication protocol type** configurations according to the SIM card. Also, set the **Debug log output level** configuration to 3.

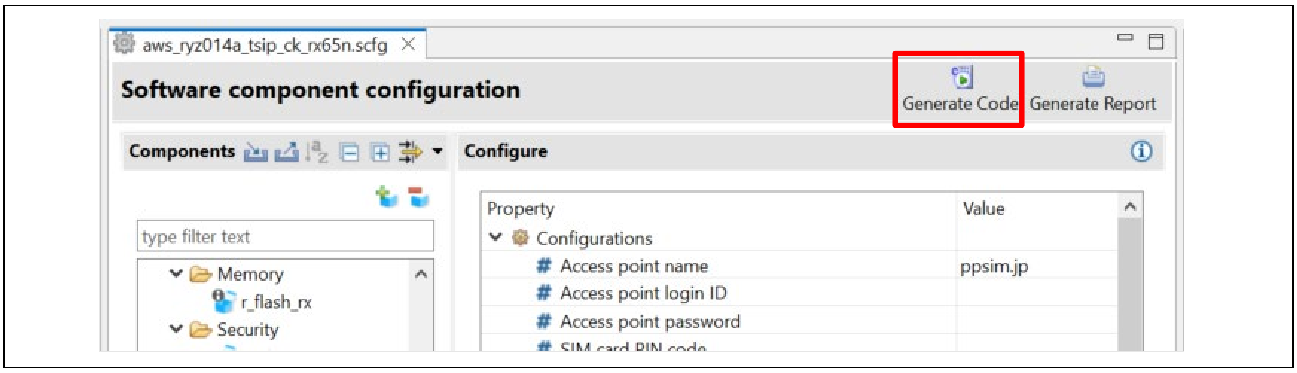


Property	Value
Configurations	
Access point name	ppsim.jp
Access point login ID	pp@sim
Access point password	jpn
SIM card PIN code	
Authentication protocol type.	2
Network status notification level.	2
Connection retry limit	600
TCP connection timeout	0
SCI interrupt priority	4
Maximum semaphore acquisition latency(msec)	15000
Reception guard time before the module transitior	100
Maximum allowable wake-up delay from PSM	5000
RING line active duration	1000
Maximum FW update latency	60
Enable user-defined URC charget functions	Disable
User URC charget function name	my_sw_urc_charget_functi
Debug log output level.	3
Reset signal logic	1

Macro definition: CELLULAR_CFG_AP_NAME
Enter the access point name.
Example: globaldata.iot, plus.4g, etc

Figure 6-4 Setting Up the “r_cellular” Module

After you have changed the settings as described above, click the **Generate Code** button at the top right of the window. The changes (made in Smart Configurator) are applied to the relevant code.



Property	Value
Configurations	
Access point name	ppsim.jp
Access point login ID	
Access point password	
SIM card PIN code	

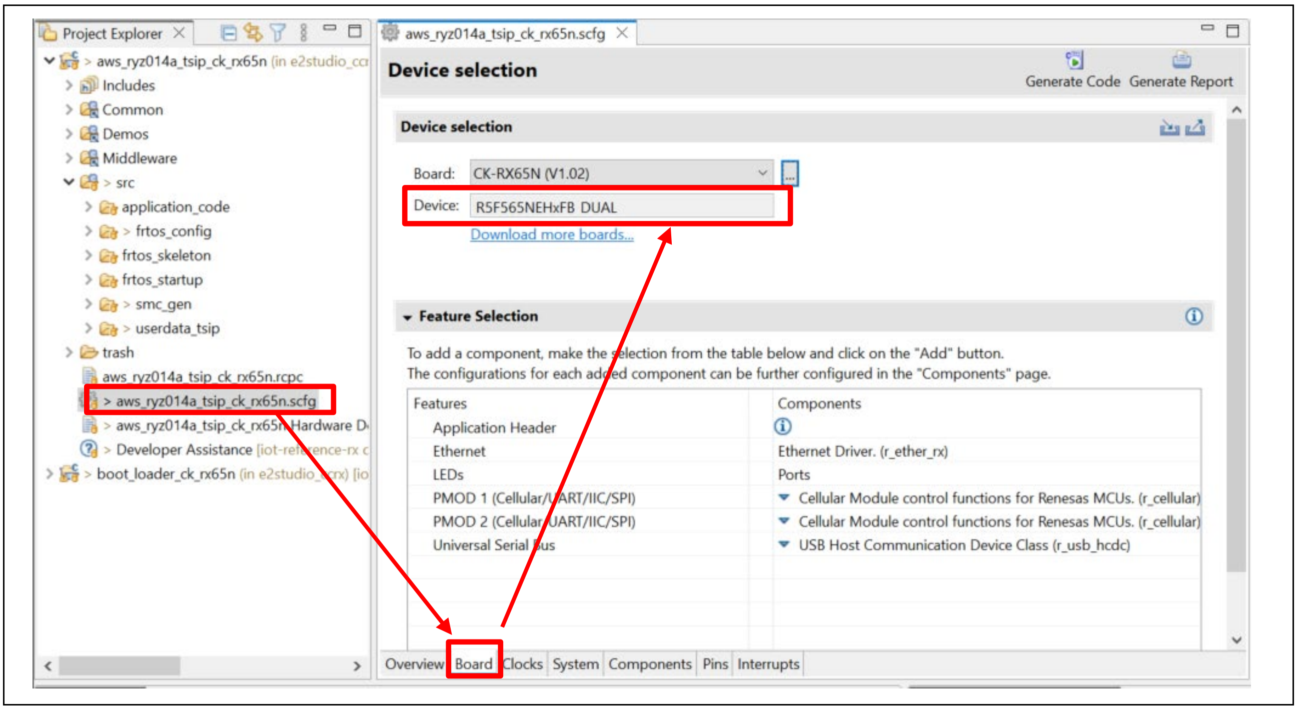
Figure 6-5 Generating Code

If you use the SIM card bundled with the CK-RX65N, activate it by referring to section 4.1.5, Activating SIM card in the following application note:

SIM activation, Creating the trial account and using Dashboard with RYZ014A or Ethernet Application for AWS - Getting Started Guide (R01QS0064)

(5) Setting the device in the firmware

1. Open Smart Configurator (aws_ryz014a_tsip_ck_rx65n.scfg), select the **Board** tab, and then check whether “R5F565NEHxFB DUAL” is set in **Device**.



Project Explorer

- aws_ryz014a_tsip_ck_rx65n (in e2studio_ccr
  - Includes
  - Common
  - Demos
  - Middleware
  - src
    - application_code
    - frtos_config
    - frtos_skeleton
    - frtos_startup
    - smc_gen
    - userdata_tsip
  - trash
  - aws_ryz014a_tsip_ck_rx65n.rcpc
  - aws_ryz014a_tsip_ck_rx65n.scfg
  - aws_ryz014a_tsip_ck_rx65n Hardware D
  - Developer Assistance [iot-reference-rx c
- boot_loader_ck_rx65n (in e2studio_ccrx) [io

aws_ryz014a_tsip_ck_rx65n.scfg

Device selection

Generate Code Generate Report

Device selection

Board: CK-RX65N (V1.02) ...

Device: R5F565NEHxFB DUAL

Download more boards...

Feature Selection

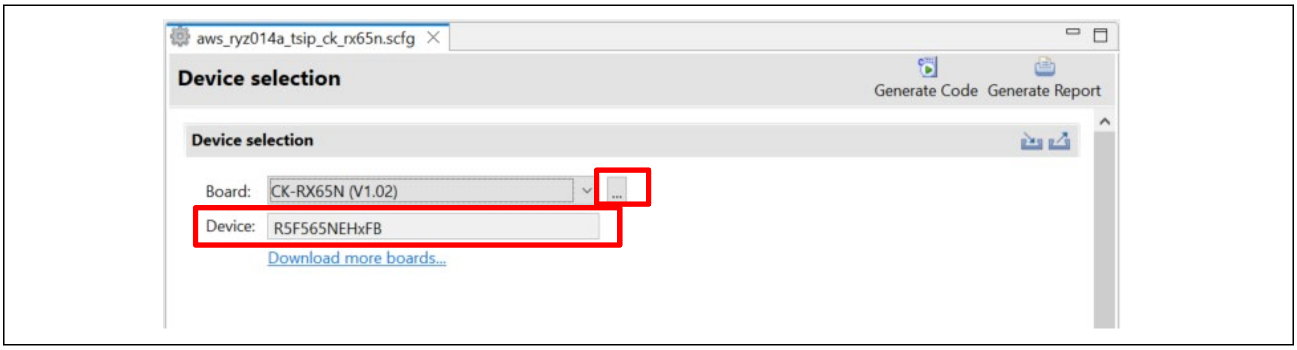
To add a component, make the selection from the table below and click on the "Add" button.
The configurations for each added component can be further configured in the "Components" page.

Features	Components
Application Header	ⓘ
Ethernet	Ethernet Driver. (r_ether_rx)
LEDs	Ports
PMOD 1 (Cellular/UART/IIC/SPI)	Cellular Module control functions for Renesas MCUs. (r_cellular)
PMOD 2 (Cellular/UART/IIC/SPI)	Cellular Module control functions for Renesas MCUs. (r_cellular)
Universal Serial Bus	USB Host Communication Device Class (r_usb_hcdc)

Figure 6-6 Confirming the Device Setting

If another device (“R5F565NEHxFB” in this example) has been set, perform the following steps to change the setting.

2. Click the **...** button to the right of the **Board** drop-down list.



aws_ryz014a_tsip_ck_rx65n.scfg

Device selection

Generate Code Generate Report

Device selection

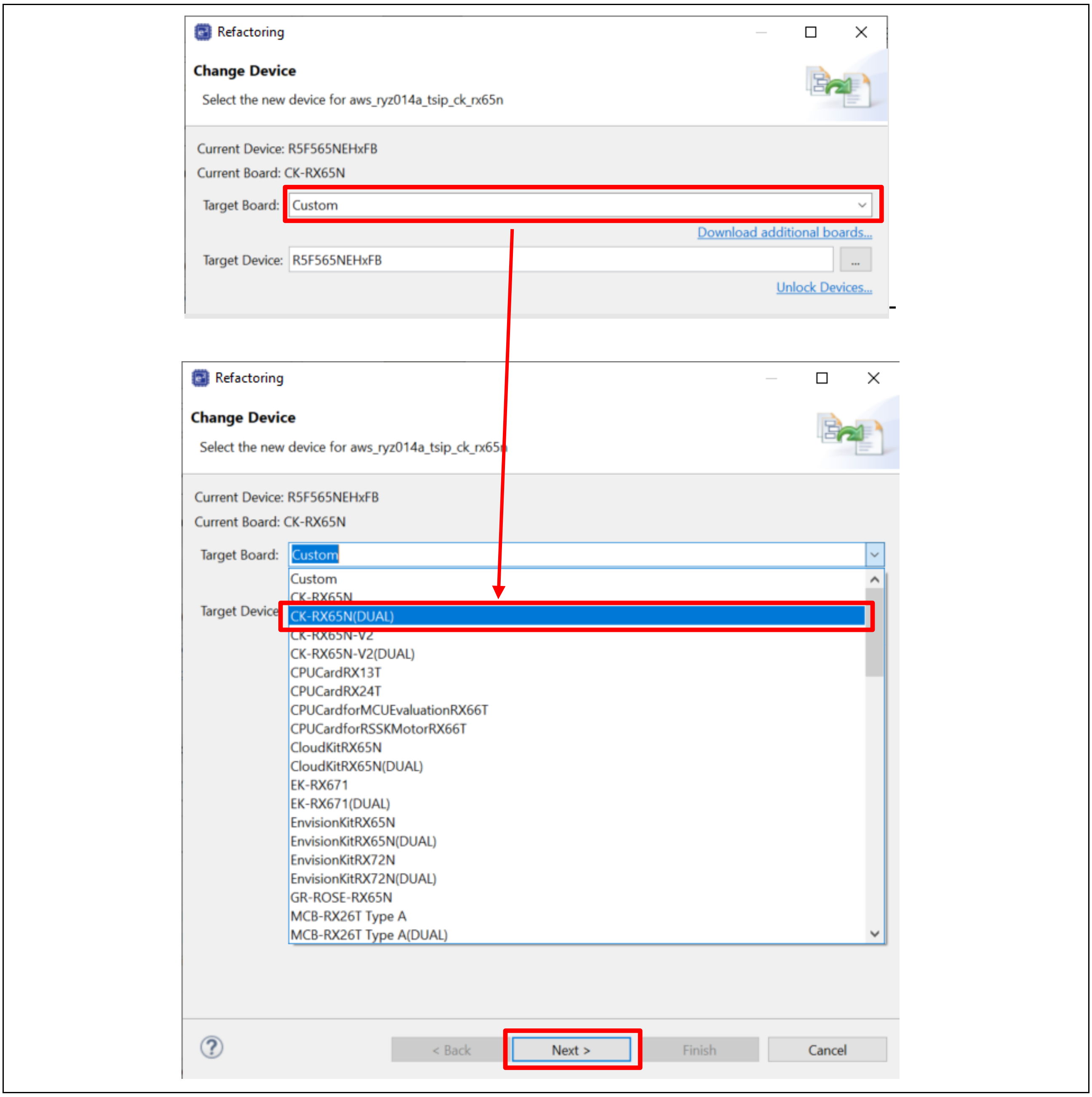
Board: CK-RX65N (V1.02) ...

Device: R5F565NEHxFB

Download more boards...

Figure 6-7 Changing the Device Setting (1)

3. The **Change Device** dialog box appears.
   From the **Target Board** drop-down list, select “CK-RX65N(DUAL)”, and then click the **Next** button.



Refactoring

Change Device

Select the new device for aws_ryz014a_tsip_ck_rx65n

Current Device: R5F565NEHxFB

Current Board: CK-RX65N

Target Board: Custom

Download additional boards...

Target Device: R5F565NEHxFB

Unlock Devices...

Refactoring

Change Device

Select the new device for aws_ryz014a_tsip_ck_rx65n

Current Device: R5F565NEHxFB

Current Board: CK-RX65N

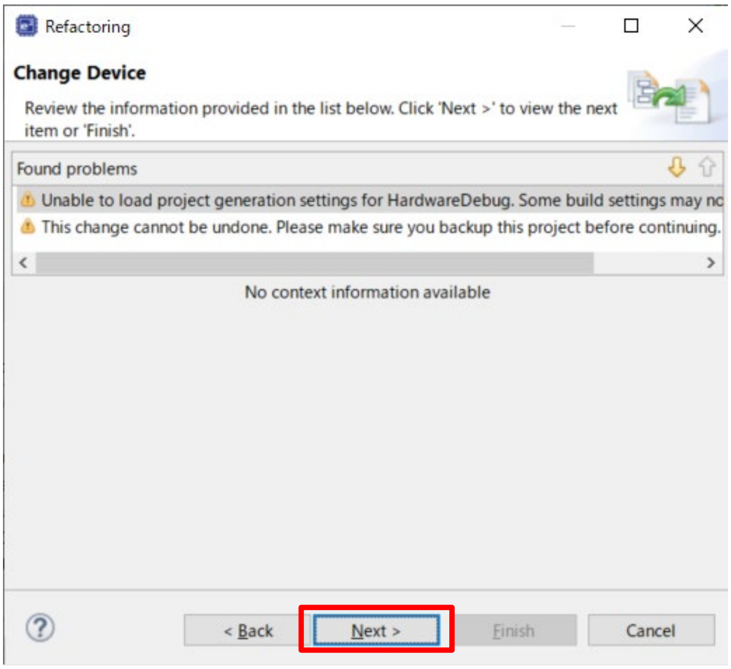
Target Board: Custom

Target Device

Custom
CK-RX65N
CK-RX65N(DUAL)
CK-RX65N-V2
CK-RX65N-V2(DUAL)
CPUCardRX13T
CPUCardRX24T
CPUCardforMCUEvaluationRX66T
CPUCardforRSSKMotorRX66T
CloudKitRX65N
CloudKitRX65N(DUAL)
EK-RX671
EK-RX671(DUAL)
EnvisionKitRX65N
EnvisionKitRX65N(DUAL)
EnvisionKitRX72N
EnvisionKitRX72N(DUAL)
GR-ROSE-RX65N
MCB-RX26T Type A
MCB-RX26T Type A(DUAL)

Figure 6-8 Change Device Dialog Box

4. If the device is changed, the following dialog box (**Found problems**) appears. In this dialog box, click **Next**.



Refactoring

Change Device

Review the information provided in the list below. Click 'Next >' to view the next item or 'Finish'.

Found problems

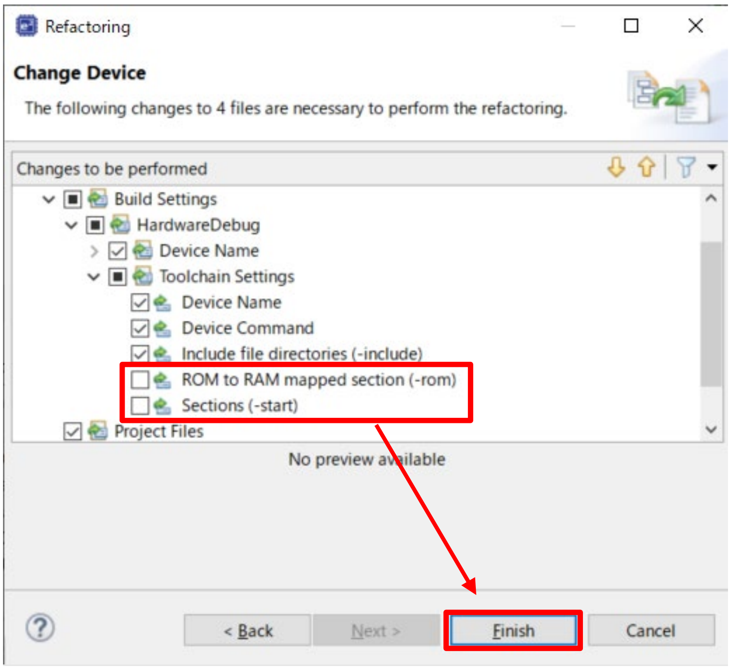
Unable to load project generation settings for HardwareDebug. Some build settings may nc

This change cannot be undone. Please make sure you backup this project before continuing.

No context information available

Figure 6-9 "Found problems" Dialog Box

5. When the following dialog box (**Change to be performed**) appears, select the following check boxes: **Build Settings** > **HardwareDebug** > **Toolchain Settings**. Then, clear the **ROM to RAM mapped section (-rom)** and **Sections (-start)** check boxes. Then, click **Finish**.



Refactoring

Change Device

The following changes to 4 files are necessary to perform the refactoring.

