



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

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 life-critical 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 non-compliance 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.

1 Configure AWS IoT Core

1.1 Set up your AWS account and Permissions

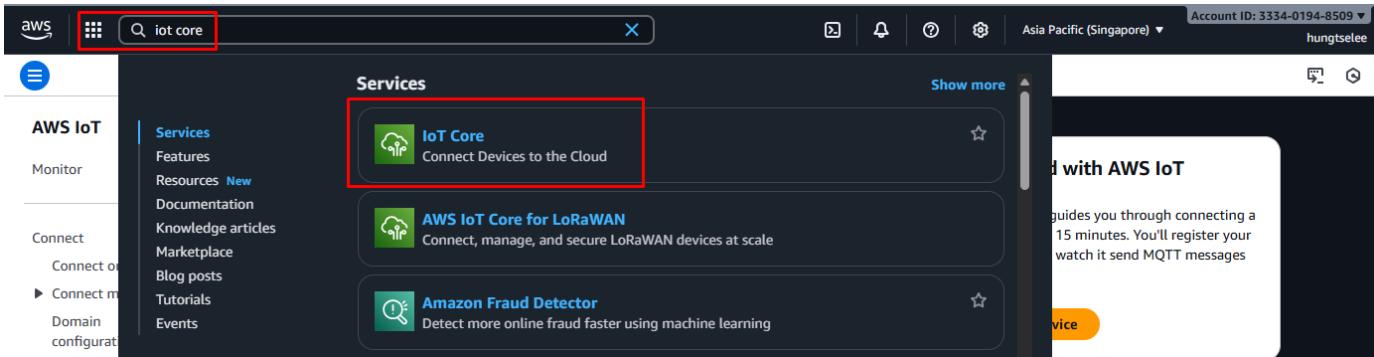
Refer to the instructions at Set up your AWS Account <https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html>. Follow the steps outlined in these sections to create your account and a user and get started:

- 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.

