

## Overview

=====

This demo demonstrates the follow function:

1. There are two parts: Android APK, AWS Wi-Fi provisioning demo (running on RT1060).
2. With Android APK running on the smart phone, the end user could view and config the Wi-Fi APs via BLE and view the Wi-Fi setting log via AWS.
3. By default, AWS Wi-Fi provisioning demo will start advertising if the Wi-Fi AP is not configured and wait the Wi-Fi AP configuration. After connected to the Android APK, the demo will execute the request from cellphone and reply the response. When the Wi-Fi Ap is configured, the Shadow demo will connect to the AWS via Wi-Fi and publish the configured Wi-Fi AP information.

Note: This demo could NOT function with the default setting provided in SDK package because an AWS account is mandatory to run to the demo, the end users must create their own AWS account and configure the IoT Console before the functionality of the demo could be used. Also, some information specified by the end customers, like Thing name, broker endpoint, etc., must be updated accordingly before the demo would work. Check "Prepare the Demo" to get the detailed guidance of the configuration steps.

## Board settings

=====

Refer to Hardware Rework Guide for MIMXRT1060-EVK and AW-AM457-uSD

## Prepare the Demo

=====

Before running the demo, some steps should be followed to configure AWS IoT Console:

1. Create AWS account: <https://console.aws.amazon.com/console/home>
2. Configure device in the AWS IoT Console as follow:
  - (1) Create one policy as the follow steps (for example: the policy name is "aws\_wifi\_provisioning\_policy").
    - a) Click "Policies"

▼ Secure

Certificates

**Policies**

CAs

Role Aliases

Authorizers

- b) Click "Create"

Policies

Create

- c) Fill the policy name and switch to "Advanced mode" fill the follow content.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

Name

aws\_wifi\_provisioning\_policy

### Add statements

Policy statements define the types of actions that can be performed by a resource.

Basic mode

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "iot:*",
7       "Resource": "*"
8     }
9   ]
10 }
```

Add statement

Create

(2) Create one "Thing" as follow (for example: the name is "aws\_wifi\_provisioning"):

a) Click "Things"

### ▼ Manage

Things

Types

Thing groups

Billing groups

Jobs

Tunnels

b) Click "Create"

Things

Create

c) Click "Create a single thing"

Register a single AWS IoT thing  
Create a thing in your registry

Create a single thing

- d) Fill the name and click “Next”

Name

aws\_wifi\_provisioning

### Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected

Create a type

### Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

Groups /

Create group Change

### Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key

Provide an attribute key, e.g. Manufacturer

Value

Provide an attribute value, e.g. Acme-Corporation

Clear

Add another

Show thing shadow

Cancel

Back

Next

- e) Click “Create certificate”

### One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

- f) Download keys, click “Active” and then click “Attach a policy”

A certificate for this thing	490291780a.cert.pem	<a href="#">Download</a>
A public key	490291780a.public.key	<a href="#">Download</a>
A private key	490291780a.private.key	<a href="#">Download</a>

You also need to download a root CA for AWS IoT:

A root CA for AWS IoT [Download](#)

[Activate](#)

Cancel Done [Attach a policy](#)

g) Select the policy and click "Register Thing"

☒ aws\_wifi\_provisioning\_policy [View](#)

1 policy selected [Register Thing](#)

3. Configure the demo's codes.

(1) Open `aws_clientcredential.h`, Fill `clientcredentialMQTT_BROKER_ENDPOINT`, `clientcredentialIOT_THING_NAME` to configure the AWS thing.

a) The `clientcredentialMQTT_BROKER_ENDPOINT` can be got as follow.

- Secure
  - Certificates
  - Policies
  - CAs
  - Role Aliases
  - Authorizers
- Defend
- Act
- Test
- Software
  - [Settings](#)

### Custom endpoint

This is your custom endpoint that allows you to connect to AWS IoT. Each of you This is also an important property to insert when using an MQTT client or the AV

Your endpoint is provisioned and ready to use. You can now start to publish a

Endpoint

`ats.iot.us-west-2.amazonaws.com`

### Logs

You can enable AWS IoT to log helpful information to CloudWatch Logs. As mess broker and the rules engine, AWS IoT logs process events which can be helpful in

Role

b) The `clientcredentialIOT_THING_NAME` is the Thing Name that is created in above steps.

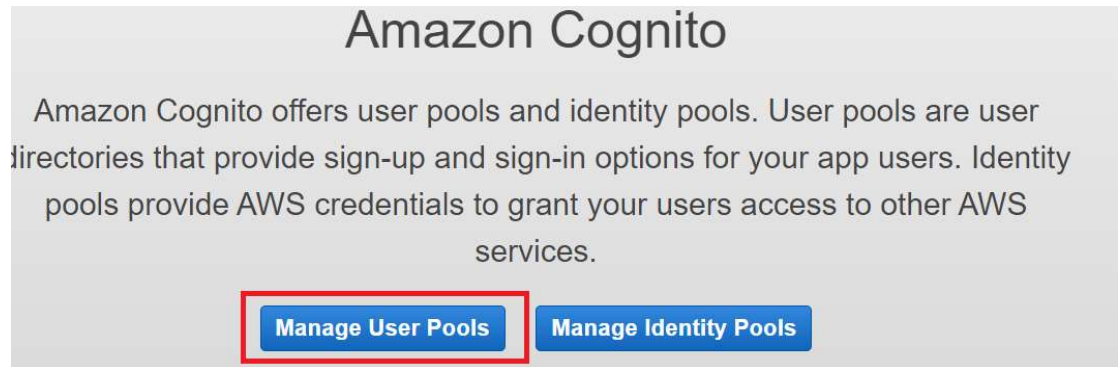
For example:

```
#define clientcredentialMQTT_BROKER_ENDPOINT "xxxxx-ats.iot.us-west-2.amazonaws.com"
#define clientcredentialIOT_THING_NAME "aws_wifi_provisioning"
```

(2) Update `aws_clientcredential_keys.h`, you can use the `CertificateConfigurator.html`

(freertos\tools\certificate\_configuration) to generate the "aws\_clientcredential\_keys.h" and replace this file. It use the key files that are download in the previous step.

4. Install the aws\_wifi\_provovisioning.apk in the android phone.
5. The Android application requires Cognito service to authorize to AWS IoT in order to access device shadows. Use Amazon Cognito to create a new user pool and identity pool.
  - a) Open the Amazon Cognito console, click "Manage User Pools"



- b) Click "Create a user pool"



- c) Fill the Identity poll name (for example: aws\_wifi\_provovisioning\_user\_pool), and select "Review defaults".

- d) Select "App clients" from the navigation pane, and then choose "Add an app client"

- e) Enter a name for the app client, like as "aws\_wifi\_provisioning\_apk", and then choose "Create app client"

☐ Legacy  
☒ Enabled (Recommended)

Set attribute read and write permissions

[Cancel](#)
[Create app client](#)

- f) From the navigation pane, choose “Review”, and then choose “Create pool”.

Name  
 Attributes  
 Policies  
 MFA and verifications  
 Message customizations  
 Tags  
 Devices  
 App clients  
 Triggers  
**Review**

Pool name: aws\_wifi\_provisioning\_user\_pool

Required attributes: email  
 Alias attributes: Choose alias attributes...  
 Username attributes: Choose username attributes...  
 Enable case insensitivity?: Yes  
 Custom attributes: Choose custom attributes...

Minimum password length: 8  
 Password policy: uppercase letters, lowercase letters, special characters, numbers  
 User sign ups allowed?: Users can sign themselves up

FROM email address: Default  
 Email Delivery through Amazon SES: Yes

MFA: Enable MFA...  
 Verifications: Email

Tags: Choose tags for your user pool

App clients: aws\_wifi\_provisioning\_apk

Triggers: Add triggers...

[Create pool](#)

- g) From the navigation pane, choose “App clients”, and then choose “Show details”. Make a note of the app client ID and app client secret.

Policies  
 MFA and verifications  
 Advanced security  
 Message customizations  
 Tags  
 Devices  
**App clients**  
 Triggers  
 Analytics  
 App integration  
 App client settings

aws\_wifi\_provisioning\_apk

App client id

[Show Details](#)

[Add another app client](#) [Return to pool details](#)

- h) Open the Amazon Cognito console, click “Manage Identity Pools”

[Manage User Pools](#)
[Manage Identity Pools](#)

- i) Click “Create new identity pool”

[Create new identity pool](#)

- j) Fill the Identity pool name (for example: aws\_wifi\_provisioning\_identity\_pool). Expand “Authentication providers”, choose the “Cognito” tab, and then enter your user pool ID and app client ID. Then Choose “Create Pool”

## Create new identity pool

Identity pools are used to store end user identities. To declare a new identity pool, enter a unique name.

Identity pool name\*

Example: My App Name

▶ Unauthenticated identities

▶ Authentication flow settings

▼ Authentication providers

Amazon Cognito supports the following authentication methods with Amazon Cognito Sign-In or any public provider. If you allow your users to authenticate using any of these public providers, you can specify your application identifiers here. Warning: Changing the application ID that your identity pool is linked to will prevent existing users from authenticating using Amazon Cognito. [Learn more about public identity providers.](#)

**Cognito** Amazon Apple Facebook Google+ Twitter / Digits OpenID SAML Custom

Configure your Cognito Identity Pool to accept users federated with your Cognito User Pool by supplying the User Pool ID and the App Client ID.

User Pool ID

App client id

Add Another Provider

\* Required

Cancel

Create Pool

- k) Expand "View Details" and make a note of the two IAM role names. Choose "Allow" to create the IAM roles for authenticated and unauthenticated identities to access Amazon Cognito.

▼ Hide Details

Role Summary

Role Description Your authenticated identities would like access to Cognito.

IAM Role

Role Name Cognito\_aws\_wifi\_provisioning\_identity\_poolIA

▶ View Policy Document

Role Summary

Role Description Your unauthenticated identities would like access to Cognito.

IAM Role

Role Name Cognito\_aws\_wifi\_provisioning\_identity\_poolIU

▶ View Policy Document

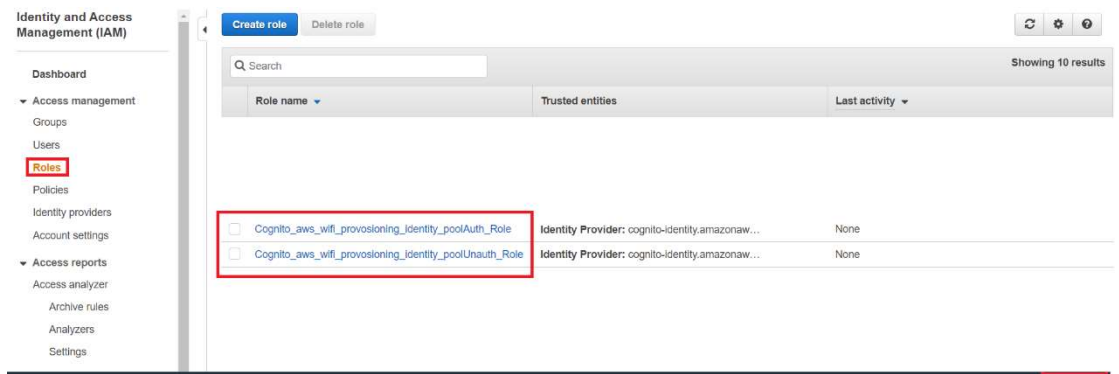
Cancel

Allow

- l) Copy the Identity pool ID. Make a note of the identity pool ID. It should be of the form us-west-2:12345678-1234-1234-1234-123456789012.

