



REALTEK

UM0207

AmebaZ2 Amazon FreeRTOS Getting Started
Guide

Abstract

This document describes the information of Mbed TLS Version management.

COPYRIGHT	4
TRADEMARKS	5
USING THIS DOCUMENT	5
REVISION HISTORY	6
1 AmebaZ2 RTL8720CM Board	7
1.1 AmebaZ2 Demo EVB	7
1.2 PCB Layout Overview	7
1.3 Pin-Out Reference	8
1.4 LED State	9
2 Configure AWS IoT Core	10
2.1 Create a New Device	10
3 Configure AmebaZ2 Amazon FreeRTOS	17
3.1 Download Source Code from github	17
3.1.1 Cloning a repository by Download ZIP	17
3.2 Get Broker Endpoint by AWS IoT Core	18
3.3 Get Thing Name	18
3.4 Setup IoT Core Information with AmebaZ2 Amazon FreeRTOS	19
3.4.1 Setup Thing's Private Key and Certificate	19
3.4.2 Enable FreeRTOS demo on AmebaZ2	21
4 Compile AmebaZ2 Amazon FreeRTOS	22
4.1 Pre-Requisite	22
4.2.1 Install IAR IDE	22
4.2.2 Compilation	22
4.3 GCC Environment	23
4.3.1 Install Cygwin on Windows (Using Cygwin)	23
4.3.2 Compile Project on Cygwin or Ubuntu/Linux	25
Open "Cygwin Terminal" or in Ubuntu/Linux	25
4.4 Generate Image Binary	25
5 ImageTool	26
5.1 Introduction	26
5.2 Environment Setup	26
5.2.1 Hardware Setup	26
5.2.2 Software Setup	27
6 MQTT Demo	29
6.1 Get Device Log	29
6.2 Run MQTT Demo	29
6.3 Monitoring MQTT messages on the cloud	31
7 Troubleshooting	32
7.1 Image Tool Download Fail	32

7.2	ERROR: Invalid Key.....	33
7.3	Failed to establish new MQTT connection.....	34
7.4	TLS_Connect fail.....	34
8	OTA Demo	35
8.1	OTA Update Prerequisites	35
8.2	Set the Firmware Version and App Version to Image File	35
8.3	How Custom Signed Image File is Created	36
8.4	How to Trigger a Custom Signed OTA Job in Amazon AWS IOT Core	37
8.5	Run OTA Demo	42
9	Troubleshooting	44
9.1	ERROR: Invalid Key	44
9.2	Failed to establish new MQTT connection	44
	TLS_Connect fail	44
10	AWS Fleet Provisioning Introduction.....	45
10.1	Create Claim Certificate Policy	45
10.2	Create Claim Certificate.....	46
10.3	Create fleet provisioning template.....	48
10.4	Configure AmebaZ2 Amazon FreeRTOS.....	51
	10.4.1 Get Broker Endpoint by AWS IoT Core	51
	10.4.2 Setup IoT Core Information with AmebaZ2 Amazon FreeRTOS.....	53

COPYRIGHT

© 2019 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

DISCLAIMER

Please Read Carefully:

Realtek Semiconductor Corp., (Realtek) reserves the right to make corrections, enhancements, improvements and other changes to its products and services. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

Reproduction of significant portions in Realtek data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Realtek is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions.

Buyers and others who are developing systems that incorporate Realtek products (collectively, "Customers") understand and agree that Customers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Customers have full and exclusive responsibility to assure the safety of Customers' applications and compliance of their applications (and of all Realtek products used in or for Customers' applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Customer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Customer agrees that prior to using or distributing any applications that include Realtek products, Customer will thoroughly test such applications and the functionality of such Realtek products as used in such applications.

Realtek's provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation kits, (collectively, "Resources") are intended to assist designers who are developing applications that incorporate Realtek products; by downloading, accessing or using Realtek's Resources in any way, Customer (individually or, if Customer is acting on behalf of a company, Customer's company) agrees to use any particular Realtek Resources solely for this purpose and subject to the terms of this Notice.

Realtek's provision of Realtek Resources does not expand or otherwise alter Realtek's applicable published warranties or warranty disclaimers for Realtek's products, and no additional obligations or liabilities arise from Realtek providing such Realtek Resources. Realtek reserves the right to make corrections, enhancements, improvements and other changes to its Realtek Resources. Realtek has not conducted any testing other than that specifically described in the published documentation for a particular Realtek Resource.

Customer is authorized to use, copy and modify any individual Realtek Resource only in connection with the development of applications that include the Realtek product(s) identified in such Realtek Resource. No other license, express or implied, by estoppel or otherwise to any other Realtek intellectual property right, and no license to any technology or intellectual property right of Realtek or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Realtek products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of Realtek Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from Realtek under the patents or other Realtek's intellectual property.

Realtek's Resources are provided "as is" and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding resources or use thereof, including but not limited to accuracy or completeness, title, any epidemic failure warranty and any implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third party intellectual property rights.

Realtek shall not be liable for and shall not defend or indemnify Customer against any claim, including but not limited to any infringement claim that related to or is based on any combination of products even if described in Realtek Resources or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek's Resources or use thereof, and regardless of whether Realtek has been advised of the possibility of such damages. Realtek is not responsible for any failure to meet such industry standard requirements.

Where Realtek specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Customers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any Realtek products in lifecritical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death. Such equipment includes, without limitation, all medical devices identified by the U.S.FDA as Class III devices and equivalent classifications outside the U.S.

Customers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Customers' own risk. Customers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Customer will fully indemnify Realtek and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's noncompliance with the terms and provisions of this Notice.

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

USING THIS DOCUMENT

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

REVISION HISTORY

Revision	Release Date	Summary
1.3	2024/09/12	Update example and build method

1 AmebaZ2 RTL8720CM Board

1.1 AmebaZ2 Demo EVB

Ameba Demo board home page: <https://www.amebaito.com/amebaz2/>

Ameba RTL8720CM Board (AMB 31)



CPU

- 32-bit Arm® Cortex®-M4, up to 100MHz

Memory

- 256KB SRAM + 4MB PSRAM

Key Features

- Integrated 802.11n Wi-Fi SoC
- Hardware SSL Engine
- Root Trust Secure Boot
- BLE4.2

[Manual / Schematic / Layout](#)

[Buy it](#)

1.2 PCB Layout Overview

RTL8720C embedded on Ameba-ZII DEV demo board, which consists of various I/O interfaces. For the details of the HDK, please contact us for further reference.

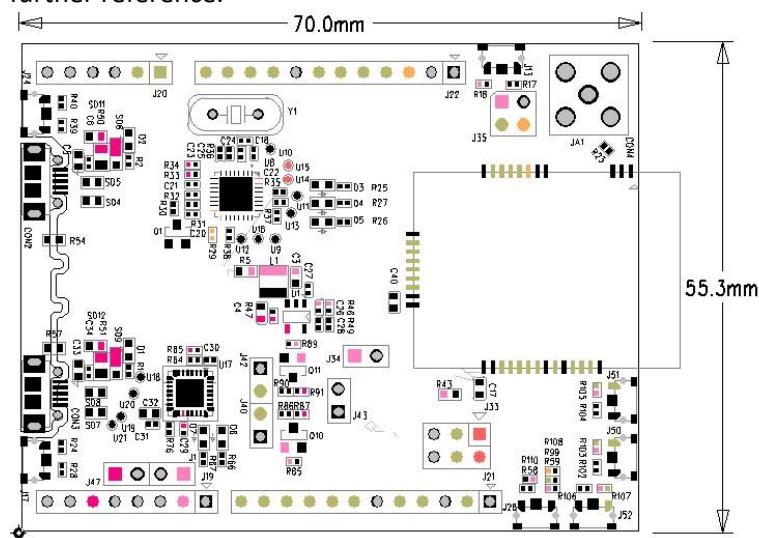
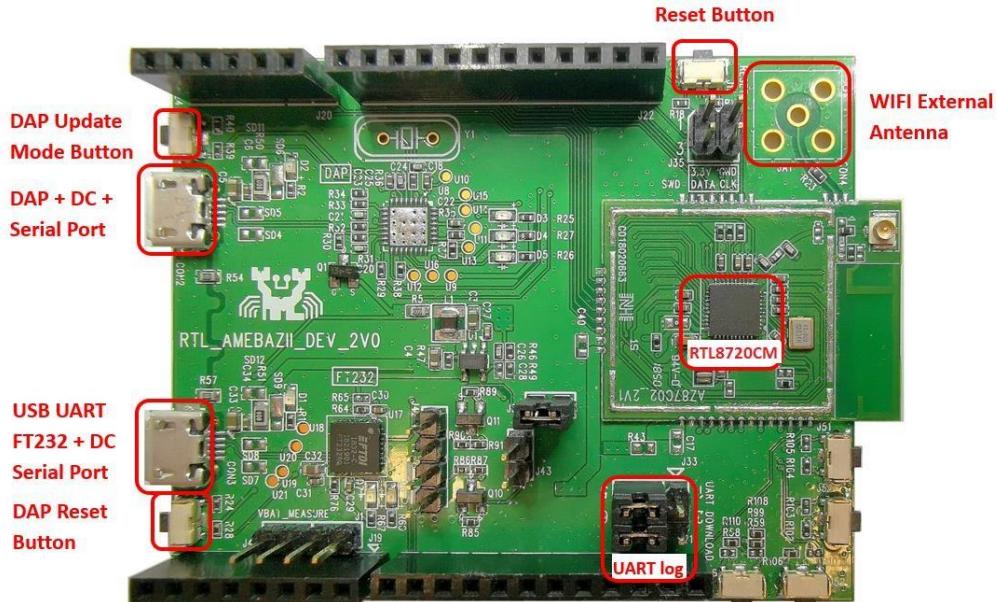


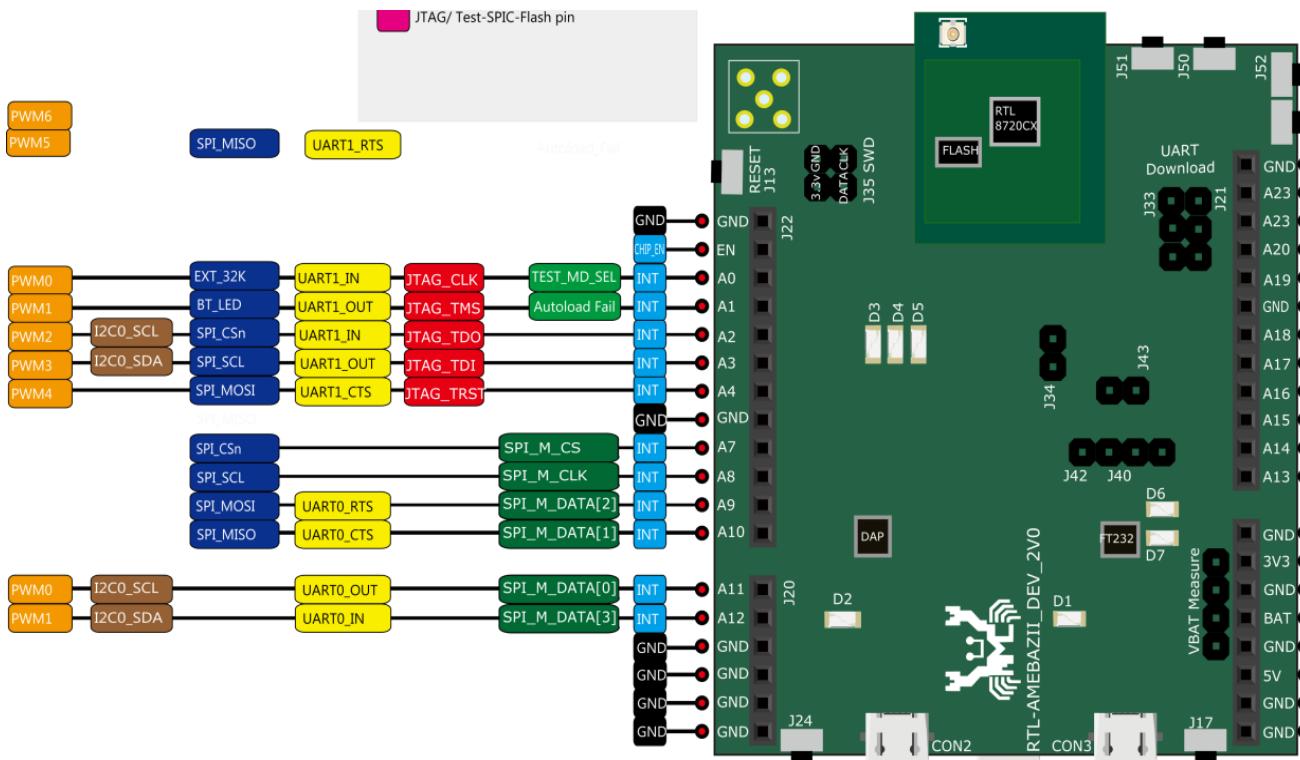
Figure 1-1 Top View of Ameba-ZII 2V0 Dev Board



Figure

1-2 Ameba-ZII 2V0 Dev Board PCB Layout

1.3 Pin-Out Reference



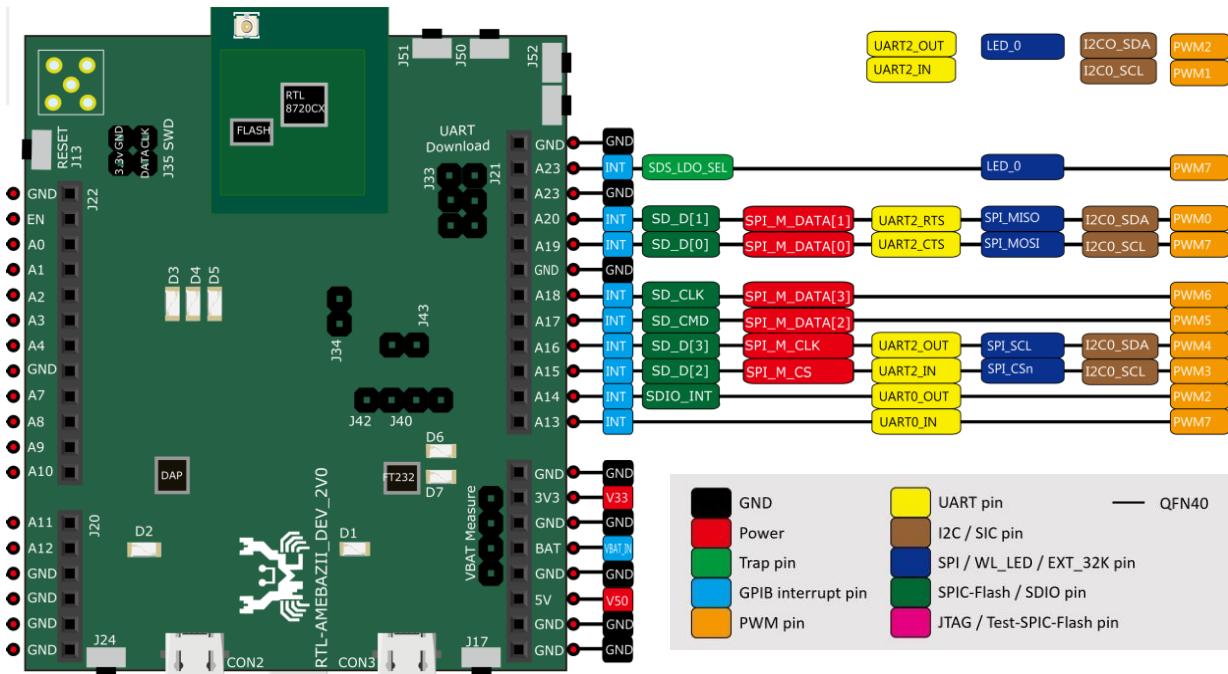
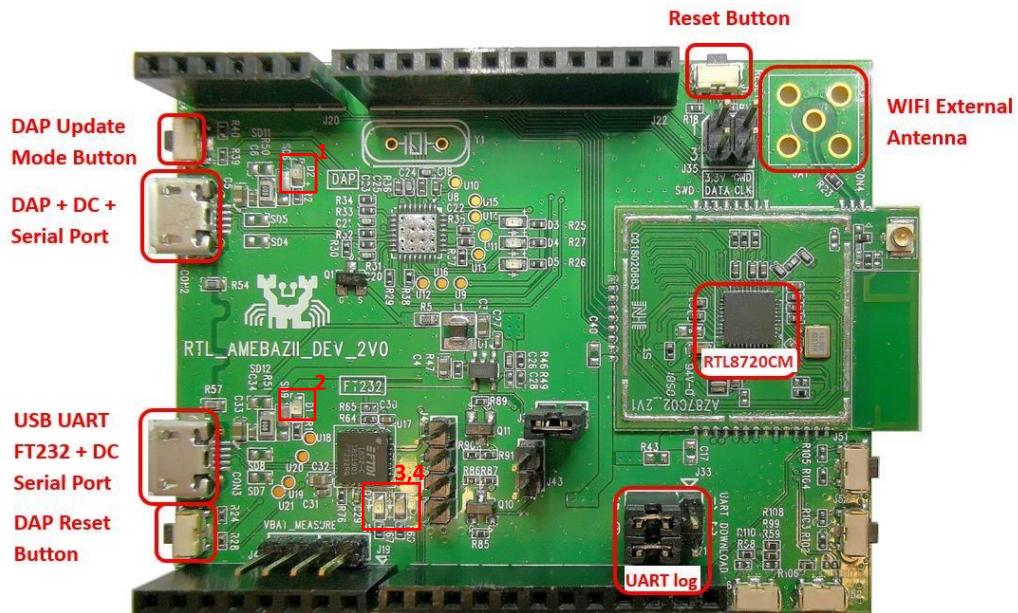