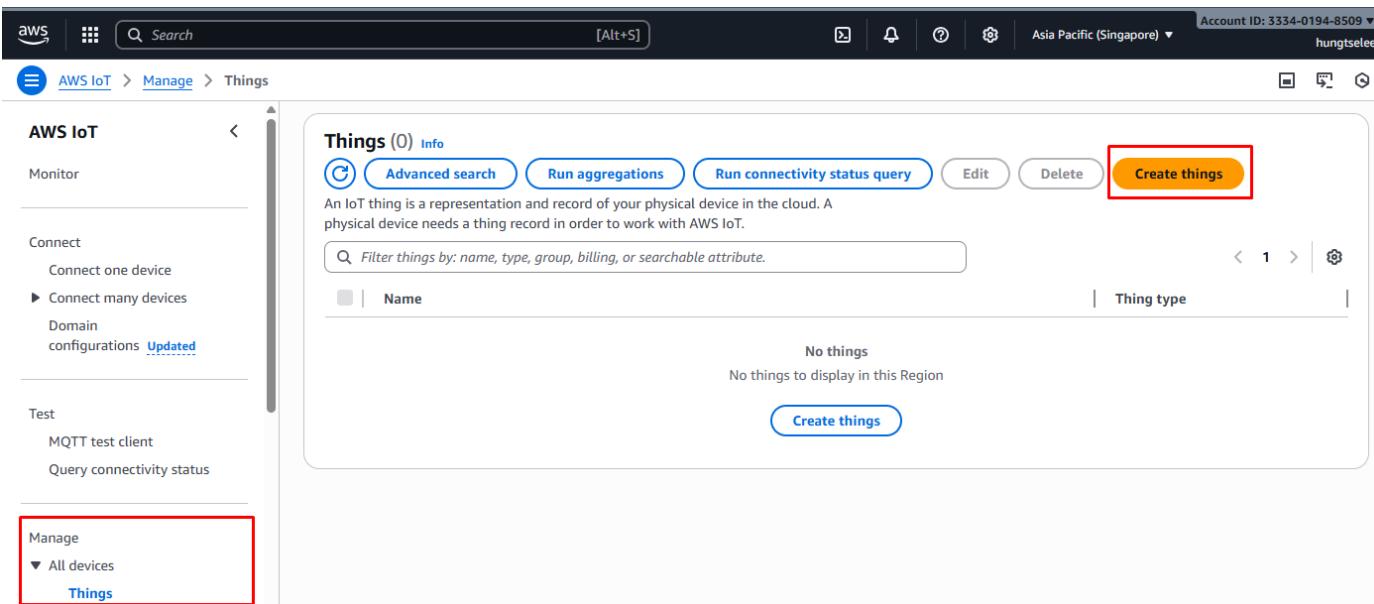
1.2 Create a New Device

First, go to IoT Core webpage



The screenshot shows the AWS IoT Core service page. The search bar at the top contains the text "iot core". In the "Services" section, there is a red box around the "IoT Core" item, which is described as "Connect Devices to the Cloud". Other services listed include "AWS IoT Core for LoRaWAN" and "Amazon Fraud Detector". The left sidebar shows navigation options like "Monitor", "Connect", and "Test". A modal window titled "Get started with AWS IoT" is partially visible on the right.

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



The screenshot shows the AWS IoT Core "Things" management page. The left sidebar has a red box around the "Manage" section, which is expanded to show "All devices" and "Things". The main area displays a table with the message "No things" and "No things to display in this Region". A red box highlights the "Create things" button. The top navigation bar shows the search bar "Search" and the AWS logo.

The screenshot shows the AWS IoT 'Create things' interface. On the left sidebar, under 'AWS IoT', there are sections for 'Monitor', 'Connect' (with 'Connect one device' and 'Connect many devices' options), 'Domain configurations' (marked as 'Updated'), and 'Test' (with 'MQTT test client'). The main content area is titled 'Create things' with a sub-section 'Number of things to create'. It contains two options: 'Create single thing' (selected, highlighted with a red box) and 'Create many things'. Below each option is a brief description. At the bottom right of the main content area are 'Cancel' and 'Next' buttons.

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

The screenshot shows the 'Create single thing' step 1 of the AWS IoT process. The left sidebar is identical to the previous screenshot. The main content area is titled 'Specify thing properties' with a sub-section 'Thing properties'. It shows a 'Thing name' input field containing 'TestDevice', which is also highlighted with a red box. Below the input field is a placeholder text: 'Enter a unique name containing only letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.' To the right of the input field, there is a section for 'Additional configurations' with options for 'Thing type - optional' and 'Searchable thing attributes - optional'.

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

Configure device certificate - optional

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.

Cancel Previous Next

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

Attach policies to certificate - optional

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.