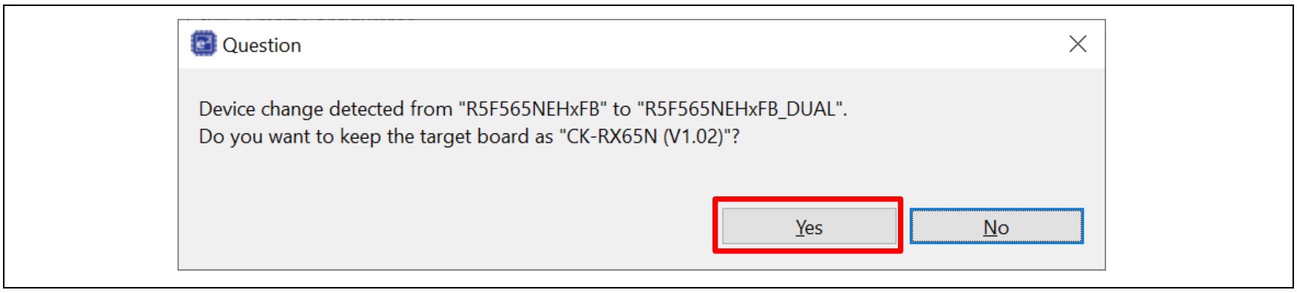
Changes to be performed

- ■ Build Settings
  - ■ HardwareDebug
    - ☑ Device Name
    - ■ Toolchain Settings
      - ☑ Device Name
      - ☑ Device Command
      - ☑ Include file directories (-include)
      - □ ROM to RAM mapped section (-rom)
      - □ Sections (-start)
- ☑ Project Files

No preview available

Figure 6-10 "Change to be performed" Dialog Box

If you change the device, the following dialog box may appear, asking you whether to retain the current target board (“CK-RX65N (V1.02)”). If this dialog box, appears, click the **Yes** button.

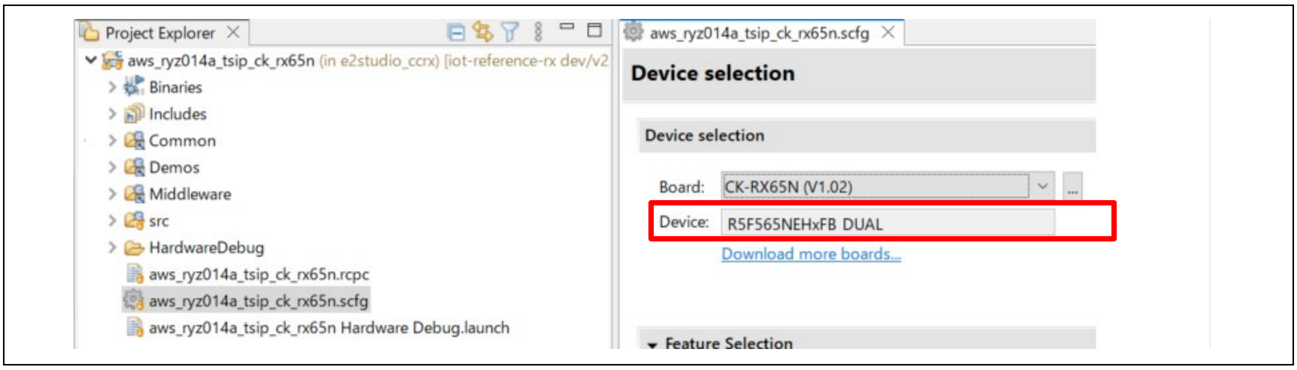


Question

Device change detected from "R5F565NEHxFB" to "R5F565NEHxFB_DUAL".
Do you want to keep the target board as "CK-RX65N (V1.02)"?

Figure 6-11 Dialog Box Asking Whether to Retain the Target Board

6. The device setting is changed to “R5F565NEHxFB DUAL”.



Project Explorer
- aws_ryz014a_tsip_ck_rx65n (in e2studio_ccrx) [iot-reference-rx dev/v2
  - Binaries
  - Includes
  - Common
  - Demos
  - Middleware
  - src
  - HardwareDebug
  - aws_ryz014a_tsip_ck_rx65n.rcpc
  - aws_ryz014a_tsip_ck_rx65n.scfg
  - aws_ryz014a_tsip_ck_rx65n Hardware Debug.launch

aws_ryz014a_tsip_ck_rx65n.scfg

Device selection

Device selection

Board: CK-RX65N (V1.02)

Device: R5F565NEHxFB DUAL

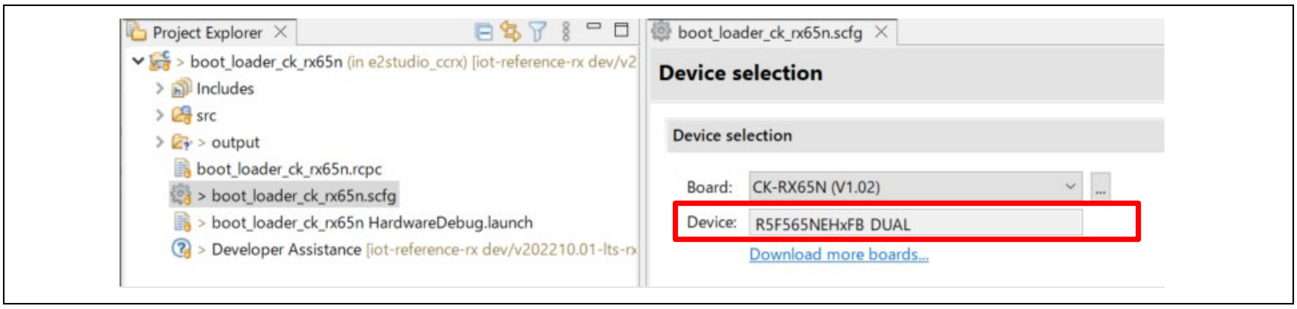
Download more boards...

Feature Selection

Figure 6-12 Changed Device Setting

(6) Confirming the device setting in the bootloader

Open **boot_loader_ck_rx65n.scfg**, and then select the **Board** tab. In the same way as in (5), confirm that “R5F565NEHxFB DUAL” is set in **Device**.



Project Explorer
- > boot_loader_ck_rx65n (in e2studio_ccrx) [iot-reference-rx dev/v2
  - Includes
  - src
  - > output
  - boot_loader_ck_rx65n.rcpc
  - > boot_loader_ck_rx65n.scfg
  - > boot_loader_ck_rx65n HardwareDebug.launch
  - > Developer Assistance [iot-reference-rx dev/v202210.01-lts-r

boot_loader_ck_rx65n.scfg

Device selection

Device selection

Board: CK-RX65N (V1.02)

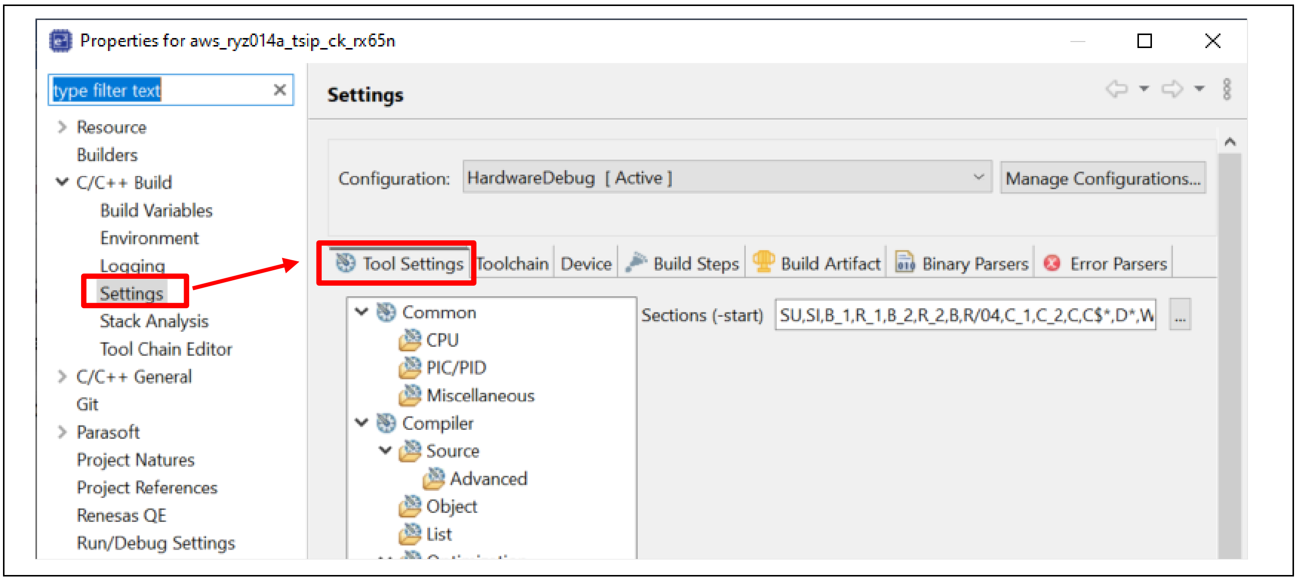
Device: R5F565NEHxFB DUAL

Download more boards...

Figure 6-13 Device Setting in the Bootloader

(7) Changing the address of the vector table in the firmware (aws_ryz014a_tsip_ck_rx65n)

In the “aws_ryz014a_tsip_ck_rx65n” project, select **Project** > **Properties**. In the properties window that appears, select **C/C++Build** > **Settings**, and then click the **Tool Settings** tab.

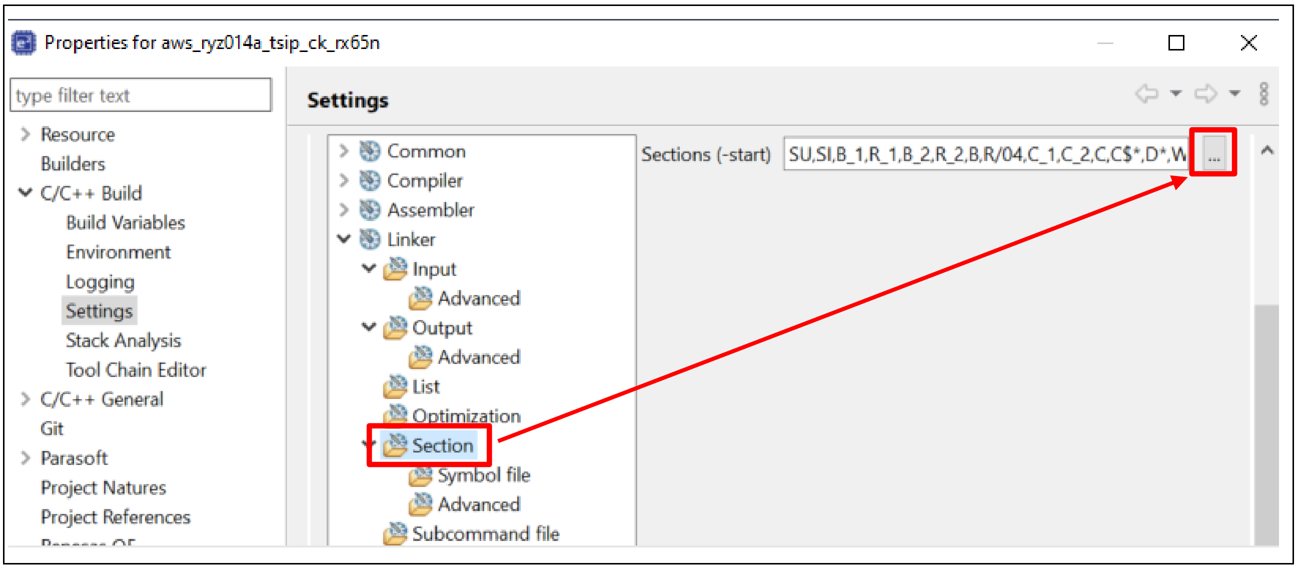


Configuration: HardwareDebug [Active]

Sections (-start) SU,SI,B_1,R_1,B_2,R_2,B,R/04,C_1,C_2,C,C$*,D*,W

Figure 6-14 Tool Settings Tab

In the **Settings** tree view, select **Linker** > **Section**, and then click **...** on the right of **Sections** to open Section Viewer.



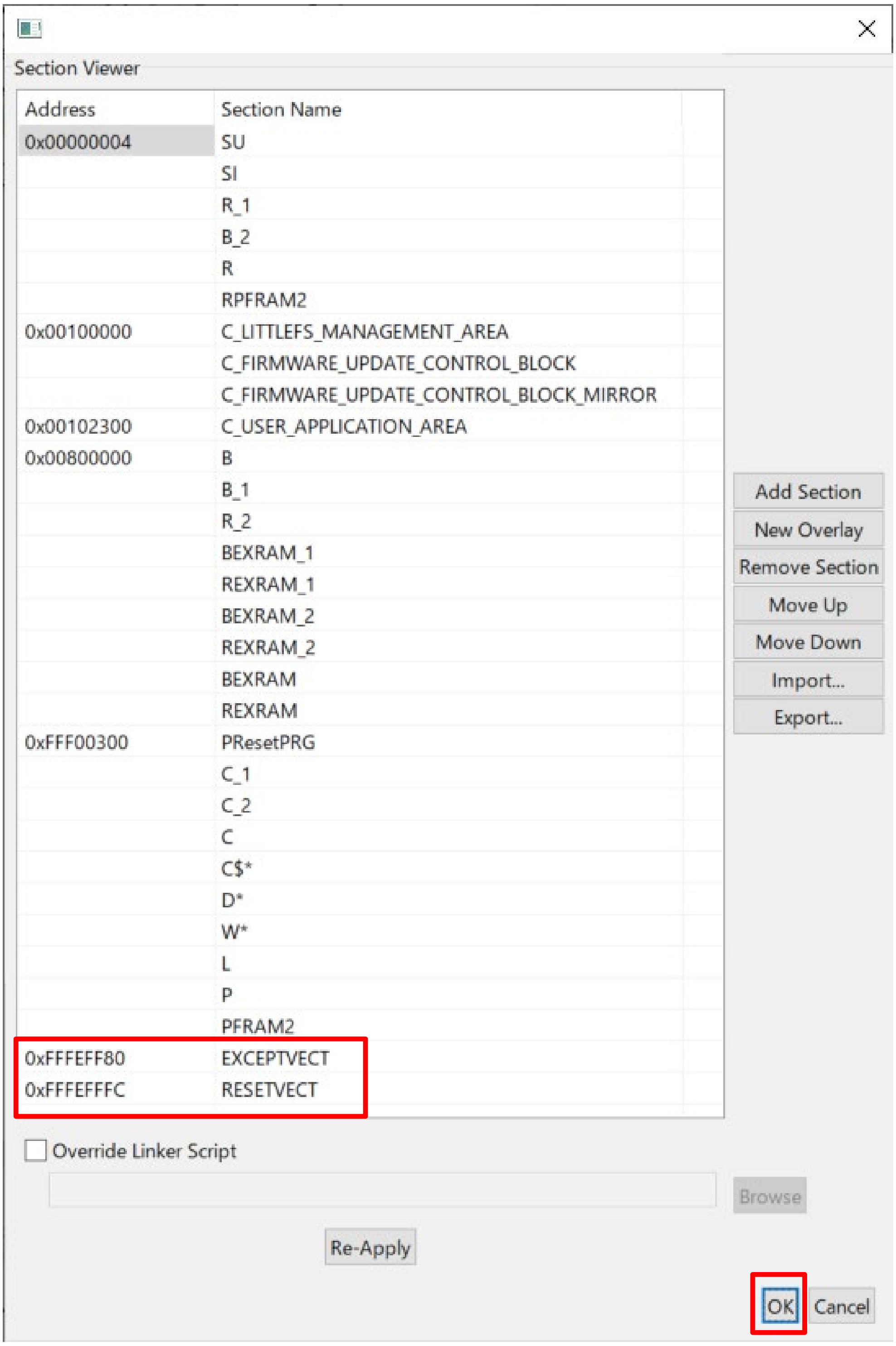
Sections (-start) SU,SI,B_1,R_1,B_2,R_2,B,R/04,C_1,C_2,C,C$*,D*,W

Figure 6-15 Settings Tree View

In the **Section Viewer** window, change the settings as follows:

- EXCEPTVECT: 0xFFFFFF80 → 0xFFFEFF80
- RESETVECT: 0xFFFFFFFC → 0xFFFEFFFC

When you have completed the settings, click the **OK** button.



Section Viewer

Address	Section Name
0x00000004	SU
	SI
	R_1
	B_2
	R
	RPFRAM2
0x00100000	C_LITTLEFS_MANAGEMENT_AREA
	C_FIRMWARE_UPDATE_CONTROL_BLOCK
	C_FIRMWARE_UPDATE_CONTROL_BLOCK_MIRROR
0x00102300	C_USER_APPLICATION_AREA
0x00800000	B
	B_1
	R_2
	BEXRAM_1
	REXRAM_1
	BEXRAM_2
	REXRAM_2
	BEXRAM
	REXRAM
0xFFF00300	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P
	PFRAM2
0xFFFEFF80	EXCEPTVECT
0xFFFEFFFC	RESETVECT

□ Override Linker Script

Figure 6-16 Section Viewer

(8) Building the projects

When you have completed all necessary settings, build the bootloader and demo application projects, and then confirm that no errors occur. The build results are stored in the following folders: In these folders, .mot files are generated as built projects.

- Bootloader
  `\iot-reference-rx\Projects\boot_loader_ck_rx65n\e2studio_ccrx HardwareDebug`
  .mot file: boot_loader_ck_rx65n.mot
- Demo application
  `\iot-reference-rx\Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\HardwareDebug`
  .mot file: aws_ryz014a_tsip_ck_rx65n.mot

6.1.3 Creating the Initial Firmware

In this section, you create the initial firmware by combining a bootloader (boot_loader_ck_rx65n) and firmware (aws_ryz014a_tsip_ck_rx65n).

Renesas Image Generator is necessary when you create firmware. Install Python and Renesas Image Generator by referring to sections 2.2 and 2.4 in the following application note:
“RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037).

Also, Renesas Flash Programmer is necessary when you write firmware to the target board.

(1) Using Renesas Image Generator to generate the initial firmware

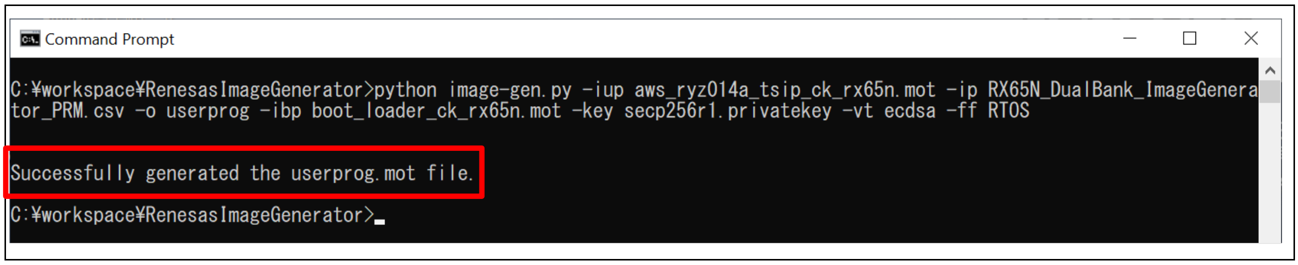
Store the following files in the Renesas Image Generator installation folder:

• Build result of the demo application created in section 6.1.2:	aws_ryz014a_tsip_ck_rx65n.mot
• Build result of the bootloader created in section 6.1.2:	boot_loader_ck_rx65n.mot
• ECDSA private key created in section 5.2:	secp256r1.privatekey

Open the Command Prompt window, change the current directory to the **RenesasImageGenerator** directory, and then generate the **userprog.mot** file by executing the following command:

```
python image-gen.py -iup aws_ryz014a_tsip_ck_rx65n.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o userprog -ibp
boot_loader_ck_rx65n.mot -key secp256r1.privatekey -vt ecdsa -ff RTOS
```

Generation of the file takes some time.