Figure 1-3 Pin Out Reference for DEV_2V0

1.4 LED State

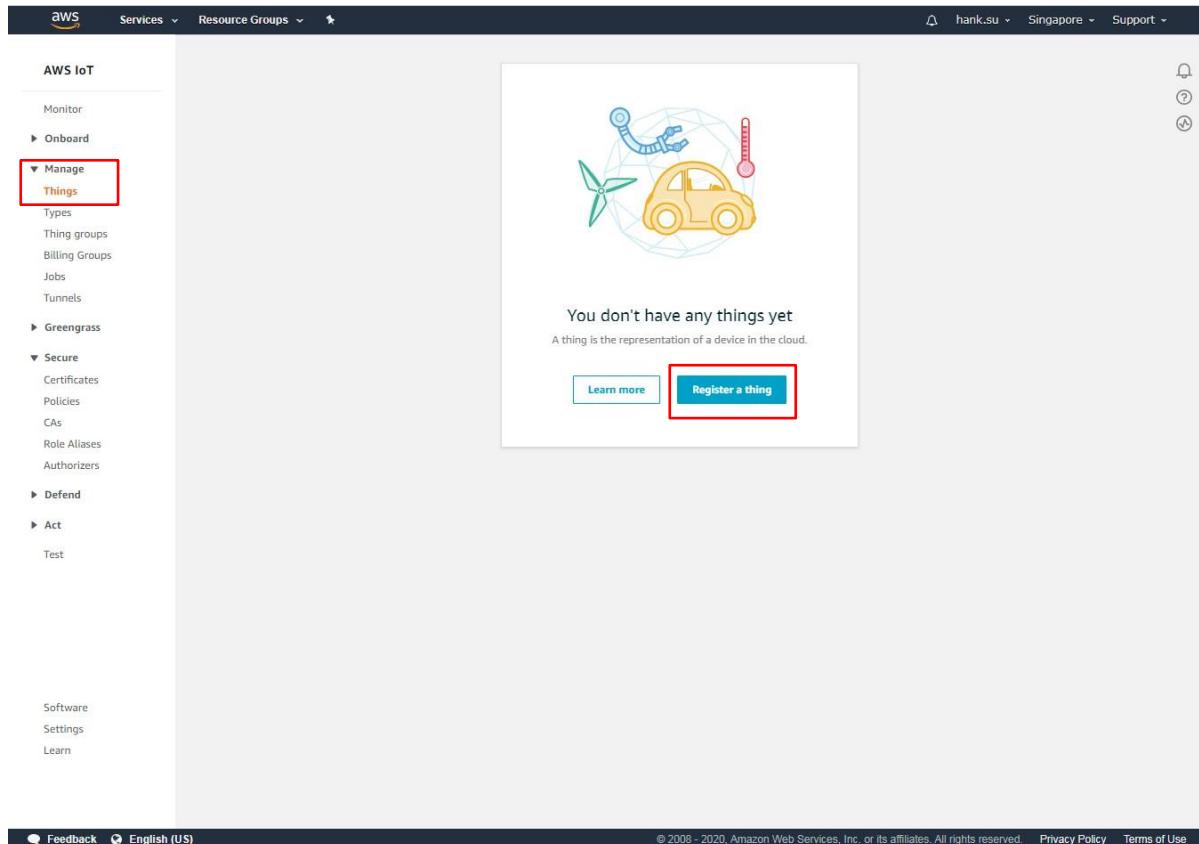
There are four LED on the AmebaZ2 EVB. LED1 lights steady green and LED2 steady red when device have power. LED3 and LED4 go with log uart, they flash red and green when uart communicating.

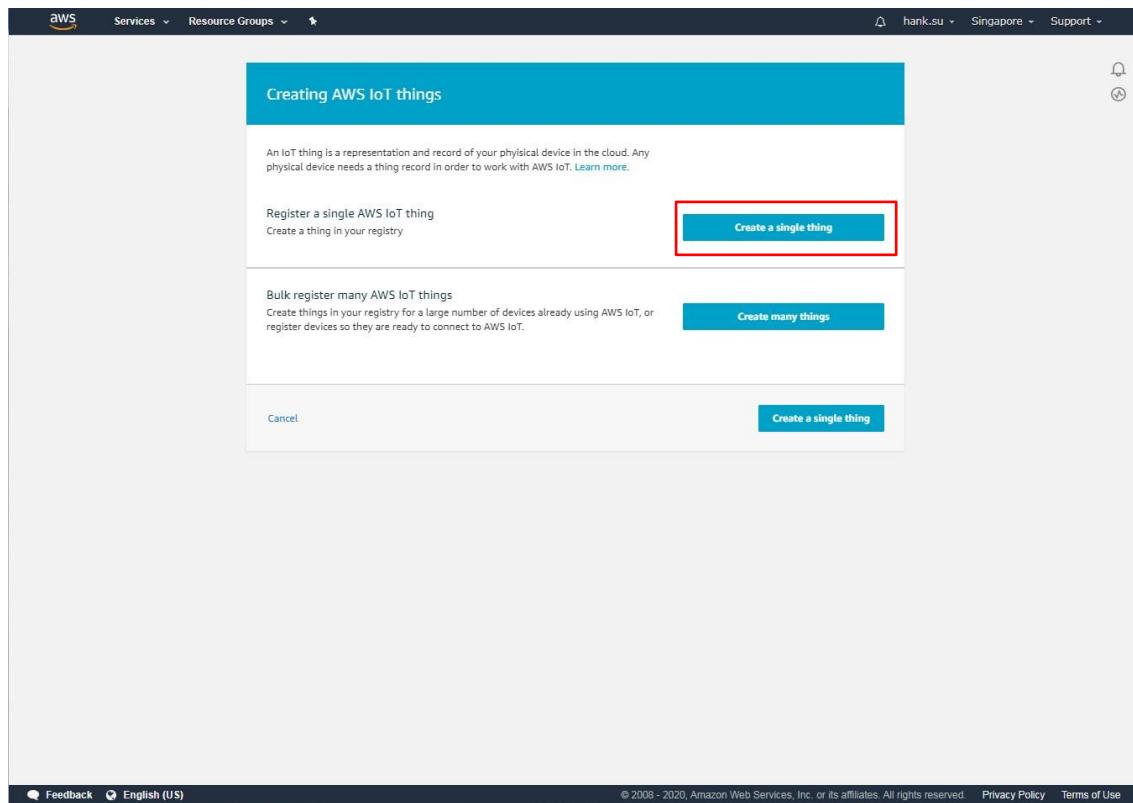


2 Configure AWS IoT Core

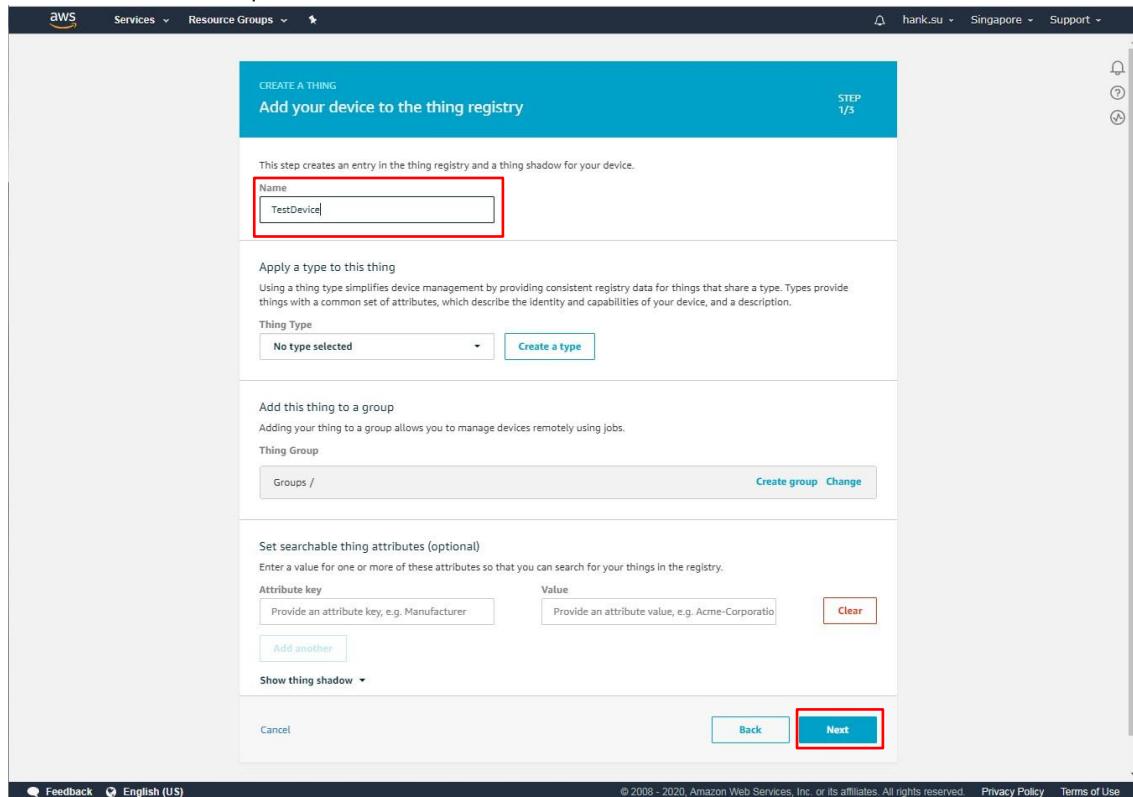
2.1 Create a New Device

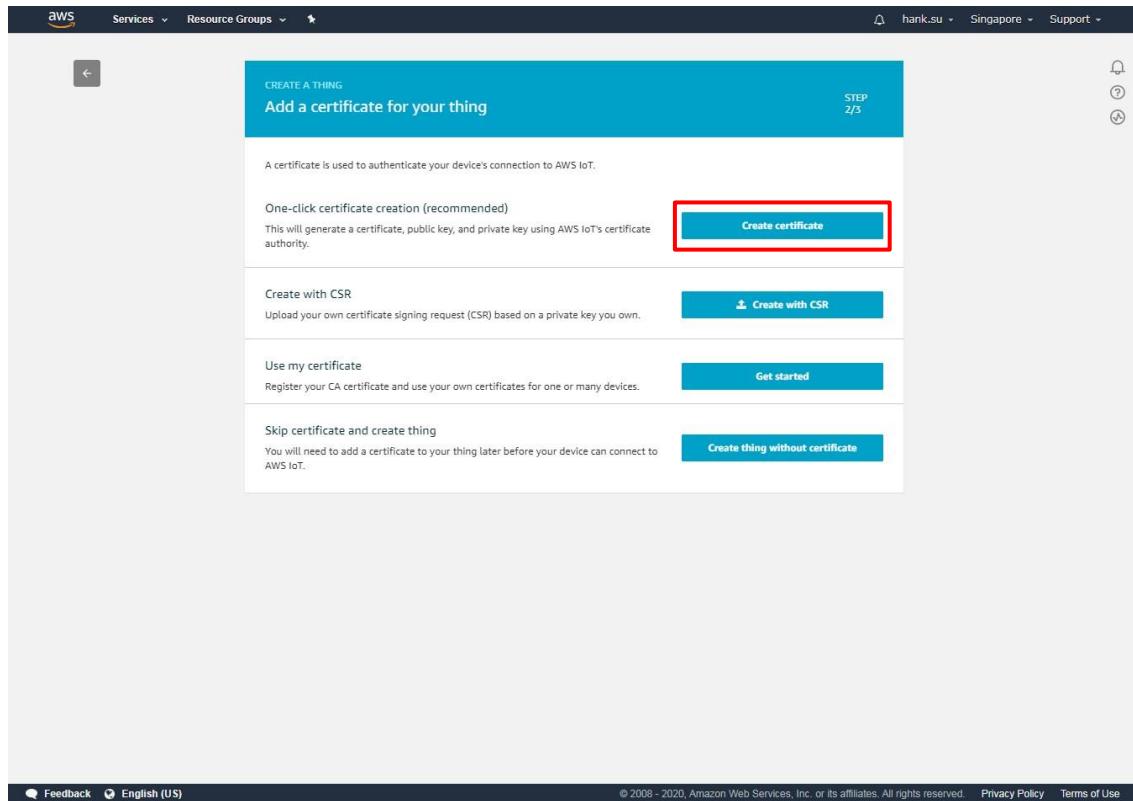
To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click “Register a thing”.



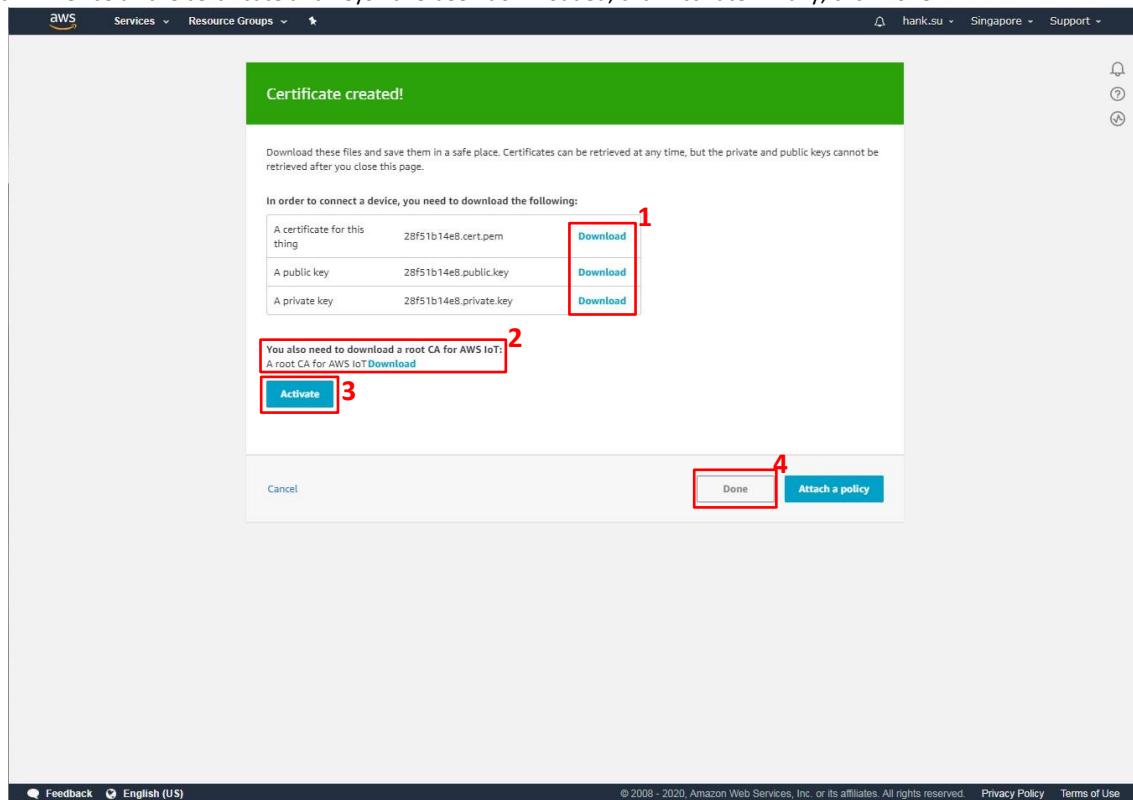


Then, name the new device. This example uses the name TestDevice.





Download the certificate, public key, and private key for the device by clicking Download. Next, download the root CA for AWS IoT by clicking to the Download link. Once all the certificate and keys have been downloaded, click Activate. Finally, click Done.



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:

VeriSign Endpoints (legacy)

- RSA 2048 bit key: [VeriSign Class 3 Public Primary G5 root CA certificate](#)

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.