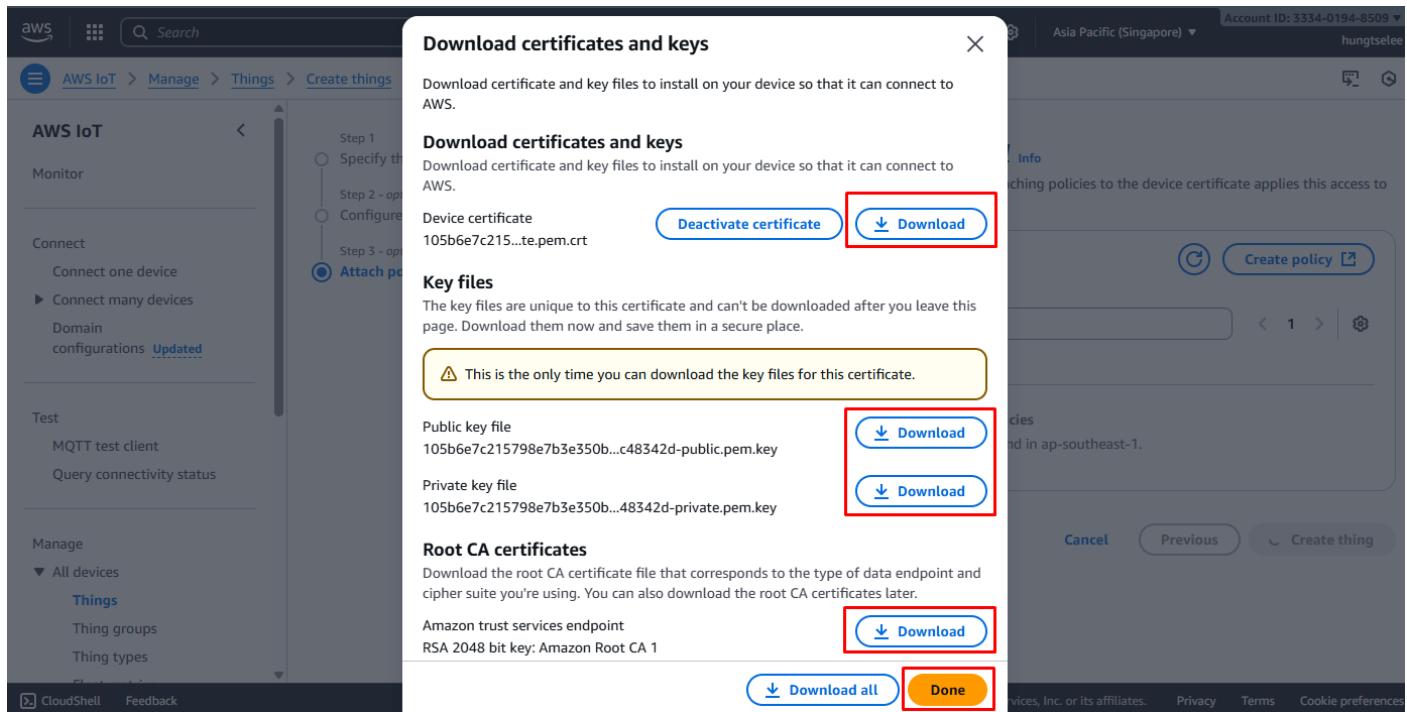
Filter policies

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

Create policy

Cancel Previous Create thing

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



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"

The screenshot shows the AWS IoT Policies page. On the left, there is a navigation sidebar with various options like Greengrass devices, Software packages, Remote actions, Message routing, Retained messages, Security (which is expanded), and others. Under Security, the 'Policies' option is selected and highlighted with a red box. In the main content area, it says 'AWS IoT policies (0) Info'. It explains that AWS IoT policies allow control access to the AWS IoT Core data plane operations. Below this is a search bar labeled 'Find policies' and a table header 'Policy name'. A message states 'No policies' and 'You don't have any AWS IoT policies in ap-southeast-1.' At the bottom right of the main area is a blue 'Create' button. Above the main area, there are standard AWS navigation icons and account information.

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

The screenshot shows the 'Create policy' wizard. The left sidebar is identical to the previous screenshot, with the 'Policies' option selected. In the main area, it says 'AWS IoT Core supports named policies so that many identities can reference the same policy document.' Below this is a 'Policy name' input field containing 'TestDevice_Policy', which is also highlighted with a red box. There is a note below the input field: '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.' Below this is a 'Tags - optional' section. At the bottom of this section is a tab bar with 'Policy statements' (which is selected and highlighted with a blue underline) and 'Policy examples'. In the 'Policy statements' tab, there is a 'Policy document' section with an 'Info' link. It says '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.' Below this is a table with three columns: 'Policy effect' (dropdown menu showing 'Allow'), 'Policy action' (dropdown menu showing '*'), and 'Policy resource' (dropdown menu showing '*'). This entire row is highlighted with a red box. At the bottom of this section is a 'Remove' button and a 'Create new statement' button. At the very bottom right of the main area are 'Cancel' and 'Create' buttons, with the 'Create' button highlighted with a red box.

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.

This screenshot shows the AWS IoT Things page. On the left sidebar, under the 'Manage' section, the 'Things' option is selected and highlighted with a red box. In the main content area, there is a summary for 'Things (1)'. Below it, a table lists one item: 'TestDevice'. The 'Name' column contains 'TestDevice', and the 'Thing type' column is empty. A red box highlights the 'TestDevice' entry in the table.

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

This screenshot shows the AWS IoT TestDevice page for 'TestDevice'. The left sidebar shows the 'Manage' section with 'Things' selected and highlighted with a red box. In the main content area, the 'Thing details' section shows the device's name, ARN, and other metadata. Below it, the navigation tabs include 'Attributes', 'Certificates' (which is selected and highlighted with a red box), 'Thing groups', 'Command history', 'Device Shadows', 'Activity', and 'Packages and vers'. Under the 'Certificates' tab, there is a table titled 'Certificates (1)'. It shows one certificate entry: '105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d'. The 'Status' column shows 'Active' with a green checkmark, and the 'Certificate to thing association' column shows 'Non-Exclusive'. A red box highlights the certificate ID in the table.

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)

Actions

- Activate
- Deactivate
- Revoke
- Accept transfer
- Reject transfer
- Start transfer
- Attach policy**
- Attach to things
- Download
- Delete

Details

Attach policies to the certificate

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

Choose AWS IoT policy

TestDevice_Policy

Cancel

Attach policies

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

The screenshot shows the AWS IoT Settings page. A green success message at the top states: "Successfully attached the policy TestDevice_Policy to certificate 105b6e7c215798e7b3e350ba066f0ae471680fc24e71fd9e052b5be31c48342d." Below this, two notifications are displayed:

- 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.
- 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.

