AmebaPro2 Amazon FreeRTOS-LTS - Getting Started Guide

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211. Fax: +886-3-577-6047
www.realtek.com

1 Configure AWS IoT Core

1.1 Set up your AWS account and Permissions

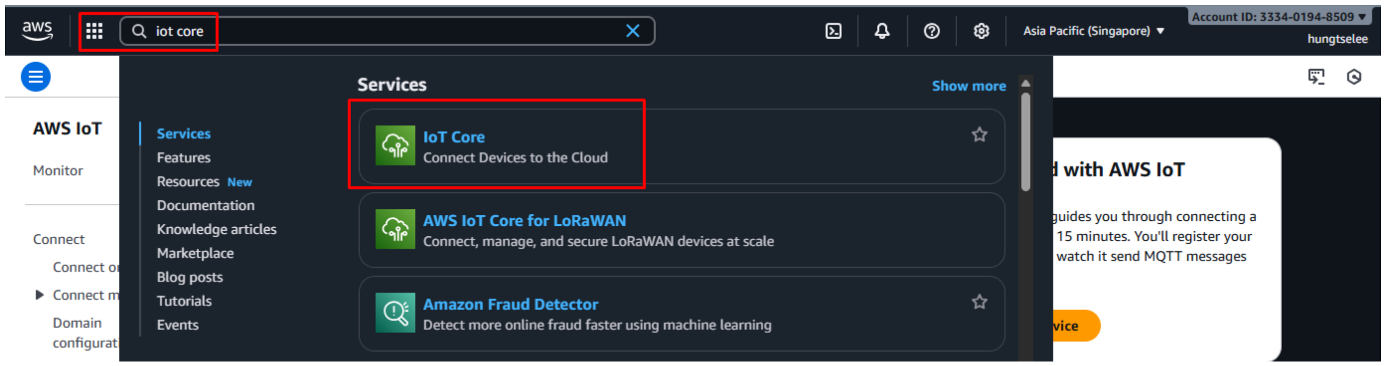
Refer to the instructions at Set up your AWS Account https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html. Follow the steps outlined in these sections to create your account and a user and get started:

- Sign up for an AWS account
- Create a user and grant permissions
- Open the AWS IoT console

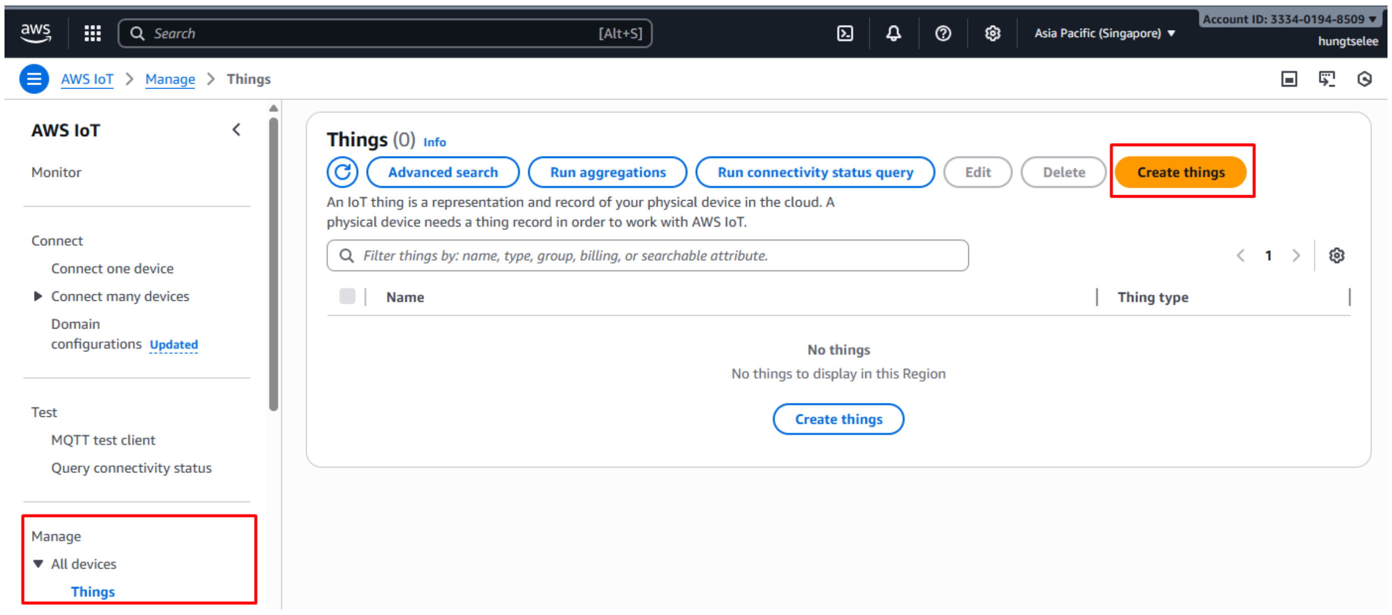
Please pay special attention to the Notes in AWS webpage.

1.2 Create a New Device

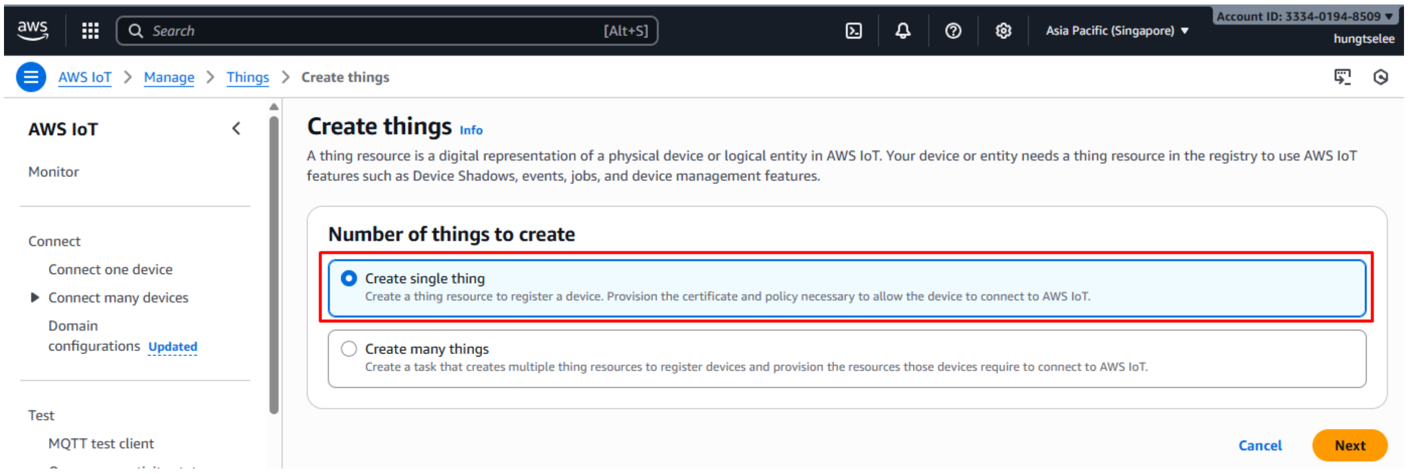
First, go to IoT Core webpage



To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click "Create things".



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



Create things Info

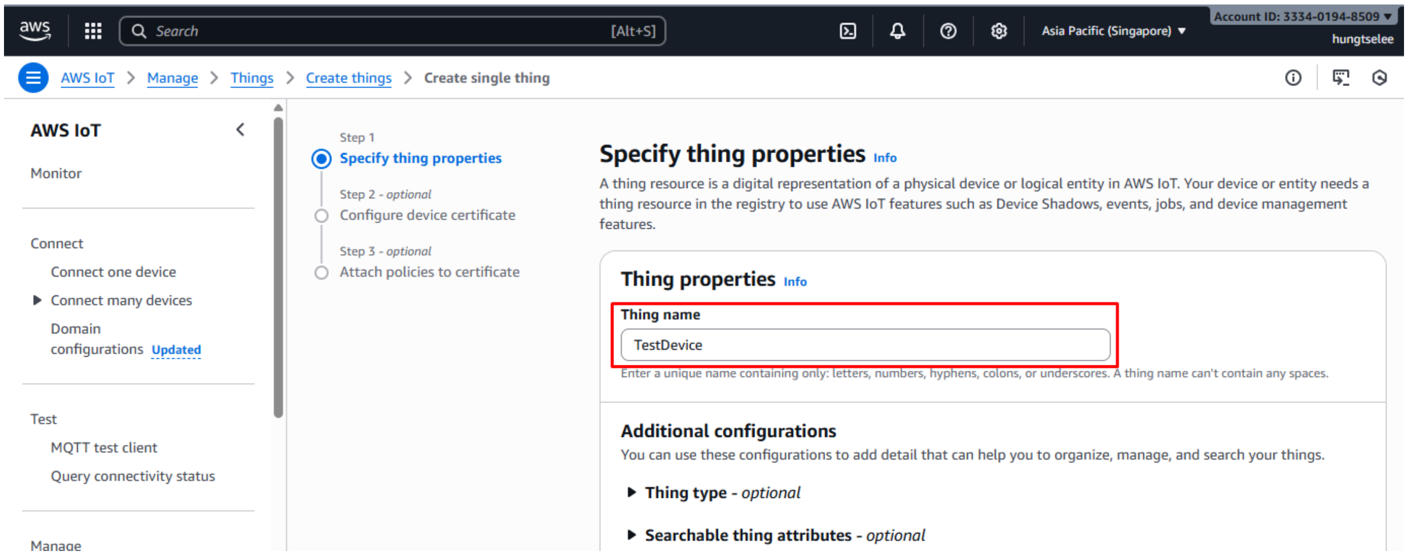
A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Number of things to create

[✓ Create single thing]
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

Create many things
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

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


aws
Search [Alt+S]
Asia Pacific (Singapore)
Account ID: 3334-0194-8509
hungtselee
AWS IoT > Manage > Things > Create things > Create single thing
AWS IoT
Monitor
Connect
Connect one device
Connect many devices
Domain configurations Updated
Test
MQTT test client
Query connectivity status
Manage
Step 1
Specify thing properties
Step 2 - optional
Configure device certificate
Step 3 - optional
Attach policies to certificate