On this page

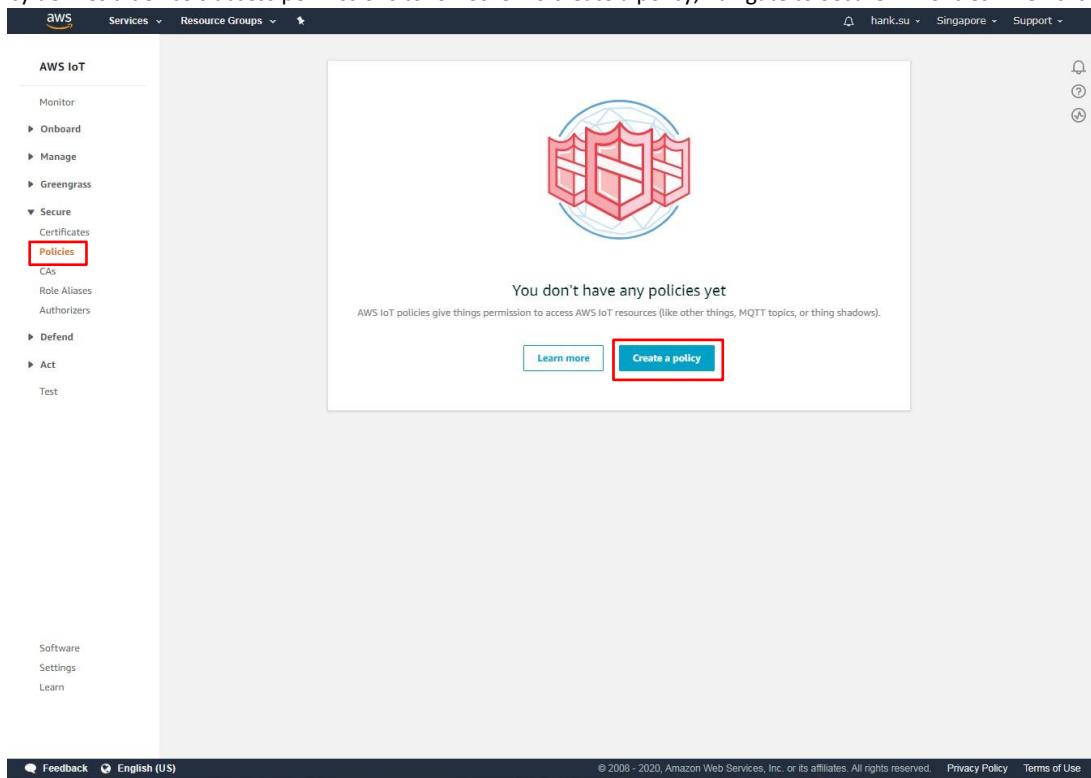
Endpoint types

CA certificates for server authentication

Server authentication guidelines

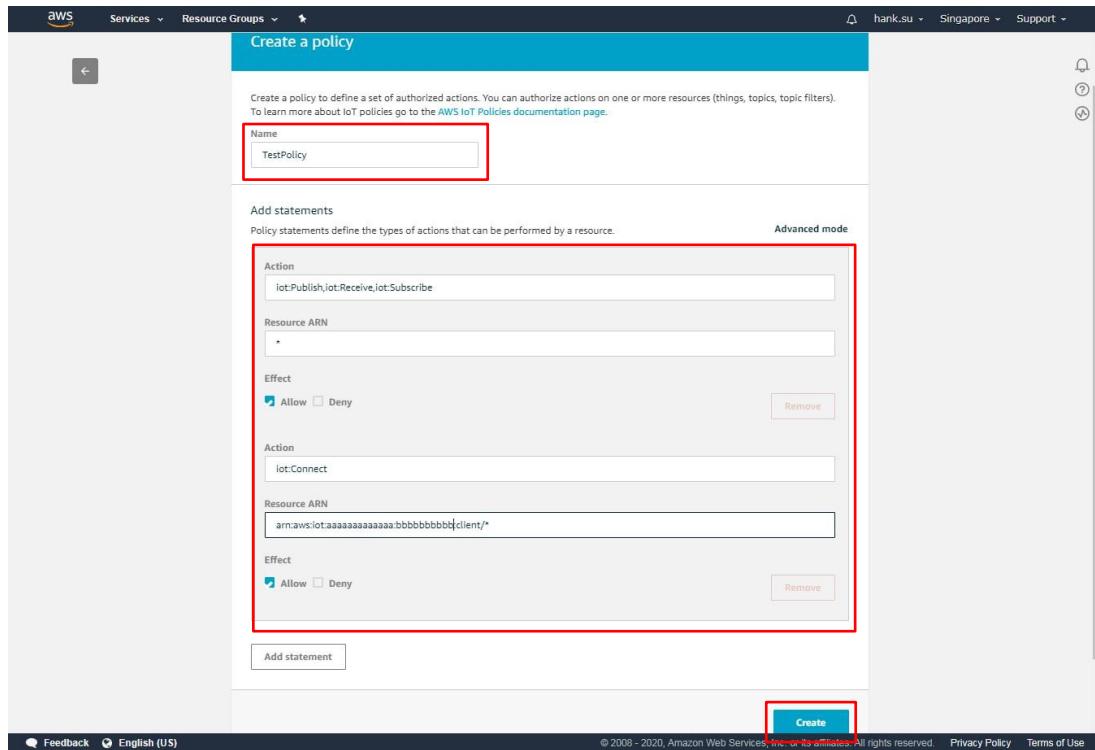
2.2 Create a policy

A policy defines a device's access permissions to IoT Core. To create a policy, navigate to Secure -> Policies. Then click "Create a policy"



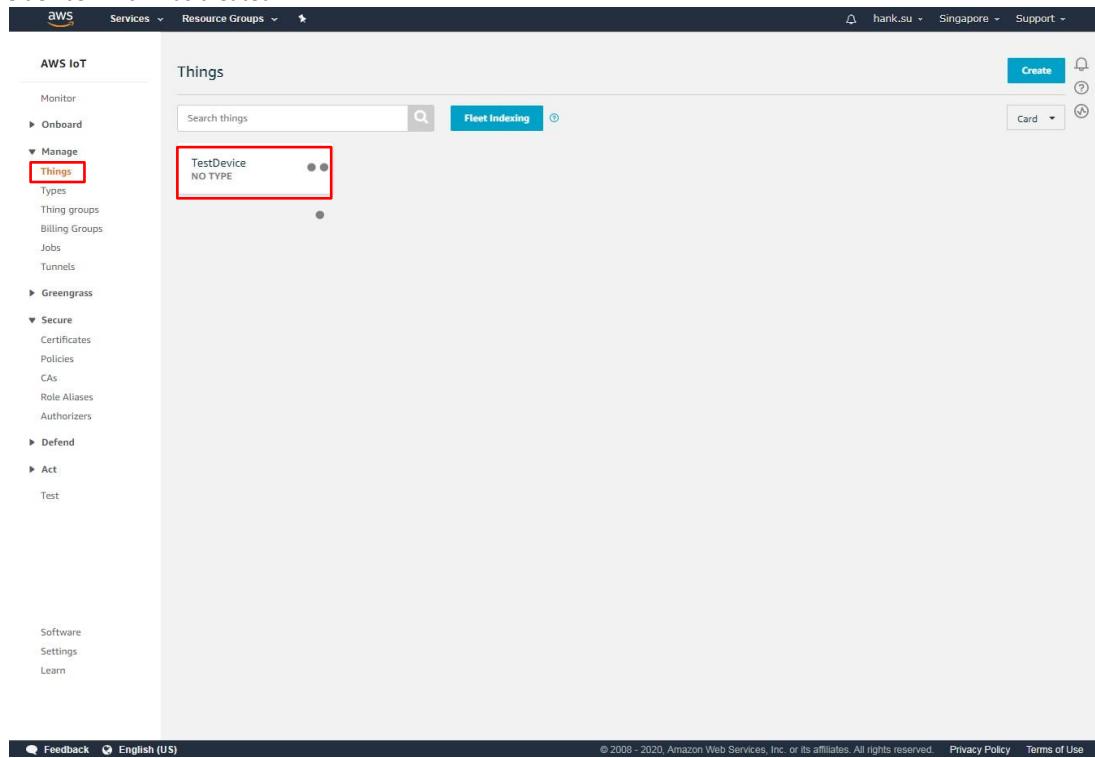
NOTE – this policy grants unrestricted access for all iot operations, and is to be used only in a development environment. For non-dev environments, all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions such as publishing messages or subscribing to topics with specific scope and context. The specific permission policies can vary for your use cases. Identify the permission policies that best meet your business and security requirements.

For sample policies, refer to <https://docs.aws.amazon.com/iot/latest/developerguide/example-iot-policies.html>. Also refer to <https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html>

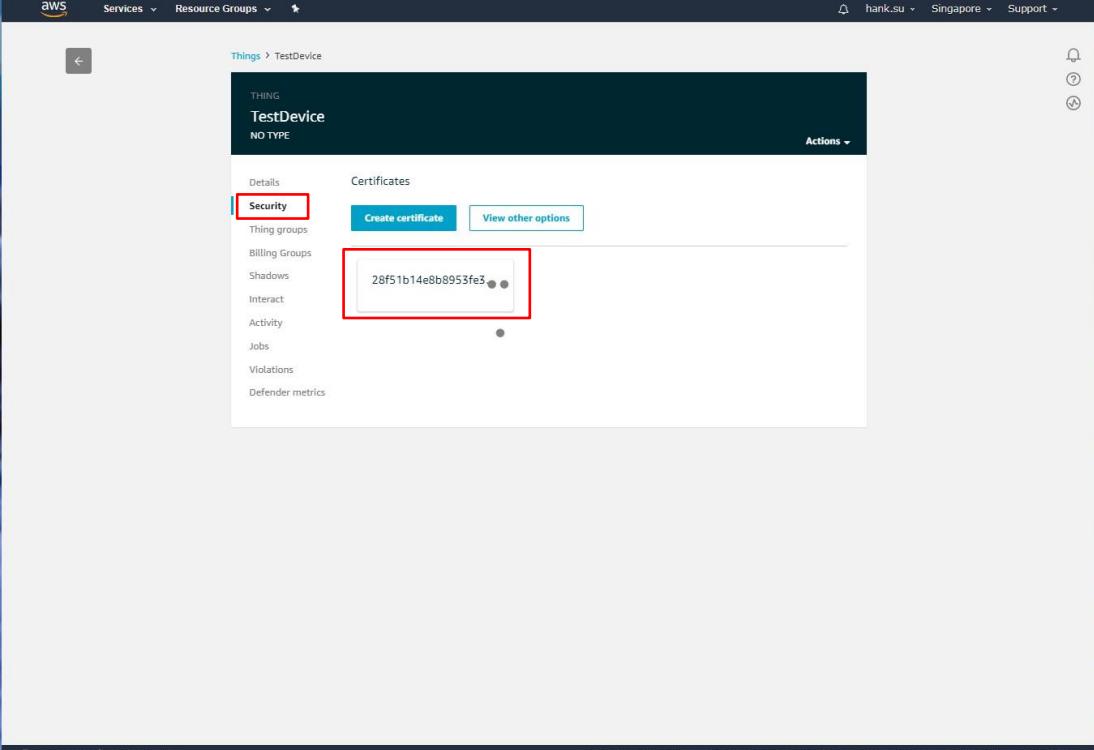


2.3 Attach Policy

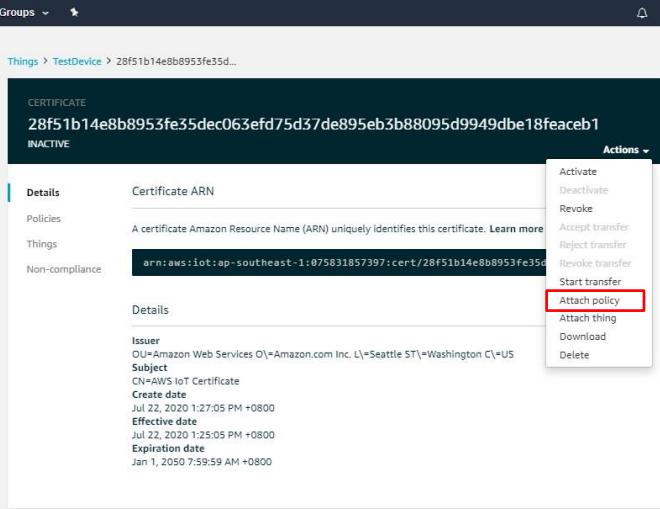
The last step to configuring the device is attaching a policy to new device, navigate to Manage -> Things. Then click on the device which was created.



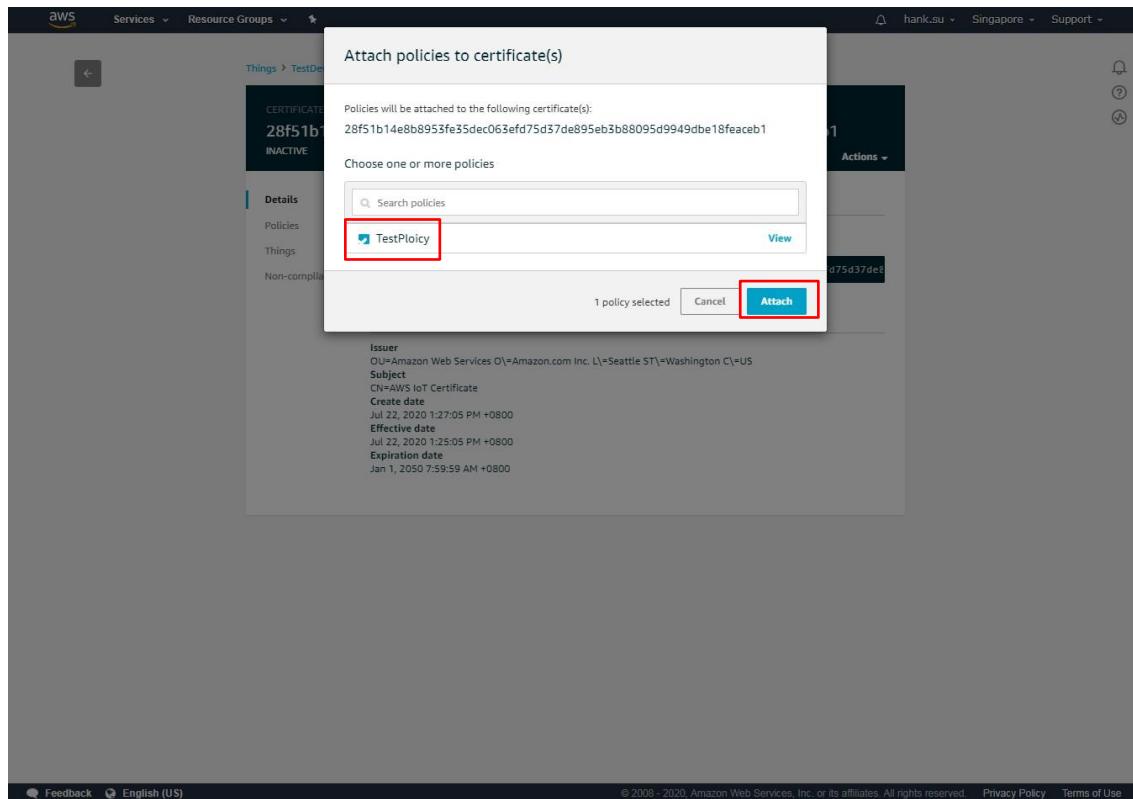
Click Security, then click the certificate create in previous step.



The screenshot shows the AWS IoT Things console. In the top navigation bar, the user is in the 'Services' section under 'Resource Groups'. The main content area displays a 'THING' named 'TestDevice' with 'NO TYPE'. On the left sidebar, there are tabs for 'Details', 'Security' (which is currently selected), 'Thing groups', 'Billing Groups', 'Shadows', 'Interact', 'Activity', 'Jobs', 'Violations', and 'Defender metrics'. In the 'Certificates' section, there are two buttons: 'Create certificate' and 'View other options'. Below these buttons, a certificate ARN is listed: '28f51b14e8b8953fe3...'. This entire row is highlighted with a red box. At the bottom right of the page, there are links for 'Feedback', 'English (US)', and 'Privacy Policy'.



This screenshot shows the AWS IoT Things console with the URL 'Things > TestDevice > 28f51b14e8b8953fe3...' in the address bar. The main content area is titled 'CERTIFICATE' and shows a certificate ARN: '28f51b14e8b8953fe35dec063efd75d37de895eb3b88095d9949dbe18feaceb1'. The status is 'INACTIVE'. On the left, there are tabs for 'Details', 'Policies', 'Things', and 'Non-compliance'. The 'Details' tab is selected. In the 'Details' section, there is a table with columns for 'Issuer', 'Subject', 'Create date', 'Effective date', and 'Expiration date'. The 'Actions' button at the top right has a dropdown menu with options: 'Activate', 'Deactivate', 'Revoke', 'Accept transfer', 'Reject transfer', 'Revoke transfer', 'Start transfer', 'Attach policy' (which is highlighted with a red box), 'Attach thing', 'Download', and 'Delete'. At the bottom right of the page, there are links for 'Feedback', 'English (US)', and 'Privacy Policy'.



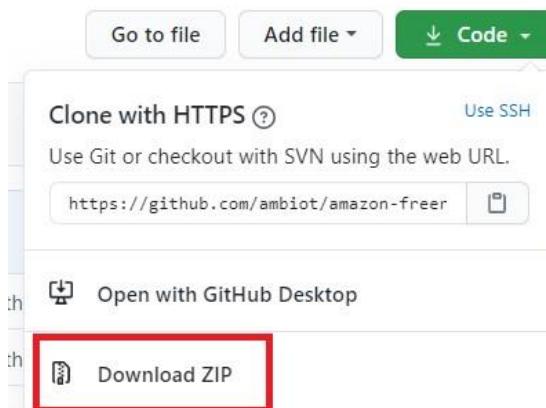
3 Configure AmebaZ2 Amazon FreeRTOS

3.1 Download Source Code from github

Open source link: <https://github.com/Ameba-AIoT/ameba-amazon-freertos> and select amebaZ2-7.1d-202107.00-LTS for get newest source code. The stable version could be found in "Releases" page.

3.1.1 Cloning a repository by Download ZIP

1. On GitHub, navigate to the main page of the repository.
2. Above the list of files, click **Code**.
3. Click **Download ZIP** to get source code.



For more information, please refer "[Cloning a repository from GitHub to GitHub Desktop](#)."

3.2 Get Broker Endpoint by AWS IoT Core

The screenshot shows the AWS IoT Core Settings page. On the left sidebar, under the Software section, the 'Settings' option is highlighted with a red box. The main content area displays the 'Custom endpoint' configuration. A red box highlights the 'Endpoint' input field, which contains '.amazonaws.com'. To the right of the input field, the text 'Broker Endpoint' is written in red. The 'Custom endpoint' section has a green 'ENABLED' status indicator.

3.3 Get Thing Name

The screenshot shows the AWS IoT Core Things page. On the left sidebar, under the Manage section, the 'Things' option is highlighted with a red box. The main content area displays a list of things. One item, 'TestDevice', is highlighted with a red box and labeled 'Thing Name' in red text. The 'TestDevice' entry also includes the text 'NO TYPE'.