Command Prompt

```
C:¥workspace¥RenesasImageGenerator>python image-gen.py -iup aws_ryz014a_tsip_ck_rx65n.mot -ip RX65N_DualBank_ImageGenera
tor_PRM.csv -o userprog -ibp boot_loader_ck_rx65n.mot -key secp256r1.privatekey -vt ecdsa -ff RTOS

Successfully generated the userprog.mot file.

C:¥workspace¥RenesasImageGenerator>_
```

Figure 6-17 Generating the Initial Firmware

Generation is complete when the following message is displayed on the command line: “Successfully generated the userprog.mot file.”

The initial firmware is created with the following file name:

- userprog.mot

(2) Using Renesas Flash Programmer to write the initial firmware to the target board (CK-RX65N)

1. Install the flash memory programming tool (Renesas Flash Programmer).
   Download "Renesas Flash Programmer V3.14.00 Windows" from the download website of the flash memory programming tool, and then install the tool.
2. Connect the CK-RX65N v1 to a PC by referring to section 2.5 in the following application note: "RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)" (R01AN7037).
3. Start Renesas Flash Programmer, and then open the device erase project (**erase.rpj**).
   The **erase.rpj** project is stored in the following folder of the sample program:
   `\Projects\aws_ryz014a_tsip_ck_rx65n\flash_project\erase_from_bank1`

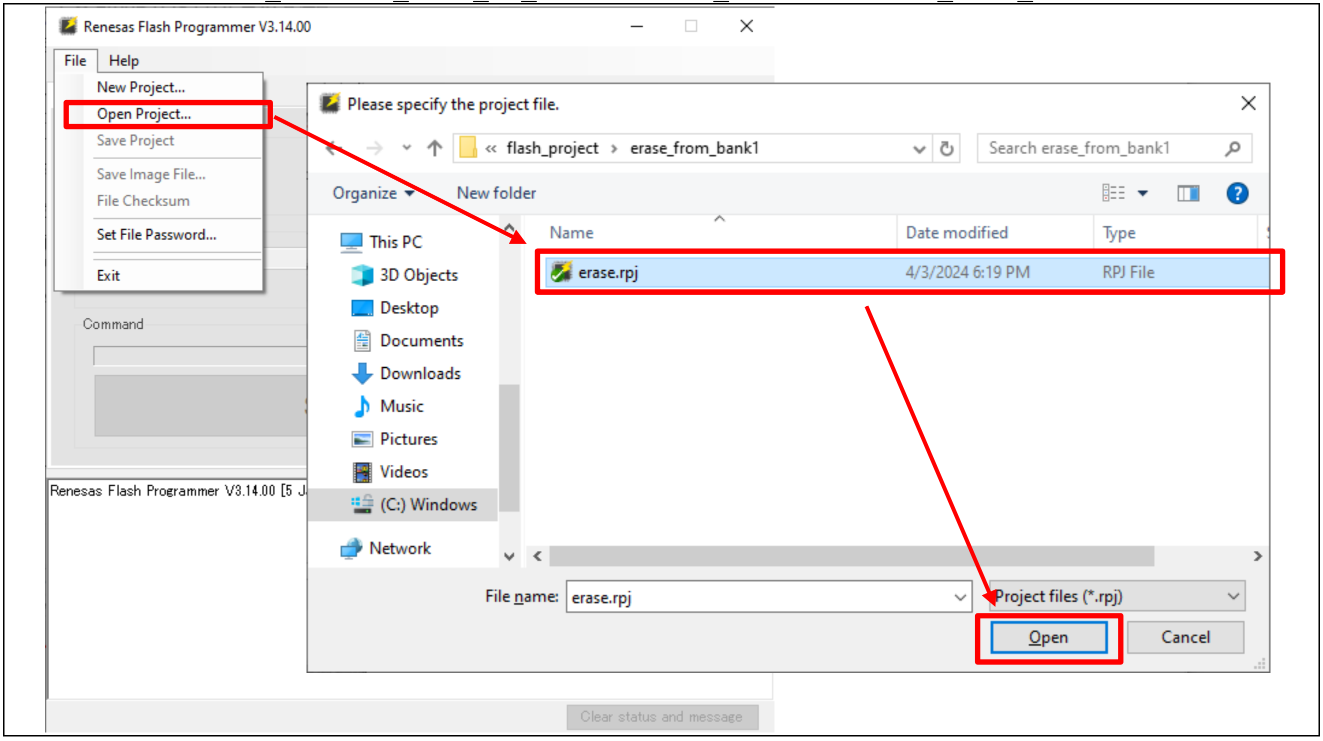
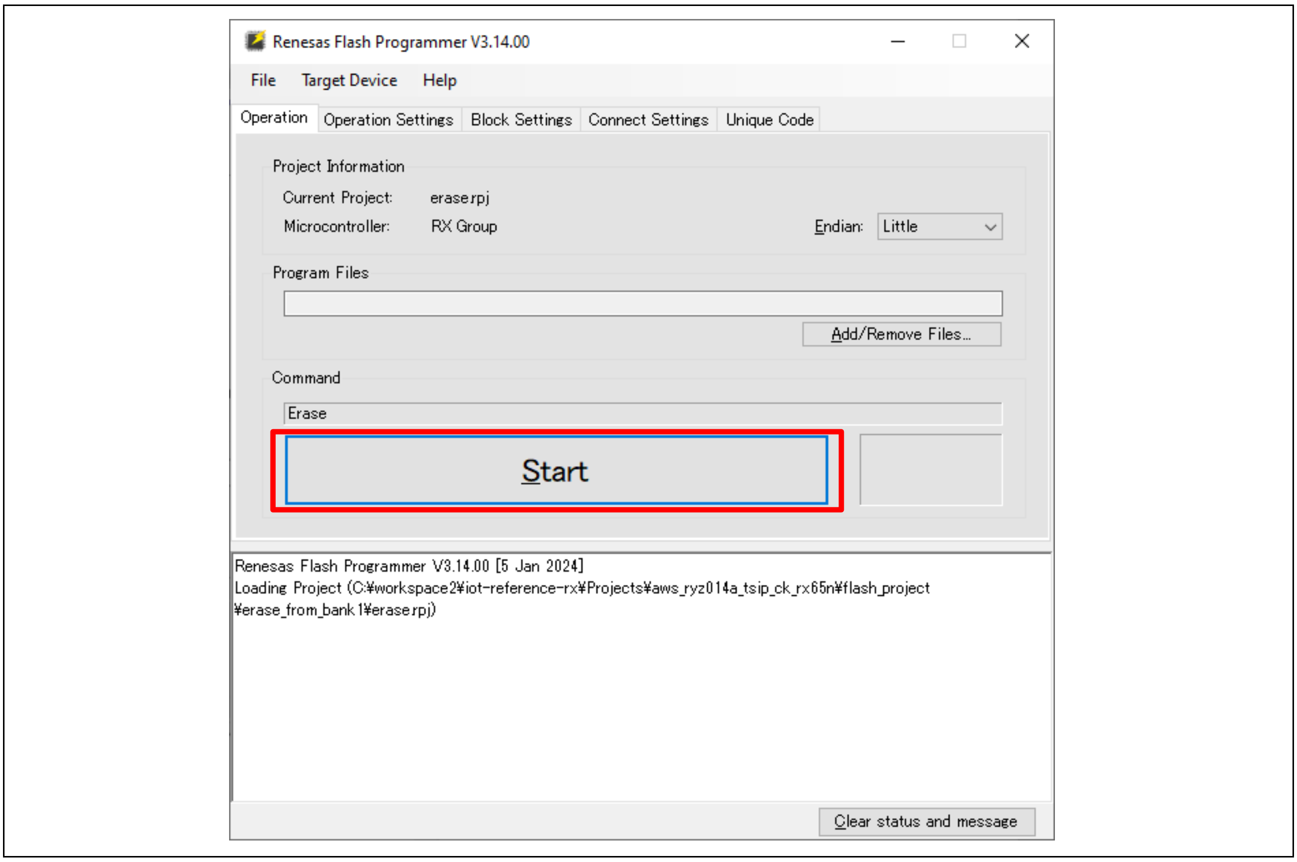


Figure 6-18 Procedure for Opening "erase.rpj"

4. Click the **Start** button to start erasure for the device.



Current Project: erase.rpj

Microcontroller: RX Group

Endian: Little

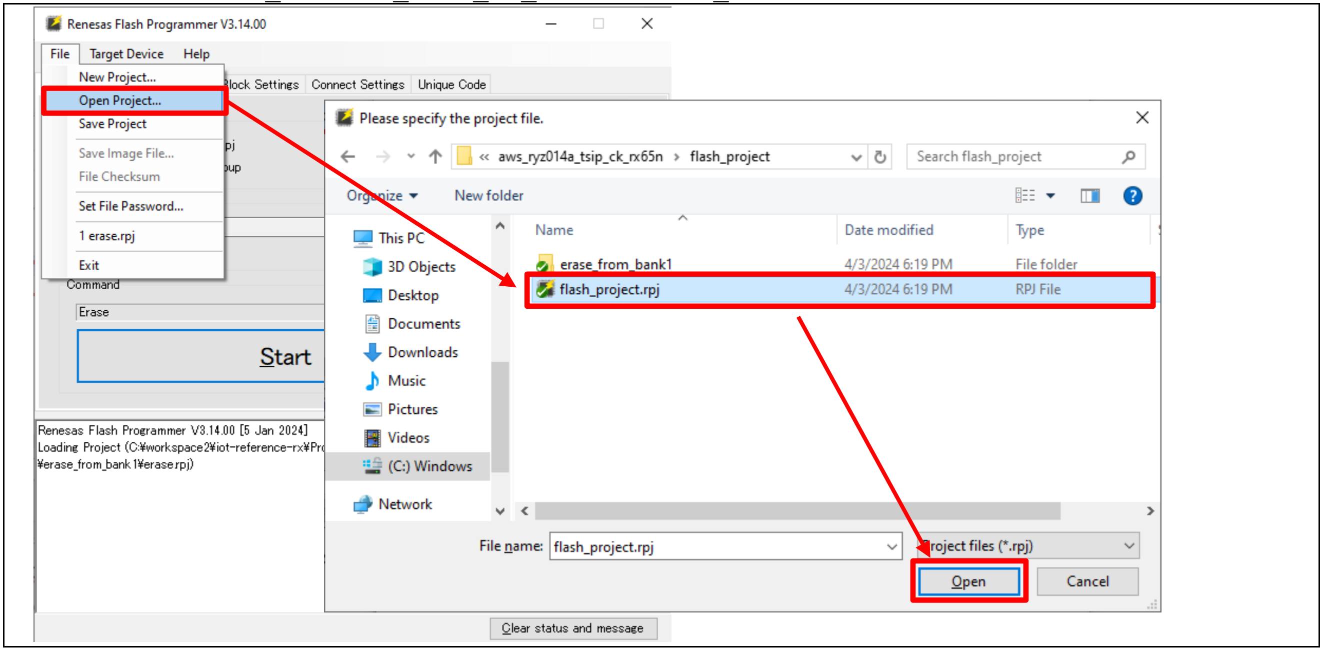
Command: Erase

```
Renesas Flash Programmer V3.14.00 [5 Jan 2024]
Loading Project (C:¥workspace2¥iot-reference-rx¥Projects¥aws_ryz014a_tsip_ck_rx65n¥flash_project
¥erase_from_bank1¥erase.rpj)
```

Figure 6-19 Starting Erasure for the Device

If the E3000107 error message, which states that the device conflicts with the connection information, is output, go to step 5.

5. Open the flash memory programming project (**flash_project.rpj**).
   The **flash_project.rpj** project is stored in the following folder of the sample program:
   `\Projects\aws_ryz014a_tsip_ck_rx65n\flash_project\`



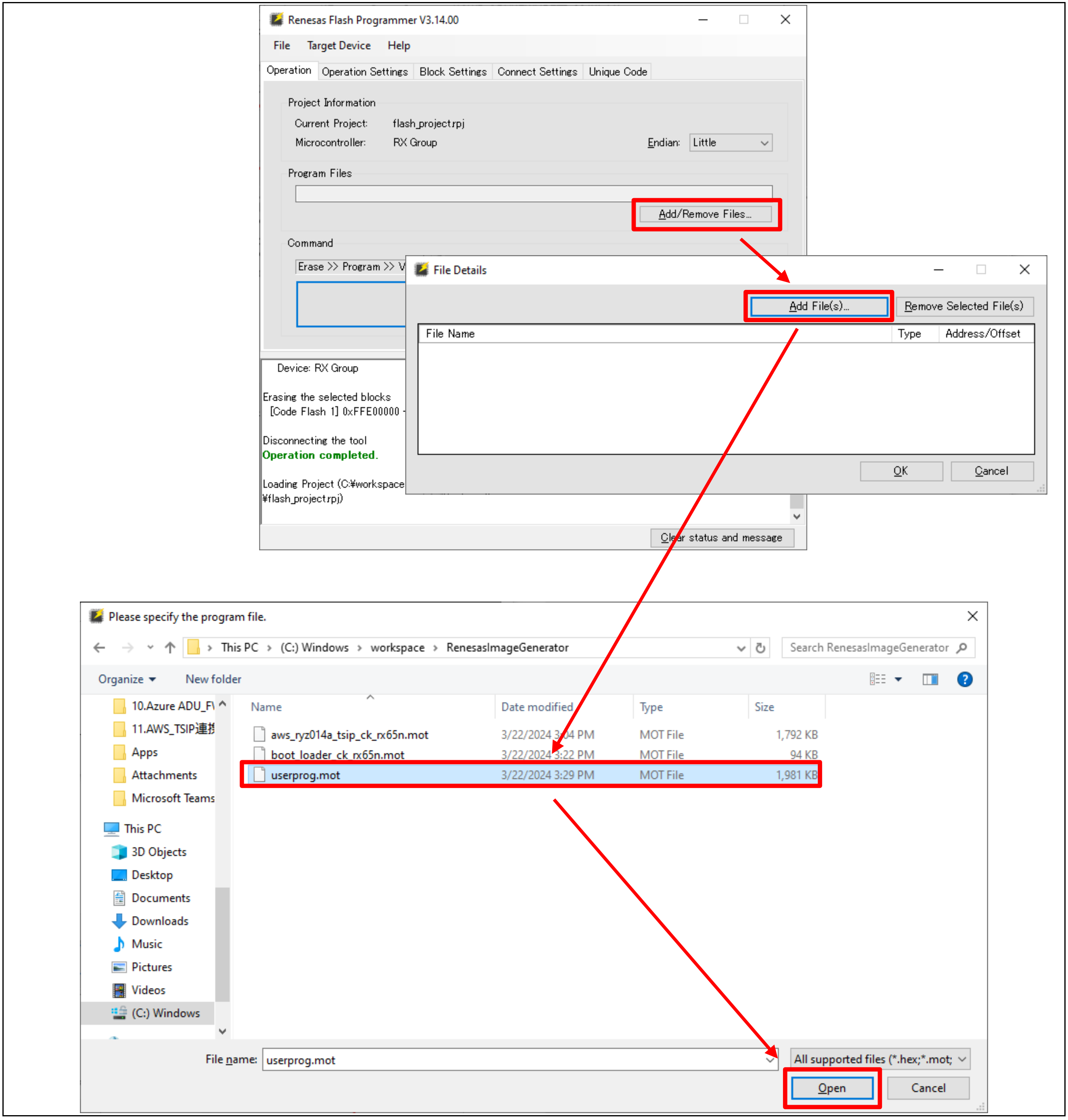
Name	Date modified	Type
erase_from_bank1	4/3/2024 6:19 PM	File folder
flash_project.rpj	4/3/2024 6:19 PM	RPJ File

File name: flash_project.rpj

```
Renesas Flash Programmer V3.14.00 [5 Jan 2024]
Loading Project (C:¥workspace2¥iot-reference-rx¥Pr
¥erase_from_bank1¥erase.rpj)
```

Figure 6-20 Procedure for Opening “flash_project.rpj”

6. Select the firmware to be written.
   Click the **Add/Remove Files** button, and then **Add Files** button. In the dialog box that appears, select **userprog.mot**, which is the initial firmware created in section 6.1.3(1).



Current Project: flash_project.rpj

Microcontroller: RX Group

Endian: Little

Device: RX Group

Erasing the selected blocks
[Code Flash 1] 0xFFE00000 -

Disconnecting the tool
Operation completed.