Specify thing properties Info

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties Info

Thing name: TestDevice

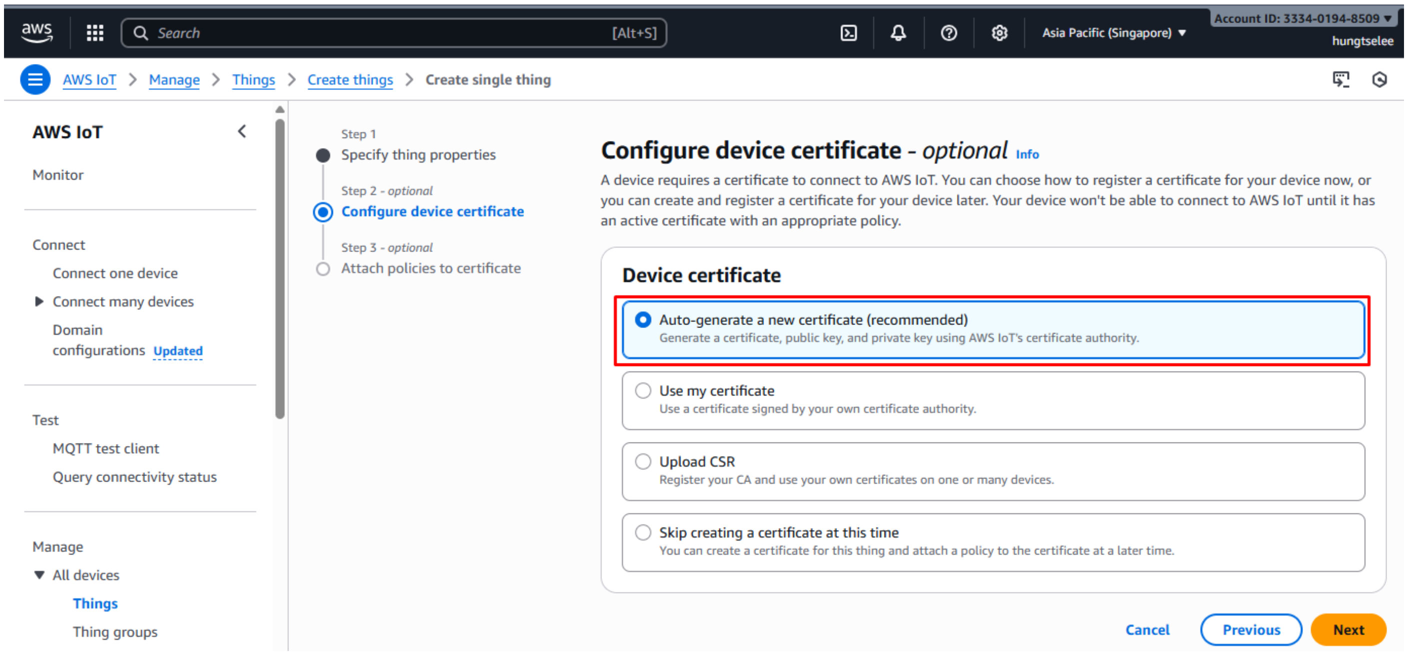
Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

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*



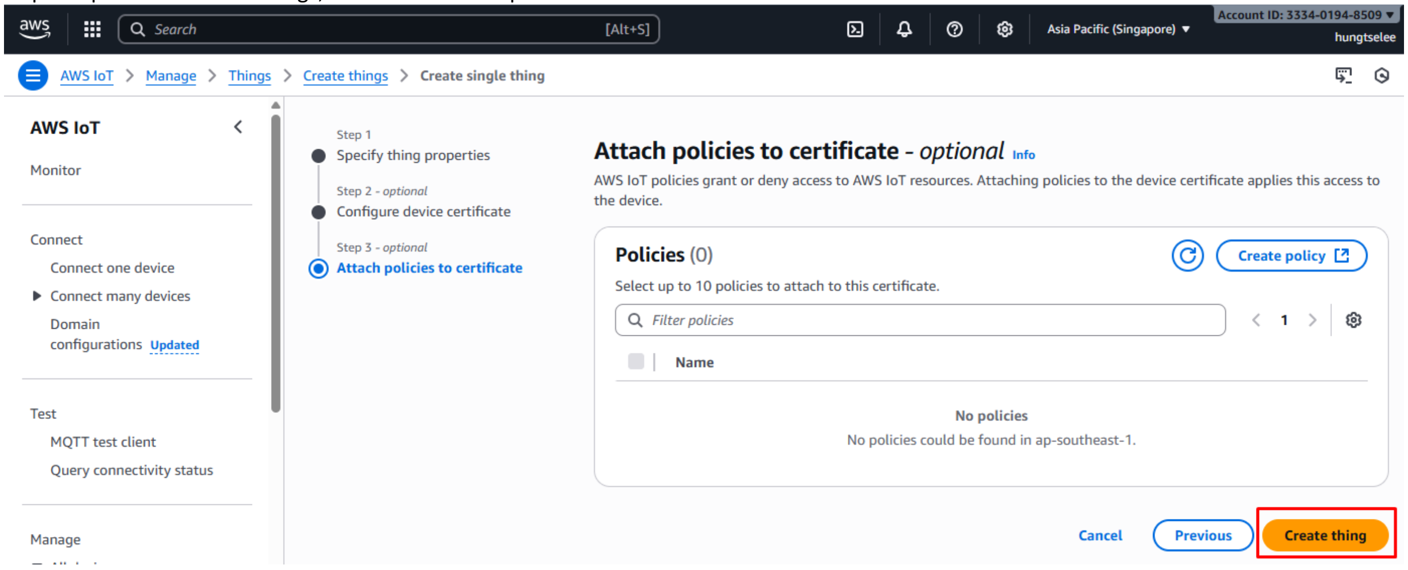
Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

- [✓ Auto-generate a new certificate (recommended)] Generate a certificate, public key, and private key using AWS IoT's certificate authority.
- Use my certificate — Use a certificate signed by your own certificate authority.
- Upload CSR — Register your CA and use your own certificates on one or many devices.
- Skip creating a certificate at this time — You can create a certificate for this thing and attach a policy to the certificate at a later time.

Skip this part and "Create thing", we will attach the policies to certificate later.


aws
Search [Alt+S]
Asia Pacific (Singapore)
Account ID: 3334-0194-8509
hungtselee
AWS IoT > Manage > Things > Create things > Create single thing
AWS IoT
Monitor
Connect
Connect one device
Connect many devices
Domain configurations Updated
Test
MQTT test client
Query connectivity status
Manage
Step 1
Specify thing properties
Step 2 - optional
Configure device certificate
Step 3 - optional
Attach policies to certificate

Attach policies to certificate - *optional* Info

AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

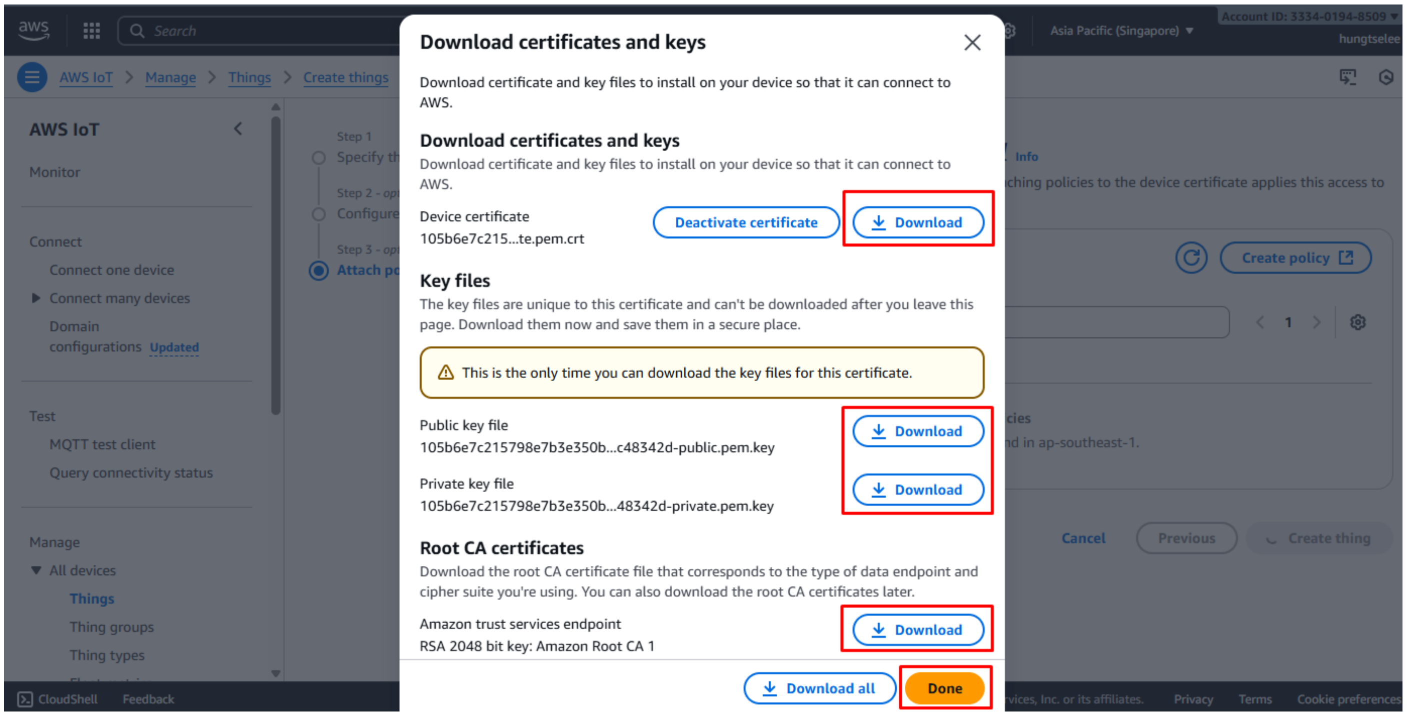
Policies (0)

Select up to 10 policies to attach to this certificate.

No policies

No policies could be found in ap-southeast-1.

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 Done



Download certificates and keys

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

Download certificates and keys

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

Device certificate
105b6e7c215...te.pem.crt

Key files

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

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

Public key file
105b6e7c215798e7b3e350b...c48342d-public.pem.key

Private key file
105b6e7c215798e7b3e350b...48342d-private.pem.key

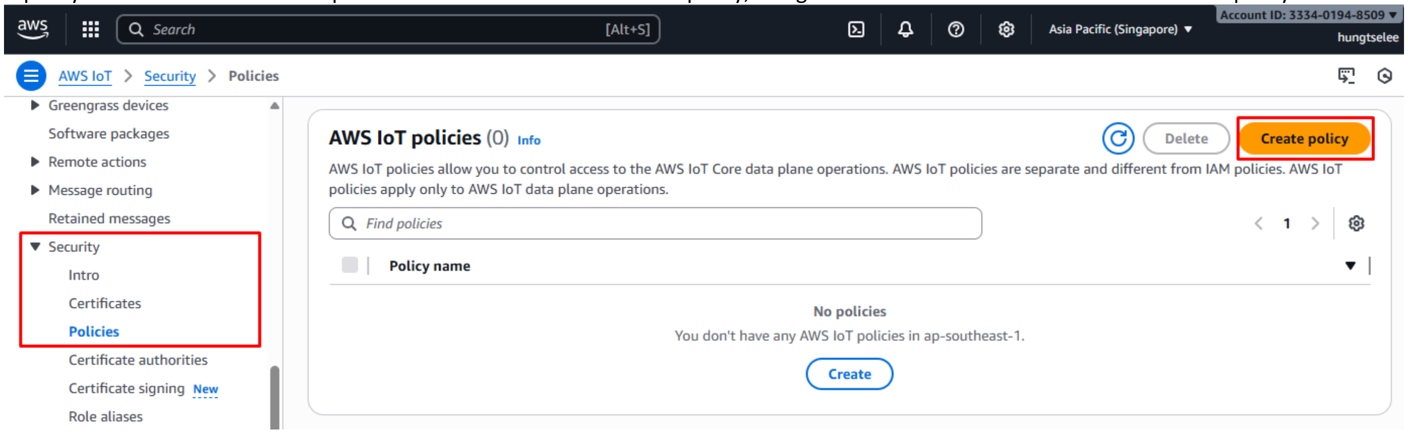
Root CA certificates

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

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

1.3 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 policy"

