# Using AWS IoT Fleet Provisioning with RAFW Connectivity Devices

This document shows how to use the reference application of AWS IoT Fleet Provisioning.

## Contents


**1. Introduction** ..... **3**
1.1 Software Requirements ..... 3
1.2 Hardware Requirements ..... 3
**2. Hardware Setup and Configuration** ..... **4**
**3. Reference Application Setup** ..... **4**
3.1 Set Up Hardware ..... 4
3.2 Program Firmware Image ..... 4
3.3 Prepare AWS Account ..... 4
**4. AWS IoT Fleet Provisioning Example** ..... **5**
4.1 Run Fleet Provisioning Reference Application ..... 6
4.2 AT Command List ..... 8
**5. Revision History** ..... **9**


## Figures


Figure 1. Launching MQTT test client ..... 7


## Tables


Table 1. Basic set of AT commands from MCU to RA6W1/RA6W2 ..... 8
Table 2. AT commands to write certificate ..... 8



© 2026 Renesas Electronics

# Terms and Definitions

| | |
|---|---|
| ARN | Amazon Resource Name |
| AT | Attention |
| AWS | Amazon Web Services |
| CA | Certificate Authority |
| EVK | Evaluation Kit |
| FSP | Flexible Software Package |
| GCC | GNU Compiler Collection |
| ID | Identity |
| IDE | Integrated Development Environment |
| IoT | Internet of Things |
| MQTT | Message Queuing Telemetry Transport |
| NVRAM | Non-Volatile Random-Access Memory |
| PC | Personal Computer |
| TLS | Transport Layer Security |
| USB | Universal Serial Bus |
| Wi-Fi® | Wireless Fidelity |

# 1. Introduction

Start using the AWS IoT Fleet Provisioning feature with a ready-to-use reference application. The provided application project template (`awsiot_wifi_fleet_provisioning_ek_ra6w1 and awsiot_wifi_fleet_provisioning_ek_ra6w2`) includes pre-configured project settings, middleware, and example code, which helps save time and ensures that the environment is correctly set up for AWS IoT development.
This application demonstrates how to implement AWS Fleet Provisioning using the Fleet Provisioning by Claim method. The AWS IoT Fleet Provisioning reference application showcases a way to provision a fleet of IoT devices with unique certificates and register them with AWS IoT Core using the Fleet Provisioning feature. It shows how devices with the ability to generate a public-private key pair on board can use a common claim certificate (across the entire fleet of devices) to request unique certificates from AWS IoT Core for their generated key pairs and register themselves with AWS IoT Core as AWS IoT thing resources.
The following are the main components of AWS IoT Fleet Provisioning reference application:

- RA6W1/RA6W2 EVK
- AWS IoT Fleet Provisioning reference application package
- Terminal application
- Router with an Internet connection
- AWS account.

## 1.1 Software Requirements

The following software tools and packages are required to complete the project:

- Renesas Electronics Flexible Software Package (FSP): Version 2.2.0 or later.
- $e^2$ studio IDE: Version 2026-04.3 or later.
- GCC Arm Embedded Toolchain: Version 13.3.1.arm-13-24.
- Serial Terminal Application: Tera Term or similar tools for logging and monitoring output.

## 1.2 Hardware Requirements

- Renesas RA6W1/RA6W2 Evaluation Kit (EVK).
- RA6W1/RA6W2 MCUwith ultra-low power consumption.
- USB Type-C Cable: For programming and debugging.
- Jumper Wire (Female to Female): For making proper motherboard configurations for the application.
- Host PC: For programming, viewing console logs, opening MQTT test client and giving AT commands.
- Wi-Fi Access Point (AP): With network connectivity.

# 2. Hardware Setup and Configuration

To connect hardware:

1. Connect the RA6W1/RA6W2 MCU to the Evaluation Kit (EVK) using the extension socket.
2. Connect the USB Debug port on the EVK to the host PC for programming and debugging using a USB Type-C cable.
3. Set up jumpers as following:
   - UART (RX): P0_00
   - UART (TX): P0_01
   - J102: {2-3} [Current Measurement]
   - J105: {2,3} [Voltage Selector]
   - J106: {2,3} [Voltage Selector]
   - J107: {2,3} [Voltage Selector]
   - J210: {1,2} [Reset]
   - SWCLK(c): SWCLK
   - SWDIO(c): SWDIO

# 3. Reference Application Setup

Before starting the application, ensure your hardware environment is prepared, the firmware is programmed, and your AWS account is configured. See the following sections to finalize the application setup.