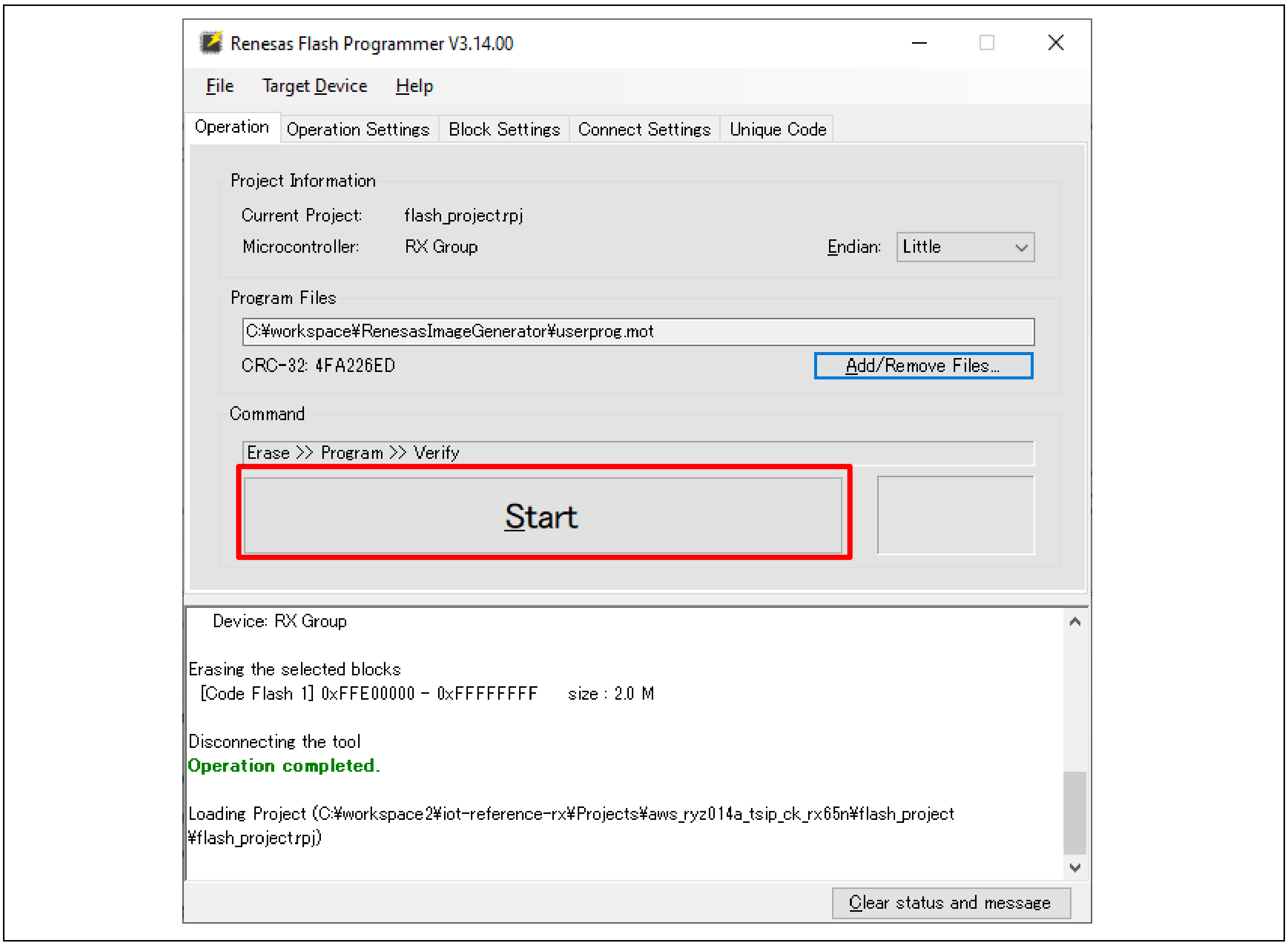
Loading Project (C:¥workspace
¥flash_project.rpj)

Name	Date modified	Type	Size
aws_ryz014a_tsip_ck_rx65n.mot	3/22/2024 3:04 PM	MOT File	1,792 KB
boot_loader_ck_rx65n.mot	3/22/2024 3:22 PM	MOT File	94 KB
userprog.mot	3/22/2024 3:29 PM	MOT File	1,981 KB

File name: userprog.mot

Figure 6-21 Selecting the Initial Firmware

7. Click the **Start** button to start writing the firmware.



Current Project: flash_project.rpj

Microcontroller: RX Group

Endian: Little

C:¥workspace¥RenesasImageGenerator¥userprog.mot

CRC-32: 4FA226ED

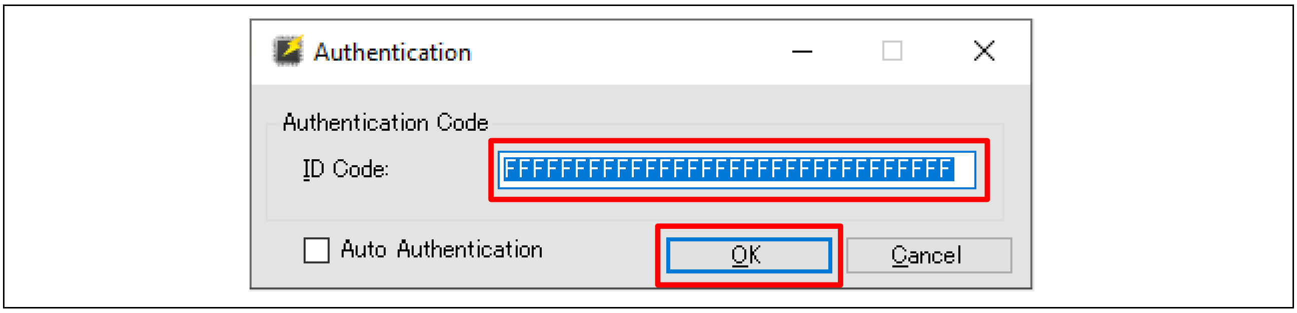
Erase >> Program >> Verify

```
 Device: RX Group

Erasing the selected blocks
  [Code Flash 1] 0xFFE00000 - 0xFFFFFFFF    size : 2.0 M

Disconnecting the tool
Operation completed.

Loading Project (C:¥workspace2¥iot-reference-rx¥Projects¥aws_ryz014a_tsip_ck_rx65n¥flash_project
¥flash_project.rpj)
```

Figure 6-22 Button to Start Writing the Firmware

If the **Authentication** dialog box (shown below) appears when you start a write, enter the preset ID code, and then click the **OK** button.

If no ID code has been preset, use the initial value as is.

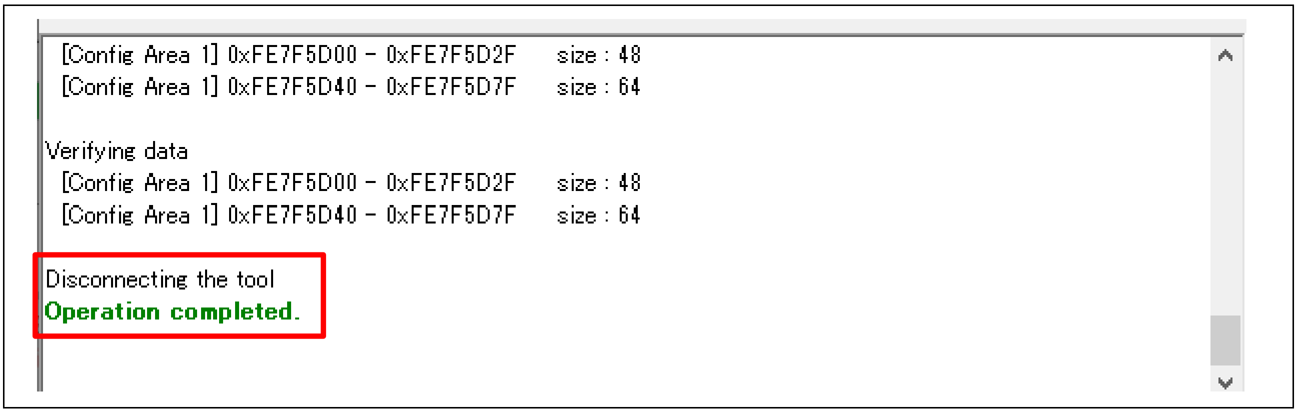


ID Code: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

□ Auto Authentication

Figure 6-23 Authorization by an ID Code

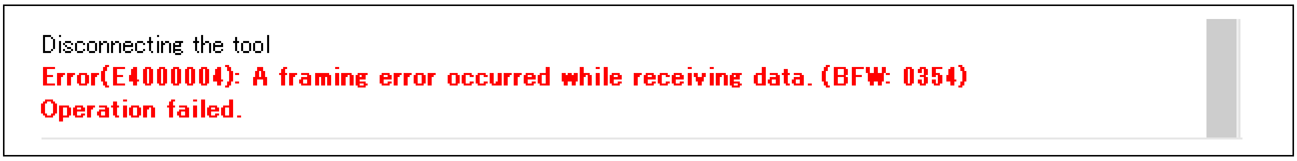
Writing the firmware starts. The write is completed successfully when the "Operation completed" message is displayed at the bottom of the screen as shown in the following figure.



```
  [Config Area 1] 0xFE7F5D00 - 0xFE7F5D2F    size : 48
  [Config Area 1] 0xFE7F5D40 - 0xFE7F5D7F    size : 64

Verifying data
  [Config Area 1] 0xFE7F5D00 - 0xFE7F5D2F    size : 48
  [Config Area 1] 0xFE7F5D40 - 0xFE7F5D7F    size : 64

Disconnecting the tool
Operation completed.
```

Figure 6-24 Message Appearing When Writing Firmware Is Completed

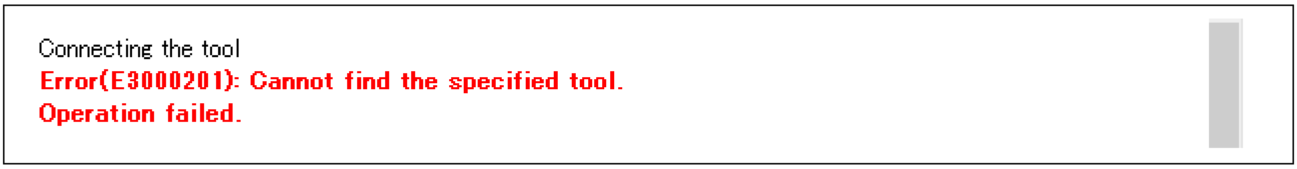
When you write the firmware, a communication error, such as the one shown in the following figure, may occur at the time of connection to the target board. If such an error occurs, try again by clicking the **Start** button. If connection fails again, detach and re-attach the USB cable of the target board, and then try again.



```
Disconnecting the tool
Error(E4000004): A framing error occurred while receiving data. (BFW: 0354)
Operation failed.
```

Figure 6-25 Example of a Communication Error (Framing Error)

After you perform demonstration by setting the J16 for **RUN** mode on the target board (CK-RX65N), you may want to write the initial firmware again from Renesas Flash Programmer. In this case, if you fail to set the J16 for **DEBUG** mode, the error shown in the following figure occurs.
When you use Renesas Flash Programmer to write firmware, make sure that the J16 is set for **DEBUG** mode.



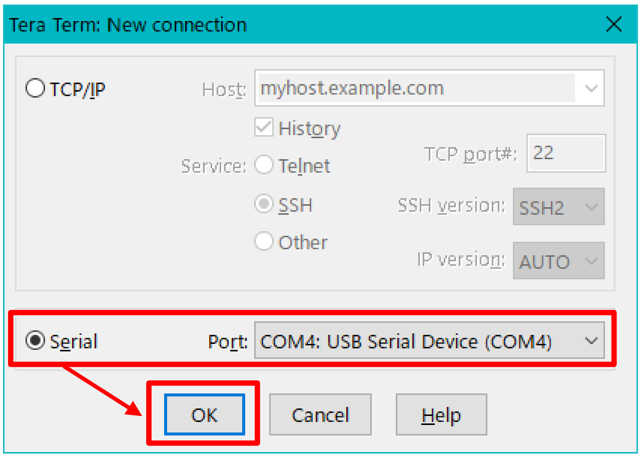
```
Connecting the tool
Error(E3000201): Cannot find the specified tool.
Operation failed.
```

Figure 6-26 Example of a Connection Error

6.1.4 Executing the Initial Firmware

The following explains how to set AWS IoT information in the Tera Term terminal software by running aws_ryz014a_tsip_ck_rx65n. The information set by this process is written to data flash memory. Install Tera Term by referring to section 2.1 in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037).

(1) Start Tera Term, and then, from the **File** menu, select **New Connection**. In the dialog box that opens, select the **Serial** radio button, and then click **OK**.

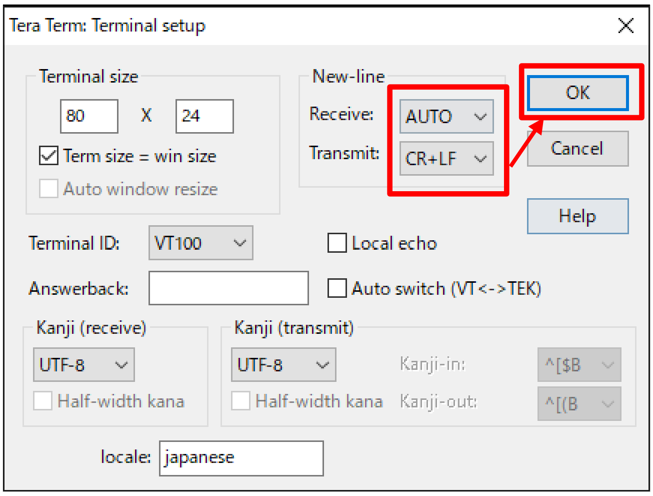

Tera Term: New connection

○ TCP/IP   Host: myhost.example.com
☑ History
Service: ○ Telnet   TCP port#: 22
● SSH   SSH version: SSH2
○ Other   IP version: AUTO

● Serial   Port: COM4: USB Serial Device (COM4)

Figure 6-27 Selecting “Serial”

(2) From the **Setup** menu, select **Terminal**. In the **New-line** area of the dialog box that appears, select **AUTO** for **Receive** and **CR+LF** for **Transmit**.


Tera Term: Terminal setup

Terminal size: 80 X 24
☑ Term size = win size
□ Auto window resize

New-line
Receive: AUTO
Transmit: CR+LF

Terminal ID: VT100   □ Local echo
Answerback: □ Auto switch (VT<->TEK)

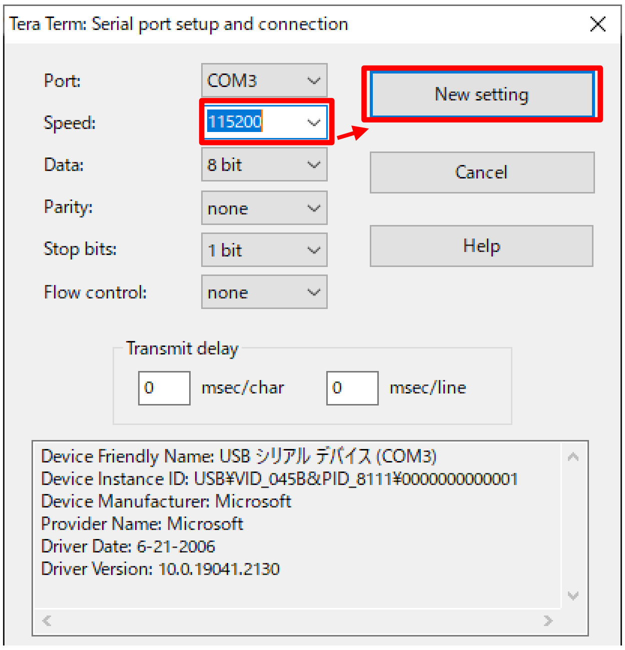
Kanji (receive): UTF-8
□ Half-width kana

Kanji (transmit): UTF-8   Kanji-in: ^[$B
□ Half-width kana   Kanji-out: ^[(B

locale: japanese

Figure 6-28 Settings in the Terminal Software

(3) From the **Setup** menu, select **Serial port**. In the dialog box that appears, set **Speed** to 115200, and then click **New setting**. For the other settings, do not change their initial values.



Tera Term: Serial port setup and connection

Port: COM3
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Device Friendly Name: USB シリアル デバイス (COM3)
Device Instance ID: USB¥VID_045B&PID_8111¥0000000000001
Device Manufacturer: Microsoft
Provider Name: Microsoft
Driver Date: 6-21-2006
Driver Version: 10.0.19041.2130

Figure 6-29 Communication Speed Settings

(4) On the CK-RX65N, set the J16 for **RUN** mode, and then press the **RESET** switch. After a hardware reset occurs, the program starts.
After the program starts, the operating status is displayed in the Tera Term window.

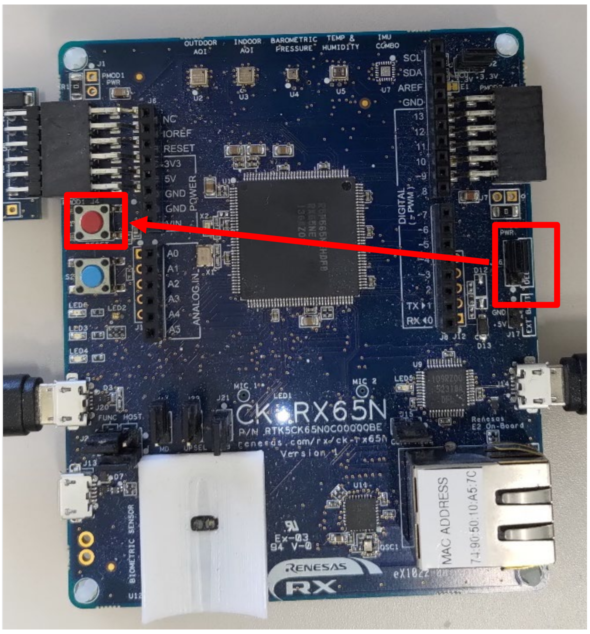
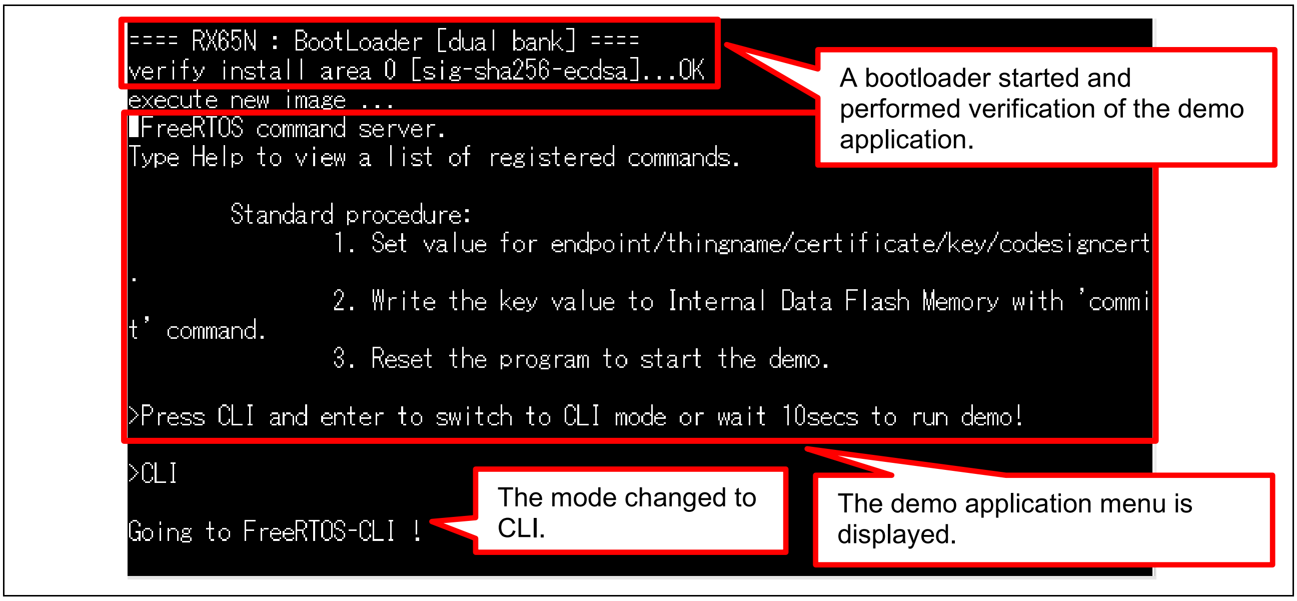


Figure 6-30 Setting RUN Mode and Performing a Hardware Reset

(5) The bootloader starts, and then, after verification finishes, the demo application starts.
When the demo application starts, a menu is displayed. When this menu is displayed, enter “CLI” within 10 seconds, and then press the **Enter** key. The application enters CLI mode. In CLI mode, you can register various kinds of information in the program by using commands.
If you do not enter “CLI” for 10 seconds or more, a sequence for connection to AWS starts.



```
==== RX65N : BootLoader [dual bank] ====
verify install area 0 [sig-sha256-ecdsa]...OK
execute new image ...
FreeRTOS command server.
Type Help to view a list of registered commands.

        Standard procedure:
                1. Set value for endpoint/thingname/certificate/key/codesigncert
.
                2. Write the key value to Internal Data Flash Memory with 'commi
t' command.
                3. Reset the program to start the demo.

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

>CLI

Going to FreeRTOS-CLI !
```

A bootloader started and performed verification of the demo application.