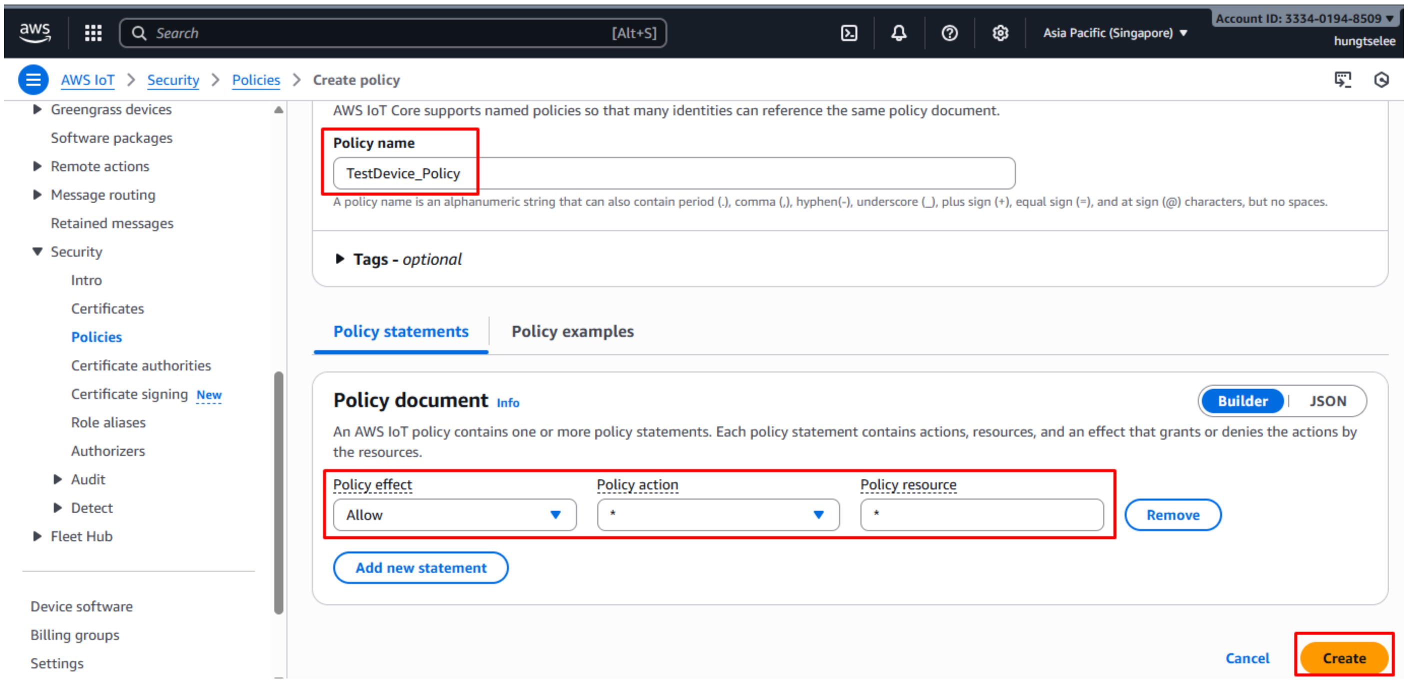


AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

No policies
You don't have any AWS IoT policies in ap-southeast-1.

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

Name the policy and set allowed "action" and "resource" as "*", then create it



AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name: TestDevice_Policy

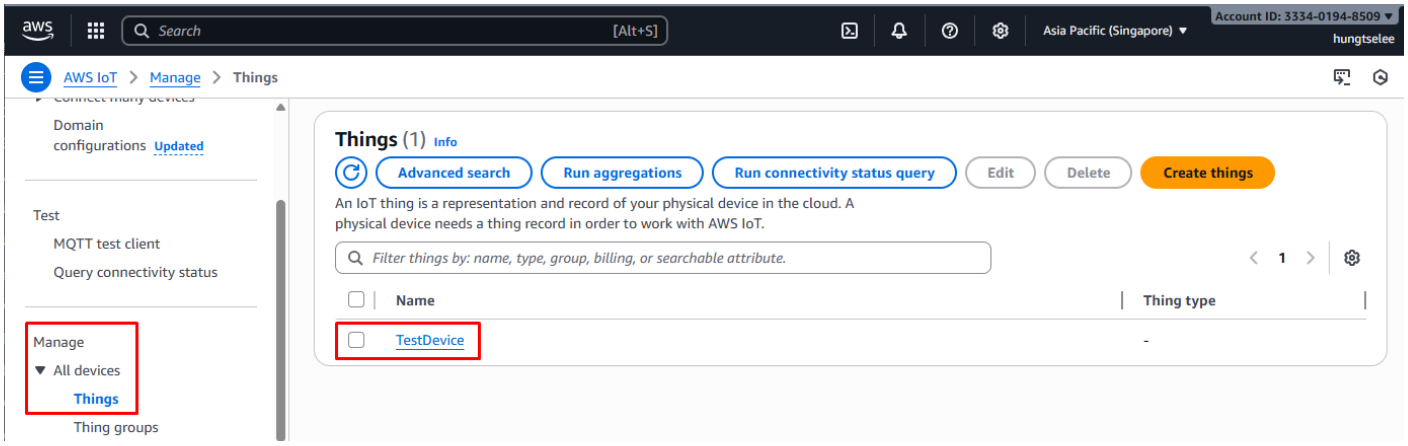
A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

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

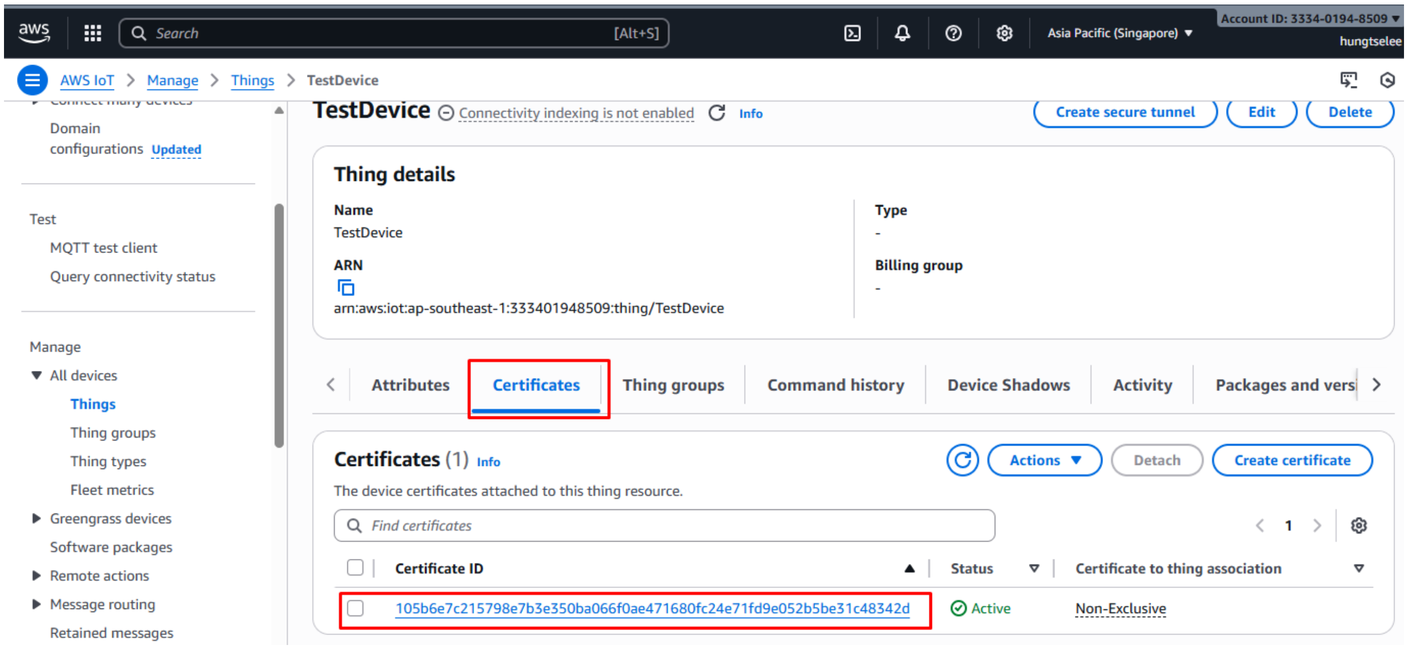
Policy effect	Policy action	Policy resource
Allow	*	*

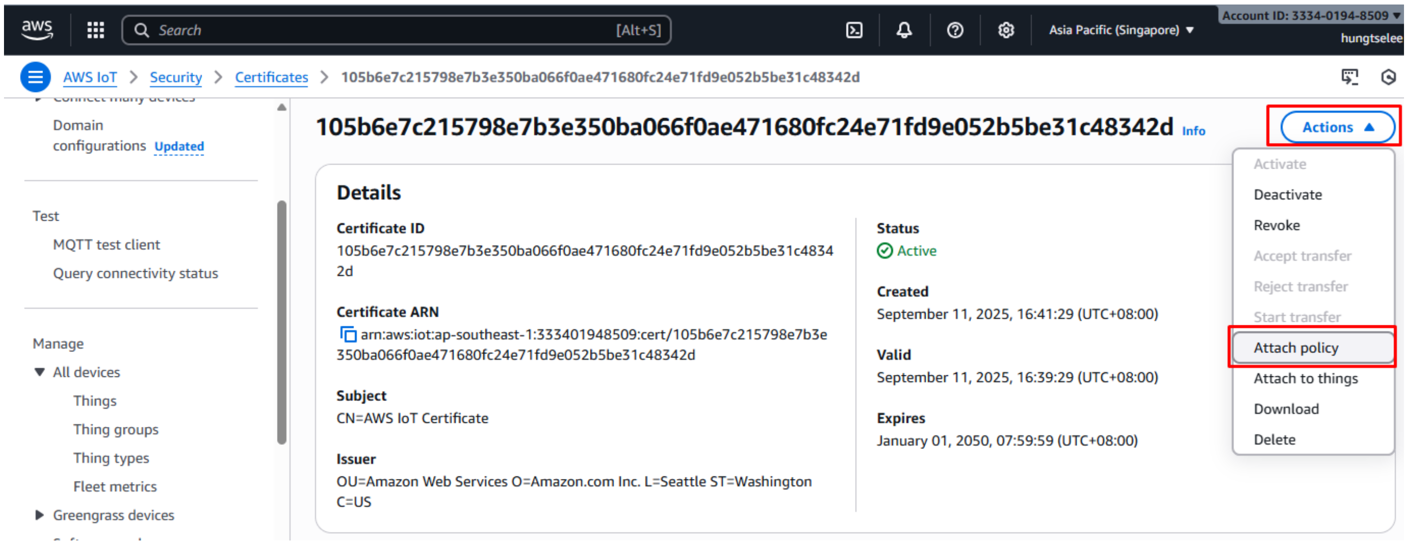
1.4 Attach Policy

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


aws
Search [Alt+S]
Asia Pacific (Singapore)
Account ID: 3334-0194-8509
hungtselee
AWS IoT > Manage > Things
Domain configurations Updated
Test
MQTT test client
Query connectivity status
Manage
All devices
Things
Thing groups
Things (1) Info
Advanced search
Run aggregations
Run connectivity status query
Edit
Delete
Create things
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.
Filter things by: name, type, group, billing, or searchable attribute.
Name
Thing type
TestDevice
-

Click Certificate, then choose the certificate create in previous step.