3.4 Setup IoT Core Information with AmebaZ2 Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in
 "amazonfreertos/blob/master/demos/include/aws_clientcredential.h"

```

/*
 * @brief Host name.
 *
 * @todo Set this to the unique name of your IoT Thing.
 */
#define clientcredentialMQTT_BROKER_ENDPOINT      "xxxxxxxxxxxxxx.amazonaws.com"

/*
 * @brief Port number the MQTT broker is using.
 */
#define clientcredentialMQTT_BROKER_PORT          8883

/*
 * @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
 */
#define clientcredentialGREENGRASS_DISCOVERY_PORT  8443

/*
 * @brief Wi-Fi network to join.
 *
 * @todo If you are using Wi-Fi, set this to your network name.
 */
#define clientcredentialWIFI_SSID                  "TestAP"

/*
 * @brief Password needed to join Wi-Fi network.
 * @todo If you are using WPA, set this to your network password.
 */
#define clientcredentialWIFI_PASSWORD              "password"

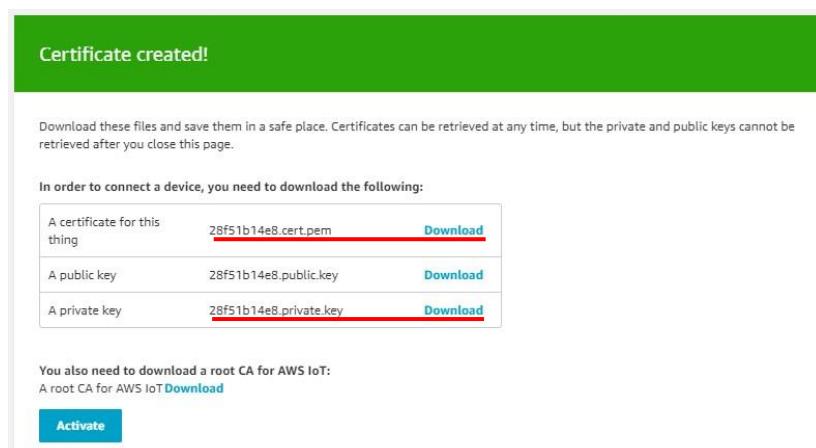
/*
 * @brief Wi-Fi network security type.
 *
 * @see WiFiSecurity_t.
 *
 * @note Possible values are eWiFiSecurityOpen, eWiFiSecurityWEP, eWiFiSecurityWPA,
 * eWiFiSecurityWPA2 (depending on the support of your device Wi-Fi radio).
 */
#define clientcredentialWIFI_SECURITY              eWiFiSecurityWPA2

#endif /* ifndef __AWS_CLIENTCREDENTIAL_H__ */

```

3.4.1 Setup Thing's Private Key and Certificate

Filled keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in
 "amazonfreertos/blob/master/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-certificate.pem and xxxxxxxx-private.pem.key.



It can done by amazon-freertos/tools/certificate_configuration/CertificateConfigurator.html

Certificate Configuration Tool

FreeRTOS Developer Demos

Provide client certificate and private key PEM files downloaded from the AWS IoT Console.

Certificate PEM file:

選擇檔案 未選擇任何檔案

Private Key PEM file:

選擇檔案 未選擇任何檔案

Generate and save aws_clientcredential_keys.h

 Save the generated header file to the `demos/common/include` folder of the demo project.

Copyright (C) 2017 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Final aws_clientcredential_keys.h overview.

3.4.2 Enable FreeRTOS demo on AmebaZ2

Find platform_opts_amazon.h in component\common\application\amazon\amazon-freertos\demos\include and enable **CONFIG_EXAMPLE_AMAZON_FREERTOS**

```
/* For Amazon FreeRTOS SDK example */
#define CONFIG_EXAMPLE_AMAZON_FREERTOS 1
```

Fine aws_demo_config.h in component\common\application\amazon\amazon-freertos\vendors\realtek\boards\amebaZ2\aws_demos\config_files and add **CONFIG_MQTT_DEMO_ENABLED**

```
/* To run a particular demo you need to define one of these.
 * Only one demo can be configured at a time
 *
 *     CONFIG_MQTT_DEMO_ENABLED
 *     CONFIG_SHADOW_DEMO_ENABLED
 *     CONFIG_OTA_UPDATE_DEMO_ENABLED
 *
 * These defines are used in iot_demo_runner.h for demo selection */

#define CONFIG_MQTT_DEMO_ENABLED
```

Now you can start to compile AmebaZ2 Amazon FreeRTOS

4 Compile AmebaZ2 Amazon FreeRTOS

4.1 Pre-Requisite

- Required AmebaZ2 source code(<https://github.com/Ameba-AIoT/ameba-rtos-z2>)
- Required source code. (<https://github.com/Ameba-AIoT/ameba-amazon-freertos>), please select the branch name with AmebaZ2 which should be download and paste under ameba-rtos-z2\ component\common\application\amazon
- AmebaZ2 Demo board
- Realtek Image Tool
- IAR Embedded Workbench ver.8.30.1 OR Cygwin 32bit for windows

4.2 IAR Build Environment Setup

The IAR IDE (integrated development environment) only supports Windows OS, this section is applicable for **Windows OS only**.

4.2.1 Install IAR IDE

IAR IDE provides the toolchain for Ameba-ZII. It allows users to write programs, compile and upload them to your board. Also, it supports stepby-step debug function.

User can visit the official website of **IAR Embedded Workbench** and install the IDE by following its instructions.

Note: Please use IAR version **8.30** or above.

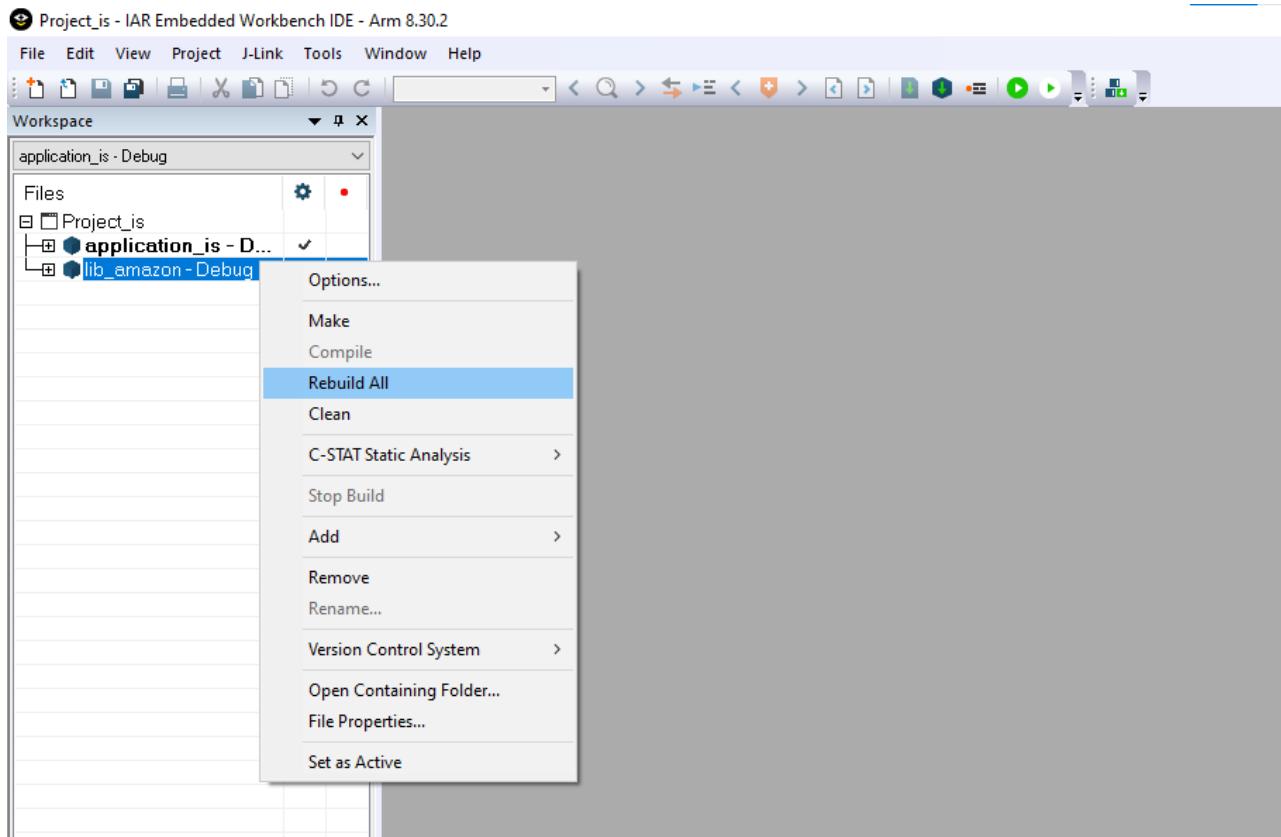
4.2.2 Compilation

- 1) Run setup_amazon.sh.

This script will rename the Project_is_amazon.eww to Project_is.eww, and rename application_is_amazon.ewp to application_is.ewp.

And download the freertos10.4.0 to ameba-rtos-z2/component/os/freertos, and download ameba-amazon-freertos to ameba-rtos-z2/component/common/application/amazon/amazon-freertos

- 2) Open /project/realtek_amebaz2_v0_example/EWARM-RELEASE /Project_is.eww.
- 3) Confirm ‘lib_amazon’ in Work Space, right click ‘lib_amazon’ and choose “**Rebuild All**” to compile.
- 4) Confirm ‘application_is’ in Work Space, right click ‘application_is’ and choose “**Rebuild All**” to compile.
- 5) Make sure there is no error after compile.



4.3 GCC Environment

4.3.1 Install Cygwin on Windows (Using Cygwin)

Cygwin is a large collection of GNU and Open Source tools which provide similar functionality as a Linux distribution on Windows. It provides the GCC toolchain for Ameba-ZII to compile projects.

Users can visit the official website of Cygwin and install the software. Please use **Cygwin 32-bit** version.

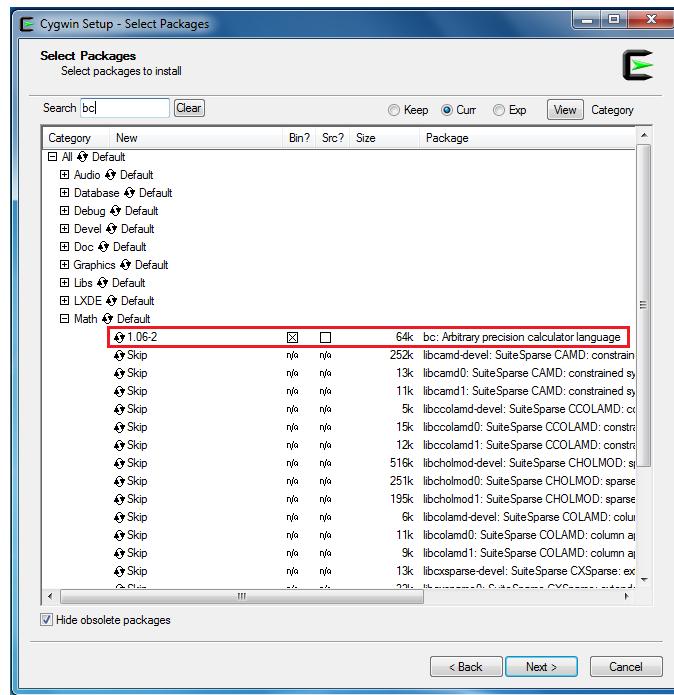
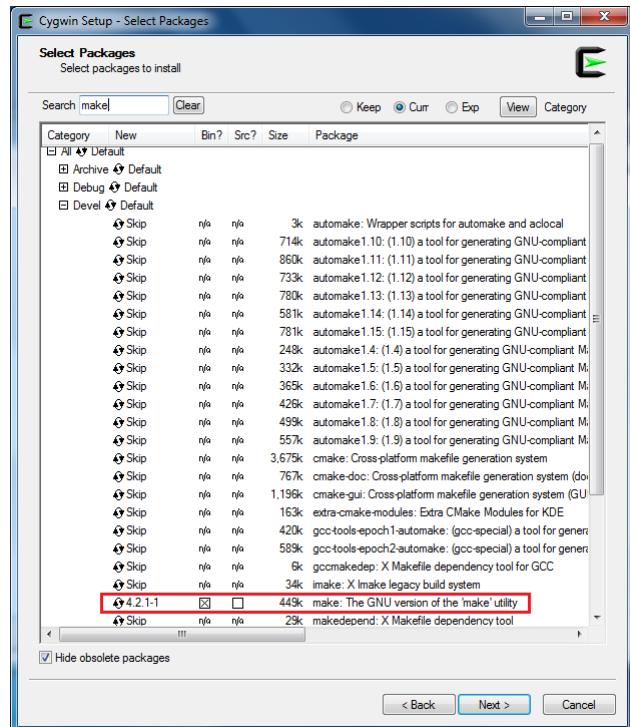
Click <http://cygwin.com> and download the Cygwin package setup-x86.exe for your Windows platform.

1. 32-bit Cygwin is supported both for 32-bit Windows and 64-bit Windows.

Note: If you can not installed 32-bit Cygwin, refer to Q: How can I install the last Cygwin version for an old, unsupported Windows?. Installation at CMD:

```
setup-2.912.x86.exe --allow-unsupported-windows --site  
http://ctm.crouchingtigerhiddenfruitbat.org/pub/cygwin/circa/2022/11/23/063457
```

2. During the installation of Cygwin package, include Devel -> make and Math -> bc utilities on the Select Packages page, as below shows.

**Note:**

- During the Cygwin installation, please install “math” “**bc: Arbitrary precision calculator language**”
- During the Cygwin installation, please install “devel” “**make: The GNU version of the ‘make’ utility**”

4.3.2 Compile Project on Cygwin or Ubuntu/Linux

Open “Cygwin Terminal” or in Ubuntu/Linux

- 1) Direct to compile path. Enter command “**cd /SDK/project/realtek_amebaz2_v0_example/GCC-RELEASE**”
- 2) Clean up previous compilation files. Enter command “**make clean**”
- 3) Build the amazon library. Enter command “**make amazon**”
- 4) Build the application. Enter command “**make is**”
- 5) Make sure there are no errors after compilation.

4.4 Generate Image Binary

After compile, the images **partition.bin**, **bootloader.bin**, **firmware_is.bin** and **flash_is.bin** can be seen in the `amazonfreertos/project/realtek_amebaz2_v0_example/EWARM-RELEASE/Debug/Exe`.

- 1) **partition.bin** stores partition table, recording the address of Boot image and firmware image;
- 2) **bootloader.bin** is bootloader image;
- 3) **firmware_is.bin** is application image;
- 4) **flash_is.bin** links `partition.bin`, `bootloader.bin` and `firmware_is.bin`. Users need to choose `flash_is.bin` when downloading the image to board by Image Tool

5 ImageTool

The tool can be find in ameba-rtos-z2/tools/AmebaZ2/Image_Tool/AmebaZ2_PGTool_v1.2.46

5.1 Introduction

This chapter introduces how to use Image Tool to generate and download images. As show in picture below, Image Tool has two menu pages:

- Download: used as image download server to transmit images to Ameba through UART.

Note: If you need to download code via external uart, must use **FT232** USB to connect UART dongle.

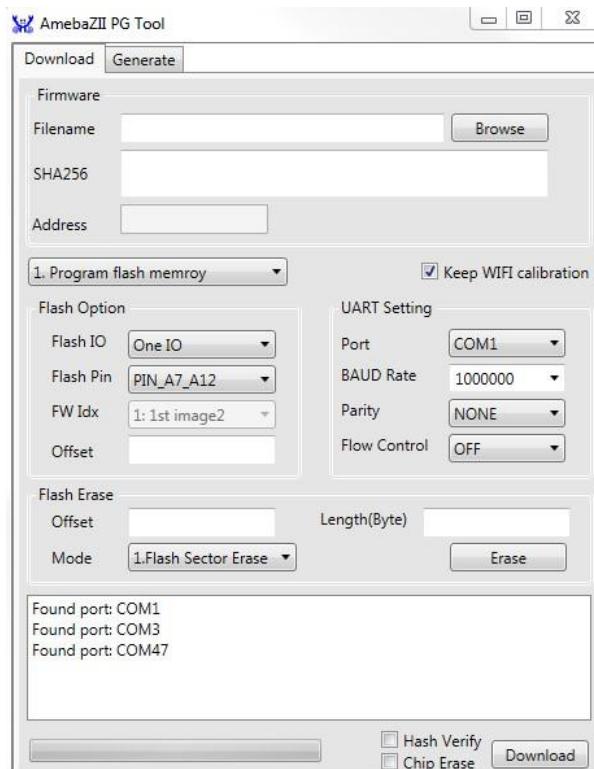


Fig 5-1 AmebaZ2 ImageTool UI

5.2 Environment Setup

5.2.1 Hardware Setup

User needs to connect CON3 to user's PC via a Micro USB cable. Add jumpers for J34 and J33 (J33 is for log UART which has two jumpers) if there is no connection.

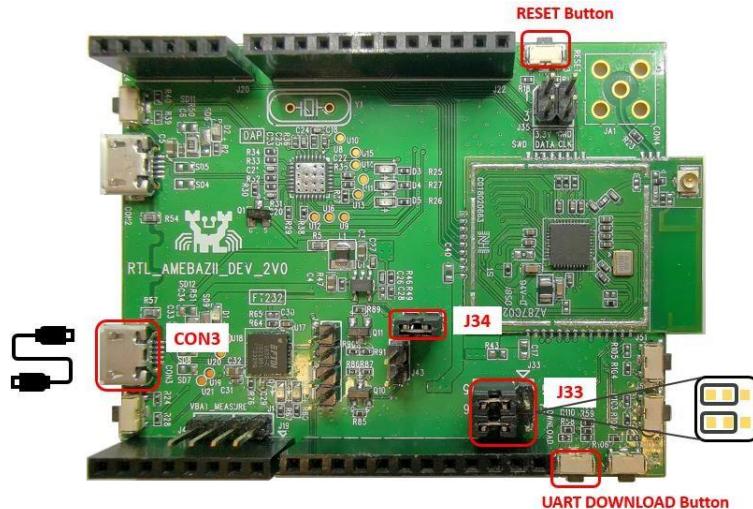


Fig 5-2 Ameba-ZII EVB V2.0 Hardware Setup

5.2.2 Software Setup

- Environment Requirements: EX. WinXP, Win 7 Above, Microsoft .NET Framework 3.5
- AmebaZII_PGTool_v1.2.47.exe

5.3 Image Download

User can download the image to demo board by following steps: 1)

Trigger Ameba-ZII chip enter **UART download mode** by:

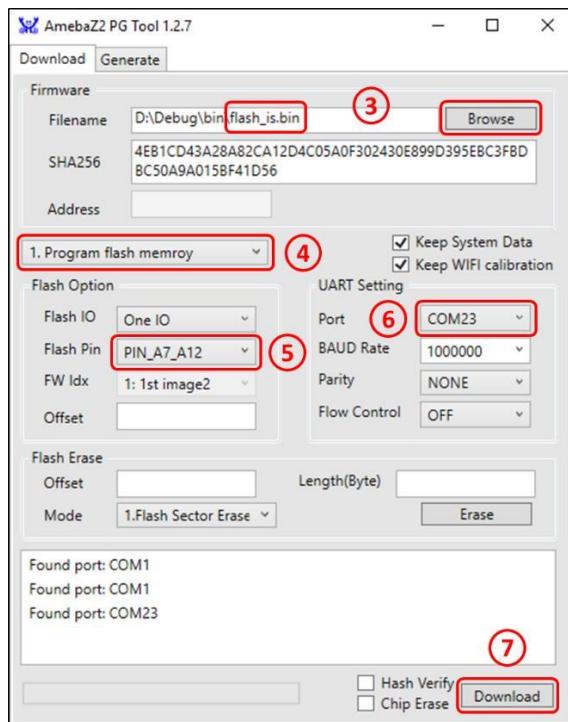
- a. Press and hold the **UART DOWNLOAD** button then press the **RESET** button and release both buttons. And make sure the log UART is connected properly.
- b. If the chip enters **download mode**, the below log should be shown on log UART console.

```
== Rt18710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x0
Download Image over UART2[tx=16,rx=15] baud=115200
```

Fig 5-3 Ameba-ZII UART download mode

- c. After confirming it is in download mode, **remember to disconnect the log UART console before using Image Tool to download**, because the tool will also need to connect to this log UART port.