The demo application menu is displayed.

The mode changed to CLI.

Figure 6-31 Screen for Executing the Demo Application

6.1.5 Registering the AWS IoT Information

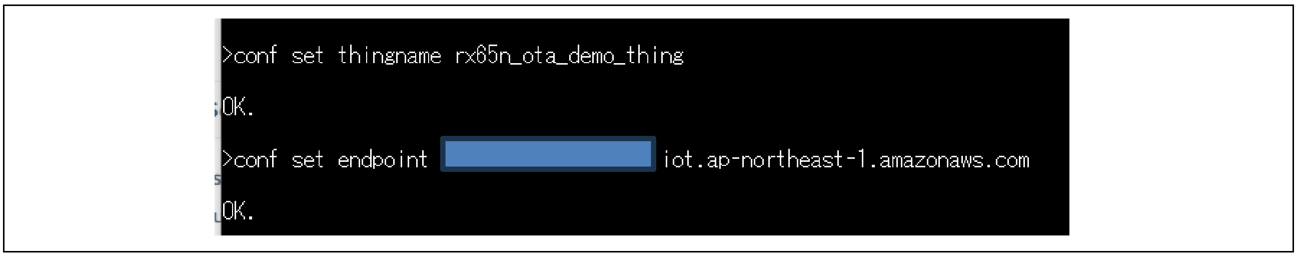
This section describes how to register various information necessary to establish a connection to AWS via TLS communication using the TSIP. After starting the initial firmware, enter CLI mode, and specify the necessary settings.
In CLI mode, you can use commands to set values and check set values.
The values set in CLI mode are saved in data flash memory. Therefore, the settings are maintained even when the power to the target board is turned off.

(1) Setting the AWS connection information

Register the name and endpoint of the device (thing) set in Chapter 3, AWS Setup in CLI mode.

Use Tera Term to enter CLI mode, and then execute the following commands:

```
conf set thingname thing-name [Enter]
conf set endpoint endpoint-name [Enter]
```



```
>conf set thingname rx65n_ota_demo_thing

OK.

>conf set endpoint [redacted] iot.ap-northeast-1.amazonaws.com

OK.
```

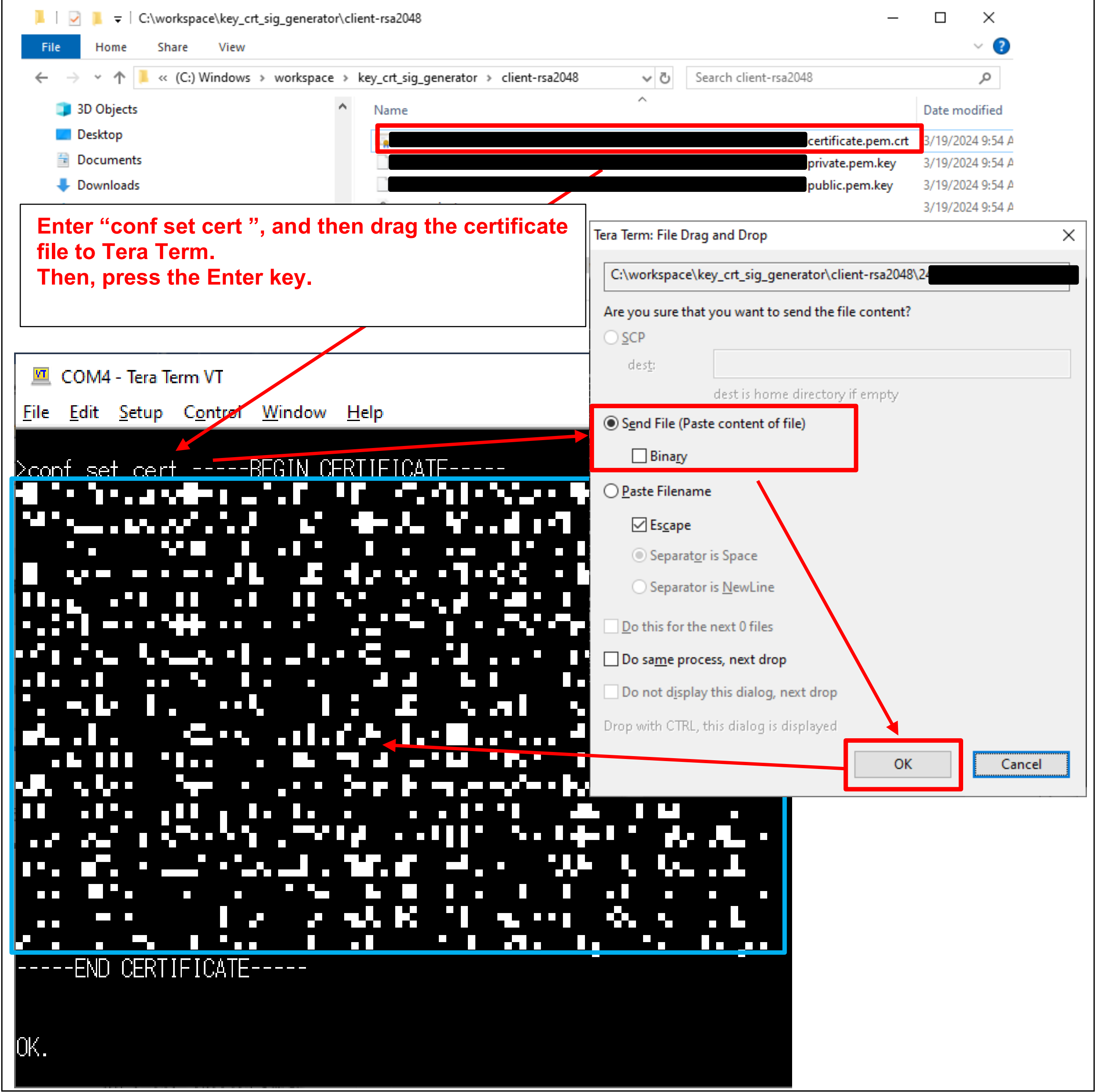
Figure 6-32 Inputting AWS Connection Information from CLI

(2) Registering the client certificate

In CLI mode, register the client certificate that was downloaded from AWS in section 5.1.3, Obtaining a Key Pair and Client Certificate for RSA.

In Tera Term, enter “conf set cert ”, and then transmit the client certificate file (***xxxx*-certificate.pem.crt**) by dragging it to Tera Term. (Note: Place a halfwidth space after “cert”.)

After dragging the file, move the focus to Tera Term, and then press the **Enter** key.



C:\workspace\key_crt_sig_generator\client-rsa2048
File
Home
Share
View
« (C:) Windows › workspace › key_crt_sig_generator › client-rsa2048
Search client-rsa2048
3D Objects
Desktop
Documents
Downloads
Name
Date modified

Name	Date modified
[redacted]certificate.pem.crt	3/19/2024 9:54 A
[redacted]private.pem.key	3/19/2024 9:54 A
[redacted]public.pem.key	3/19/2024 9:54 A
	3/19/2024 9:54 A

Enter “conf set cert ”, and then drag the certificate file to Tera Term.
Then, press the Enter key.

COM4 - Tera Term VT
File Edit Setup Control Window Help

```
>conf set cert -----BEGIN CERTIFICATE-----
[redacted]
-----END CERTIFICATE-----

OK.
```

Tera Term: File Drag and Drop

C:\workspace\key_crt_sig_generator\client-rsa2048\2[redacted]

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

○ SCP

dest:

dest is home directory if empty

● Send File (Paste content of file)

□ Binary

○ Paste Filename

☑ Escape

○ Separator is Space

○ Separator is NewLine

□ Do this for the next 0 files

□ Do same process, next drop

□ Do not display this dialog, next drop

Drop with CTRL, this dialog is displayed

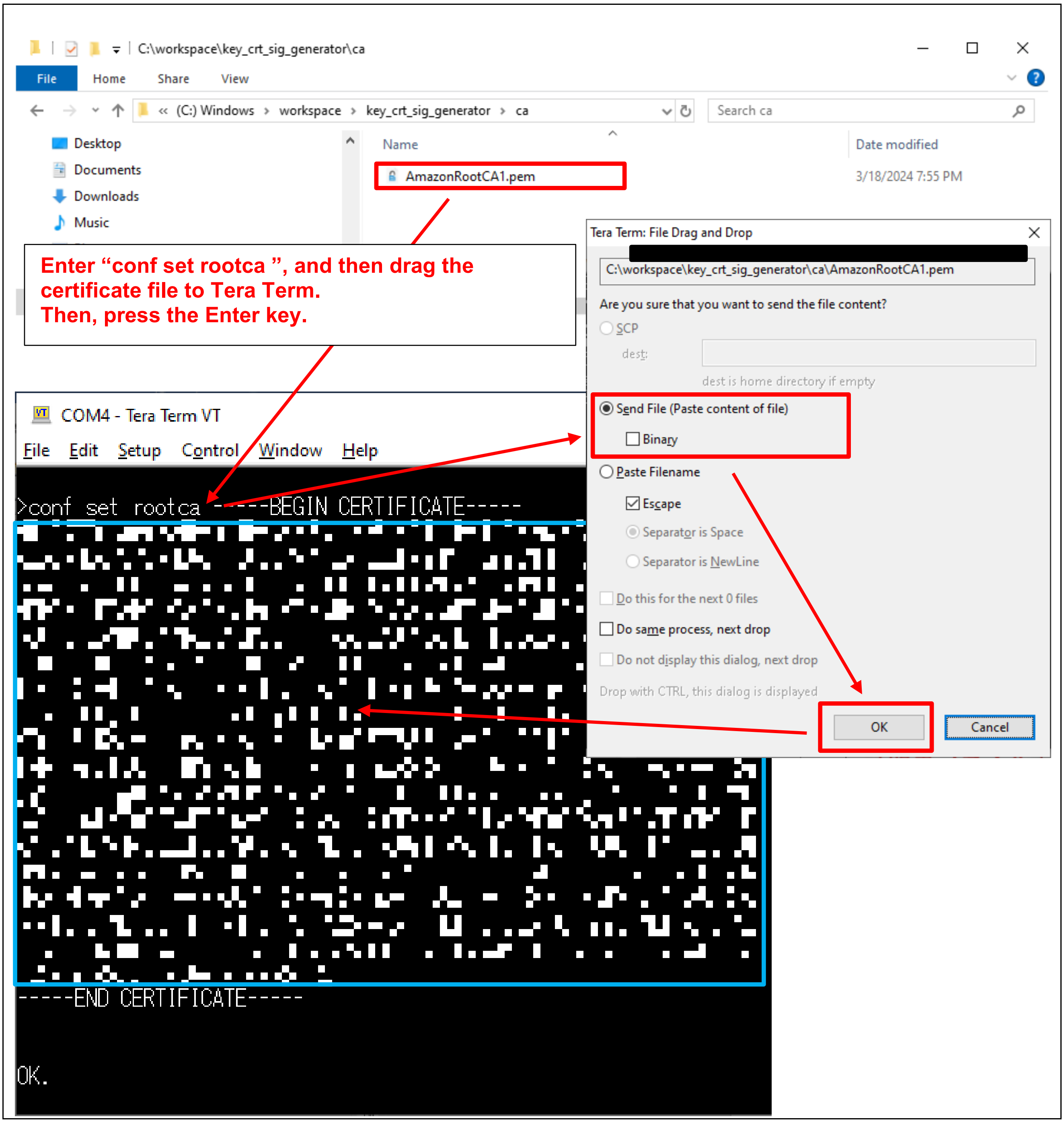
Figure 6-33 Inputting the Client Certificate

(3) Registering the root CA certificate

In CLI mode, register the root CA certificate that was downloaded from AWS in section 5.1.2, Obtaining a Root CA Certificate.

In Tera Term, enter "conf set rootca ", and then transmit the root CA certificate file (**AmazonRootCA1.pem**) by dragging it to Tera Term. (Note: Place a halfwidth space after "rootca".)

After dragging the file, move the focus to Tera Term, and then press the **Enter** key.



Enter "conf set rootca ", and then drag the certificate file to Tera Term.
Then, press the Enter key.

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

OK.
```

[redacted]

C:\workspace\key_crt_sig_generator\ca\AmazonRootCA1.pem

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

○ SCP

dest:

dest is home directory if empty

● Send File (Paste content of file)

□ Binary

○ Paste Filename

☑ Escape

○ Separator is Space

○ Separator is NewLine

□ Do this for the next 0 files

□ Do same process, next drop

□ Do not display this dialog, next drop

Drop with CTRL, this dialog is displayed

Figure 6-34 Inputting the Root CA Certificate

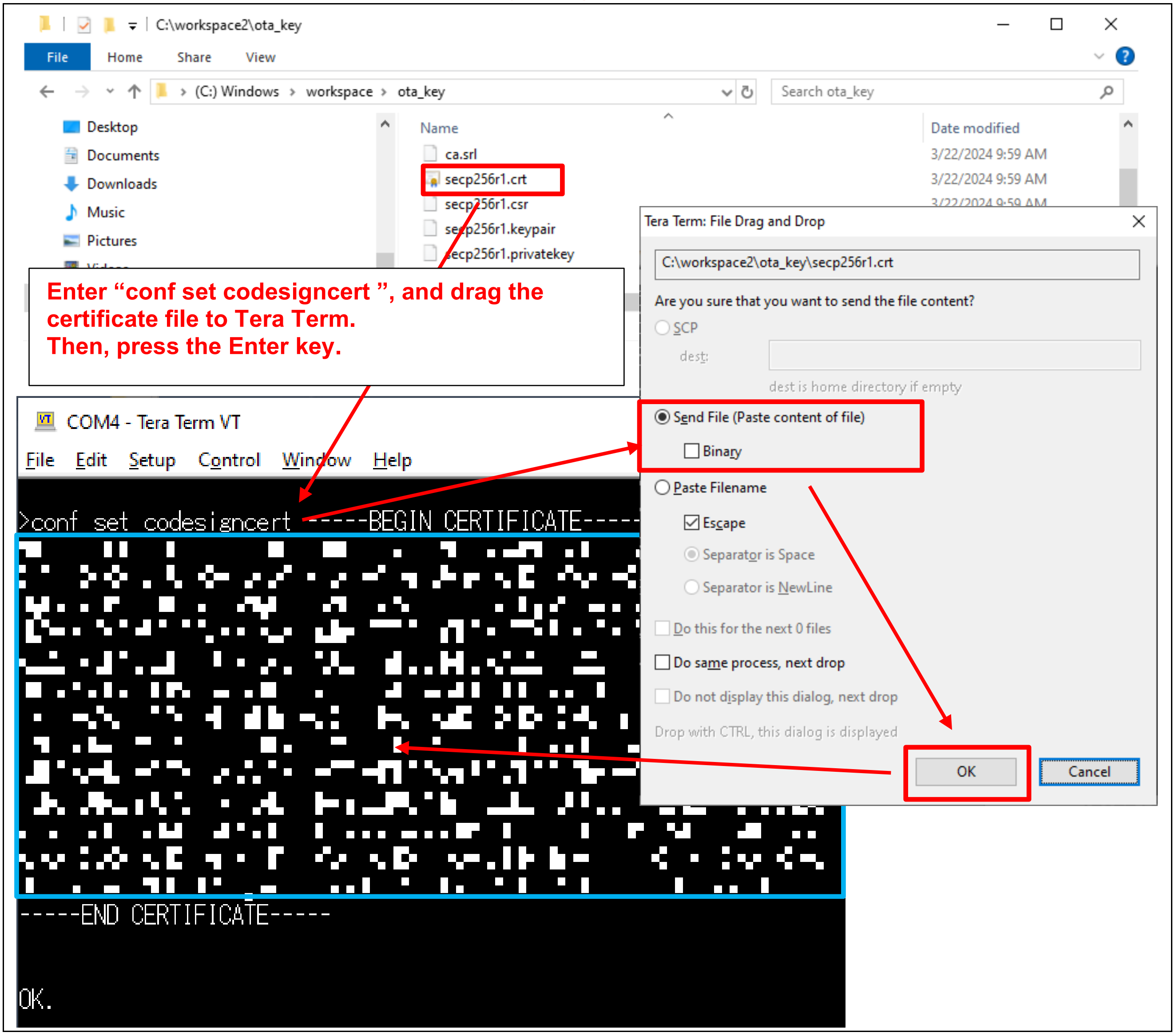
(4) Registering the key pair certificate (ECDSA certificate) for an OTA update

In CLI mode, register the key pair certificate that was generated in section 5.2, Generating a Key Pair and Certificates for an OTA Update.

In Tera Term, enter “conf set codesigncert ”, and then transmit the key pair certificate file (**secp256r1.crt**) by dragging it to Tera Term. (Note: Place a halfwidth space after “codesigncert”.)

After dragging the file, move the focus to Tera Term, and then press the **Enter** key.

Note: Before transmitting the file, make sure that the line break code in the certificate file is LF. If the line break code is not LF, convert it by using a text editor.



Enter “conf set codesigncert ”, and drag the certificate file to Tera Term.
Then, press the Enter key.

Tera Term: File Drag and Drop

C:\workspace2\ota_key\secp256r1.crt

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

○ SCP

dest:

dest is home directory if empty

● Send File (Paste content of file)

□ Binary

○ Paste Filename

☑ Escape

○ Separator is Space

○ Separator is NewLine

□ Do this for the next 0 files

□ Do same process, next drop

□ Do not display this dialog, next drop

Drop with CTRL, this dialog is displayed

OK   Cancel

```
>conf set codesigncert -----BEGIN CERTIFICATE-----
[redacted]
-----END CERTIFICATE-----