105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d Info

Details

Certificate ID
105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d

Certificate ARN
arn:aws:iot:ap-southeast-1:333401948509:cert/105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d

Subject
CN=AWS IoT Certificate

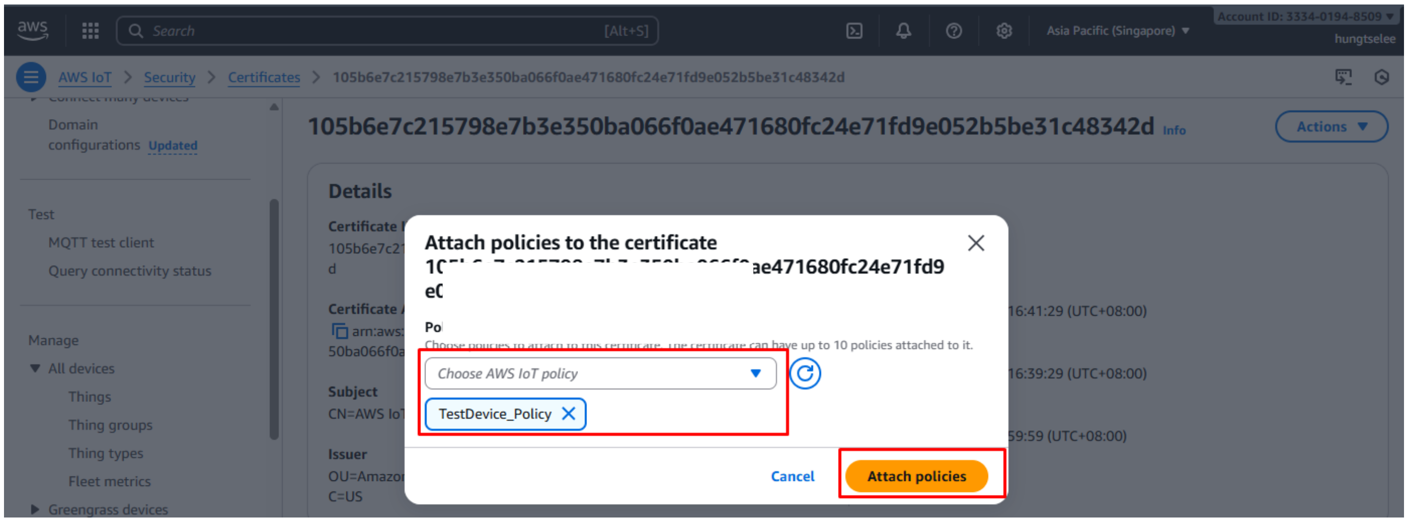
Issuer
OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US

Status
Active

Created
September 11, 2025, 16:41:29 (UTC+08:00)

Valid
September 11, 2025, 16:39:29 (UTC+08:00)

Expires
January 01, 2050, 07:59:59 (UTC+08:00)



Attach policies to the certificate 105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d

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

2 Configure AmebaPro2 Amazon FreeRTOS

2.1 Download FreeRTOS-LTS Library Source Code from Github

Open source link: https://github.com/ambiot/amazon-freertos/tree/amebaPro2-9.x-202107.00-LTS
branch: **amebaPro2-9.x-202107.00-LTS**

2.1.1 Download Source Code of Required Libraries to SDK

Go to “AmebaPro2_SDK/project/realtek_amebapro2_v0_example/src”:

```
$ cd project/realtek_amebapro2_v0_example/src
$ git clone --recurse-submodules -b amebaPro2-9.x-202107.00-LTS https://github.com/ambiot/amazon-freertos.git aws_iot_freertos_lts
```

2.1.2 Modify FreeRTOSConfig.h

Copy & paste below configurations to the end of FreeRTOSConfig.h in “project\realtek_amebapro2_v0_example\inc”:

```
/* Sets the length of the buffers into which logging messages are written - so
     * also defines the maximum length of each log message. */
    #define configLOGGING_MAX_MESSAGE_LENGTH            512

    /* Set to 1 to prepend each log message with a message number, the task name,
     * and a time stamp. */
    #define configLOGGING_INCLUDE_TIME_AND_TASK_NAME    1

    /* Map the FreeRTOS printf() to the logging task printf. */
    /* The function that implements FreeRTOS printf style output, and the macro
     * that maps the configPRINTF() macros to that function. */
    #define configPRINTF( X )    vLoggingPrintf X

    /* Non-format version thread-safe print. */
    #define configPRINT( X )     vLoggingPrint( X )

    /* Map the logging task's printf to the board specific output function. */
    #define configPRINT_STRING( X )        printf( X )

    #define iotconfigUSE_PORT_SPECIFIC_HOOKS
```

2.1.3 Configure Mbedtls Setting

In this project, we use mbedtls-2.16.6, same as KVS webrtc. Set mbedtls version to 2.16.6 in “project/realtek_amebapro2_v0_example/GCC-RELEASE/config.cmake”

```
set(mbedtls "mbedtls-2.16.6")
```

You have to modify some mbedtls config before running aws-iot demo, go to “component/ssl/mbedtls-2.16.6/include/mbedtls/config_rsa.h” check the following setting:

```
#define MBEDTLS_THREADING_ALT
//#define MBEDTLS_DEBUG_C
#define MBEDTLS_THREADING_C
```

The default mbedtls version of AmebaPro2 is 3.0.0. However, for the aws iot demo, we use mbedtls version 2.16.6 in default. It might be easier for user to use it with AWS KVS service now.
If user want to use the aws-iot with mbedtls-3.0.0 or mbedtls-2.4.0, user can compare the config file between mbedtls-2.16.6 and mbedtls-3.0.0, mbedtls-2.4.0

2.1.4 Multiple Definition Issue

There might be multiple definition of “vApplicationGetIdleTaskMemory” and “vApplicationGetTimerTaskMemory”.
Since aws demo runner have the same function that have been defined in SDK, so we should comment one of them, go to “component\os\freertos\freertos_cb.c” and comment these two functions

```
//void vApplicationGetIdleTaskMemory(...)
//{
//    ...
//}

//void vApplicationGetTimerTaskMemory(...)
//{
//    ...
//}
```

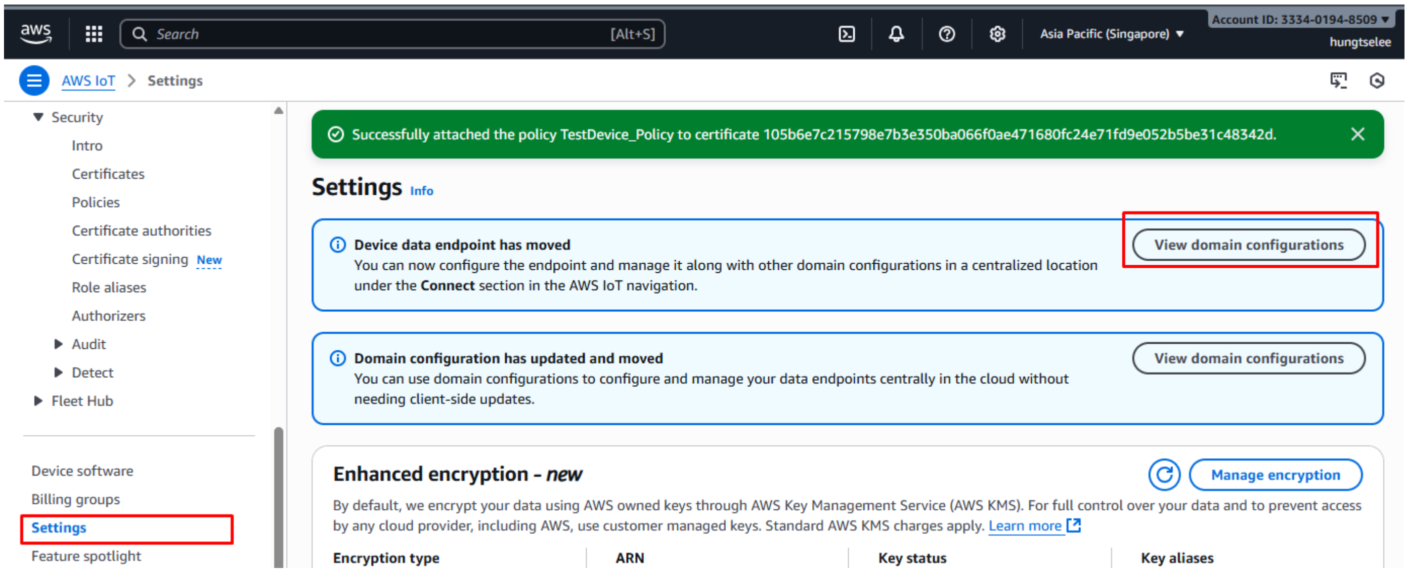
2.1.5 Configure NVM interface for PKCS11

User should select a non-volatile memory (NVM) interface such as SD card and flash for the PKCS11 library
1. **SD card**: used by default, so please insert a SD card to the device
2. **Flash**: user can select the flash for pkcs11 in “aws_iot_freertos_lts/vendors/realtek/boards/amebaPro2/ports/pkcs11”

```
#define PKCS11_NVM_INTERFACE PKCS11_AMEBA_FLASH
```

If using flash as NVM for PKCS11, please arrange a proper flash address(AWSIOT_PKCS11_DATA) in platform_opt.h to store pkcs11 data.

2.2 Get Broker Endpoint by AWS IoT Core



Account ID: 3334-0194-8509
hungtselee
AWS IoT > Settings

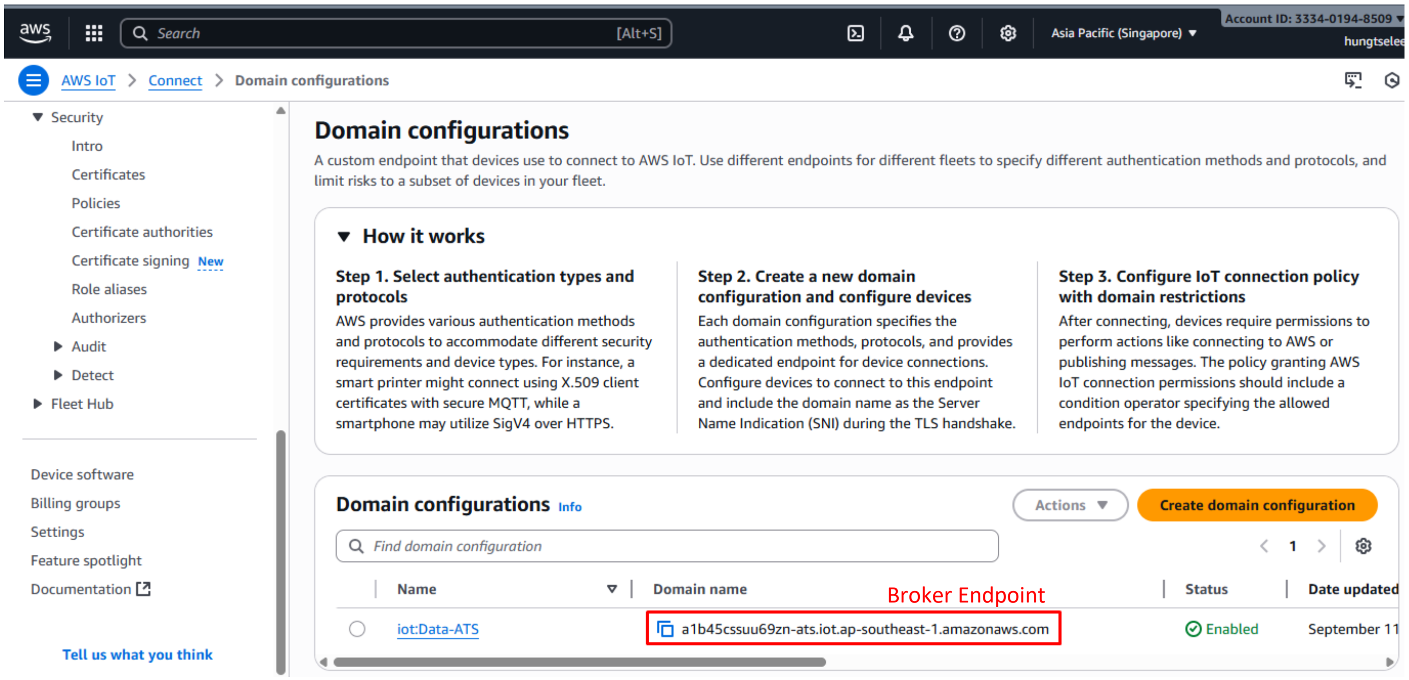
Successfully attached the policy TestDevice_Policy to certificate 105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d.