2) Open AmebaZ2 PG Tool



- 3) "Browse" to choose the image to be downloaded (amazon-freertos/projects/realtek/amebaZ2/IAR/aws_demos/Debug/Exe /flash_is.bin)
 - 4) Choose "1. Program flash memory"
 - 5) Choose correct "Flash Pin" according to the IC part number
- | Flash Pin | IC part number |
|------------|---------------------|
| PIN_A7_A12 | RTL8710CX/RTL8720CM |
| PIN_B6_B12 | RTL8720CF |
- 6) Choose the correct **UART port** (use **rescan** to update the port list)
 - 7) Click "**Download**" to start downloading image. While downloading, the status will be shown on the left bar. **Note:** It's recommended to use the default settings unless user is familiar with them.

6 MQTT Demo

6.1 Get Device Log

Install Tera Term to get device log

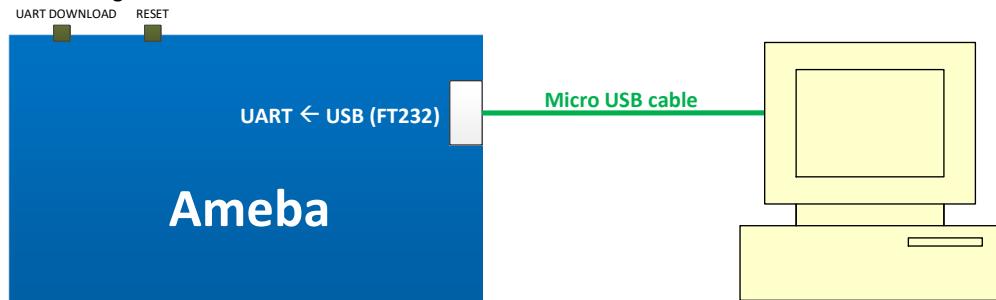
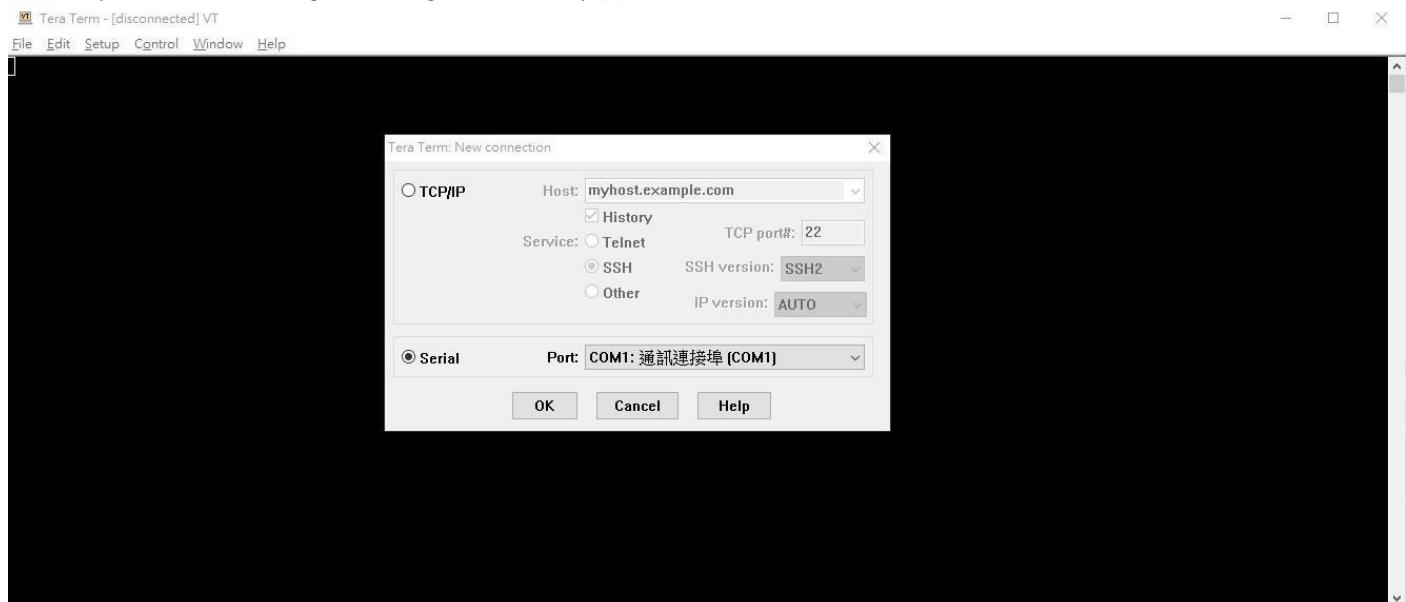


Fig 6-1 Hardware setup

The serial port is same with ImageTool that get from 5.4 step (6).



6.2 Run MQTT Demo

Default setting of SDK are enable MQTT demo. Once the AmebaZ2 EVB has rebooted, the application will automatically start run MQTT demo and communicate to IoT Core.

```

COM6 - Tera Term VT
File Edit Setup Control Window Help
#calibration_ok:[2:19:11]
#interface 0 is initialized
interface 1 is initialized

Initializing WIFI ...
WIFI is not running
WIFI initialized

init_thread(58), Available heap 0x24ac0
0 56 [example_a] Wi-Fi module initialized. Connecting to AP...
WIFI is already running
Joining BSS by SSID RealEZ-2.4G...

RTL8721D[Driver]: set ssid [RealEZ-2.4G]

RTL8721D[Driver]: rtw_set_wpa_ie[1136]: AuthKeyMgmt = 0x2
RTL8721D[Driver]: rtw_restruct_sec_ie[3763]: no pmksa cached
RTL8721D[Driver]: start auth to 80:2a:a8:d4:93:c4
RTL8721D[Driver]: auth alg = 2
RTL8721D[Driver]:
OnAuthClient:algthm = 0, seq = 2, status = 0, sae_msg_len = 0
RTL8721D[Driver]: auth success, start assoc
RTL8721D[Driver]: association success(res=4)
wlan1: 1 DL RSVD page success! DLBcnCount:01, poll:00000001
RTL8721D[Driver]: ClientSendEAPOL[1522]: no use cache pmksa
RTL8721D[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)
RTL8721D[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:2
1 8000 [example_a] Wi-Fi Connected to AP. Creating tasks which use network...
2 8007 [example_a] IP Address acquired 192.168.89.151
3 8019 [example_a] Write certificate...
4 8080 [iot_threa] [INFO ][DEMO][8079] -----STARTING DEMO-----
5 8086 [iot_threa] [INFO ][INIT][8086] SDK successfully initialized.

...
6 15504 [iot_threa] [INFO ][DEMO][15504] Successfully initialized the demo. Network type for the demo: 1
7 15513 [iot_threa] [INFO ][MQTT][15513] MQTT library successfully initialized.
8 15522 [iot_threa] [INFO ][DEMO][15522] MQTT demo client identifier is ameba-ota (length 9).
9 17272 [iot_threa] [INFO ][MQTT][17272] Establishing new MQTT connection.
Interface 0 IP address : 192.168.89.15110 17283 [iot_threa] [INFO ][MQTT][17283] Anonymous metrics (SDK language, SDK version) will be provided to AWS IoT. Recompile with AWS_IOT_MQTT_ENABLE_METRICS to 0 to disable.
11 17302 [iot_threa] [INFO ][MQTT][17302] (MQTT connection 100337e0, CONNECT operation 100339a0) Waiting for operation completion.
12 17421 [iot_threa] [INFO ][MQTT][17421] (MQTT connection 100337e0, CONNECT operation 100339a0) Wait complete with result SUCCESS.
13 17433 [iot_threa] [INFO ][MQTT][17433] New MQTT connection 100381d0 established.
14 17443 [iot_threa] [INFO ][MQTT][17443] (MQTT connection 100337e0) SUBSCRIBE operation scheduled.
15 17452 [iot_threa] [INFO ][MQTT][17452] (MQTT connection 100337e0, SUBSCRIBE operation 100339e0) Waiting for operation completion.
16 17612 [iot_threa] [INFO ][MQTT][17612] (MQTT connection 100337e0, SUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
17 17624 [iot_threa] [INFO ][DEMO][17624] All demo topic filter subscriber subscriptions accepted.
18 17632 [iot_threa] [INFO ][DEMO][17632] Publishing messages 0 to 1.
19 17640 [iot_threa] [INFO ][MQTT][17640] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
20 17650 [iot_threa] [INFO ][MQTT][17650] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
21 17659 [iot_threa] [INFO ][DEMO][17659] Waiting for 2 publishes to be received.
22 17752 [iot_threa] [INFO ][DEMO][17752] MQTT PUBLISH 0 successfully sent.
23 17784 [iot_threa] [INFO ][DEMO][17784] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 0!
24 17804 [iot_threa] [INFO ][MQTT][17804] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
25 17814 [iot_threa] [INFO ][DEMO][17814] Acknowledgment message for PUBLISH 0 will be sent.
26 17825 [iot_threa] [INFO ][DEMO][17825] MQTT PUBLISH 1 successfully sent.
27 17841 [iot_threa] [INFO ][DEMO][17840] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 1!
28 17861 [iot_threa] [INFO ][MQTT][17861] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
29 17870 [iot_threa] [INFO ][DEMO][17870] Acknowledgment message for PUBLISH 1 will be sent.
30 17883 [iot_threa] [INFO ][DEMO][17883] 2 publishes received.
31 17889 [iot_threa] [INFO ][DEMO][17889] Publishing messages 2 to 3.
32 17897 [iot_threa] [INFO ][MQTT][17897] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
33 17907 [iot_threa] [INFO ][MQTT][17907] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
34 17916 [iot_threa] [INFO ][DEMO][17916] Waiting for 2 publishes to be received.
35 18021 [iot_threa] [INFO ][DEMO][18021] MQTT PUBLISH 3 successfully sent.
36 18030 [iot_threa] [INFO ][DEMO][18029] MQTT PUBLISH 2 successfully sent.
37 18039 [iot_threa] [INFO ][DEMO][18038] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
Publish topic name: iotdemo/topic/4

```

```

...
Publish payload: Hello world 16!
132 19827 [iot_threa] [INFO ][MQTT][19827] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
133 19837 [iot_threa] [INFO ][DEMO][19837] Acknowledgment message for PUBLISH 16 will be sent.
134 19851 [iot_threa] [INFO ][DEMO][19851] 2 publishes received.
135 19857 [iot_threa] [INFO ][DEMO][19857] Publishing messages 18 to 19.
136 19865 [iot_threa] [INFO ][MQTT][19865] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
137 19876 [iot_threa] [INFO ][MQTT][19876] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
138 19885 [iot_threa] [INFO ][DEMO][19885] Waiting for 2 publishes to be received.
139 19953 [iot_threa] [INFO ][DEMO][19953] MQTT PUBLISH 18 successfully sent.
140 19980 [iot_threa] [INFO ][DEMO][19980] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/3
Publish topic name: iotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 18!
141 20001 [iot_threa] [INFO ][MQTT][20001] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
142 20011 [iot_threa] [INFO ][DEMO][20011] Acknowledgment message for PUBLISH 18 will be sent.
143 20053 [iot_threa] [INFO ][DEMO][20053] MQTT PUBLISH 19 successfully sent.
144 20069 [iot_threa] [INFO ][DEMO][20068] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
Publish topic name: iotdemo/topic/4
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 19!
145 20089 [iot_threa] [INFO ][MQTT][20089] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
146 20099 [iot_threa] [INFO ][DEMO][20099] Acknowledgment message for PUBLISH 19 will be sent.
147 20108 [iot_threa] [INFO ][DEMO][20108] 2 publishes received.
148 20116 [iot_threa] [INFO ][MQTT][20116] (MQTT connection 100337e0) UNSUBSCRIBE operation scheduled.
149 20129 [iot_threa] [INFO ][MQTT][20128] (MQTT connection 100337e0, UNSUBSCRIBE operation 100339e0) Waiting for operation completion.
150 20322 [iot_threa] [INFO ][MQTT][20321] (MQTT connection 100337e0, UNSUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
151 20335 [iot_threa] [INFO ][MQTT][20335] (MQTT connection 100337e0) Disconnecting connection.
152 20347 [iot_threa] [INFO ][MQTT][20347] (MQTT connection 100337e0, DISCONNECT operation 100339e0) Waiting for operation completion.
153 20359 [iot_threa] [INFO ][MQTT][20359] (MQTT connection 100337e0, DISCONNECT operation 100339e0) Wait complete with result SUCCESS.
154 20371 [iot_threa] [INFO ][MQTT][20371] (MQTT connection 100337e0) Connection disconnected.
155 20380 [iot_threa] [INFO ][MQTT][20380] (MQTT connection 100337e0) Network connection closed.
156 21622 [iot_threa] [INFO ][MQTT][21622] (MQTT connection 100337e0) Network connection destroyed.
157 21631 [iot_threa] [INFO ][MQTT][21631] MQTT library cleanup done.
158 21637 [iot_threa] [INFO ][DEMO][21637] Demo completed successfully.

LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
159 21772 [iot_threa] [INFO ][INIT][21772] SDK cleanup done.
160 21777 [iot_threa] [INFO ][DEMO][21777] -----DEMO FINISHED-----

```

6.3 Monitoring MQTT messages on the cloud

To subscribe to the MQTT topic with the AWS IoT MQTT client 1.

Sign in to the AWS IoT console.

2. In the navigation pane, choose Test to open the MQTT client.
3. In Subscription topic, enter “+/example/topic”, and then choose Subscribe to topic.

The screenshot shows the AWS IoT MQTT test client interface. On the left, there's a sidebar with navigation links for Connect, Test, Manage, Device software, and AWS IoT. Under Test, the 'MQTT test client' is selected. The main area has two tabs: 'Subscribe to a topic' (selected) and 'Publish to a topic'. In the 'Subscribe to a topic' tab, there's a 'Topic filter' input field containing '+/example/topic', an 'Additional configuration' section with a 'Number of messages to keep' input set to 100, and a 'Quality of service' section with radio buttons for QoS 0 (selected) and QoS 1. Below these are sections for 'MQTT payload display' (with options for JSON, strings, or raw binary) and a 'Subscribe' button. The 'Publish to a topic' tab shows a list of subscriptions: 'iotdemo/#' and '+/example/topic'. The '+/example/topic' entry has a message box with an info icon stating 'You cannot publish messages to a wildcard topic. Please select a different topic to publish messages to.' Below this, there are three message entries from 'wanglu_test/example/topic': one from July 31, 2024, at 12:01:11 (UTC+0800) with the message 'Hello world!', another from July 31, 2024, at 12:01:08 (UTC+0800) with the message 'Hello world!', and a third from July 31, 2024, at 12:01:05 (UTC+0800) with the message 'Hello world!'. Each message entry has a 'Properties' link.

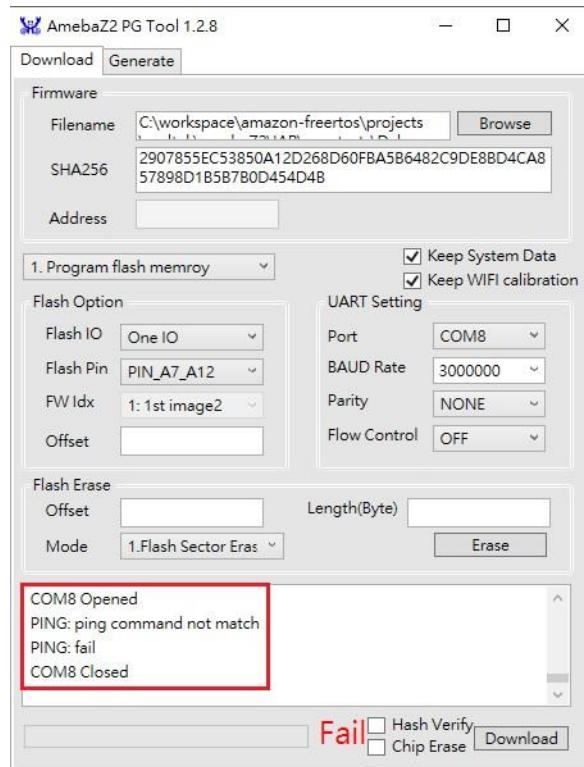
7 Troubleshooting

If these steps don't work, look at the device log in the serial terminal. You should see some text that indicates the source of the problem.