OK.
```

Figure 6-35 Inputting the Key Pair Certificate for OTA Update

(5) Committing the AWS IoT settings (writing the settings to data flash memory)

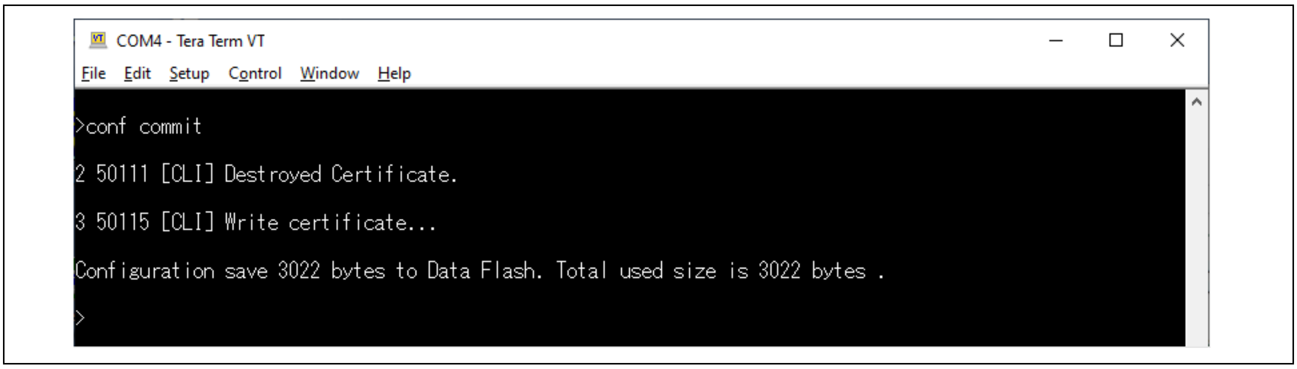
After the settings are input from CLI, write them to data flash memory. The input settings are not saved until “commit” is executed. Always execute “commit” the settings before turning off the power to the board.

Once you commit the settings, they are retained even after turning power off.

In Tera Term, execute the following command:

```
conf commit[enter]
```

When you execute “commit”, the following text is displayed:



```
>conf commit
2 50111 [CLI] Destroyed Certificate.
3 50115 [CLI] Write certificate...
Configuration save 3022 bytes to Data Flash. Total used size is 3022 bytes .
>
```

Figure 6-36 Writing the Settings to Data Flash by Executing “commit”

(6) Deleting the written settings from the data flash memory

After data is written to the data flash memory by executing “commit”, you can delete the data.
When you input new settings and execute “commit” again, the existing settings are replaced by the new settings. Therefore, you do not need to delete the written data ordinarily. Perform deletion when you want to clear the data flash memory.

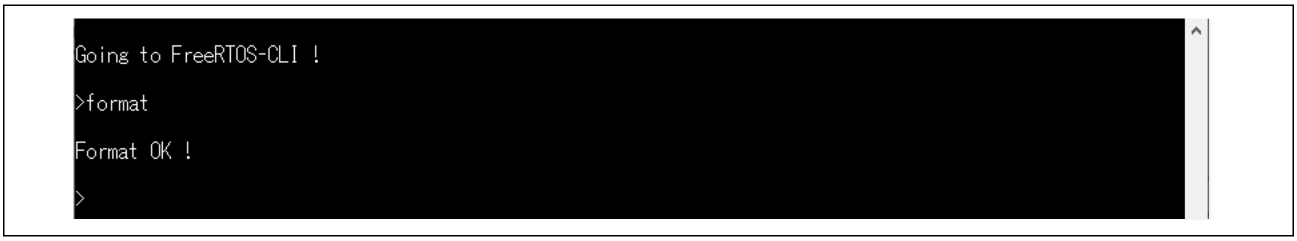
Note that the three keys used for the TSIP (root CA certificate signature verification public key, client certificate public key, and client certificate private key) are written to the data flash memory when the program is executed for the first time.
These keys cannot be accessed from CLI. Therefore, if you want to change them, perform deletion to clear the data flash memory.

Note: If you re-execute the script in section 5.1.4 or regenerate the three types of keys in section 5.1.5, be sure to clear the data flash with this operation.
Also, after clearing the data flash, execute steps (1) to (4) again.
As a result, when the program runs, three new recreated keys are written to the data flash.

In Tera Term, execute the following command:

```
format[enter]
```

When you perform deletion, the following text is displayed. Because all settings in the data flash memory are cleared, specify new settings.



```
Going to FreeRTOS-CLI !
>format
Format OK !
>
```

Figure 6-37 Deleting the Settings in the Data Flash Memory

(7) Resetting the program and executing the initial firmware

After the settings are input completely, when you reset the program, connection to AWS starts.

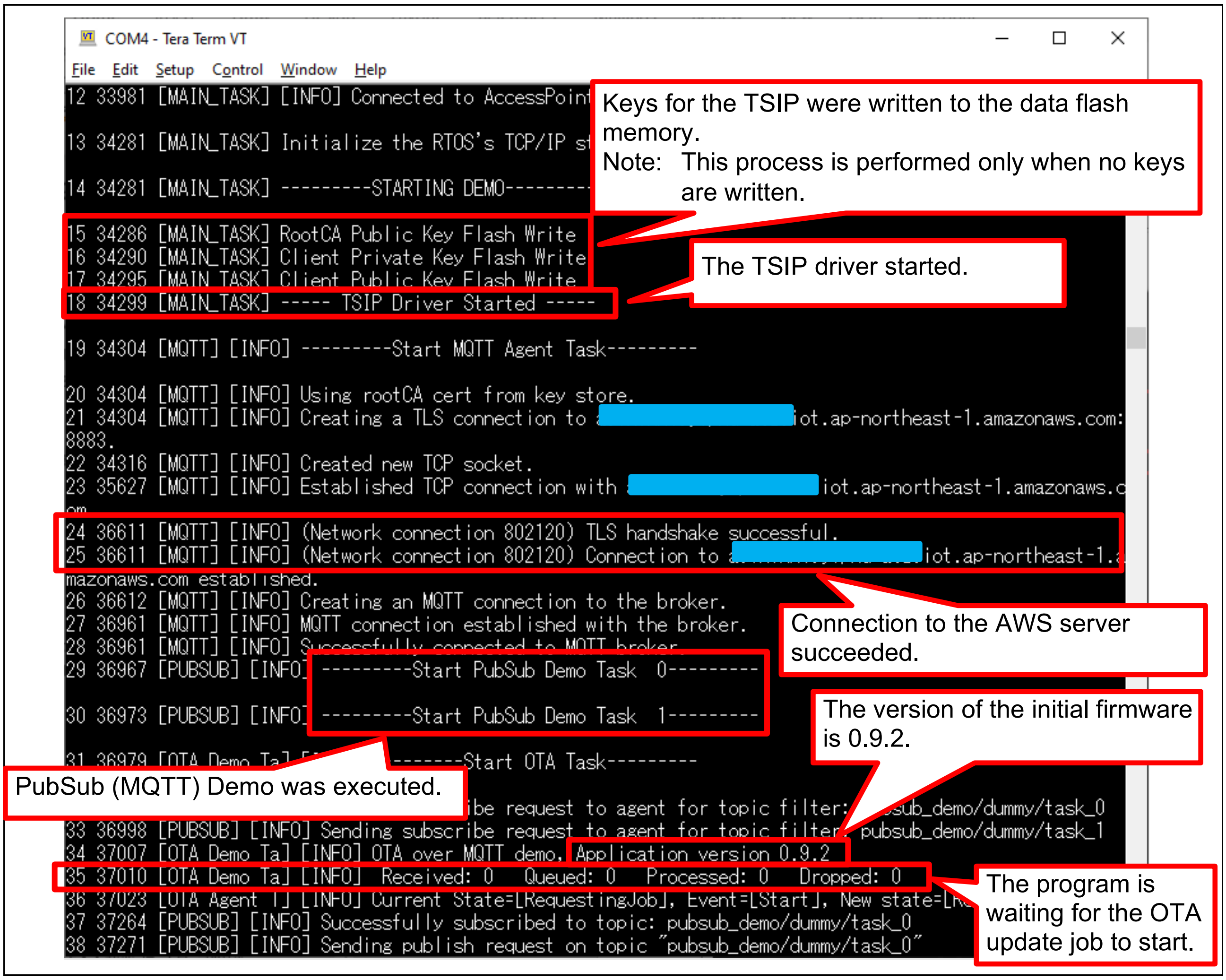
In Tera Term, execute the following command: A software reset occurs, and the program restarts from the beginning.

```
reset[enter]
```

When “reset” is executed, Tera Term displays the communication log data as shown in the following figure.

After TLS communication using the TSIP starts, PubSub Demo and OTA Demo tasks are performed. PubSub Demo demonstrates verification of MQTT communication, and OTA Demo demonstrates a firmware update.

In the log, confirm that the program already executed PubSub Demo and is waiting for the OTA update job to start.



```
12 33981 [MAIN_TASK] [INFO] Connected to AccessPoint

13 34281 [MAIN_TASK] Initialize the RTOS's TCP/IP st

14 34281 [MAIN_TASK] ---------STARTING DEMO--------

15 34286 [MAIN_TASK] RootCA Public Key Flash Write
16 34290 [MAIN_TASK] Client Private Key Flash Write
17 34295 [MAIN_TASK] Client Public Key Flash Write
18 34299 [MAIN_TASK] ----- TSIP Driver Started -----

19 34304 [MQTT] [INFO] ---------Start MQTT Agent Task---------

20 34304 [MQTT] [INFO] Using rootCA cert from key store.
21 34304 [MQTT] [INFO] Creating a TLS connection to [redacted]iot.ap-northeast-1.amazonaws.com:
8883.
22 34316 [MQTT] [INFO] Created new TCP socket.
23 35627 [MQTT] [INFO] Established TCP connection with [redacted]iot.ap-northeast-1.amazonaws.c
om
24 36611 [MQTT] [INFO] (Network connection 802120) TLS handshake successful.
25 36611 [MQTT] [INFO] (Network connection 802120) Connection to [redacted]iot.ap-northeast-1.a
mazonaws.com established.
26 36612 [MQTT] [INFO] Creating an MQTT connection to the broker.
27 36961 [MQTT] [INFO] MQTT connection established with the broker.
28 36961 [MQTT] [INFO] Successfully connected to MQTT broker.
29 36967 [PUBSUB] [INFO] ---------Start PubSub Demo Task  0---------

30 36973 [PUBSUB] [INFO] ---------Start PubSub Demo Task  1---------

31 36979 [OTA Demo Ta] [I --------Start OTA Task---------

                                ibe request to agent for topic filter: sub_demo/dummy/task_0
33 36998 [PUBSUB] [INFO] Sending subscribe request to agent for topic filter: pubsub_demo/dummy/task_1
34 37007 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.2
35 37010 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
36 37023 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[R
37 37264 [PUBSUB] [INFO] Successfully subscribed to topic: pubsub_demo/dummy/task_0
38 37271 [PUBSUB] [INFO] Sending publish request on topic "pubsub_demo/dummy/task_0"
```

Keys for the TSIP were written to the data flash memory.
Note: This process is performed only when no keys are written.

The TSIP driver started.

Connection to the AWS server succeeded.

The version of the initial firmware is 0.9.2.

PubSub (MQTT) Demo was executed.

The program is waiting for the OTA update job to start.

Figure 6-38 Executing the Initial Firmware

6.1.6 Verifying the Status of MQTT Communication

This section describes how to verify the status of MQTT communication.

The status of MQTT communication can be verified in AWS. Before executing the program, specify the AWS monitor settings by using the procedures described below.

(1) Signing in to AWS Management Console

Sign in to AWS Management Console (https://aws.amazon.com/console/), and then open the IoT Core window by using the AWS menu. To do this, in the menu to the left of IoT Core, under **Test**, select **MQTT test client** to open the MQTT test client.

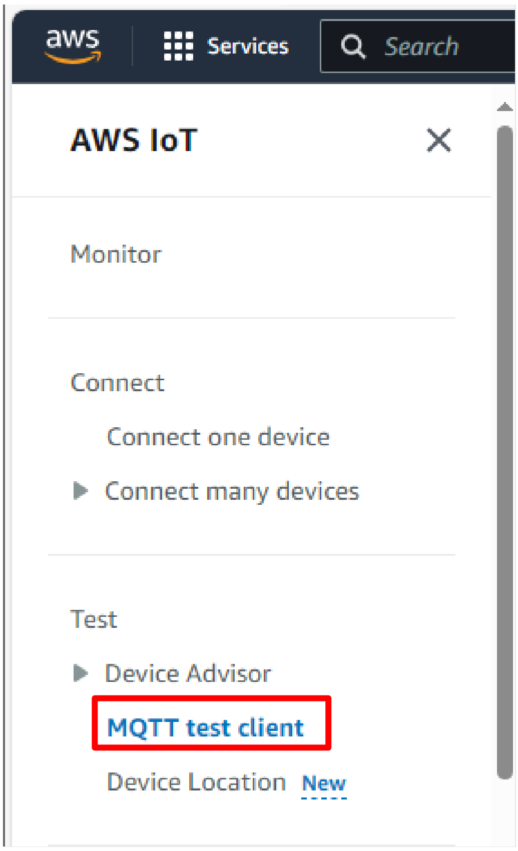
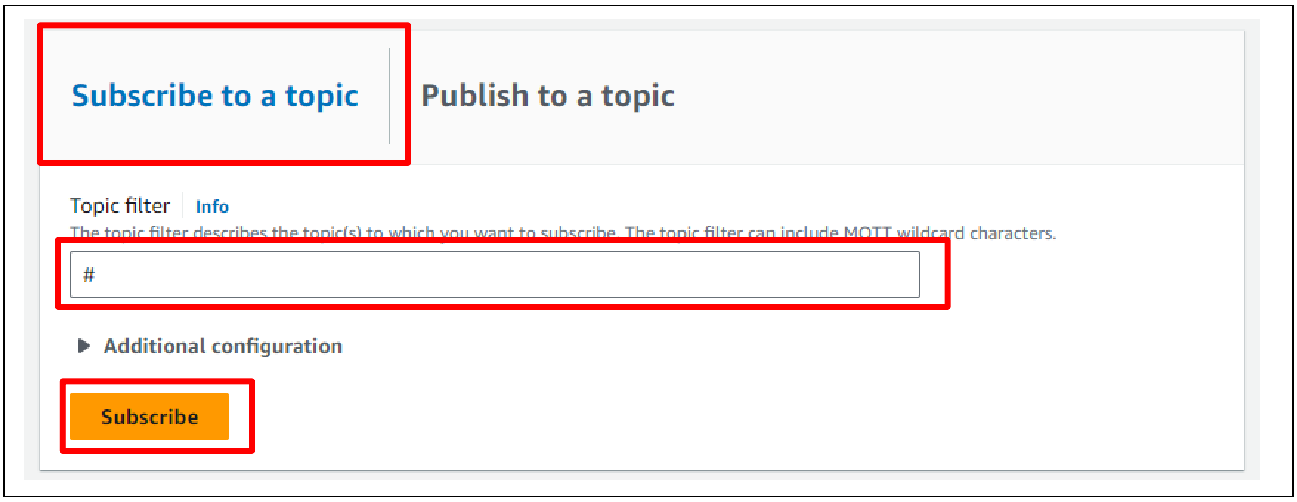


Figure 6-39 AWS IoT Menu

(2) Subscribing to a topic

Click the **Subscribe to a topic** tab, enter a hashmark (#) as a wildcard in **Topic filter**, and then click **Subscribe**.



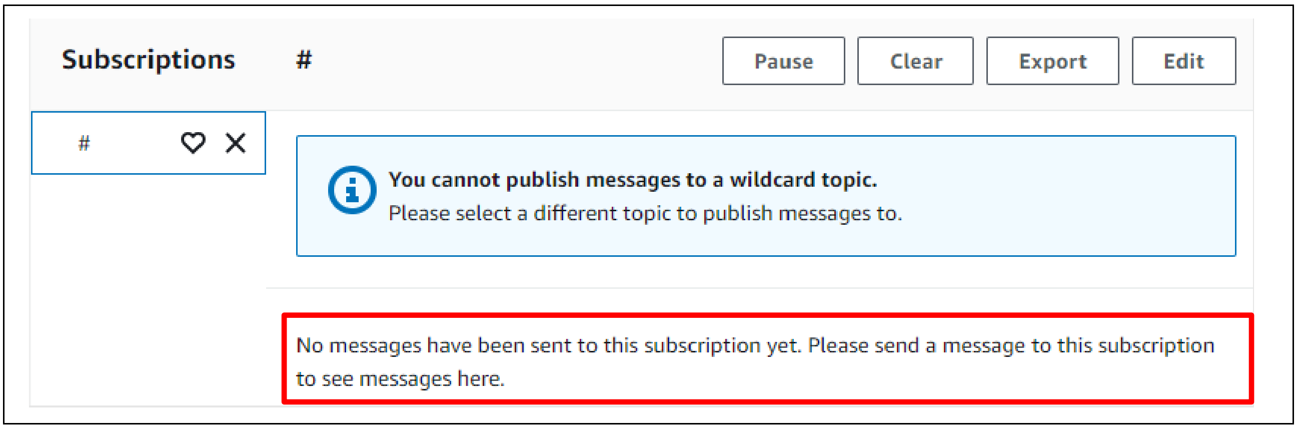
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

#

Figure 6-40 Specifying the MQTT Test Client Settings

(3) Confirming that a blank console appears

Confirm that a blank console is displayed at the bottom of the window as shown in the following figure.



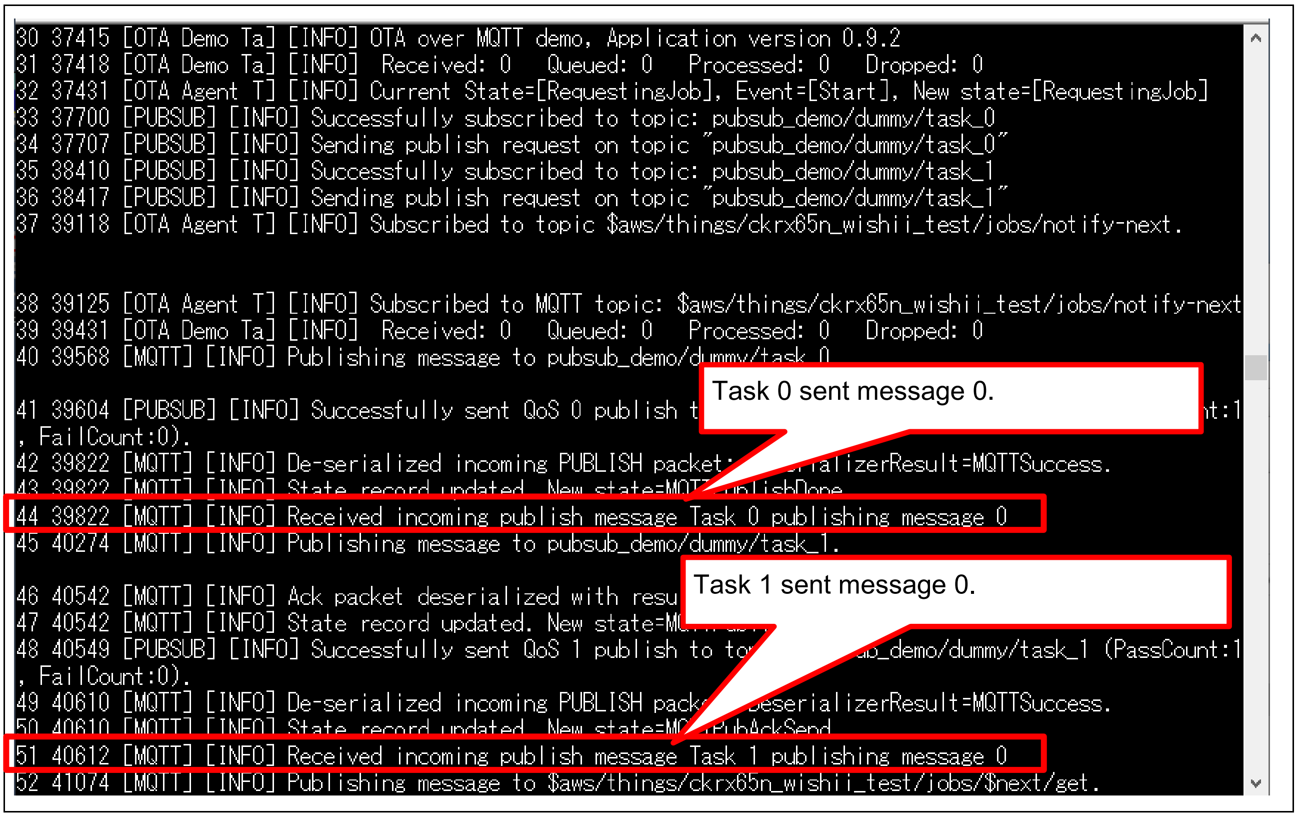
You cannot publish messages to a wildcard topic.
Please select a different topic to publish messages to.

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

Figure 6-41 Confirming That a Blank Console Is Displayed

(4) Executing the program

Press the reset switch on the target board to reset and restart the program by referring to section 6.1.4(4).
When the program starts, the progress of executing PubSub Demo (MQTT communication tasks) is displayed as shown in the following figure.
PubSub Demo performs two MQTT communication tasks, Task 0 and Task 1.
Each PubSub task sends 10 messages (messages 0 to 9).