Settings Info

Device data endpoint has moved
You can now configure the endpoint and manage it along with other domain configurations in a centralized location under the **Connect** section in the AWS IoT navigation.
View domain configurations

Domain configuration has updated and moved
You can use domain configurations to configure and manage your data endpoints centrally in the cloud without needing client-side updates.
View domain configurations

Enhanced encryption - *new*
By default, we encrypt your data using AWS owned keys through AWS Key Management Service (AWS KMS). For full control over your data and to prevent access by any cloud provider, including AWS, use customer managed keys. Standard AWS KMS charges apply. Learn more



Account ID: 3334-0194-8509
hungtselee
AWS IoT > Connect > Domain configurations

Domain configurations

A custom endpoint that devices use to connect to AWS IoT. Use different endpoints for different fleets to specify different authentication methods and protocols, and limit risks to a subset of devices in your fleet.

How it works

Step 1. Select authentication types and protocols
AWS provides various authentication methods and protocols to accommodate different security requirements and device types. For instance, a smart printer might connect using X.509 client certificates with secure MQTT, while a smartphone may utilize SigV4 over HTTPS.

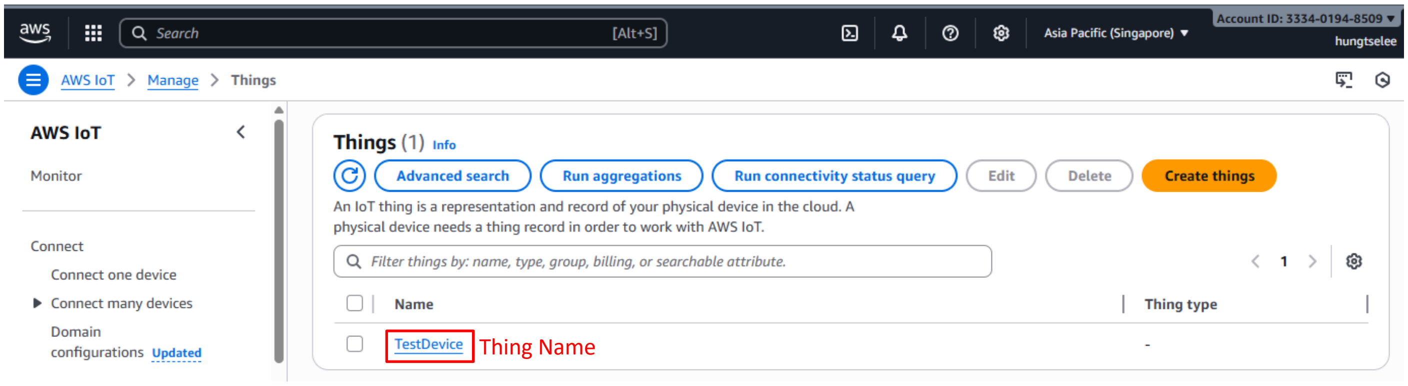
Step 2. Create a new domain configuration and configure devices
Each domain configuration specifies the authentication methods, protocols, and provides a dedicated endpoint for device connections. Configure devices to connect to this endpoint and include the domain name as the Server Name Indication (SNI) during the TLS handshake.

Step 3. Configure IoT connection policy with domain restrictions
After connecting, devices require permissions to perform actions like connecting to AWS or publishing messages. The policy granting AWS IoT connection permissions should include a condition operator specifying the allowed endpoints for the device.

Domain configurations Info

Name	Domain name	Status	Date updated
iot:Data-ATS	a1b45cssuu69zn-ats.iot.ap-southeast-1.amazonaws.com	Enabled	September 11

2.3 Get Thing Name



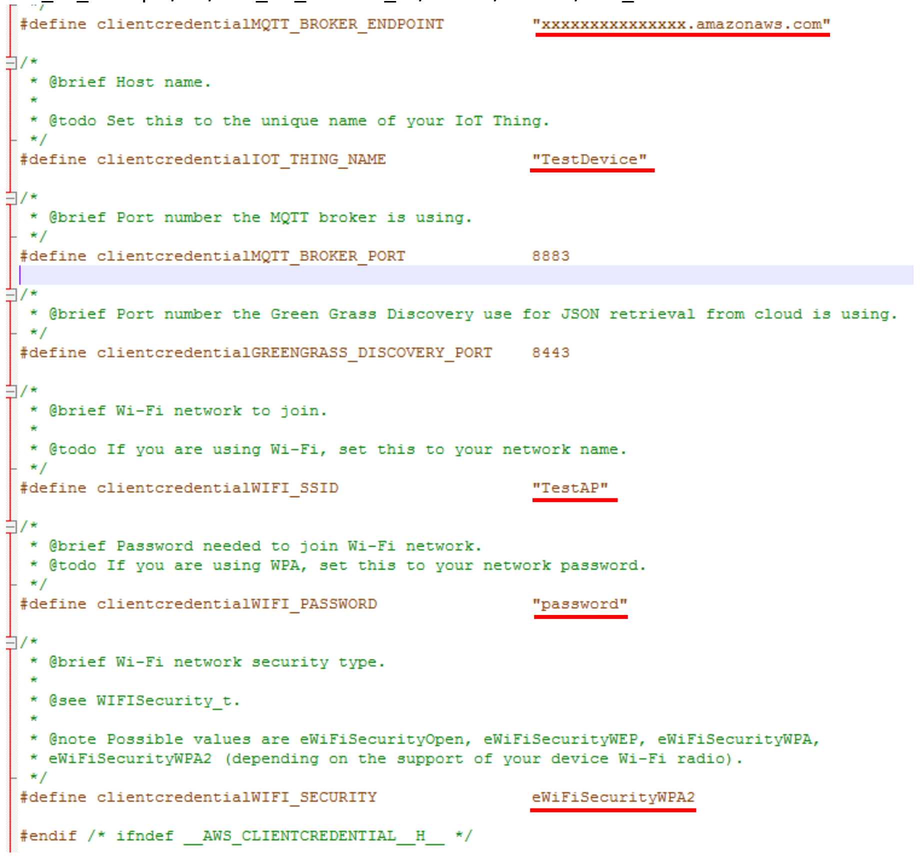
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

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

Name	Thing type
TestDevice	-

2.4 Setup IoT Core Information with AmebaPro2 Amazon FreeRTOS

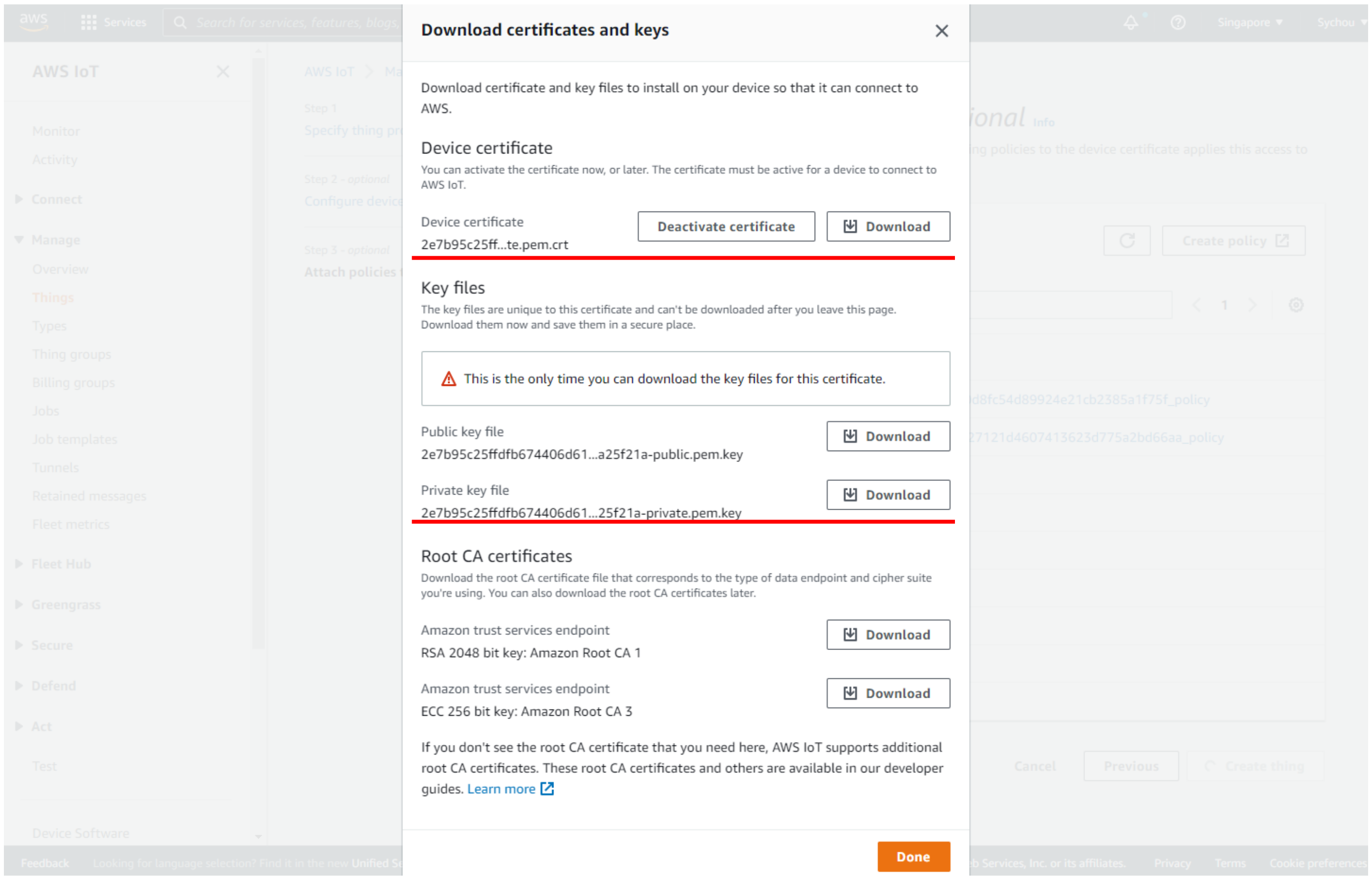
Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in
“project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential.h”



```
#define clientcredentialMQTT_BROKER_ENDPOINT        "xxxxxxxxxxxxxxx.amazonaws.com"

/*
 * @brief Host name.
 *
 * @todo Set this to the unique name of your IoT Thing.
 */
#define clientcredentialIOT_THING_NAME              "TestDevice"

/*
 * @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__ */
```

2.4.1 Setup Thing’s Private Key and Certificate

Fill keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in “project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential_keys.h” by xxxxxxxx-certifiacte.pem and xxxxxxxx-private.pem.key.