For general troubleshooting information about Getting Started with FreeRTOS, see [Troubleshooting getting started](#).

7.1 Image Tool Download Fail

Please check device in UART_DOWNLOAD mode or not. Refer 5.3 for more detail.



7.2 ERROR: Invalid Key

Please check **WIFI_SSID** and **WIFI_PASSWORD** in ameba-rtos-z2\component\common\application\amazon\amazon-freertos\demos\include\aws_clientcredential.h

```

Enter SSID for Soft AP started
3 1098 [example_a] Wi-Fi configuration successful.
4 1108 [iot_threa] [INFO ][DEMO][1108] -----STARTING DEMO-----
5 1115 [iot_threa] [INFO ][INIT][1115] SDK successfully initialized.

LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
WIFI deinitialized
Initializing WIFI ...
WIFI initialized

Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...

```

7.3 Failed to establish new MQTT connection

Please check **clientcredentialMQTT_BROKER_ENDPOINT** in ameba-rtos-z2\component\common\application\amazon\amazon-freertos\demos\include\aws_clientcredential.h

```

6 12508 [iot_threa] [INFO ][DEMO][12508] Successfully initialized the demo. Network type for the demo: 1
7 12517 [iot_threa] [INFO ][MQTT][12517] MQTT library successfully initialized.
8 12524 [iot_threa] [INFO ][DEMO][12524] MQTT demo client identifier is ameba-ota (length 9).
9 12624 [iot_threa] [ERROR][NET][12624] Failed to resolve [REDACTED].amazonaws.com.
10 12934 [iot_threa] [ERROR][MQTT][12934] Failed to establish new MQTT connection, error NETWORK ERROR.
11 12943 [iot_threa] [ERROR][DEMO][12943] MQTT CONNECT returned error NETWORK ERROR.
12 12951 [iot_threa] [INFO ][MQTT][12950] MQTT library cleanup done.
13 12957 [iot_threa] [ERROR][DEMO][12957] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP DHCP: dhcp stop.
Deinitializing WIFI ...
14 13094 [iot_threa] [INFO ][INIT][13094] SDK cleanup done.
15 13099 [iot_threa] [INFO ][DEMO][13099] -----DEMO FINISHED-----

```

7.4 TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in ameba-rtos-z2\component\common\application\amazon\amazon-freertos\demos\include\aws_clientcredential_keys.h

```

8 13501 [iot_threa] [INFO ][DEMO][13501] Successfully initialized the demo. Network type for the demo: 1
9 13511 [iot_threa] [INFO ][MQTT][13511] MQTT library successfully initialized.
10 13518 [iot_threa] [INFO ][DEMO][13518] MQTT demo client identifier is ameba-ota (length 9).
11 20102 [iot_threa] [ERROR] Private key not found. 12 20107 [iot_threa] TLS Connect fail (0x7d4, [REDACTED].amazonaws.com)
13 20115 [iot_threa] [ERROR][NET][20115] Failed to establish new connection. Socket status: -1.
14 20424 [iot_threa] [ERROR][MQTT][20424] Failed to establish new MQTT connection, error NETWORK ERROR.
15 20433 [iot_threa] [ERROR][DEMO][20433] MQTT CONNECT returned error NETWORK ERROR.
16 20441 [iot_threa] [INFO ][MQTT][20441] MQTT library cleanup done.
17 20447 [iot_threa] [ERROR][DEMO][20447] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
18 20586 [iot_threa] [INFO ][INIT][20586] SDK cleanup done.
19 20591 [iot_threa] [INFO ][DEMO][20591] -----DEMO FINISHED-----

```

8 OTA Demo

8.1 OTA Update Prerequisites

Please refer to the AWS official guide (<https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html>) and finish the following steps:

- Step 1. Prerequisites for OTA updates using MQTT
- Step 2. Create an Amazon S3 bucket to store your update
- Step 3. Create an OTA Update service role
- Step 4. Create an OTA user policy
- Step 5. Create esdsasigner.key and ecdfsasigner.crt by openSSL you can create the key and certification by running:

```
$ sudo openssl ecparam -name prime256v1 -genkey -out ecdfa-sha256-signer.key.pem
$ sudo openssl req -new -x509 -days 3650 -key ecdfa-sha256-signer.key.pem -out ecdfa-sha256-signer.crt.pem
```

...
- Step 6. Add certificate pem(ecdfa-sha256-signer.crt.pem) into component\common\application\amazon\amazon-freertos\vendors\realtek\boards\amebaZ2\aws_demos\config_files\ota_demo_config.h

```
#ifndef __AWS_CODESIGN_KEYS_H__
#define __AWS_CODESIGN_KEYS_H__

/*
 * PEM-encoded code signer certificate
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"
 * "...base64 data...\n"
 * "-----END CERTIFICATE-----";
 */
static const char signingcredentialsIGNING_CERTIFICATE_PEM[] =
"-----BEGIN CERTIFICATE-----\n"
[REDACTED]
"-----END CERTIFICATE-----";

#endif
```

8.2 Set the Firmware Version and App Version to Image File

Set “serial” to **101** from 100 for the serial number in “amebaZ2_firmware_is.json”, this determines the new firmware version of AmebaZ2:

```
"FWHS": {
  "source": "Debug/Exe/application_is.out",
  "header": {
    "next": null,
    "__comment_type": "Support Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP,M0,CPFW",
    "type": "FWHS_S",
    "enc": false,
    "user_key2": "BB0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F",
    "comment_pkey_idx": "assign by program, no need to configurate",
    "serial": 100
  }
}
```

The app number in “aws_application_version.h” decides the version of application code:

```
#ifndef _AWS_APPLICATION_VERSION_H_
#define _AWS_APPLICATION_VERSION_H_

#include "iot_appversion32.h"
extern const AppVersion32_t xAppFirmwareVersion;

#define APP_VERSION_MAJOR....100
#define APP_VERSION_MINOR....9
#define APP_VERSION_BUILD....1

#endif
```

The APP_VERSION_MAJOR in “component\common\application\amazon\amazon-freertos\vendors\realtek\boards\amebaZ2\aws_demos\config_files\ota_demo_config.h” and FW version (serial number) in “amebaz2_firmware_is.json” must be the **same**.

In this example, set APP_VERSION_MAJOR to **101**

Please note that the newer image file must have the bigger version number. So now, you need two image file to perform this demo.

- One image with older version should be downloaded to your AmebaZ2, and wait the OTA job coming.
- Another image with newer version will be uploaded to S3 bucket. Then, create a new job for OTA.

Note: newer version image file should be signed by a private key before uploading. Next section will introduce how to sign the image.

8.3 How Custom Signed Image File is Created

We use custom signing feature provided by amazon to manually sign the OTA binary and attach the signatures along with the **firmware_is.bin**:

1. The firmware_is.bin is manually signed using the ECDSA P-256 key provided by user.
2. The ECDSA signatures are then appended to the end of the firmware_is_sig.bin with signature sizes.
3. The signatures are received as a separate packet and formatted accordingly to verify the OTA image which was updated.

The custom signing process is executed by a python script – **python_custom_ecdsa_Z2.py**, that provided in the folder (component\common\application\amazon\amazon_ota_tools\python_custom_ecdsa_Pro.py)

The python script requires the following pre-requisites to work

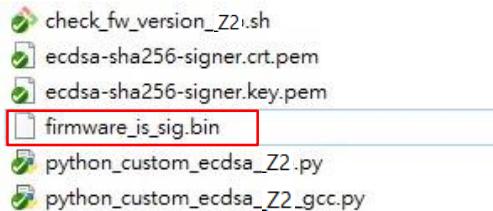
1. Python must be installed in the windows system with version 3.7.x or later
2. Pyopenssl library must be installed using ‘pip install pyopenssl’
3. The ECDSA signing key and the Certificate pair must be present in the same folder as the python script and must be named ‘ecdsasha256-signer.key.pem’ and ‘ecdsa-sha256-signer.crt.pem’ respectively.
(**Note:** The key pair in SDK are just for example, please generated new key by openssl !)

Run the python script in folder: component\common\application\amazon\amazon_ota_tools

- cmd command after IAR build : **\$ python python_custom_ecdsa_Z2.py**
- shell command after GCC build : **\$ python3 python_custom_ecdsa_Z2_gcc.py**

There might be some error if there are packages lack in your environment (like openssl...). Please install the package and run the script again.

Once all these are present and the python script is run, it will generate a custom signed binary named **firmware_is_sig.bin** inside the component\common\application\amazon\amazon_ota_tools folder.



After getting the custom signed **firmware_is_sig.bin**, you can upload it to the S3 bucket.

8.4 How to Trigger a Custom Signed OTA Job in Amazon AWS IOT Core

Go to AWS IoT Core <https://console.aws.amazon.com/iot?p=icr&cp=bn&ad=c>. Then, follow the following steps to create an AWS OTA task for AmebaZ2:

Step 1. Click on 'Create OTA update job', select your job type and then click

The screenshots show the AWS IoT Core interface. The top one is the main dashboard with a sidebar for AWS IoT. The 'Jobs' item in the sidebar is highlighted with a red box. The bottom one is a detailed view of the 'Jobs' list, showing a single entry named 'AFR_OTA-Pro_OTA_test_0615_1' with a status of 'Completed'. The 'Create job' button at the top right of the list table is also highlighted with a red box.

Job type

- Create custom job
Send a request to acquire an executable job file from one of your S3 buckets to one or more devices connected to AWS IoT.
- Create FreeRTOS OTA update job
Send a request to acquire an executable job file from one of your S3 buckets to one or more devices connected to AWS IoT.
- Create Greengrass V1 Core update job
Create a snapshot job to update one or more Greengrass V1 Core devices with the latest Greengrass V1 Core or OTA agent version.

Cancel **Next**

- For Job properties, give a unique name to your OTA job, then click next.

OTA job properties Info

Job properties

Job name

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

Description - optional

▶ Tags - optional

Cancel **Next**

Step 3. In the following page, choose your device to update and select the protocol for file transfer.

If CONFIG_OTA_MQTT_UPDATE_DEMO_ENABLED is set. Please select MQTT.

If CONFIG_OTA_HTTP_UPDATE_DEMO_ENABLED is set. Please select MQTT and HTTP.

AWS IoT > Jobs > Create job > OTA job

OTA file configuration Info

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

Devices to update
Choose things and/or thing groups
 X

Select the protocol for file transfer
Select the protocol that your device supports.

MQTT HTTP

File Info

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

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

- Step 4. In the following page, choose the option 'Use my custom signed firmware image'.
In the signature field just enter '/'. Choose hash algorithm as 'SHA-256'. Choose encryption algorithm as 'ECDSA'. In "pathname of code signing certificate", enter '/'

File Info

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

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

Code signing information
Enter information about your file and how it was signed so that your devices can verify its authenticity before they install it.

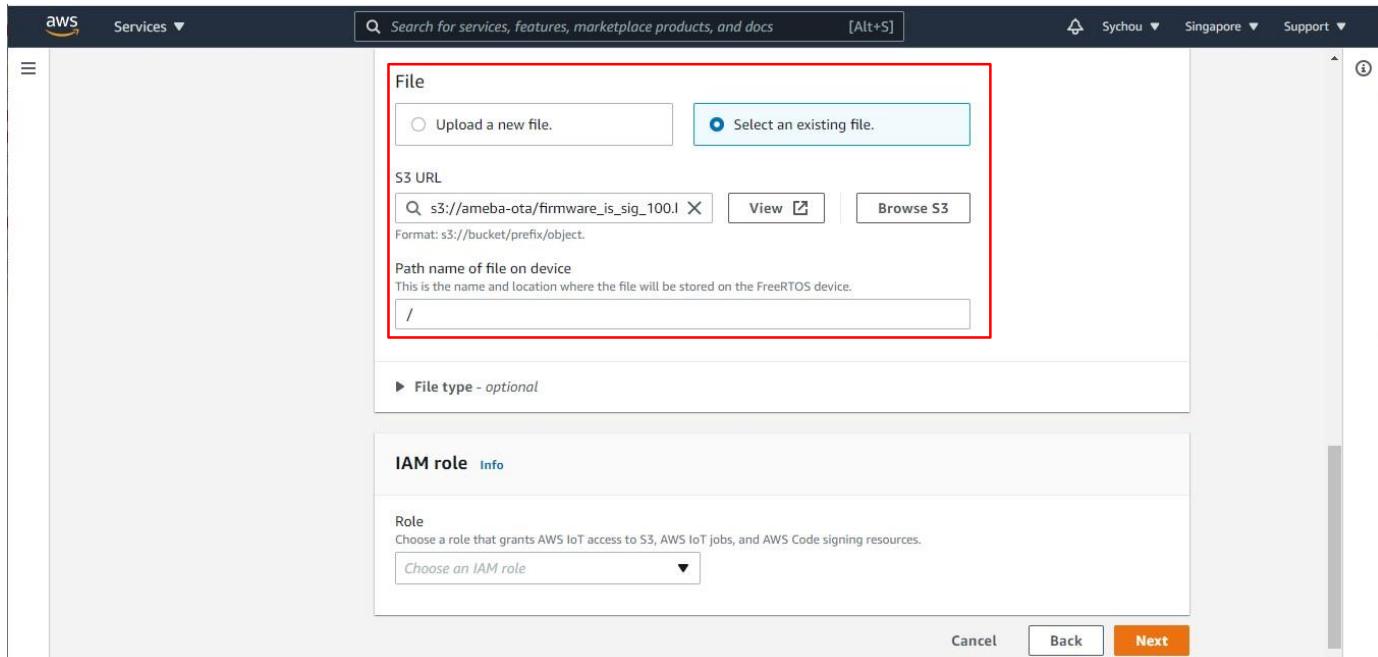
Signature

Original hash algorithm
Choose the hash algorithm that was used to create your file signature.

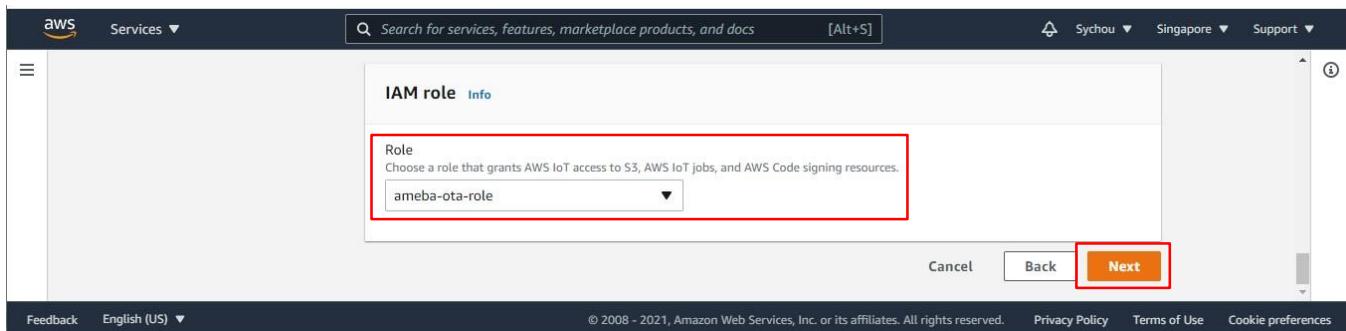
Original encryption algorithm
Choose the encryption algorithm that was used to create your file signature.