```
30 37415 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.2
31 37418 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
32 37431 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
33 37700 [PUBSUB] [INFO] Successfully subscribed to topic: pubsub_demo/dummy/task_0
34 37707 [PUBSUB] [INFO] Sending publish request on topic "pubsub_demo/dummy/task_0"
35 38410 [PUBSUB] [INFO] Successfully subscribed to topic: pubsub_demo/dummy/task_1
36 38417 [PUBSUB] [INFO] Sending publish request on topic "pubsub_demo/dummy/task_1"
37 39118 [OTA Agent T] [INFO] Subscribed to topic $aws/things/ckrx65n_wishii_test/jobs/notify-next.


38 39125 [OTA Agent T] [INFO] Subscribed to MQTT topic: $aws/things/ckrx65n_wishii_test/jobs/notify-next
39 39431 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
40 39568 [MQTT] [INFO] Publishing message to pubsub_demo/dummy/task_0

41 39604 [PUBSUB] [INFO] Successfully sent QoS 0 publish t                                        nt:1
, FailCount:0).
42 39822 [MQTT] [INFO] De-serialized incoming PUBLISH packet:      ializerResult=MQTTSuccess.
43 39822 [MQTT] [INFO] State record updated. New state=MQT  ublishDone.
44 39822 [MQTT] [INFO] Received incoming publish message Task 0 publishing message 0
45 40274 [MQTT] [INFO] Publishing message to pubsub_demo/dummy/task_1.

46 40542 [MQTT] [INFO] Ack packet deserialized with resu
47 40542 [MQTT] [INFO] State record updated. New state=M
48 40549 [PUBSUB] [INFO] Successfully sent QoS 1 publish to to          b_demo/dummy/task_1 (PassCount:1
, FailCount:0).
49 40610 [MQTT] [INFO] De-serialized incoming PUBLISH pack     eserializerResult=MQTTSuccess.
50 40610 [MQTT] [INFO] State record updated. New state=M   PubAckSend.
51 40612 [MQTT] [INFO] Received incoming publish message Task 1 publishing message 0
52 41074 [MQTT] [INFO] Publishing message to $aws/things/ckrx65n_wishii_test/jobs/$next/get.
```

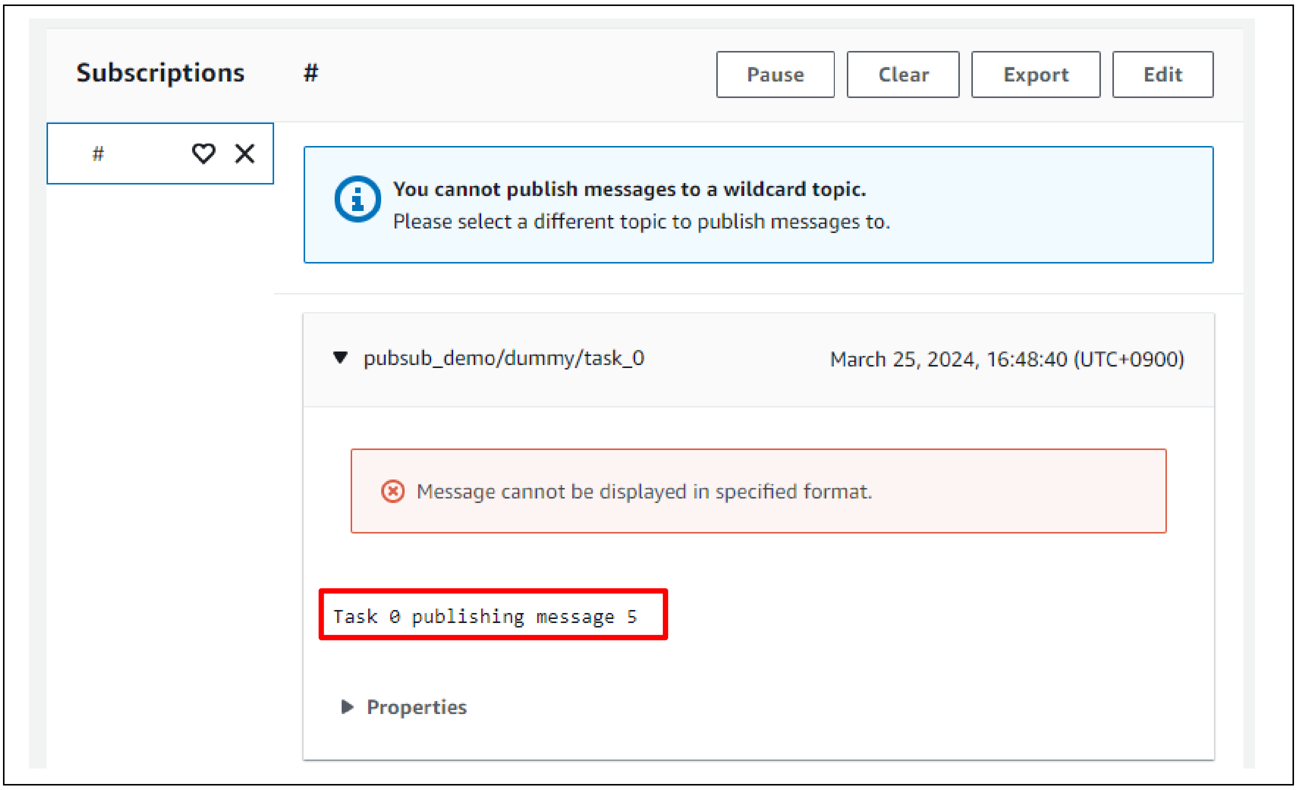
Task 0 sent message 0.

Task 1 sent message 0.

Figure 6-42 Progress of Executing PubSub Demo

(5) Viewing the communication log in the MQTT test client window

If you open the MQTT test client window of AWS during execution of PubSub Demo, you can view the communication log.
When PubSub Demo successfully connects to AWS, a message to that effect is output to the communication log on the MQTT client.
The following figure shows an example of a message output when message 5 is received from Task 0.



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

▼ pubsub_demo/dummy/task_0 March 25, 2024, 16:48:40 (UTC+0900)

Message cannot be displayed in specified format.

```
Task 0 publishing message 5
```

Figure 6-43 Viewing the MQTT Communication Log

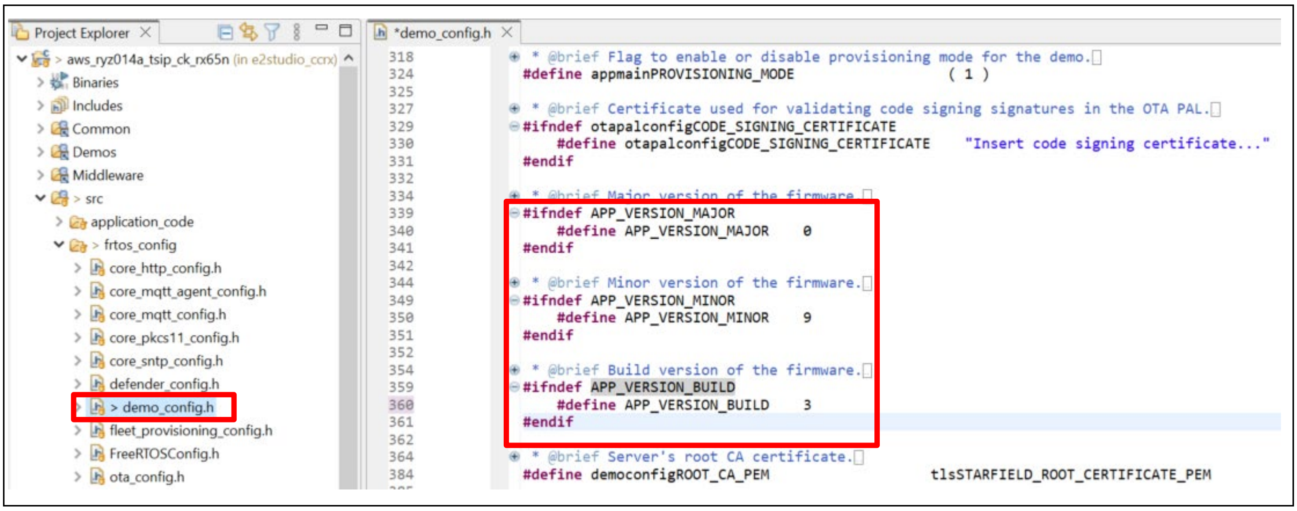
6.2 Building and Executing Update Firmware

This section describes how to create update firmware to be used in an OTA update. In this application note, you create update firmware by only changing the version number of the project for the initial firmware “aws_ryz014a_tsip_ck_rx65n” created in section 6.1.

6.2.1 Creating Update Firmware

(1) Changing the firmware version to 0.9.3

The following figure shows the section to be changed in the “aws_ryz014a_tsip_ck_rx65n” project.
In the **aws_ryz014a_tsip_ck_rx65n\src\frtos_config\demo_config.h** file, change the value of the APP_VERSION_BUILD definition to 3 so that the version number changes to 0.9.3. When the change is completed, rebuild the project.



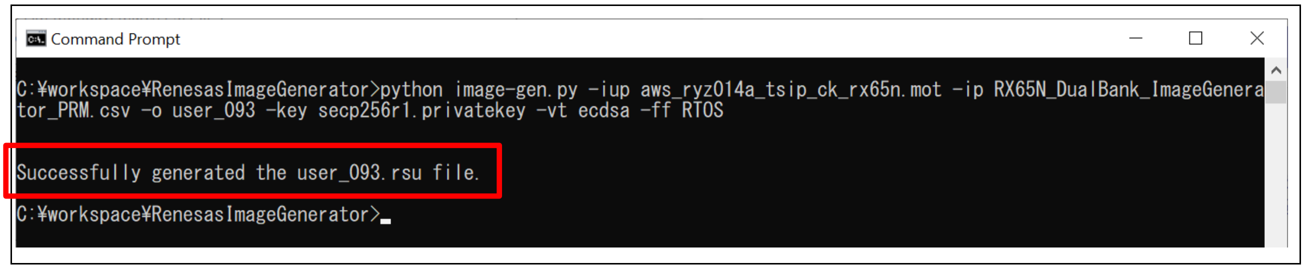
```
 * @brief Flag to enable or disable provisioning mode for the demo.
#define appmainPROVISIONING_MODE                          ( 1 )

 * @brief Certificate used for validating code signing signatures in the OTA PAL.
#ifndef otapalconfigCODE_SIGNING_CERTIFICATE
    #define otapalconfigCODE_SIGNING_CERTIFICATE    "Insert code signing certificate..."
#endif

 * @brief Major version of the firmware.
#ifndef APP_VERSION_MAJOR
    #define APP_VERSION_MAJOR    0
#endif

 * @brief Minor version of the firmware.
#ifndef APP_VERSION_MINOR
    #define APP_VERSION_MINOR    9
#endif

 * @brief Build version of the firmware.
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    3
#endif

 * @brief Server's root CA certificate.
#define democonfigROOT_CA_PEM                          tlsSTARFIELD_ROOT_CERTIFICATE_PEM
```

Figure 6-44 Changing the Version Number of the Update Firmware

(2) Using Renesas Image Generator to generate update firmware

Confirm that you have update firmware that was rebuilt in section 6.2.1(1) (aws_ryz014a_tsip_ck_rx65n.mot). Then, copy it to the Renesas Image Generator folder, overwriting the existing file with the same name, and then execute the following command in the Command Prompt window:

```
python image-gen.py -iup aws_ryz014a_tsip_ck_rx65n.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o user_093 -key secp256r1.privatekey -
vt ecdsa -ff RTOS
```



```
C:¥workspace¥RenesasImageGenerator>python image-gen.py -iup aws_ryz014a_tsip_ck_rx65n.mot -ip RX65N_DualBank_ImageGenera
tor_PRM.csv -o user_093 -key secp256r1.privatekey -vt ecdsa -ff RTOS

Successfully generated the user_093.rsu file.

C:¥workspace¥RenesasImageGenerator>_
```

Figure 6-45 Creating Update Firmware

Generation is complete when the following message is displayed on the command line: “Successfully generated the user_093.rsu file.”

The update firmware is created with the following file name:

- user_093.rsu

6.2.2 Updating the Firmware

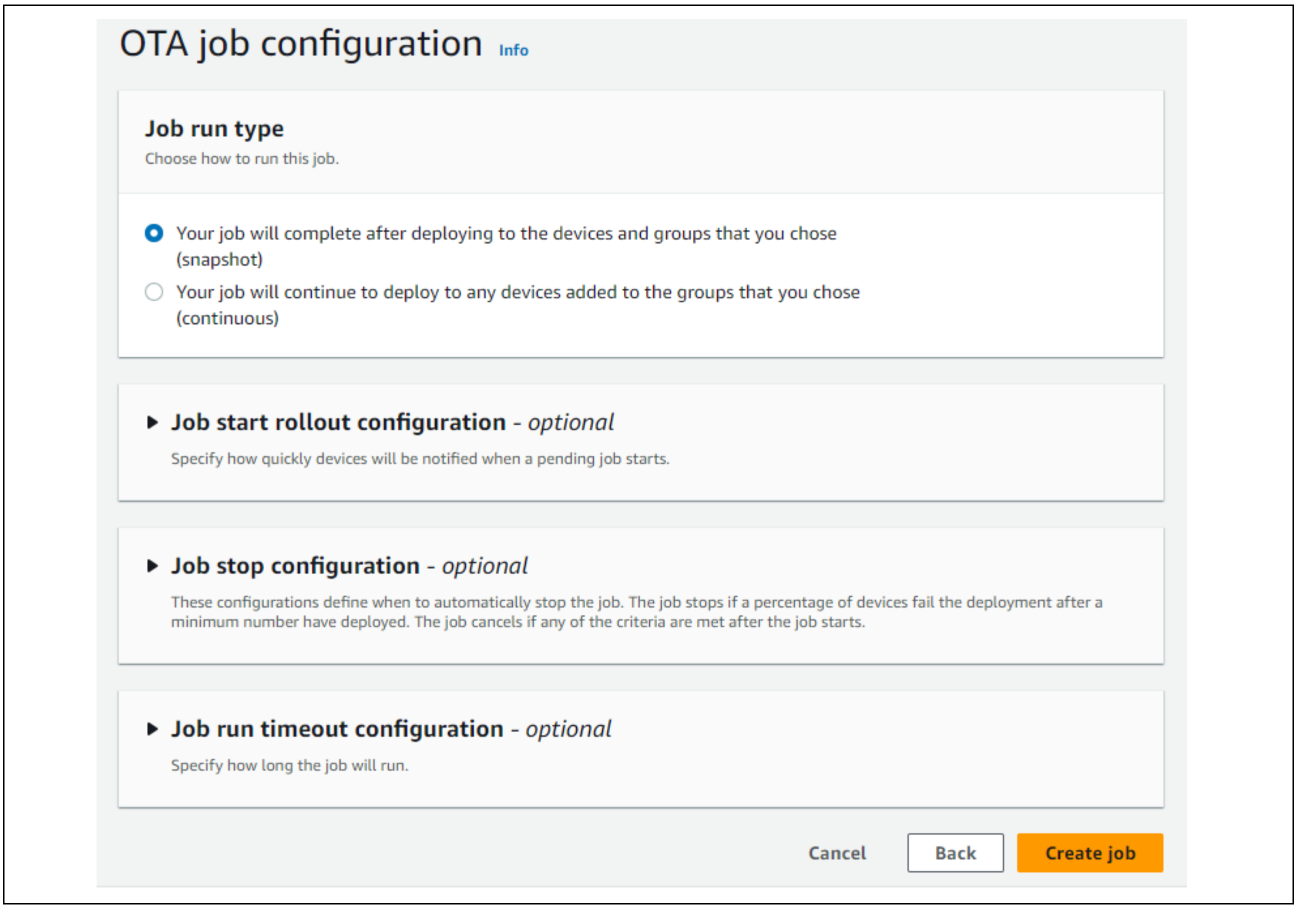
This section describes how to create an OTA update job that updates firmware. You create this job in AWS.

Before you create the job, execute the initial firmware on the target board and make sure that the program is waiting for the OTA update job to start by using the procedure described in section 6.1.5(7).

(1) Creating a firmware update job in AWS IoT Core

Before creating a firmware update job, register the update firmware in AWS. For details on the procedure, refer to “5.2 Updating the firmware” in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037).

Note that creation of an OTA update job requires the ECDSA certificates and keys created in section 5.2, Generating a Key Pair and Certificates for an OTA Update. Be sure to use the same data that was generated and used when the initial firmware was created.
When the OTA update job is created, OTA update is executed to update the firmware.



OTA job configuration Info

Job run type

Choose how to run this job.

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

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

▶ **Job start rollout configuration** - *optional*

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

▶ **Job stop configuration** - *optional*

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

▶ **Job run timeout configuration** - *optional*

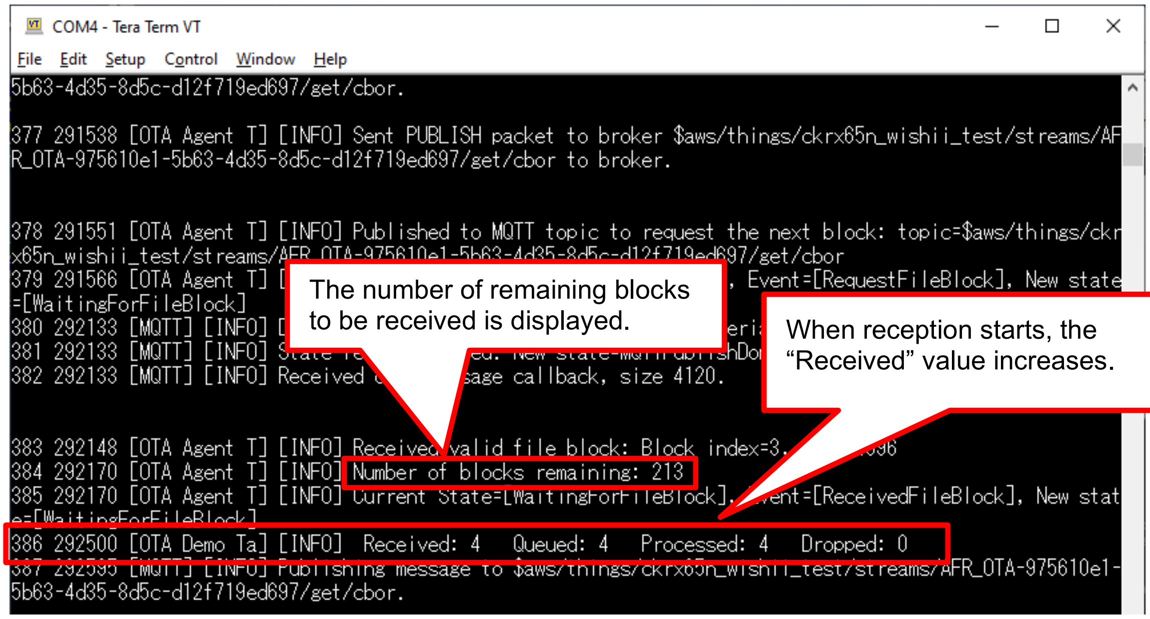
Specify how long the job will run.

Figure 6-46 Executing the OTA Update Job

(2) Receiving firmware

When the OTA update job starts, processing to receive firmware starts on the target board.

The firmware is received in units of blocks. The value of “Received” is incremented each time a block is received. When the value of “Number of blocks remaining” becomes 0, reception is complete.