## 3.1 Set Up Hardware

To run the reference application, you need an RA6W1 or RA6W2 evaluation kit (EK-RA6W1/RA6W2). An external access point with internet access is required for the final Wi-Fi connectivity.

## 3.2 Program Firmware Image

The firmware image for reference application is generated and flashed onto the EK-RA6W1/RA6W2 using e$^2$ studio with J-Link support. The e$^2$ studio provides an integrated build and flash environment, eliminating the need for external flash programming scripts.

To flash the firmware image:

1. Open the e$^2$ studio and connect your evaluation board to the host PC using the USB (using the inbuilt SEGGER J-Link).
2. Load the reference application to the e$^2$ studio. The reference application for AWS Fleet Provisioning is in the following path: `application_projects/r19an0471/AWS_Fleet_Provisioning_Reference_Application/`. For more information on how to load the template, see the Load Project Template in e$^2$ Studio section in *RA6W1*/RRQ61001*Application Development Guide*.
3. Flash the image using the debugger. For more information, see the Flash and Debug using e$^2$ Studio section in *RA6W1/RRQ61001 Application Development Guide*.

When flashing is complete, the device starts running the firmware.

## 3.3 Prepare AWS Account

To prepare the AWS account for fleet provisioning, complete the steps described in the AWS Settings for Fleet Provisioning section in *RA6W1/RRQ61001 AWS® IoT Core Quick Start Guide* or *RA6W2/RRQ61051 AWS® IoT Core Quick Start Guide*.

# 4. AWS IoT Fleet Provisioning Example

In the Fleet Provisioning reference application, the Fleet Provisioning by Claim method was used. This method can be further differentiated by the process in which it obtains a permanent certificate and private key from AWS.

- **CreateKeysAndCertificate** – Call `CreateKeysAndCertificate` to create a new certificate and private key using the AWS certificate authority.
- **CreateCertificateFromCsr** – Call `CreateCertificateFromCsr` to generate a certificate from a certificate signing request that keeps its private key secure.

In the reference application, the `CreateCertificateFromCsr` method was used.
The following is the flow of Fleet Provisioning by Claim method:

1. Device opens TLS connection to AWS IoT using initial Claim certificate and Claim private key and connects to AWS IoT MQTT broker.
2. Generates a private/public key pair.
3. Keeps private key secure by saving it in the device.
4. Creates a Certificate Signing Request (CSR) and publishes to `$aws/certificates/create-from-csr/<payload-format>`. The message payload format is CBOR or JSON.

```
Payload: {
    "certificateSigningRequest": "string"
    }
```

5. AWS responds with: New device certificate, Certificate ID, Certificate ARN, Certificate ownership token in the topic: `$aws/certificates/create-from-csr/<payload-format>/accepted`.

```
Payload: {
    "certificateOwnershipToken": "string",
    "certificateId": "string",
    "certificatePem": "string"
    }
```

   This certificate is unique per device, and it is saved in the device securely.
   If the request is not accepted, the response is received in the topic: `$aws/certificates/create-from-csr/<payload-format>/rejected`.

```
Payload: {
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
    }
```

6. Device calls RegisterThing, publishes to `$aws/provisioning-templates/<template-name>/provision/<payload-format>`.

```
Payload: {
    "certificateOwnershipToken": "string",
    "parameters": {
        "string": "string",
        ...
    }
}
```

   The RegisterThing response is received in the topic: `$aws/provisioning-templates/<templateName>/provision/<payload-format>/accepted`.

```
Payload: {
    "deviceConfiguration": {
        "string": "string",
        ...
    },
    "thingName": "string"
}
```

If the RegisterThing request fails, an error response is received in the topic: `$aws/provisioning-templates/<templateName>/provision/<payload-format>/rejected`.

```
Payload: {
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
    }
```

7. AWS creates the Thing, attaches a certificate, attaches a policy, and activates a certificate.

Now the device is ready to disconnect and connect again using the provisioned unique certificate and key.