Path name of code signing certificate on device

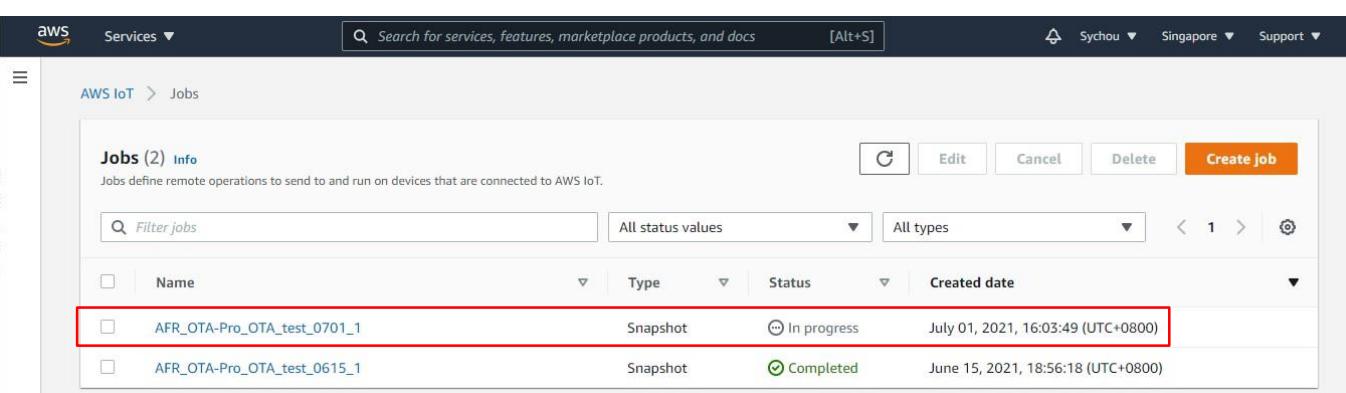
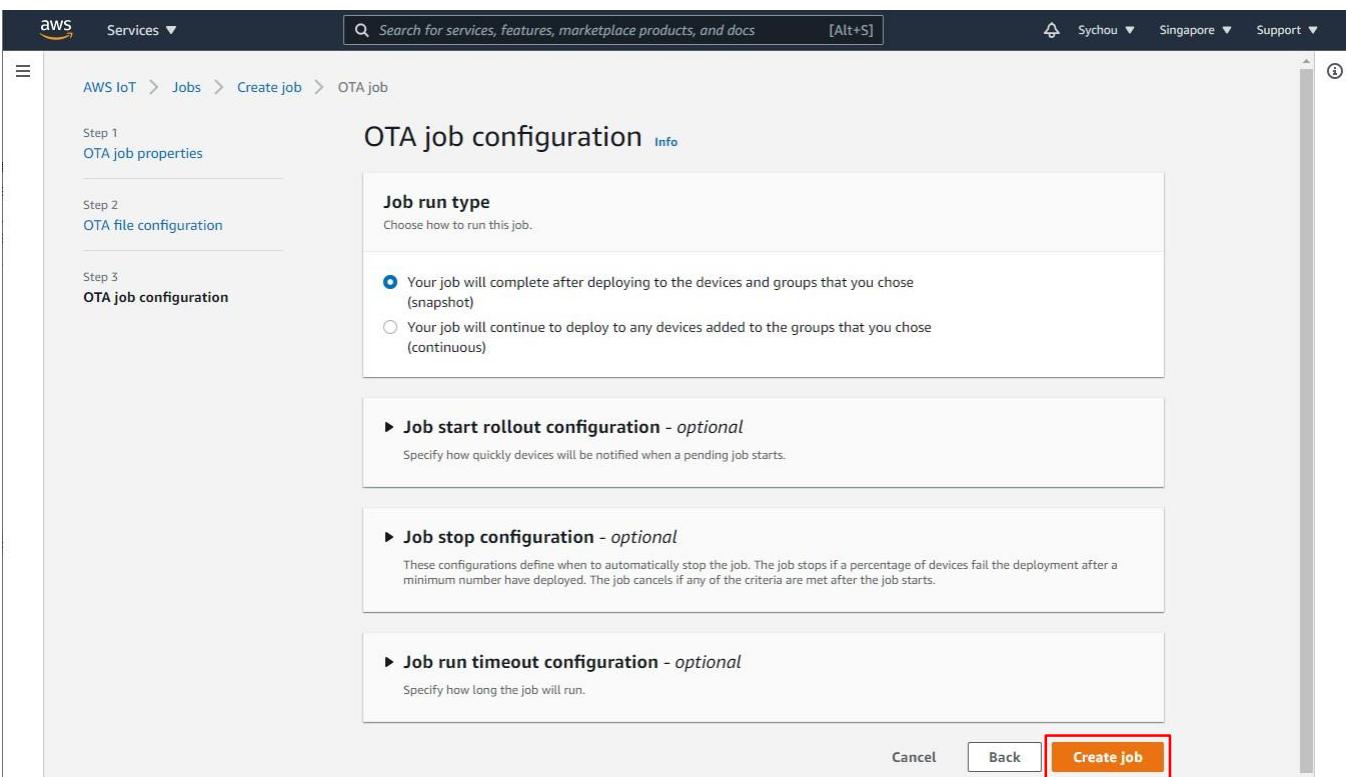
- Step 5. Choose your custom signed firmware binary that was generated by the python script from S3 bucket.
In "Pathname of file on device", enter '/'



Step 6. Choose the IAM role for OTA update job. (This is the same IAM role as any OTA update job)



Step 7. Click next, and create your OTA job.



8.5 Run OTA Demo

Now we can see that the status of OTA job on AWS IoT Core is “in progress”. It means that it is waiting AmebaZ2 to request the update.

Next, download the image file with older version number to AmebaZ2 and then reboot the device, the application will automatically start run OTA demo.

In the beginning, we can check the app version of this running firmware, and the OTA process by the job ID:

We can see that the OTA process start!

```
60 8174 [iot_thread] [INFO ] [DEMO][8174] State: RequestingJob Received: 1 Queued: 0 Processed: 0 Dropped: 0
61 8427 [OTA Agent] [prvIngestDataBlock] Received file block 5, size 1024
[update_ota_prepare_addr] Get loaded_fw_idx 1
[update_ota_prepare_addr] NewFWAddr 00240000
[update_ota_erase_upg_region] NewFWLen 1712128
[update_ota_erase_upg_region] NewFWBlkSize 418 0x1a2[OTA][prvPAL_WriteBlock_amebaPro] NewImg2BlkSize 418
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0x1400: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x241400 (0x241400)
62 23029 [OTA Agent] [prvIngestDataBlock] Remaining: 1672
63 23033 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
64 23048 [OTA Agent] [INFO ] [MQTT][23048] (MOTT connection 7026ef40) MQTT PUBLISH operation queued.
65 23055 [OTA Agent] [prvRequestFileBlock_Mqtt] OK: $aws/things/ameba-otastreams/AFR OTA-d7c5e0c5-6d12-45c3-8fe4-7167753724dc/get/cbor
66 23067 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [RequestTimer] New state [WaitingForFileBlock]
67 23081 [OTA Agent] [prvIngestDataBlock] Received file block 4, size 1024
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0x1000: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x241000 (0x241000)
68 23101 [OTA Agent] [prvIngestDataBlock] Remaining: 1671
69 23105 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
70 23120 [OTA Agent] [prvIngestDataBlock] Received file block 3, size 1024
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0xc00: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x240c00 (0x240c00)
```

After receiving the final block, the signature will be checked if valid or not. If signature is valid, the OTA process is successful ! Then, the device will reboot with new firmware automatically.

```
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0x1a2000: iBlockSize: 0x48
[OTA] Final block with signature arrived
[OTA][prvPAL_WriteBlock_amebaPro] OTA1 Sig Size is 71
[OTA] image download is already done, dropped, aws_ota_imgsZ=0x1A2000, ImgLen=0x1A2000
10226 140083 [OTA Agent] [prvIngestDataBlock] Received final expected block of file.
10227 140091 [OTA Agent] [prvStopRequestTimer] Stopping request timer.
[OTA] Authenticating and closing file.
10228 140100 [OTA Agent] [prvPAL_CloseFile] Authenticating and closing file.
10229 140107 [OTA Agent] [prvPAL_CheckFileSignature_amebaPro] Started sig-sha256-ecdsa signature verification, file: /
10230 140118 [OTA Agent] Assume Cert - No such file: /. Using header file
10231 140682 [OTA Agent] [prvIngestDataBlock] File receive complete and signature is valid.
10232 140689 [OTA Agent] [prvStopRequestTimer] Stopping request timer.
10233 140695 [OTA Agent] [prvPublishStatusMessage] Msg: {"status": "IN_PROGRESS", "statusDetails": {"self_test": "ready", "updatedBy": "0x63090002"}}
10234 140709 [OTA Agent] [INFO ][MQTT][140709] (MQTT connection 7026ef40) MQTT PUBLISH operation queued.
10235 140717 [OTA Agent] [INFO ][MQTT][140717] (MQTT connection 7026ef40, PUBLISH operation 7026f3c0) Waiting for operation completion.
10236 140735 [iot_threa] [INFO ][DEMO][140735] State: WaitingForFileBlock Received: 3343 Queued: 0 Processed: 0 Dropped: 0

10237 140955 [OTA Agent] [INFO ][MQTT][140955] (MQTT connection 7026ef40, PUBLISH operation 7026f3c0) Wait complete with result SUCCESS.
10238 140965 [OTA Agent] [prvPublishStatusMessage] 'IN_PROGRESS' to $aws/things/ameba-ota/jobs/AFR_OTA-Pro_OTA_test_0701_4/update
10239 140977 [OTA Agent] [INFO ][DEMO][140977] Received eOTA_JobEvent_Activate callback from OTA Agent.

10240 140986 [OTA Agent] [INFO ][MQTT][140986] (MQTT connection 7026ef40) Disconnecting connection.
10241 140995 [OTA Agent] [INFO] [MQTT] [core_mqtt.c:2149] 10242 141000 [OTA Agent] Disconnected from the broker.10243 141004 [OTA Agent]
10244 141007 [OTA Agent] [INFO ][MQTT][141007] (MQTT connection 7026ef40) Network connection closed.
10245 141015 [OTA Agent] [INFO ][DEMO][141015] Mqtt disconnected due to invoking disconnect function.

10246 141025 [OTA Agent] [INFO ][MQTT][141025] (MQTT connection 7026ef40) Network connection destroyed.

[update_ota_prepare_addr] Get loaded_fw_idx 1

[update_ota_prepare_addr] NewFWAddr 00240000

[update_ota_signature] Append OTA signature
[update_ota_signature] signature:
E6 8C 79 0B 33 80 72 F7 C3 D2 63 D1 A5 BE 05 CA
66 EE 58 4B BD F9 69 A2 53 9E C9 F9 0B DA 46 CC[OTA] [prvPAL_ActivateNewImage_amebaPro] Update OTA success!
[OTA] Resetting MCU to activate new image.
```

After booting with newer image, the device will start a self-test mode to check the app version which is latest.

We can see that the version now is 101.9.1, which is greater than old version 100.9.1.

```
46 7625 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: "/" }]}... ]  
47 7633 [OTA Agent] [prvParseJobDoc] In self test mode.  
48 7638 [OTA Agent] [prvValidateUpdateVersion] The update version is newer than the version on device.  
49 7647 [OTA Agent] [prvParseJobDoc] Setting image state to Testing for file ID 0  
50 7654 [OTA Agent] [prvSetImageState_amebaPro] Testing image.  
51 7660 [OTA Agent] [prvPublishStatusMessage] Msg: {"status": "IN_PROGRESS", "statusDetails": {"self_test": "active", "updatedBy": "0x64090002"} }  
52 7673 [OTA Agent] [INFO ][MQTT][7673] (MQTT connection 7026af20) MQTT PUBLISH operation queued.  
53 7681 [OTA Agent] [INFO ][MQTT][7681] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Waiting for operation completion.  
54 7916 [OTA Agent] [INFO ][MQTT][7916] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Wait complete with result SUCCESS.  
55 7926 [OTA Agent] [prvPublishStatusMessage] 'IN_PROGRESS' to $aws/things/ameba-ota/jobs/AFR_OTA-Pro_OTA_test_0701_4/update  
56 7937 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForJob] Event [ReceivedJobDocument] New state [CreatingFile]  
57 7949 [OTA Agent] [prvInSelfTestHandler] prvInSelfTestHandler, platform is in self-test.  
58 7957 [OTA Agent] [prvPAL_GetPlatformImageState_amebaPro] Image current state (0x01)  
59 7964 [OTA Agent] [INFO ][DEMO][7964] Received eOTA_JobEvent_StartTest callback from OTA Agent.  
  
60 7974 [OTA Agent] [prvSetImageState_amebaPro] Accepted and committed final image.  
  
[update_ota_prepare_addr] Get loaded_fw_idx 2  
  
[update_ota_prepare_addr] NewFWAddr 00040000  
61 7989 [OTA Agent] [prvStopSelfTestTimer] Stopping the self test timer.  
62 7995 [OTA Agent] [prvPublishStatusMessage] Msg: {"status": "SUCCEEDED", "statusDetails": {"reason": "accepted v101.9.1"} }  
63 8007 [OTA Agent] [INFO ][MQTT][8007] (MQTT connection 7026af20) MQTT PUBLISH operation queued.  
64 8015 [OTA Agent] [INFO ][MQTT][8014] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Waiting for operation completion.  
65 8026 [iot_threa] [INFO ][DEMO][8026] State: RequestingJob Received: 1 Queued: 0 Processed: 0 Dropped: 0  
  
66 8243 [OTA Agent] [INFO ][MQTT][8243] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Wait complete with result SUCCESS.  
67 8253 [OTA Agent] [prvPublishStatusMessage] 'SUCCEEDED' to $aws/things/ameba-ota/jobs/AFR_OTA-Pro_OTA_test_0701_4/update  
68 8264 [OTA Agent] [prvExecuteHandler] Called handler. Current State [CreatingFile] Event [StartSelfTest] New state [WaitingForJob]  
69 9036 [iot_threa] [INFO ][DEMO][9036] State: CreatingFile Received: 1 Queued: 0 Processed: 0 Dropped: 0
```

In the final, the log imply that the OTA status is changed to “SUCCEEDED” !

9 Troubleshooting

If these steps don't work, look at the device log in the serial terminal. You should see some text that indicates the source of the problem.

For general troubleshooting information about Getting Started with FreeRTOS, see [Troubleshooting getting started](#).

9.1 ERROR: Invalid Key

Please check **WIFI_SSID** and **WIFI_PASSWORD** in in “component/common/application/amazon/v202012/demos/include/aws_clientcredential.h”

```
Enter SSID for Soft AP started
3 1098 [example_a] Wi-Fi configuration successful.
4 1108 [iot_threa] [INFO ][DEMO][1108] -----STARTING DEMO-----
5 1115 [iot_threa] [INFO ][INIT][1115] SDK successfully initialized.

LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
WIFI deinitialized
Initializing WIFI ...
WIFI initialized

Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...
```

9.2 Failed to establish new MQTT connection

Please check **clientcredentialMQTT_BROKER_ENDPOINT** in “component/common/application/amazon/v202012/demos/include/aws_clientcredential.h”

```
6 12508 [iot_threa] [INFO ][DEMO][12508] Successfully initialized the demo. Network type for the demo: 1
7 12517 [iot_threa] [INFO ][MQTT][12517] MQTT library successfully initialized.
8 12524 [iot_threa] [INFO ][DEMO][12524] MQTT demo client identifier is ameba-ota (length 9).
9 12624 [iot_threa] [ERROR][NET][12624] Failed to resolve [REDACTED].amazonaws.com.
10 12934 [iot_threa] [ERROR][MQTT][12934] Failed to establish new MQTT connection, error NETWORK ERROR.
11 12943 [iot_threa] [ERROR][DEMO][12943] MQTT CONNECT returned error NETWORK ERROR.
12 12951 [iot_threa] [INFO ][MQTT][12950] MQTT library cleanup done.
13 12957 [iot_threa] [ERROR][DEMO][12957] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
14 13094 [iot_threa] [INFO ][INIT][13094] SDK cleanup done.
15 13099 [iot_threa] [INFO ][DEMO][13099] -----DEMO FINISHED-----
```

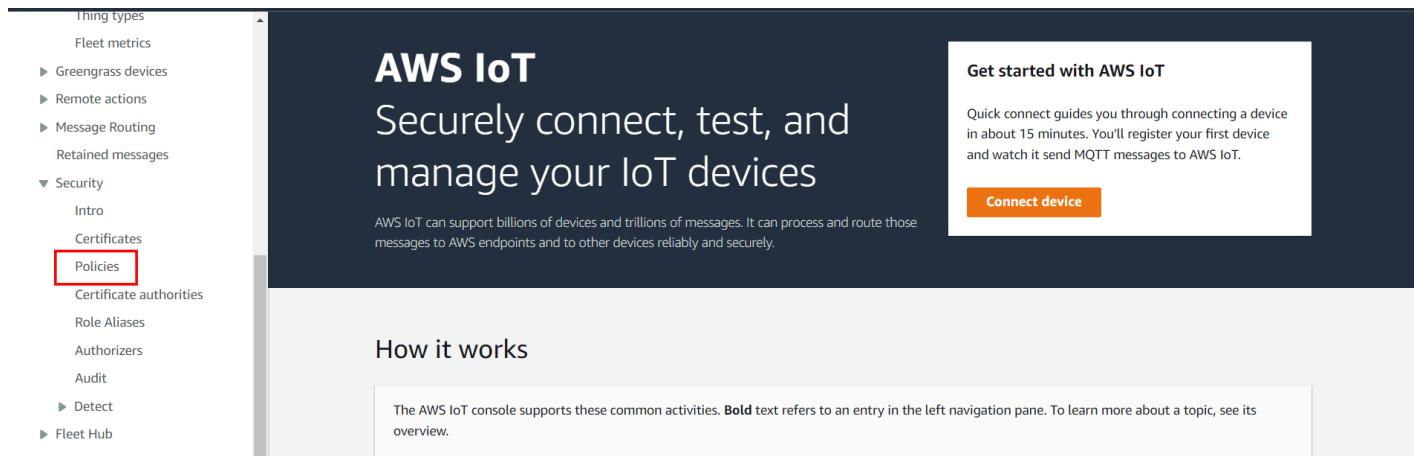
TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in “component/common/application/amazon/v202012/demos/include/aws_clientcredential_keys.h”

10 AWS Fleet Provisioning Introduction

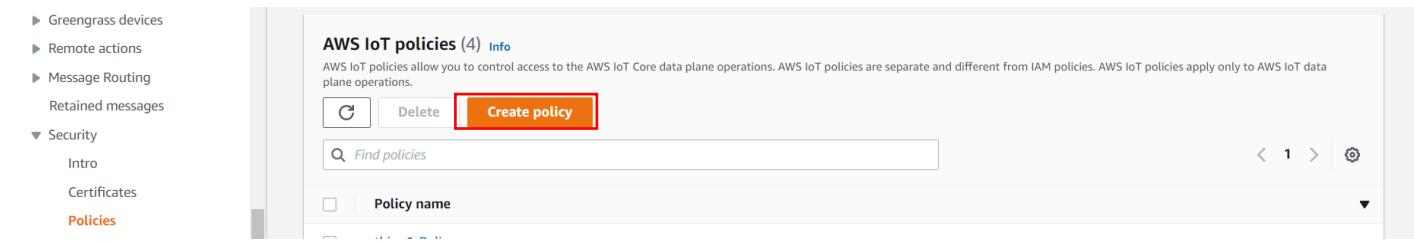
Fleet Provisioning is first introduced in version AWS v202012. Using an automated template, end users can now do the provisioning instead of the manufacturers. This guide introduces Fleet Provisioning **by Claim**. For more information visit [AWS by Claim](#).

10.1 Create Claim Certificate Policy



The screenshot shows the AWS IoT console interface. On the left, a navigation pane lists various IoT features: Thing types, Fleet metrics, Greengrass devices, Remote actions, Message Routing, Retained messages, Security (with sub-options like Intro, Certificates, Policies, and Certificate authorities), Role Aliases, Authorizers, Audit, Detect, and Fleet Hub. The 'Policies' option under the Security section is highlighted with a red box. The main content area displays the AWS IoT logo and the tagline 'Securely connect, test, and manage your IoT devices'. Below this, a text box states: 'AWS IoT can support billions of devices and trillions of messages. It can process and route those messages to AWS endpoints and to other devices reliably and securely.' To the right, a 'Get started with AWS IoT' section includes a 'Connect device' button. At the bottom, a 'How it works' section provides an overview of common activities supported by the console.