```
5b63-4d35-8d5c-d12f719ed697/get/cbor.

377 291538 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/ckrx65n_wishii_test/streams/AF
R_OTA-975610e1-5b63-4d35-8d5c-d12f719ed697/get/cbor to broker.


378 291551 [OTA Agent T] [INFO] Published to MQTT topic to request the next block: topic=$aws/things/ckr
x65n_wishii_test/streams/AFR_OTA-975610e1-5b63-4d35-8d5c-d12f719ed697/get/cbor
379 291566 [OTA Agent T] [                                       , Event=[RequestFileBlock], New state
=[WaitingForFileBlock]
380 292133 [MQTT] [INFO] [                                     eri
381 292133 [MQTT] [INFO] State re                  ed. New state=MQTTPublishDo
382 292133 [MQTT] [INFO] Received                 sage callback, size 4120.


383 292148 [OTA Agent T] [INFO] Received valid file block: Block index=3.              096
384 292170 [OTA Agent T] [INFO] Number of blocks remaining: 213
385 292170 [OTA Agent T] [INFO] Current State=[WaitingForFileBlock],   ent=[ReceivedFileBlock], New stat
e=[WaitingForFileBlock]
386 292500 [OTA Demo Ta] [INFO]  Received: 4   Queued: 4   Processed: 4   Dropped: 0
387 292595 [MQTT] [INFO] Publishing message to $aws/things/ckrx65n_wishii_test/streams/AFR_OTA-975610e1-
5b63-4d35-8d5c-d12f719ed697/get/cbor.
```

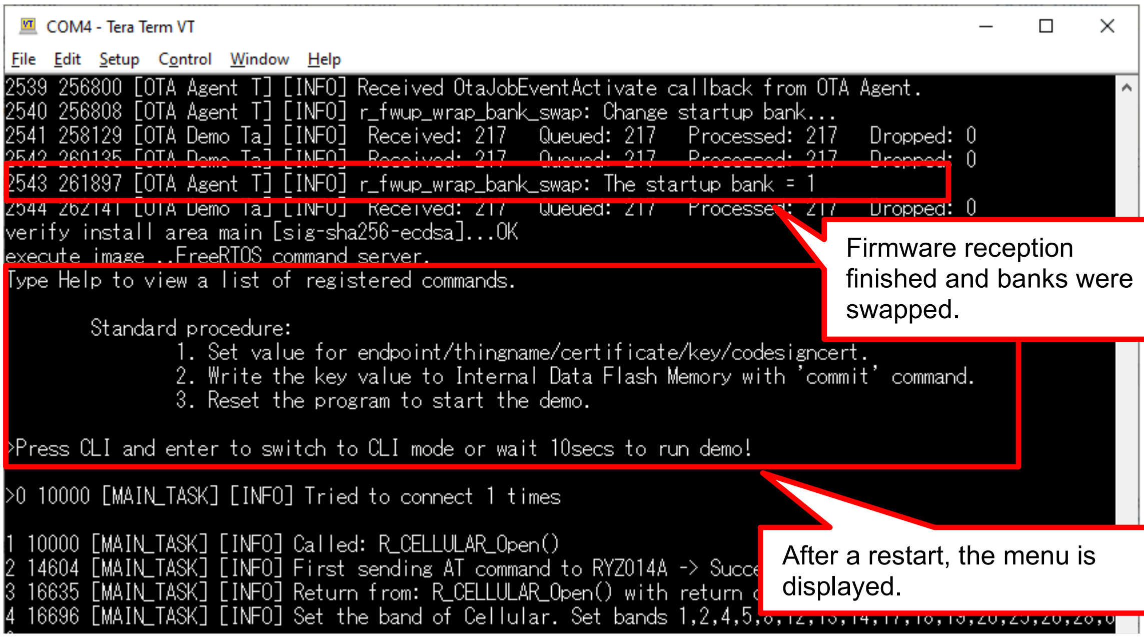
The number of remaining blocks to be received is displayed.

When reception starts, the “Received” value increases.

Figure 6-47 Log Data Displayed During Firmware Reception

(3) Completion of firmware reception

When all the firmware data is received and the firmware is successfully verified, the firmware is written, the banks are swapped, and then the update firmware is executed.
When the update firmware is executed normally, the initial menu is displayed.



```
2539 256800 [OTA Agent T] [INFO] Received OtaJobEventActivate callback from OTA Agent.
2540 256808 [OTA Agent T] [INFO] r_fwup_wrap_bank_swap: Change startup bank...
2541 258129 [OTA Demo Ta] [INFO]  Received: 217   Queued: 217   Processed: 217   Dropped: 0
2542 260135 [OTA Demo Ta] [INFO]  Received: 217   Queued: 217   Processed: 217   Dropped: 0
2543 261897 [OTA Agent T] [INFO] r_fwup_wrap_bank_swap: The startup bank = 1
2544 262141 [OTA Demo Ta] [INFO]  Received: 217   Queued: 217   Processed: 217   Dropped: 0
verify install area main [sig-sha256-ecdsa]...OK
execute image ...FreeRTOS command server.
Type Help to view a list of registered commands.

        Standard procedure:
                1. Set value for endpoint/thingname/certificate/key/codesigncert.
                2. Write the key value to Internal Data Flash Memory with 'commit' command.
                3. Reset the program to start the demo.

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

>0 10000 [MAIN_TASK] [INFO] Tried to connect 1 times

1 10000 [MAIN_TASK] [INFO] Called: R_CELLULAR_Open()
2 14604 [MAIN_TASK] [INFO] First sending AT command to RYZ014A -> Succe
3 16635 [MAIN_TASK] [INFO] Return from: R_CELLULAR_Open() with return c
4 16696 [MAIN_TASK] [INFO] Set the band of Cellular. Set bands 1,2,4,5,8,12,13,14,17,18,19,20,25,26,28,6
```

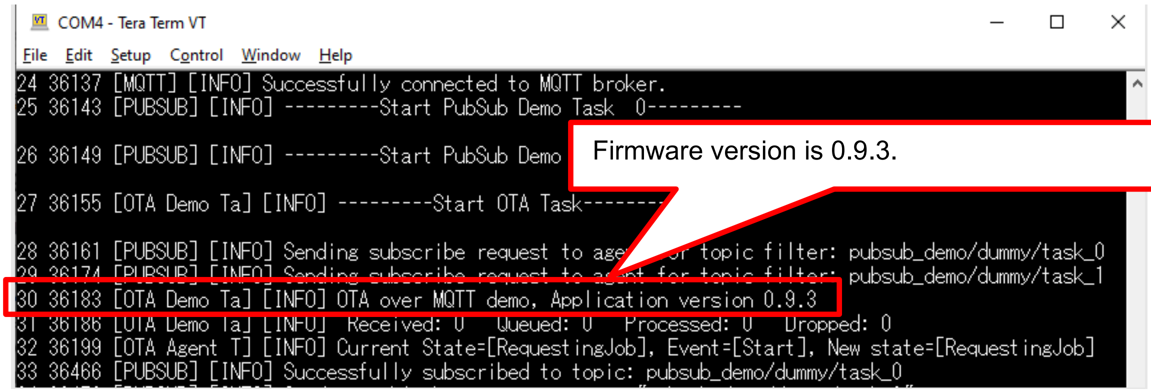
Firmware reception finished and banks were swapped.

After a restart, the menu is displayed.

Figure 6-48 Completion of Firmware Reception

(4) Confirming the firmware version

Confirm that the version number of the firmware is 0.9.3, which is the version number of the update firmware. If the PubSub Demo and OTA Demo operate in the same way as the initial firmware, the firmware update is complete.



```
24 36137 [MQTT] [INFO] Successfully connected to MQTT broker.
25 36143 [PUBSUB] [INFO] ---------Start PubSub Demo Task  0---------

26 36149 [PUBSUB] [INFO] ---------Start PubSub Demo

27 36155 [OTA Demo Ta] [INFO] ---------Start OTA Task-------

28 36161 [PUBSUB] [INFO] Sending subscribe request to age    or topic filter: pubsub_demo/dummy/task_0
29 36174 [PUBSUB] [INFO] Sending subscribe request to agent for topic filter: pubsub_demo/dummy/task_1
30 36183 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.3
31 36186 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
32 36199 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
33 36466 [PUBSUB] [INFO] Successfully subscribed to topic: pubsub_demo/dummy/task_0
```

Firmware version is 0.9.3.

Figure 6-49 Confirming the Firmware Version

7. Appendix

7.1 Notes on Executing the Sample Program on Multiple Devices Concurrently in the Same LAN Environment

The sample code (ethernet version) includes MAC addresses assigned to vendor IDs of Renesas Electronics.

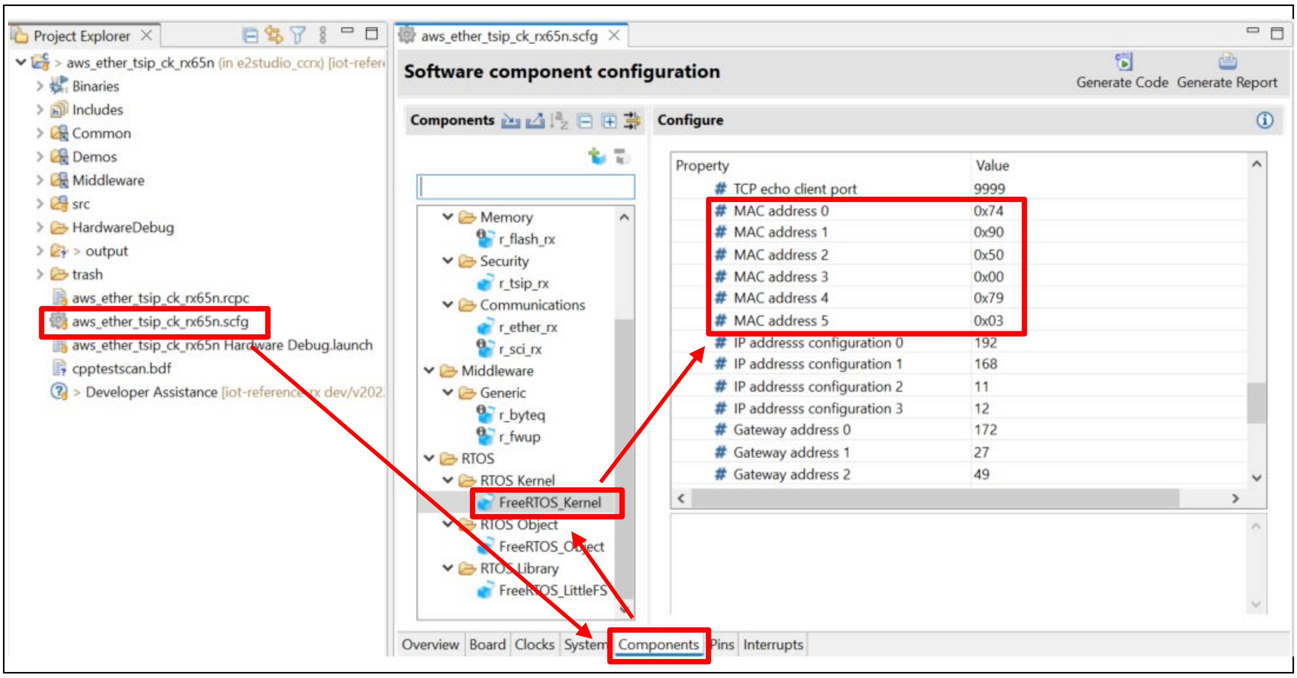
If you execute the sample program on multiple devices concurrently in the same LAN environment, make sure that the MAC addresses of each instance of the sample program are unique.

If multiple instances of the sample program having the same MAC addresses are executed on multiple devices concurrently, they may not operate correctly.

The following shows the procedure for changing the MAC addresses.

Open Smart Configurator (aws_ether_tsip_ck_rx65n.scfg), and then select the **Components** tab.
In the tree view, select **RTOS** > **RTOS Kernel** > **FreeRTOS_Kernel**. Then, in the list box that appears in the right pane, change the value of the “MAC address 0” to “MAC address 5” properties to any hexadecimal values of your choice.
Specify “0x” followed by a two-digit hexadecimal number for each property.

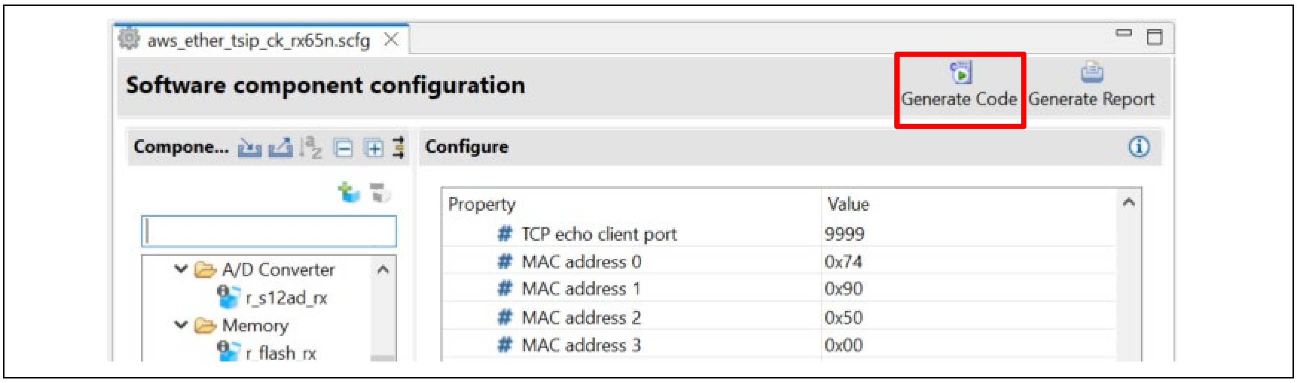
Customers who create their own products from the sample program must use MAC addresses that the customers themselves obtained from IEEE.



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

Figure 7-1   MAC Address Settings

After you have changed the settings as described above, click the **Generate Code** button at the top right of the window. The changes (made in Smart Configurator) are applied to the relevant code.



Property	Value
TCP echo client port	9999
MAC address 0	0x74
MAC address 1	0x90
MAC address 2	0x50
MAC address 3	0x00

Figure 7-2 Generating Code

8. Troubleshooting

The following table shows problems that may occur when the sample program is executed and their solutions.

Table 8-1 Troubleshooting (1)

No.	Problem	Cause	Solution	Reference
1	The command that creates the initial firmware fails.	The path to Python is unknown.	Re-install Python. At this time, make sure that the **Add python.exe to PATH** check box is selected. *2	*1
2		No cryptographic library is installed.	Install a cryptographic library.	*2
3	The initial firmware cannot be written.	The CK-RX65N is not set in DEBUG mode.	On the CK-RX65N, make sure that J16 pins 1 and 2 are closed to set DEBUG mode.	*3
4	The initial firmware cannot be started.	The CK-RX65N is not set in RUN mode.	On the CK-RX65N, make sure that J16 pins 2 and 3 are closed (RUN mode).	6.1.4(4)
5	Cellular communication cannot be started.	The RYZ014A PMOD expansion board is not connected correctly.	Review the connection of the RYZ014A PMOD expansion board.	*3
6		No SIM card is inserted.	Insert a SIM card.	*3
7		The SIM card settings are not specified correctly.	Review the configuration settings of the “r_cellular” module.	6.1.2(4)
8		The SIM card bundled with the CK-RX65N is used and is not activated.	Activate the SIM card.	6.1.2(4)
9	An error occurs during cellular communication.	The communication environment is poor.	Connect an antenna and power supply to the RYZ014A PMOD expansion board. Also make sure that the antenna is placed at a window or another location where communication quality is good.	*3

Notes: 1. Section 2.2 in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037)

2. Section 2.2(5) in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037)

3. Section 2.5 in the following application note: “RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.3 or later)” (R01AN7037)

Table 8-2 Troubleshooting (2)

No.	Problem	Cause	Solution	Reference
10	Connection to AWS fails.	AWS IoT information is not set or incorrect AWS IoT information is set.	Set the AWS IoT information again.	6.1.4
11		The data of certificates and keys is not created or registered normally.	Create the data of certificates and keys again by using the correct procedures.	5.1
12	Firmware does not start after the bootloader is started.	A public key is not correctly set in the bootloader.	Review the public key settings in the bootloader.	6.1.2(1)
13	Firmware does not start after OTA update is performed.	A public key is not correctly set in the firmware.	Review the public key settings in the firmware.	6.2.1(2)
14		The device selection settings are not correctly specified.	Review the device settings in the firmware and bootloader.	6.1.5

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	June 14, 2024	—	First edition issued

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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