A red box highlights the "View domain configurations" button next to the second notification. On the left sidebar, the "Settings" option is also highlighted with a red box.

The screenshot shows the AWS IoT Connect Domain configurations page. The left sidebar includes the "Settings" option, which is highlighted with a red box. The main content area is titled "Domain configurations" and describes a custom endpoint for device connections. It includes three steps:

- 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.

The "Domain configurations" table lists one entry:

Name	Domain name	Broker Endpoint	Status	Date updated
iot:Data-ATS	a1b45cssuu69zn-ats.iot.ap-southeast-1.amazonaws.com	<input checked="" type="checkbox"/>	Enabled	September 11

2.3 Get Thing Name

The screenshot shows the AWS IoT Things management interface. On the left, there's a sidebar with 'AWS IoT' selected under 'Manage'. The main area shows a list titled 'Things (1) Info'. It has buttons for 'Advanced search', 'Run aggregations', and 'Run connectivity status query'. There are 'Edit', 'Delete', and 'Create things' buttons. A note says '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.' Below this is a search bar with 'Filter things by: name, type, group, billing, or searchable attribute.' A table lists one item: 'Name' (TestDevice) and 'Thing Name' (TestDevice). The 'Thing Name' column is highlighted with a red box.

2.4 Setup IoT Core Information with AmebaPro2 Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in
"project/realtek_ameapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential.h"

```
#define clientcredentialMQTT_BROKER_ENDPOINT      "xxxxxxxxxxxxxx.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_ameapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-certifiacte.pem
and xxxxxxxx-private.pem.key.

The screenshot shows two overlapping AWS console pages. The left page is titled 'Download certificates and keys' under 'AWS IoT > Manage > Connect'. It displays a certificate named '2e7b95c25ff...te.pem.crt' with a red underline. Below it are sections for 'Key files' (public and private key files) and 'Root CA certificates' (Amazon trust services endpoint). A 'Done' button is at the bottom right. The right page is partially visible, showing policy names like 'd8fc54d89924e21cb23b5a1f75f_policy' and '27121d4607413623d775a2bd6aa_aa_policy'.

It can done by [CertificateConfigurator.html](#) and it can be found here: <https://yona75.github.io/credformatter/>

The screenshot shows the 'Certificate Configuration Tool' interface. It has a blue header with the title 'Certificate Configuration Tool' and 'FreeRTOS Developer Demos'. Below the header, a message says 'Provide client certificate and private key PEM files downloaded from the AWS IoT Console.' There are two input fields: 'Certificate PEM file:' and 'Private Key PEM file:', both currently showing '未選擇任何檔案'. At the bottom is a blue button labeled 'Generate and save aws_clientcredential_keys.h'. Below the button, a note says '⚠ Save the generated header file to the demos/common/include folder of the demo project.' and 'Copyright (C) 2017 Amazon.com, Inc. or its affiliates. All Rights Reserved.'

Final aws_clientcredential_keys.h overview.