## 4.1 Run Fleet Provisioning Reference Application

To run Fleet Provisioning Reference Application:

1. Import the Fleet Provisioning Example project into e$^2$ studio.
2. Generate and build the Example project.
3. Connect the RA6W1/RA6W2 EVK Debug port to the host PC using a USB Type-C cable.
4. Debug or flash the project to the RA6W1/RA6W2 board. When flashing is complete, press the reset button.
5. Use any terminal application for logging.
6. On initial boot, the Wi-Fi setup menu is displayed. Configure the Wi-Fi settings required for the Internet connection.
7. Write the RootCA, Claim Certificate, and Claim Private Key (which were created in Section 3.3 Prepare AWS Account) to the NVRAM using CLI commands:
   a. RootCA – `net cert write ca6`
   b. Claim Certificate – `net cert write initcert6`
   c. Claim Private Key – `net cert write initkey6`
8. Write the Device ID and Fleet Provisioning Template Name (which were configured in Section 3.3 Prepare AWS Account) to the NVRAM using CLI commands.
   a. `nvram setenv appcfg fp_dev_id <DeviceID>`
   b. `nvram setenv appcfg fp_tmpl_name <TemplateName>`

9. Open the MQTT test client in AWS IoT console in a web browser and subscribe to the topic (Figure 1 explains how to launch the MQTT test client):
   `$aws/things/<Thingname>/shadow/update/delta`.