Download certificates and keys

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

Device certificate

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

Device certificate
2e7b95c25ff...te.pem.crt

Key files

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

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

Public key file
2e7b95c25ffdfb674406d61...a25f21a-public.pem.key

Private key file
2e7b95c25ffdfb674406d61...25f21a-private.pem.key

Root CA certificates

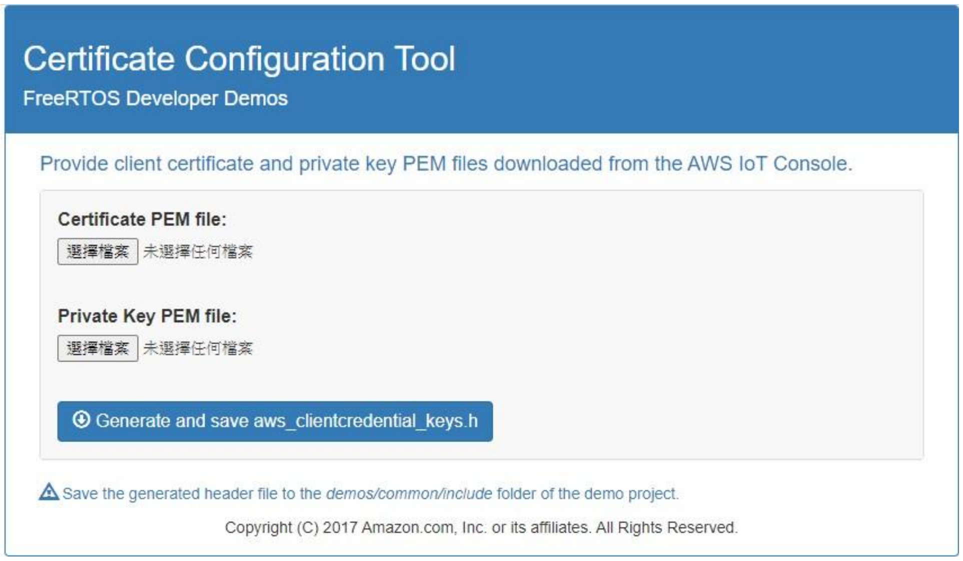
Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

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

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

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

It can done by CertificateConfigurator.html and it can be found here: https://yona75.github.io/credformatter/


Certificate Configuration Tool

FreeRTOS Developer Demos

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

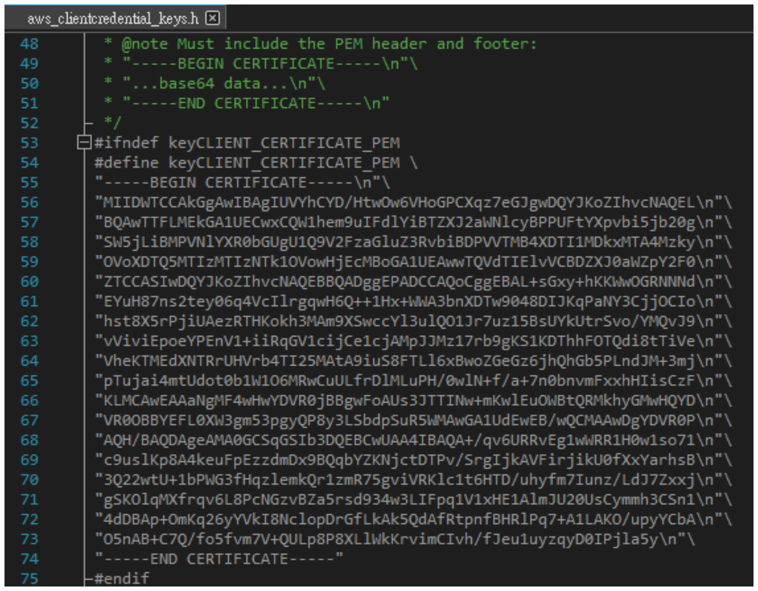
Certificate PEM file:
選擇檔案 未選擇任何檔案

Private Key PEM file:
選擇檔案 未選擇任何檔案

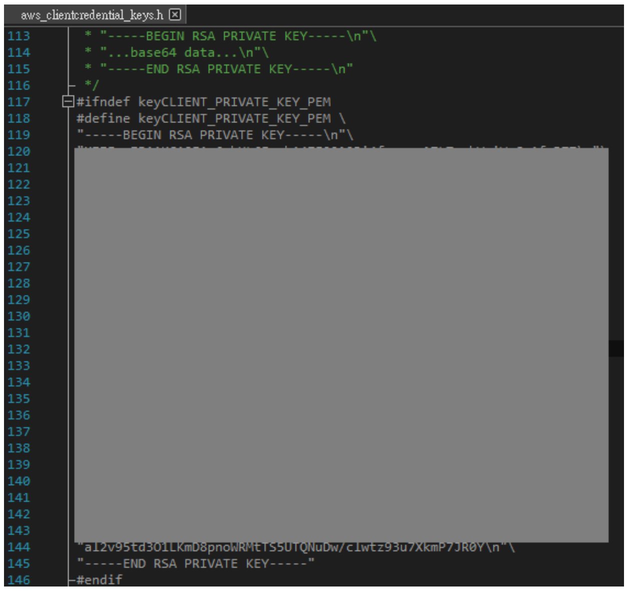
Save the generated header file to the *demos/common/include* folder of the demo project.

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

Final aws_clientcredential_keys.h overview.



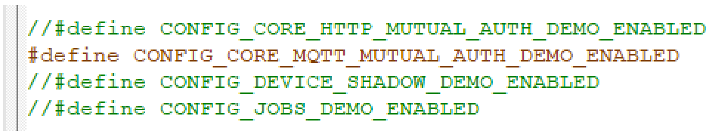
```
 * @note Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"\
 * "...base64 data...\n"\
 * "-----END CERTIFICATE-----\n"
 */
#ifndef keyCLIENT_CERTIFICATE_PEM
#define keyCLIENT_CERTIFICATE_PEM \
"-----BEGIN CERTIFICATE-----\n"\
"MIIDWTCCAkGgAwIBAgIUVYhCYD/HtwOw6VHoGPCXqz7eGJgwDQYJKoZIhvcNAQEL\n"\
"BQAwTTFLMEkGA1UECwxCQW1hem9uIFdlYiBTZXJ2aWNlcyBPPUFtYXpvbi5jb20g\n"\
"SW5jLiBMPVNlYXR0bGUgU1Q9V2FzaGluZ3RvbiBDPVVTMB4XDTI1MDkxMTA4Mzky\n"\
"OVoXDTQ5MTIzMTIzNTk1OVowHjEcMBoGA1UEAwwTQVdTIElvVCBDZXJ0aWZpY2F0\n"\
"ZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL+sGxy+hKKWwOGRNNNd\n"\
"EYuH87ns2tey06q4VcIlrgqwH6Q++1Hx+WWA3bnXDTw9048DIJKqPaNY3CjjOCIo\n"\
"hst8X5rPjiUAezRTHKokh3MAm9XSwccYl3ulQO1Jr7uz15BsUYkUtrSvo/YMQvJ9\n"\
"vViviEpoeYPEnV1+iiRqGV1cijCe1cjAMpJJMz17rb9gKS1KDThhFOTQdi8tTiVe\n"\
"VheKTMEdXNTRrUHVrb4TI25MAtA9iuS8FTLl6xBwoZGeGz6jhQhGb5PLndJM+3mj\n"\
"pTujai4mtUdot0b1W1O6MRwCuULfrDlMLuPH/0wlN+f/a+7n0bnvmFxxhHIisCzF\n"\
"KLMCAwEAAaNgMF4wHwYDVR0jBBgwFoAUs3JTTINw+mKwlEuOWBtQRMkhyGMwHQYD\n"\
"VR0OBBYEFL0XW3gm53pgyQP8y3LSbdpSuR5WMAwGA1UdEwEB/wQCMAAwDgYDVR0P\n"\
"AQH/BAQDAgeAMA0GCSqGSIb3DQEBCwUAA4IBAQA+/qv6URRvEg1wWRR1H0w1so71\n"\
"c9uslKp8A4keuFpEzzdmDx9BQqbYZKNjctDTPv/SrgIjkAVFirjikU0fXxYarhsB\n"\
"3Q22wtU+1bPWG3fHqzlemkQr1zmR75gviVRKlc1t6HTD/uhyfm7Iunz/LdJ7Zxxj\n"\
"gSKOlqMXfrqv6L8PcNGzvBZa5rsd934w3LIFpq1V1xHE1AlmJU20UsCymmh3CSn1\n"\
"4dDBAp+OmKq26yYVkI8NclopDrGfLkAk5QdAfRtpnfBHRlPq7+A1LAKO/upyYCbA\n"\
"O5nAB+C7Q/fo5fvm7V+QULp8P8XLlWkKrvimCIvh/fJeu1uyzqyD0IPjla5y\n"\
"-----END CERTIFICATE-----"
#endif
```



```
 * "-----BEGIN RSA PRIVATE KEY-----\n"\
 * "...base64 data...\n"\
 * "-----END RSA PRIVATE KEY-----\n"
 */
#ifndef keyCLIENT_PRIVATE_KEY_PEM
#define keyCLIENT_PRIVATE_KEY_PEM \
"-----BEGIN RSA PRIVATE KEY-----\n"\
[redacted]
"-----END RSA PRIVATE KEY-----"
#endif
```

2.4.2 Enable FreeRTOS demo on AmebaPro2

For example, if you would like to run MQTT mutual authentication demo, please find aws_demo_config.h in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/vendors/realtek/boards/amebaPro2/aws_demos/config_files" and enable **CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED**



```
//#define CONFIG_CORE_HTTP_MUTUAL_AUTH_DEMO_ENABLED
#define CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
//#define CONFIG_DEVICE_SHADOW_DEMO_ENABLED
//#define CONFIG_JOBS_DEMO_ENABLED
```

Now you can start to compile AmebaPro2 Amazon FreeRTOS project !

3 Compile AmebaPro2 Amazon FreeRTOS

3.1 Compile Program with GCC Toolchain

Run following commands to build the image with option `-DEXAMPLE=amazon_freertos`

```
$ cd project/realtek_amebapro2_v0_example/GCC-RELEASE
$ mkdir build
$ cd build
$ cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DEXAMPLE=amazon_freertos
$ cmake --build . --target flash -j4
```

After successfully build, there should be an image file **flash_ntz.bin** located in "build/" directory.