```
aws_clientcredential_keys.h [x]
48  * @note Must include the PEM header and footer:
49  * "----BEGIN CERTIFICATE----\n"
50  * ....base64 data...\n"
51  * "----END CERTIFICATE----\n"
52  */
53 #ifndef keyCLIENT_CERTIFICATE_PEM
54 #define keyCLIENT_CERTIFICATE_PEM \
55 "-----BEGIN CERTIFICATE-----\n"
56 "MIIDWTCCAkGgAwIBAgIUVVhCYD/Htw0w6H0gPDXQz7eGJgwDQYJKoZIhvcNAQEL\n"
57 "BQAwtTFLMEkGA1UECxwCQW1hem9UFDlY1BTZXJ2aWlNcyBPPUfTYXpvbi5jb20g\n"
58 "S5jLbMPVmlYXR0bGUgU109V2FzaGlz3Rvb1BDPVVTMB4XDIIiMDkxMTA4Mzky\n"
59 "OVoXDTQ5MTizNTkLOvwhjEcMBoGA1UEAwwTQVdTIElvVCBDZXJ0aWZpY2F0\n"
60 "ZTCASiDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL+sGxy+hKKW0GRNNnd\n"
61 "EYuH87ns2tey06q4VcIlrgqwh6Q++1Hx+NWA3bnxDTw9048DIJKqPaNY3CjjOCIO\n"
62 "hst8XrjiUAezRTHKokh3Am9XSwccyl3u1Q01j7uz15BsUYKutrsvo/YNQvJ9\n"
63 "viviEpoeYEPnV1+iiRqGV1cijCe1cjaMpJmz17rb9gK51KDThhFOTQd18t1Ve\n"
64 "VheKTMedXNTrUHVrb4TI25MA9ius8FTLl6x8woZGeGz6jhQhb5PLndJM+3mj\n"
65 "ptUjai4mtUdotb1w106MRwCuUlfrD1MLuPH/0wLN+f/a+7n0bnvmFxHHIisCzf\n"
66 "KLMCAwEAAalGMF4whwYDVR0jBBgwfoAUs33TTINw+mKwlEu0BtQRMchyGMwHQYD\n"
67 "VR00BYEFL0XW3gm53pgyQP8y3LSbdpSuR5NWIAwGA1udEWB/wQCMAAwBgYDVR0P\n"
68 "AQH/BAQDAgeAMA0GCSqGSIb3DQEBCwUA41BAQAt/qv6URRvEg1wRR1H0wls071\n"
69 "c9us1Kp8A4keurpEzzdmDx9BQqbYZKNjctDTPv/SrgIjkAVfirjikUfxXyarhsB\n"
70 "3Q2wtU+1bPG3fHqzlemkQr1zmR75gvivRKlc1t6HTD/uhf7lunz/LdJ7xxcj\n"
71 "gSK01qMXfrqv6L8PcNGzvBZa5rsd934w3LIFpq1V1xE1almJU20UsCymmh3Cs1\n"
72 "4dDBAp+0mKq26YVvkI8Ncl0pDrGfLkAk50dAfRtpnfBHR1Pq7+A1LAKO/upyYcbA\n"
73 "05nAB+C7Q/f05fvm7V+QUlP8XllWkKrvimCivh/fJeuyzqyD0IPjla5y\n"
74 "-----END CERTIFICATE-----"
75 #endif
```

```
aws_clientcredential_keys.h [x]
113  * "----BEGIN RSA PRIVATE KEY----\n"
114  * ....base64 data...\n"
115  * "----END RSA PRIVATE KEY----\n"
116  */
117 #ifndef keyCLIENT_PRIVATE_KEY_PEM
118 #define keyCLIENT_PRIVATE_KEY_PEM \
119 "-----BEGIN RSA PRIVATE KEY----\n"
120 "-----END RSA PRIVATE KEY----\n"
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146 "a12v95td301LkmD8pnoWJM7TS5U1QNuDw/cIwtz93u/Xkmp7JR0Y\n"
"-----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_amebaipro2_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 : 192.168.1.100
3 53555 [iot_thread] [INFO ][DEMO][53555] Successfully initialized the demo. Network 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.com.
...
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.

The screenshot shows the AWS IoT console with the 'Test' menu item highlighted by a red box. The main content area is titled 'MQTT test client' and contains a 'Subscribe to a topic' section with a red box around the 'Topic filter' input field containing '+/example/topic'. Below it is a 'Publish to a topic' section. A table below lists 'Subscriptions' and 'Topic' with a note about having no subscriptions.

AWS IoT

Monitor

Activity

▶ Onboard

▶ Manage

▶ Greengrass

▶ Secure

▶ Defend

▶ Act

Test

Software

Settings

Learn

Documentation ▾

New console experience
Tell us what you think

AWS IoT > MQTT test client

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 Info **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.

The screenshot shows the AWS IoT Subscriptions interface. On the left, a sidebar lists navigation options: Monitor, Activity, Onboard, Manage, Greengrass, Secure, Defend, Act, and Test (which is currently selected). Below these are links for Software, Settings, Learn, and Documentation. At the bottom, there's a feedback link: "New console experience" with a "Tell us what you think" button.

The main content area has tabs for "Subscriptions" and "+/example/topic". The "+/example/topic" tab is active, showing five received messages:

- March 08, 2021, 17:14:36 (UTC+0800) - ameba-ota/example/topic: Hello World!
- March 08, 2021, 17:14:23 (UTC+0800) - ameba-ota/example/topic: Hello World!
- March 08, 2021, 17:14:21 (UTC+0800) - ameba-ota/example/topic: Hello World!
- March 08, 2021, 17:14:20 (UTC+0800) - ameba-ota/example/topic: Hello World!
- March 08, 2021, 17:14:17 (UTC+0800) - ameba-ota/example/topic: Hello World!

At the top right of the main content area are buttons for Pause, Clear, Export, and Edit.

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-----
```