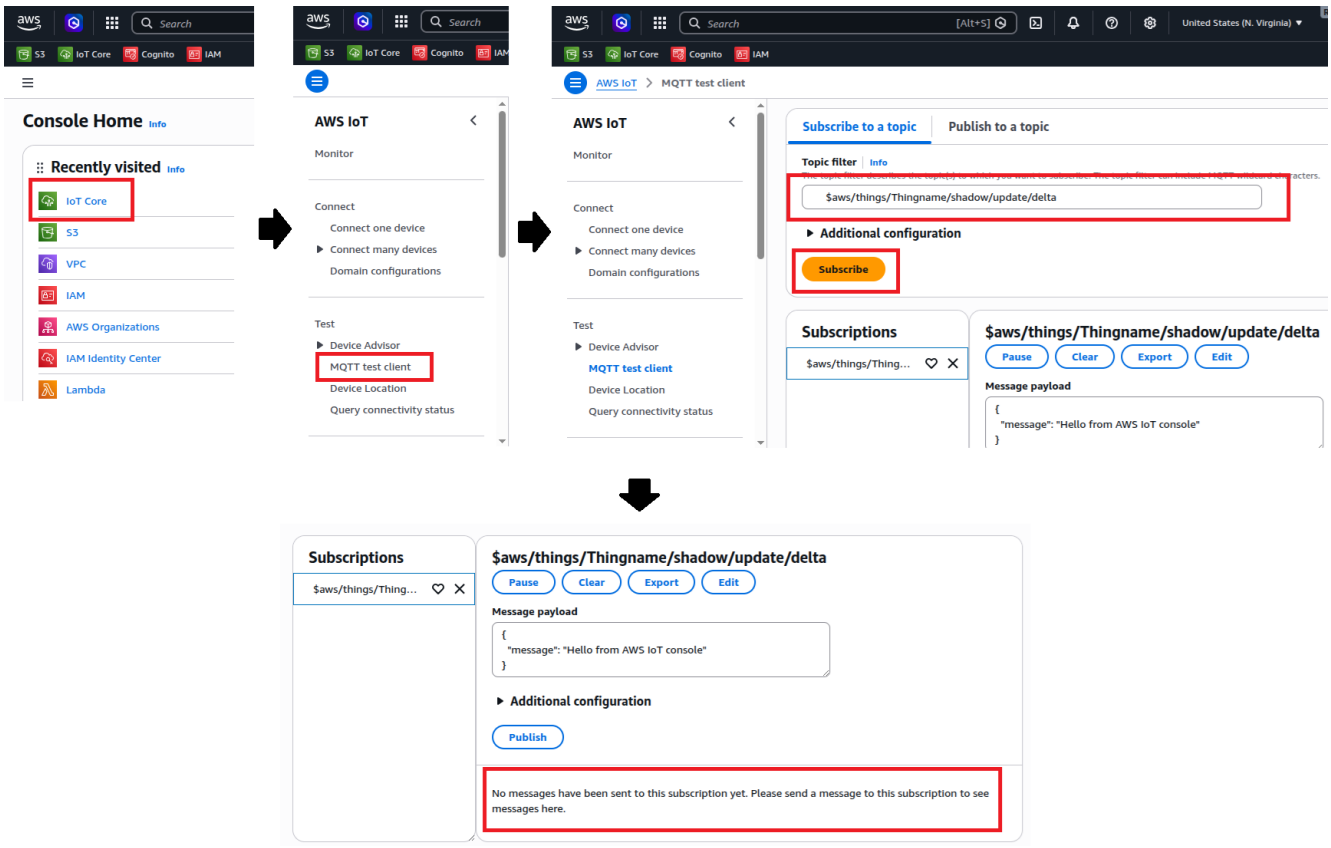



**Figure 1. Launching MQTT test client**

10. Reboot the device using the `reboot` CLI command or using the reset button.
11. After the reboot is complete, RA6W1/RA6W2 connects to the configured AP.
12. After the connection to AP is successful, RA6W1/RA6W2 checks for the AWS IoT provisioned credentials. If it is available, RA6W1/RA6W2 directly goes for AWS IoT Core connection using the provisioned credentials. If not, RA6W1/RA6W2 initiates the Fleet Provisioning by Claim process. Since this is the initial boot, RA6W1/RA6W2 goes for Fleet Provisioning.
13. After Fleet Provisioning is complete, the unique certificate received from AWS Core and the unique key generated locally are saved in the NVRAM and RA6W1/RA6W2 disconnects itself from AWS IoT Core and reconnects using the provisioned credentials.
14. After the Fleet Provisioning is complete, it publishes a message to the topic `$aws/things/<Thingname>/shadow/update/delta`.
15. Since the fleet provisioning is complete and if RA6W1/RA6W2 is rebooted, it directly connects to AWS IoT Core using the provisioned credentials and publishes to the same topic.

## 4.2 AT Command List

In addition to CLI commands, the reference application also provides AT commands for configuring the fleet provisioning parameters. The following tables list the available AT commands.

**Table 1. Basic set of AT commands from MCU to RA6W1/RA6W2**

| Head | Main | Sub | Parameters |
|---|---|---|---|
| AT+AWS= | SET | DEVICE_ID | This is the ID given specifically for each device. The device Thing Name and Client Identifier is created from this ID. |
| | | FP_TMPL_ NAME | Name of the provisioning template to use for the RegisterThing portion of the Fleet Provisioning workflow. |
| | | AWS_ BROKER | The AWS IoT broker endpoint to connect. This can be obtained from the AWS account settings. |
| | | PORT | The port to use for the application. Default is 443. |
| Example:<br>AT+AWS=SET,DEVICE_ID,RA6Wx_FP_Test_DevID_1 | | | |

**Table 2. AT commands to write certificate**

| Escape Sequence | Parameters | Description |
|---|---|---|
| <ESC>CERT | <module>,<certificate type>,<mode> [,<format>,<length>,<content>] | Save or delete certificate/CA/Private Key.<br><ESC>CERT: To enter certificate input mode.<br>■ <Module>: Module ID<br>■ 6: AWS<br>■ <Certificate type>: Certificate type<br>■ 0: CA<br>■ 1: Certificate<br>■ 2: Private Key<br>■ <mode>: Input mode<br>■ 0: Store<br>■ 1: Delete<br>■ <format>: Certificate format (only used when mode is 0)<br>■ 0: DER (currently not supported for AWS claim certificate and claim private key)<br>■ 1: PEM<br>■ <length>: Length of the certificate (only used when mode is 0)<br>■ <content>: Certificate data (only used when mode is 0)<br>Response: OK or ERROR |
| Example:<br><ESC>CERT,6,0,0,1,1187,-----BEGIN CERTIFICATE-----<br>MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF<br>-----END CERTIFICATE----- | | |

# 5. Revision History

| Revision | Date | Description |
|---|---|---|
| 1.02 | Sep 21, 2026 | Updated for FSP version 2.2.0, Reference document name update. |
| 1.01 | July 21, 2026 | Updated for FSP version 2.1.0. |
| 1.00 | May 29, 2026 | Initial release. |

## IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

**CorporateHeadquarters**

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-us/

© 2026 Renesas Electronics Corporation. All rights reserved.