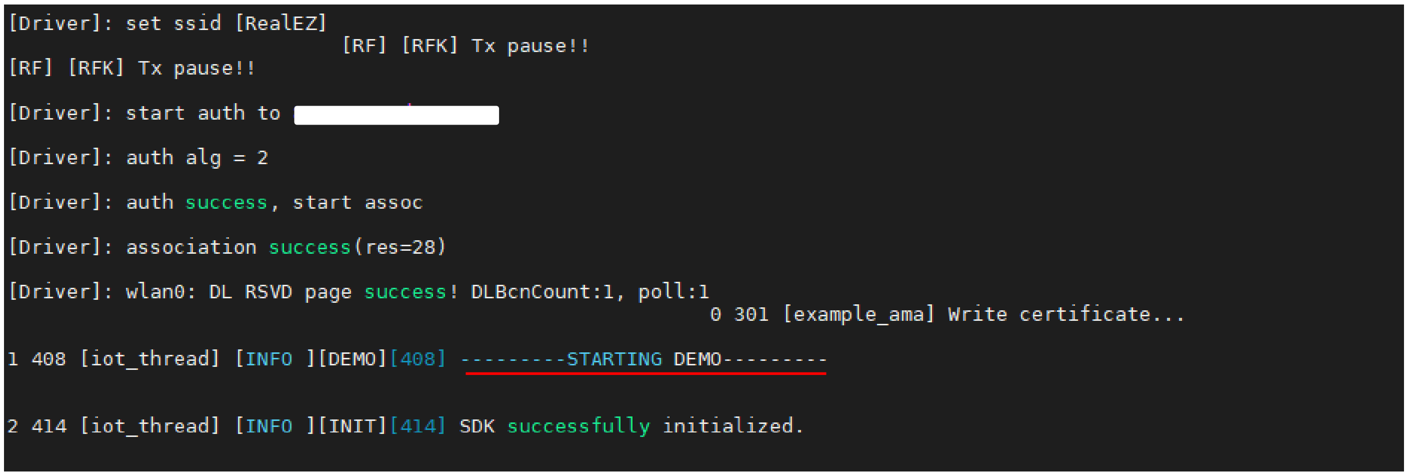
3.2 Download image to AmebaPro2

Use image tool to download the image to AmebaPro2.

4 MQTT Demo

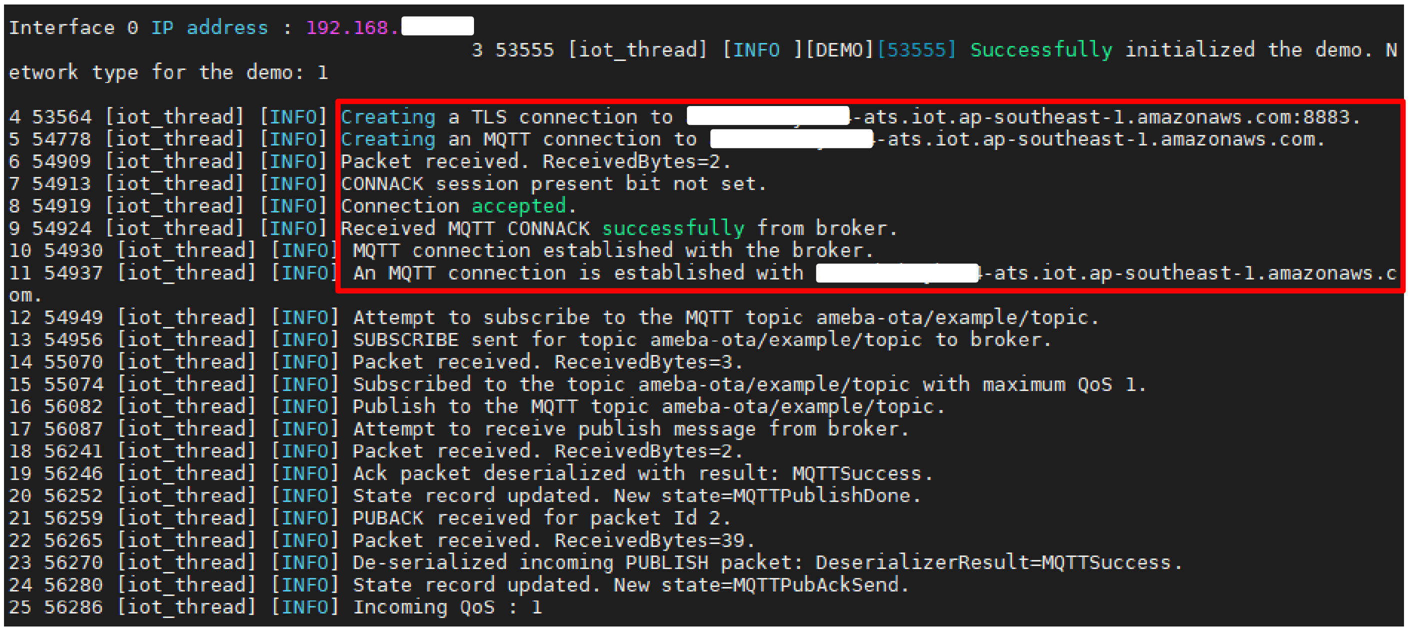
4.1 Run MQTT Demo

Default setting of SDK are enable MQTT demo. Once the AmebaPro2 EVB has rebooted, the application will automatically start run MQTT demo and communicate to IoT Core.



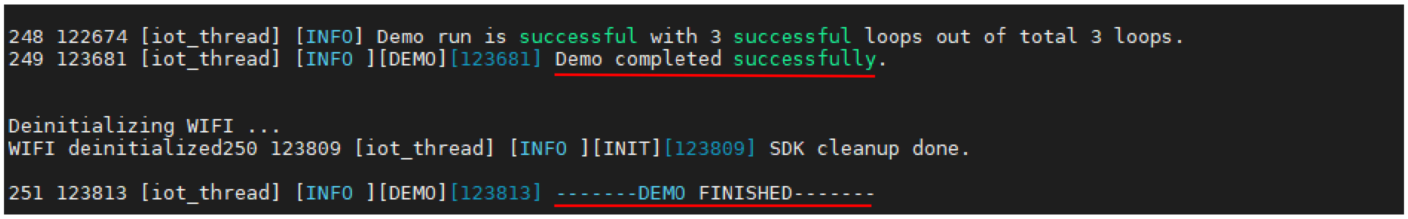
```
[Driver]: set ssid [RealEZ]
                            [RF] [RFK] Tx pause!!
[RF] [RFK] Tx pause!!

[Driver]: start auth to [redacted]

[Driver]: auth alg = 2

[Driver]: auth success, start assoc

[Driver]: association success(res=28)

[Driver]: wlan0: DL RSVD page success! DLBcnCount:1, poll:1
                                                           0 301 [example_ama] Write certificate...

1 408 [iot_thread] [INFO ][DEMO][408] ---------STARTING DEMO---------


2 414 [iot_thread] [INFO ][INIT][414] SDK successfully initialized.
```

...



```
Interface 0 IP address : 192.168.[redacted]
                                              3 53555 [iot_thread] [INFO ][DEMO][53555] Successfully initialized the demo. N
etwork type for the demo: 1

4 53564 [iot_thread] [INFO] Creating a TLS connection to [redacted]-ats.iot.ap-southeast-1.amazonaws.com:8883.
5 54778 [iot_thread] [INFO] Creating an MQTT connection to [redacted]-ats.iot.ap-southeast-1.amazonaws.com.
6 54909 [iot_thread] [INFO] Packet received. ReceivedBytes=2.
7 54913 [iot_thread] [INFO] CONNACK session present bit not set.
8 54919 [iot_thread] [INFO] Connection accepted.
9 54924 [iot_thread] [INFO] Received MQTT CONNACK successfully from broker.
10 54930 [iot_thread] [INFO] MQTT connection established with the broker.
11 54937 [iot_thread] [INFO] An MQTT connection is established with [redacted]-ats.iot.ap-southeast-1.amazonaws.c
om.
12 54949 [iot_thread] [INFO] Attempt to subscribe to the MQTT topic ameba-ota/example/topic.
13 54956 [iot_thread] [INFO] SUBSCRIBE sent for topic ameba-ota/example/topic to broker.
14 55070 [iot_thread] [INFO] Packet received. ReceivedBytes=3.
15 55074 [iot_thread] [INFO] Subscribed to the topic ameba-ota/example/topic with maximum QoS 1.
16 56082 [iot_thread] [INFO] Publish to the MQTT topic ameba-ota/example/topic.
17 56087 [iot_thread] [INFO] Attempt to receive publish message from broker.
18 56241 [iot_thread] [INFO] Packet received. ReceivedBytes=2.
19 56246 [iot_thread] [INFO] Ack packet deserialized with result: MQTTSuccess.
20 56252 [iot_thread] [INFO] State record updated. New state=MQTTPublishDone.
21 56259 [iot_thread] [INFO] PUBACK received for packet Id 2.
22 56265 [iot_thread] [INFO] Packet received. ReceivedBytes=39.
23 56270 [iot_thread] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
24 56280 [iot_thread] [INFO] State record updated. New state=MQTTPubAckSend.
25 56286 [iot_thread] [INFO] Incoming QoS : 1
```

…

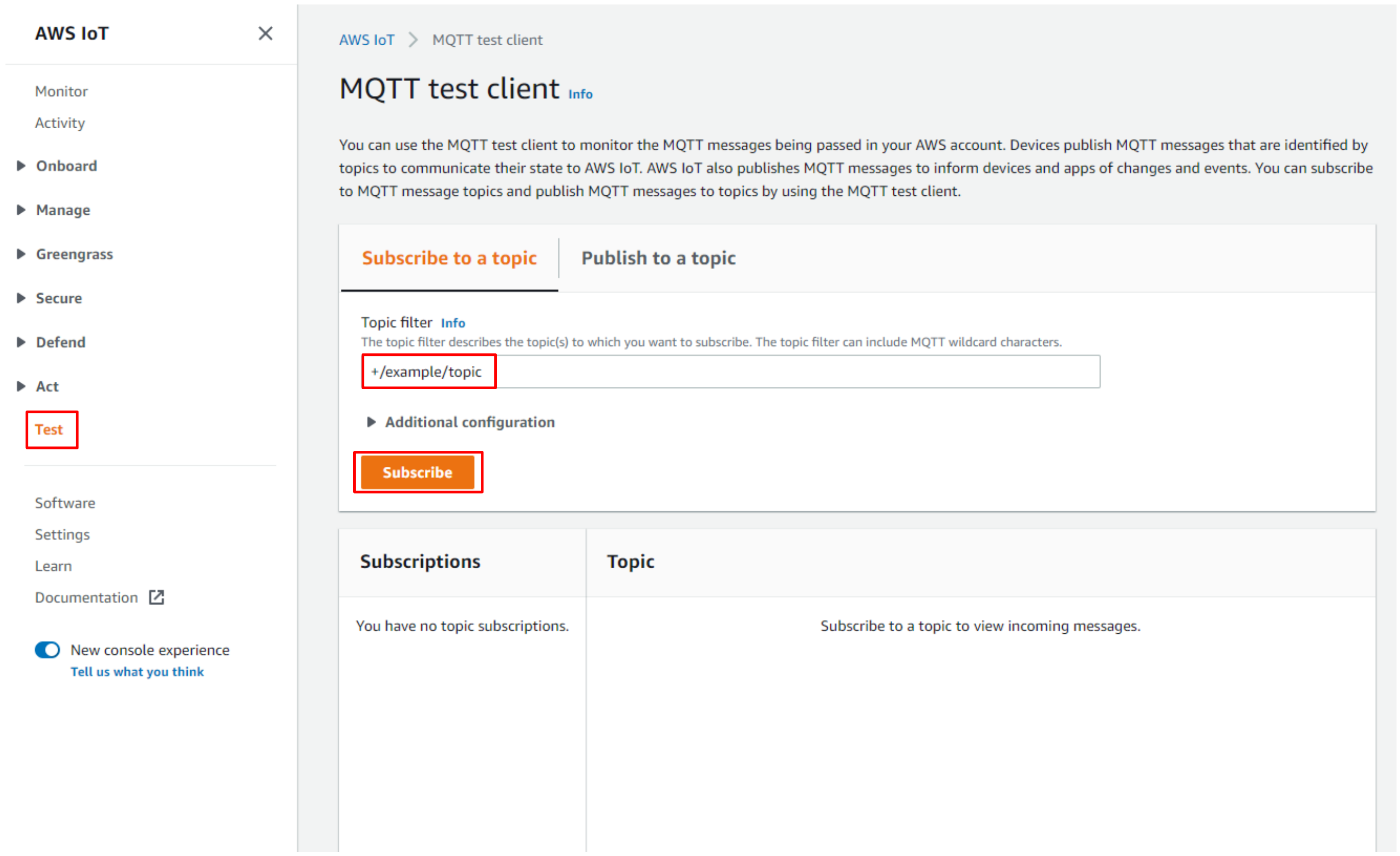


```
248 122674 [iot_thread] [INFO] Demo run is successful with 3 successful loops out of total 3 loops.
249 123681 [iot_thread] [INFO ][DEMO][123681] Demo completed successfully.


Deinitializing WIFI ...
WIFI deinitialized250 123809 [iot_thread] [INFO ][INIT][123809] SDK cleanup done.

251 123813 [iot_thread] [INFO ][DEMO][123813] -------DEMO FINISHED-------
```