Click “Policies” under Security tab



The screenshot shows the 'AWS IoT policies' page. The left navigation pane is identical to the previous screenshot, with 'Policies' selected. The main area shows a list of four existing policies. A 'Create policy' button is prominently displayed at the top of the list. Below it is a search bar labeled 'Find policies' and a table header with 'Policy name'.

Click “Create Policy”

Use the Policy name: FleetProvisioningDemoClaimPolicy

Fill in the Policy document with the following:

- **aws-region** with the AWS region of your choice (e.g. us-west-2)
- **aws-account-id** with your AWS account ID

Policy document [Info](#)

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

Builder	JSON
---------	------

Policy effect	Policy action	Policy resource
Allow	iot:Publish	arn:aws:iot: aws-region:aws-account-id:*
Allow	iot:Receive	arn:aws:iot: aws-region:aws-account-id:*
Allow	iot:Subscribe	arn:aws:iot: aws-region:aws-account-id:topicfilter/*
Allow	iot:Connect	arn:aws:iot: aws-region:aws-account-id:client/*

[Add new statement](#)

Click “Create”

10.2 Create Claim Certificate

retained messages

▼ Security

- Intro
- Certificates**
- Policies
- Certificate authorities

Certificates [Info](#)

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

Certificates	Certificates you've transferred
------------------------------	---

Select “Certificates” under Security tab & Click “Create certificate”

AWS IoT > Security > Certificates

Certificates [Info](#)

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

Certificates	Certificates you've transferred
------------------------------	---

Certificates (12)

<input type="checkbox"/> Certificate ID	Status	Created	Actions ▾	Add certificate ▲
<input type="checkbox"/>			C	Create certificate
				Register certificates

Click on “Active” & Create

Certificate

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

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

Certificate status

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

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

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

[Cancel](#) [Create](#)

Download the certificates

You can download the certificate now, or later, but the key files can only be downloaded now.

Device certificate
046d014d6bb...te.pem.crt [Download](#)

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

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

Public key file
046d014d6bb6eba201254cb...76fc2d3-public.pem.key [Download](#)

Private key file
046d014d6bb6eba201254cb...6fc2d3-private.pem.key [Download](#)

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 [Download](#)
RSA 2048 bit key: Amazon Root CA 1

Amazon trust services endpoint [Download](#)
ECC 256 bit key: Amazon Root CA 3

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

[Continue](#)

You will see certificate created similar below, click on the certificate id

<input type="checkbox"/>	Certificate ID	Status
<input type="checkbox"/>	d6ff9fc3e41395b6faf8e9ada082f9e640279f6d55f41008196b19e8b1ba5547	Active

Select "Attach policies"

Policies Things Noncompliance

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

<input type="checkbox"/>	Name
No policies You don't have any policies attached to this certificate.	

[Create policy](#) [Detach policies](#) [Attach policies](#)

Select: FleetProvisioningDemoClaimPolicy & attach policies

*This will be the temporary claim certificate in exchange for permanent cert in fleet provisioning

10.3 Create fleet provisioning template

Navigate to Connect -> Connect many devices and select "Create provisioning template"

AWS IoT X

Monitor

Connect

- Connect one device
- Connect many devices** Connect many devices
- Bulk registration

Test

MQTT test client

Manage

- All devices
- Things
- Thing groups
- Thing types
- Fleet metrics
- Greengrass devices
- Software packages New
- Remote actions
- Jobs
- Job templates
- Secure tunnels

How it works

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

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

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

Connect many devices (2) [Info](#)

[Create provisioning template](#)

To connect many devices, the provisioning template automates the provisioning experience for multiple devices.

Find provisioning templates

Select “Provisioning devices with claim certificates”

[AWS IoT](#) > [Connect](#) > [Connect many devices](#) > [Create provisioning template](#)

Create provisioning template

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

Provisioning scenario

Choose the provisioning scenario that fits your device manufacturing and installation processes the best. [Learn more](#)

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

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

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

This example uses the Template name “testcase1”

[AWS IoT](#) > [Connect](#) > [Connect many devices](#) > [Create provisioning template](#) > Provisioning by claim certificates

Step 1
Describe provisioning template

Step 2
Set provisioning actions

Step 3
Set device permissions

Step 4
Review and create

Describe provisioning template Info

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

Provisioning template properties Info

Provisioning template status

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

Inactive

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

Active

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

Provisioning template name

testcase1

The name can have up to 50 characters and must not contain spaces. Valid characters: A-Z, a-z, 0-9, and _ (underscore) and - (hyphen).

Description - optional

A description of the provisioning template you're creating.

Provisioning role

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

ameba-ota-role



Create new role

Attach managed policy to IAM role

Tags - optional

Please select the policy and certificate which create in 10.1 and 10.2

Claim certificate policy [Info](#)

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

Claim certificate provisioning policy

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

[View](#) [Create IoT policy](#)

Tip: Use multiple certificates

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

Claim certificates - optional (1/123) [Info](#)

[Activate](#) [Deactivate](#) [Delete](#) [Upload](#)

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

Certificate ID	Status
2f5e168263d6332a40383b2c330a39	Active

[Cancel](#) [Next](#)

Pre-provisioning Hook will not be used for simplicity

Pre-provisioning actions (recommended) [Info](#)

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

Pre-provisioning action

Use a pre-provisioning action (recommended)
Perform actions prior to provisioning the device. For example, to check the device against a known device database to prevent unauthorized devices from connecting to your account.

Don't use a pre-provisioning action
No actions will be performed prior to provisioning the device and the device is given access to your AWS account.

We recommend that you use a pre-provisioning action

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

[Learn more](#)

Set the things prefix.

Automatic thing creation - optional

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

Automatically create a thing resource when provisioning a device

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

thing_

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

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

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

Select the policy which create in 10.1

Set device permissions Info

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

Policies (1/18) <small>Info</small>		<input type="button" value="C"/>	<input type="button" value="Create policy"/>
Choose up to 10 policies to attach to this certificate.			
<input type="text" value="Find policies"/> < 1 > ⑤			
<input type="checkbox"/>	Policy name	ARN	▼
<input type="checkbox"/>	xuyan-Policy	arn:aws:iot:ap-southeast-1:553661462376:policy/xuyan-Policy	▼
<input type="checkbox"/>	webrtc_iot_policy_test_0308	arn:aws:iot:ap-southeast-1:553661462376:policy/webrtc_iot_policy_...	▼
<input type="checkbox"/>	wanglu_policy	arn:aws:iot:ap-southeast-1:553661462376:policy/wanglu_policy	▼
<input checked="" type="checkbox"/>	wanglu_fleet_policy	arn:aws:iot:ap-southeast-1:553661462376:policy/wanglu_fleet_policy	▼

Then select “Create template”.

10.4 Configure AmebaZ2 Amazon FreeRTOS

10.4.1 Get Broker Endpoint by AWS IoT Core

The screenshot shows the AWS IoT Settings page. On the left, there is a navigation sidebar with the following menu items:

- Get started
- Fleet provisioning templates
- Manage
- Fleet Hub
- Greengrass
- ▼ Secure
 - Overview
 - Certificates
 - Policies
 - CAs
 - Role Aliases
 - Authorizers
- Defend
- Act
 - Test
- Software
- Settings** (highlighted with a red box)
- Learn
- Feature spotlight

The main content area displays the "Device data endpoint" section, which includes a note about devices using the account's device data endpoint to connect to AWS, and a table showing the endpoint URL (redacted) and .amazonaws.com. Below this is the "Domain configurations" section, which shows a table with columns: Name, Domain name, Status, Service type, and Date updated. A message indicates "No domain configurations" and "You don't have any domain configurations.", with a "Create domain configuration" button.

10.4.2 Setup IoT Core Information with AmebaZ2 Amazon FreeRTOS

Setup BROKER_ENDPOINT, WIFI_SSID, PASSWORD in “component\common\application\amazon\amazon-freeertos\demos\include\aws_clientcredential.h”

```
26: #ifndef __AWS_CLIENTCREDENTIAL_H__
27: #define __AWS_CLIENTCREDENTIAL_H__
28:
29: /*
30:  * @brief MQTT Broker endpoint.
31:  *
32:  * @todo Set this to the fully-qualified DNS name of your MQTT broker.
33:  */
34: #define clientcredentialMQTT_BROKER_ENDPOINT "xxxxxxxxxxxxxxxxxxxxxxxxxx.amazonaws.com"
35:
36: /*
37:  * @brief Host name.
38:  *
39:  * @todo Set this to the unique name of your IoT Thing.
40:  * Please note that for convenience of demonstration only we
41:  * are using a #define here. In production scenarios the thing
42:  * name can be something unique to the device that can be read
43:  * by software, such as a production serial number, rather
44:  * than a hard-coded constant.
45: */
46: #define clientcredentialIOT_THING_NAME ""
47:
48: /*
49:  * @brief Port number the MQTT broker is using.
50:  */
51: #define clientcredentialMQTT_BROKER_PORT 8883
52:
53: /*
54:  * @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
55:  */
56: #define clientcredentialGREENGRASS_DISCOVERY_PORT 8443
57:
58: /*
59:  * @brief Wi-Fi network to join.
60:  *
61:  * @todo If you are using Wi-Fi, set this to your network name.
62: */
63: #define clientcredentialWIFI_SSID "WiFi-SSID"
64:
65: /*
66:  * @brief Password needed to join Wi-Fi network.
67:  * @todo If you are using WPA, set this to your network password.
68: */
69: #define clientcredentialWIFI_PASSWORD "WiFi-Password-Here"
```

In component\common\application\amazon\amazon-freertos\demos\fleet_provisioning_with_csr\fleet_provisioning_demo.c ,

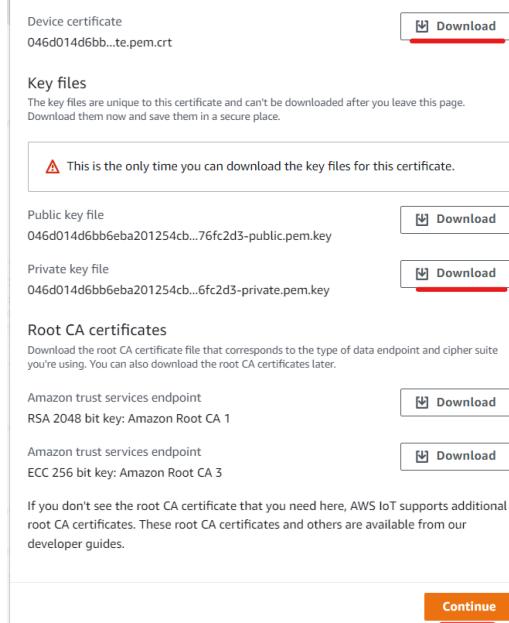
set democonfigPROVISIONING_TEMPLATE_NAME as “testcase1”

```

100: /*-----Demo configurations-----*/
101: /**
102:  ** Note: The device client certificate and private key credentials are
103:  ** obtained by the transport interface implementation (with Secure Sockets)
104:  ** from the demos/include/aws_clientcredential_keys.h file.
105:  */
106: /**
107:  * The following macros SHOULD be defined for this demo which uses both server
108:  * and client authentications for TLS session:
109:  * -----keyCLIENT_CERTIFICATE_PEM for Fleet provisioning claim CSR certificate.
110:  * -----keyCLIENT_PRIVATE_KEY_PEM for Fleet provisioning client private key.
111:  */
112: /**
113:  * @brief The length of #democonfigPROVISIONING_TEMPLATE_NAME.
114:  */
115: /**
116:  * #define democonfigPROVISIONING_TEMPLATE_NAME ... "testcase1" //Fill in Fleet provisioning template name
117:  */
118: |
119: #define fpdemoPROVISIONING_TEMPLATE_NAME_LENGTH ... ((uint16_t)(sizeof(democonfigPROVISIONING_TEMPLATE_NAME)-1))
120: /**
121: #define fpdemoMAX_THING_NAME_LENGTH ..... 128
122: /**
123: /**

```

Fill keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in “\component\common\application\amazon\amazon-freertos\demos\include\aws_clientcredential_keys.h” by xxxxxxxx-certificate.pem and xxxxxxxx-private.pem.key.



To run Fleet Provisioning demo, please find in “component\common\application\amazon\amazon-freertos\vendors\realtek\boards\amebaZ2\aws_demos\config_files\aws_demo_config.h” and enable **CONFIG_FLEET_PROVISIONING_DEMO_ENABLED**

```

26:#ifndef _AWS_DEMO_CONFIG_H_
27: #define _AWS_DEMO_CONFIG_H_
28:
29: /* To run a particular demo you need to define one of these.
30: * Only one demo can be configured at a time
31: */
32: #define CONFIG_CORE_HTTP_MUTUAL_AUTH_DEMO_ENABLED
33: #define CONFIG_CORE_HTTP_S3_DOWNLOAD_DEMO_ENABLED
34: #define CONFIG_CORE_HTTP_S3_DOWNLOAD_MULTITHREADED_DEMO_ENABLED
35: #define CONFIG_CORE_HTTP_S3_UPLOAD_DEMO_ENABLED
36: #define CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
37: #define CONFIG_CORE_MQTT_CONNECTION_SHARING_DEMO_ENABLED
38: #define CONFIG_DEVICE_SHADOW_DEMO_ENABLED
39: #define CONFIG_DEVICE_DEFENDER_DEMO_ENABLED
40: #define CONFIG_JOBS_DEMO_ENABLED
41: #define CONFIG_MQTT_BLE_DEMO_ENABLED
42: #define CONFIG_GREENGRASS_DISCOVERY_DEMO_ENABLED
43: #define CONFIG_TCP_ECHO_CLIENT_DEMO_ENABLED
44: #define CONFIG_POSIX_DEMO_ENABLED
45: #define CONFIG_OTA_UPDATE_DEMO_ENABLED
46: #define CONFIG_BLE_GATT_SERVER_DEMO_ENABLED
47: #define CONFIG_BLE_NUMERIC_COMPARISON_DEMO_ENABLED
48: #define CONFIG_FLEET_PROVISIONING_DEMO_ENABLED
49: */
50: /* These defines are used in iot_demo_runner.h for demo selection */
51:
52: // #define CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
53: // #define CONFIG_OTA_UPDATE_DEMO_ENABLED
54: // #define CONFIG_DEVICE_SHADOW_DEMO_ENABLED
55: #define CONFIG_FLEET_PROVISIONING_DEMO_ENABLED

```

Output on TeraTerm or equivalent: Sucessfully provisioning device

```

12 7686 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:665] 13 7614 [iot_threal] Subscribe to Fleet provisioning CSR Topics
14 7619 [iot_threal]
15 9878 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:686] 16 9883 [iot_threal] Publish to the MQTT topic $aws/certificates/crea
te-from-csr/chor.
17 9891 [iot_threal]
18 9896 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:698] 19 9903 [iot_threal] Attempt to receive publish message from broker.
20 9908 [iot_threal]
21 10051 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:1375] 22 10057 [iot_threal] Received accepted response from Fleet Provisi
oning CreateCertificateFromCsr API.23 10067 [iot_threal]
24 11069 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:720] 25 11075 [iot_threal] Received certificate with Id: b90b80a135ed12a0
3b56b6ccdf129eccce91c82f37271e70e788a836dd8ef18e0
26 11085 [iot_threal]
27 11176 [iot_threal] [INFO] [FleetProvisioning] [pkcs11_operations.c:660] 28 11182 [iot_threal] Writing certificate into label "Device Cert".29 1118
8 [iot_threal]
30 11271 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:738] 31 11277 [iot_threal] Unsubscribe fleet provisioning CSR Topics
32 11283 [iot_threal]
33 13382 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:764] 34 13387 [iot_threal] Subscribe to registerThing reply topics35 1339
3 [iot_threal]
36 15502 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:775] 37 15507 [iot_threal] Publish to the MQTT topic $aws/provisioning-te
mplates/testcasel/provision/chor.
38 15516 [iot_threal]
39 15521 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:798] 40 15529 [iot_threal] Attempt to receive publish message from broker
41 15535 [iot_threal]
42 15824 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:1395] 43 15830 [iot_threal] Received accepted response from Fleet Provisi
oning RegisterThing API.44 15837 [iot_threal]
45 16341 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:821] 46 16346 [iot_threal] Received AWS IoT Thing name: thing_FPDemoID15:
00:13
47 16353 [iot_threal]
48 18469 [iot_threal] [INFO] [FleetProvisioning] [fleet_provisioning_demo.c:843] 49 18474 [iot_threal] Disconnecting the MQTT connection with : [REDACTED]
50 18490 [iot_threal]
51 19490 [iot_threal] [INFO] [DEMO] [19490] Demo completed successfully.
52 wifi link err:00000000
53 LwIP_DHCP: dhcp stop.
54 wifi link err:00000000.
55 WIFI deinitialized52 19628 [iot_threal] [INFO] [INIT][19628] SDK cleanup done.
53 19632 [iot_threal] [INFO] [DEMO][19632] -----DEMO FINISHED-----

```

In addition, under the aws console, new Thing will be generated.

The screenshot shows the AWS IoT Things management interface. On the left, a sidebar menu includes options like Monitor, Activity, Connect, and Manage, with 'Things' selected. The main content area displays 'Things (2) Info' with a brief description of what an IoT thing is. Below this are buttons for Refresh, Advanced search, Run aggregations, Edit, and Delete. A search bar allows filtering by name, type, group, billing, or searchable attribute. The list of things shows two entries:

Name
thing_FPDemoID15:00:13
thing1

The entry 'thing_FPDemoID15:00:13' is highlighted with a red box.