```
// Initialize the Amazon Cognito credentials provider
CognitoCachingCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
    getApplicationContext(),
    "us-west-2:12345678-1234-1234-1234-123456789012", // Identity pool ID
    Regions.US_WEST_2 // region
);
```

6. Create and attach an IAM policy to the authenticated identity
- Open the IAM console, and from the navigation pane, choose "Roles".



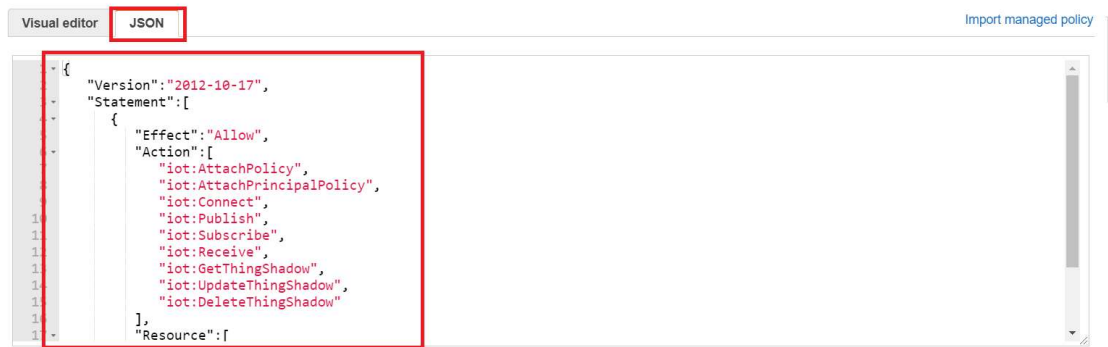
- b) Find and choose your authenticated identity's role, choose "Add inline policy".



- c) Choose the "JSON" tab and paste the following JSON. And then choose "Review policy".

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:AttachPolicy",
        "iot:AttachPrincipalPolicy",
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```





Character count: 410 of 10,240.

The current character count includes character for all inline policies in the role: `Cognito_aws_wifi_provisioning_identity_poolAuth_Role`

Cancel

Review policy

- d) Enter a name for the policy, and then choose "Create policy". This step should be applied for both Roles.

### Review policy

Before you create this policy, provide the required information and review this policy.

Name\*

Cognito\_aws\_wifi\_provisioning\_identity\_poolAuth\_Role\_policy

Maximum 128 characters. Use alphanumeric and `+=, @, _` characters.

### Summary

Filter			
Service	Access level	Resource	Request condition
Allow (1 of 239 services) Show remaining 238			
IoT	Limited: Read, Write, Permissions management	All resources	None

\* Required

Cancel

Previous

Create policy

After the above steps. The pool is created successfully.

Prepare one configuration file for the android app.

Prepare "Preferences.properties" file with yours AWS credentials. It's structure looks like this:

```

customer_specific_endpoint=<REST API ENDPOINT>
cognito_pool_id=<COGNITO POOL ID>
thing_name=<THING NAME>
region=<REGION>
policy_name=<POLICY>

```

- customer\_specific\_endpoint is the endpoint that is configured in `aws_clientcredential.h`
- cognito\_pool\_id is the copied pool id in above step.
- thing\_name is the created Thing name.
- region is the front part of the endpoint.
- policy\_name is the created Thing policy

for example:

```

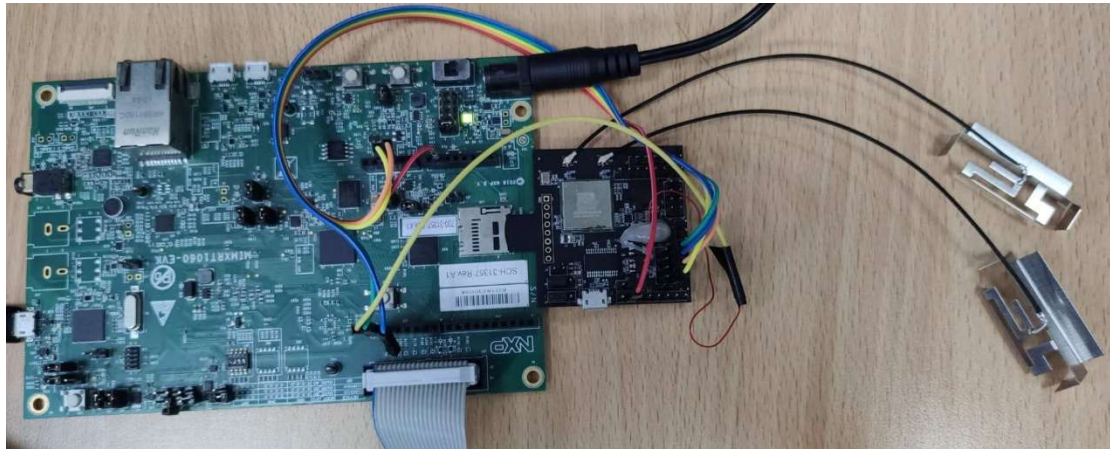
customer_specific_endpoint=xxxxxx-ats.iot.us-west-2.amazonaws.com
cognito_pool_id=us-west-2:5xxxx7-3xxxx9-4xxx5-axxf-0xxxxxxxxxxb
thing_name= aws_wifi_provisioning
region=us-west-2
policy_name=aws_wifi_provisioning_policy

```

Running the demo

=====

1. Prepare the board as follow picture.



2. Run the demo with rt1060.

(1) After running the log is as follow:

```
COM26 - PuTTY
0 17 [main task] [INFO ][MSD_FATFS][538990600] USB Host stack successfully initialized
1 194 [usb host task] [INFO ][MSD_FATFS][538990600] The USB MSD disk is attached (pid=0x1000 ,vid=0x90c) with assigned address=1
2 270 [main task] Write certificate...
3 297 [iot_thread] [INFO ][DEMO][538990600] -----STARTING DEMO-----
4 298 [iot_thread] [INFO ][INIT][538990600] SDK successfully initialized.
5 1193 [app task] [INFO ][MSD_FATFS][538990600] fatfs mount as logialcal driver 1.....
6 1193 [app task] [INFO ][MSD_FATFS][538990600] success
7 1193 [app task] [INFO ][MSD_FATFS][538990600] Get Disk information,
8 1509 [app task] [INFO ][MSD_FATFS][538990600] FAT type = FAT32
9 1510 [app task] [INFO ][MSD_FATFS][538990600] bytes per cluster = 32768; number of clusters=489411
10 1510 [app task] [INFO ][MSD_FATFS][538990600] The free size: 15313632KB, the total size:15661152KB
MAC Address: D8:C0:A6:C0:B0:4B
[net] Initialized TCP/IP networking stack
[wm_wlan] WLAN_REASON_INITIALIZED
11 5557 [EtherMind RD Ta] [INFO ][IOT_BLE_HAL_COMMON_GAP][538990600] Bluetooth ON Initialization Completed.
12 5558 [EtherMind RD Ta] [INFO ][IOT_BLE_HAL_COMMON_GAP][538990600] Stack Version - 016.002.000.
13 5564 [iot_thread] [INFO ][DEMO][538990600] No networks connected for the demo. Waiting for a network connection.
```

- (2) Open the android app and load the preferences.properties as the video. And then click "WIFI PROVISIONING" to start aws wifi provisioning test.