4.2 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.



MQTT test client Info

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

Subscribe to a topic | Publish to a topic

Topic filter Info

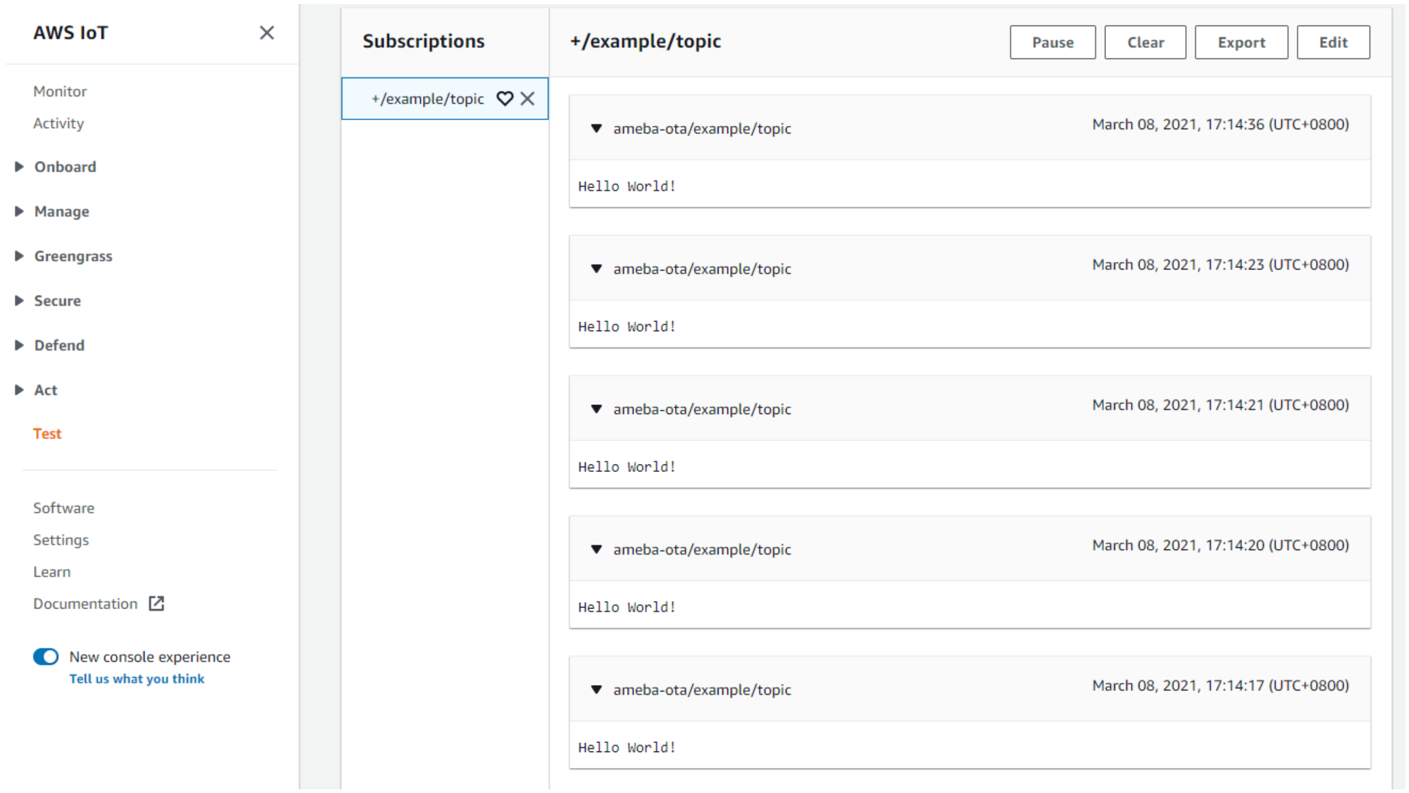
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

+/example/topic

Additional configuration

Subscribe

Subscriptions	Topic
You have no topic subscriptions.	Subscribe to a topic to view incoming messages.



Subscriptions: +/example/topic

+/example/topic

```
ameba-ota/example/topic    March 08, 2021, 17:14:36 (UTC+0800)
Hello World!

ameba-ota/example/topic    March 08, 2021, 17:14:23 (UTC+0800)
Hello World!

ameba-ota/example/topic    March 08, 2021, 17:14:21 (UTC+0800)
Hello World!

ameba-ota/example/topic    March 08, 2021, 17:14:20 (UTC+0800)
Hello World!

ameba-ota/example/topic    March 08, 2021, 17:14:17 (UTC+0800)
Hello World!
```

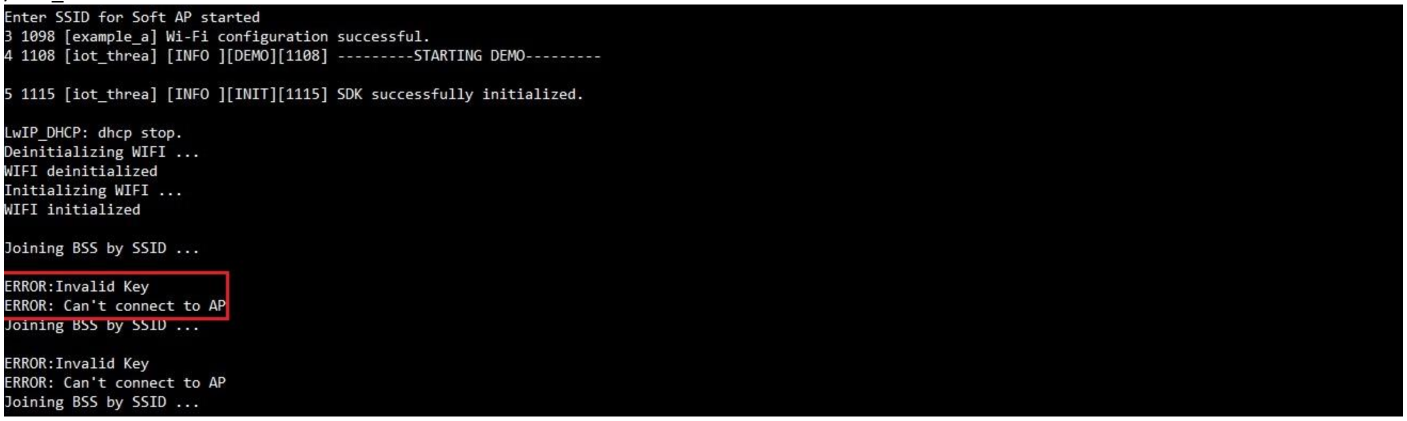
5 Troubleshooting

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.

5.1 ERROR: Invalid Key

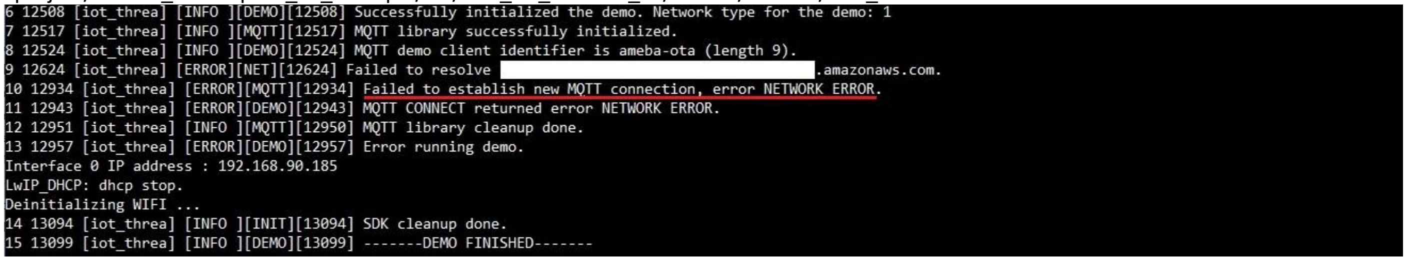
Please check **WIFI_SSID** and **WIFI_PASSWORD** in in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include /aws_clientcredential.h"



```
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 ...
```

5.2 Failed to establish new MQTT connection

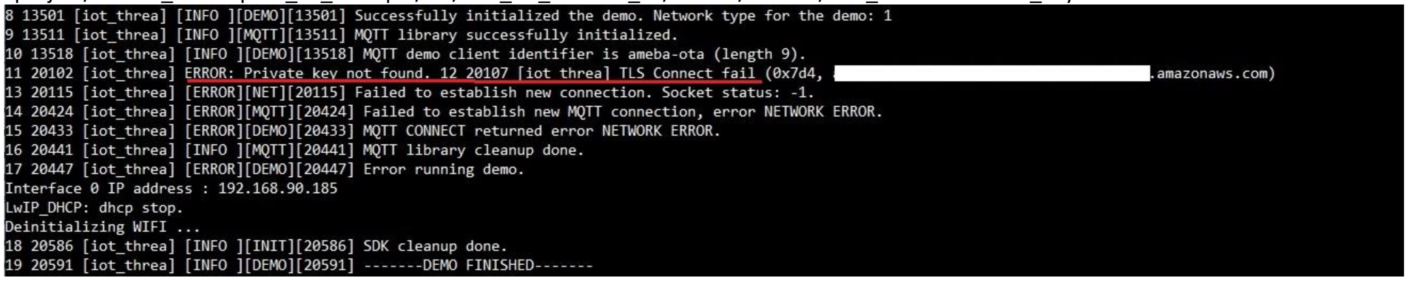
Please check **clientcredentialMQTT_BROKER_ENDPOINT** in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential.h"



```
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-------
```

5.3 TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/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-------
```