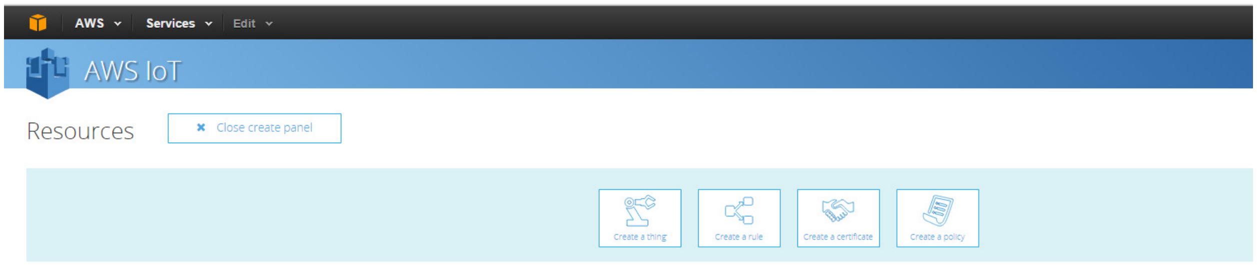
Quick guide for LinkitOne with AWS IoT

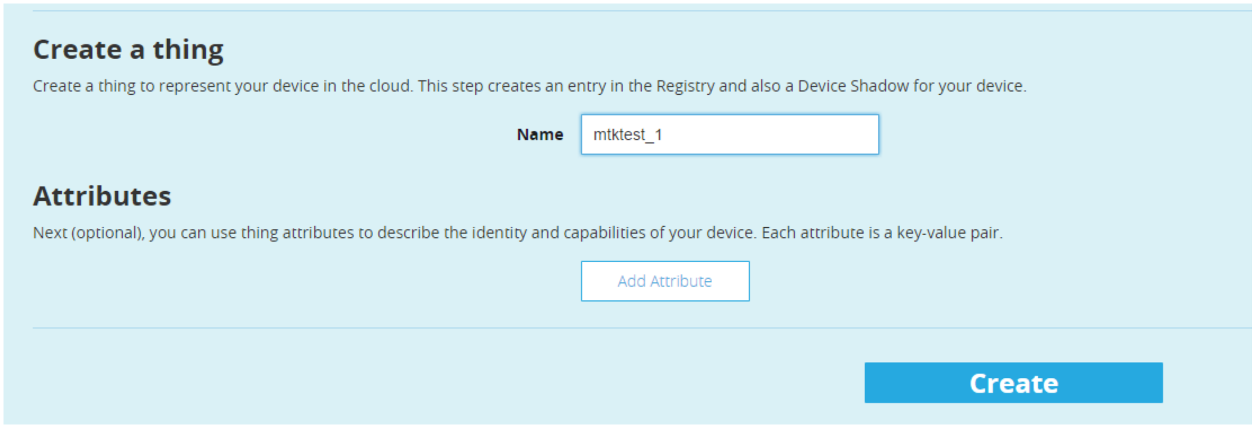
This is temporary guide for using LinkitOne with AWS IoT server. Official guide will be published very soon

MQTT Shadow Example

1. Create an AWS Account.
2. Go to AWS IoT and open up the AWS IoT Dashboard



3. Choose "Create thing"
4. Create a name for the thing



Create a thing

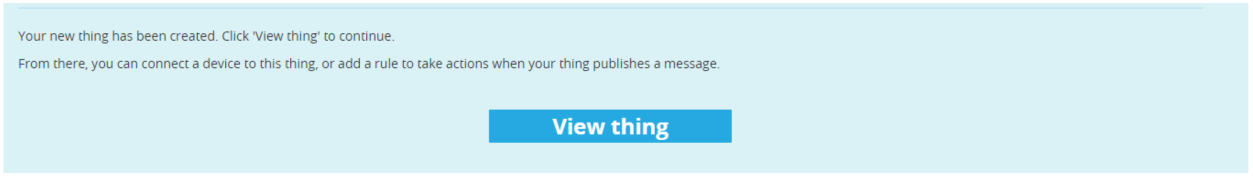
Create a thing to represent your device in the cloud. This step creates an entry in the Registry and also a Device Shadow for your device.

Name: mtktest_1

Attributes

Next (optional), you can use thing attributes to describe the identity and capabilities of your device. Each attribute is a key-value pair.

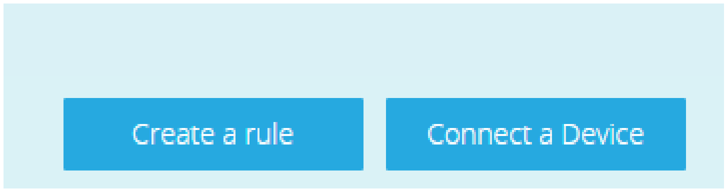
5. Click view thing



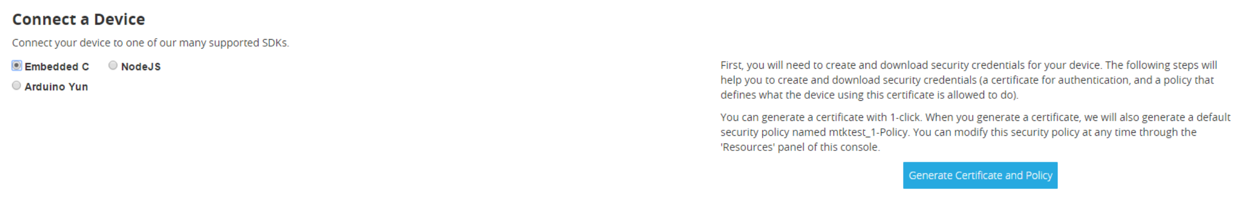
Your new thing has been created. Click 'View thing' to continue.

From there, you can connect a device to this thing, or add a rule to take actions when your thing publishes a message.

6. Choose "Connect a Device"



7. Please do like following screenshot and click "Generate Certificate and Policy"



Connect a Device

Connect your device to one of our many supported SDKs.

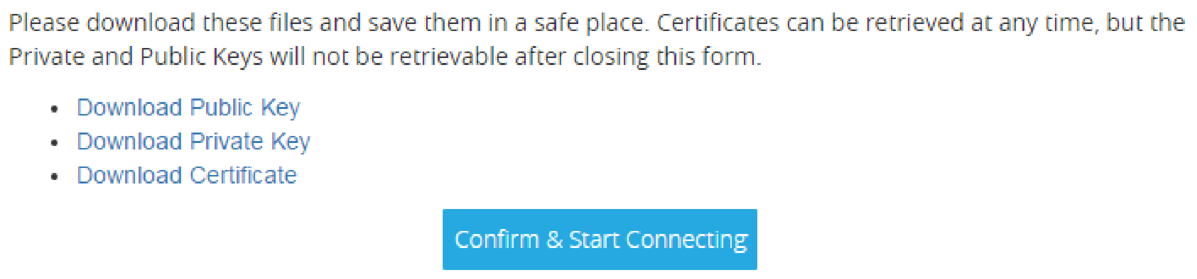
[✓ Embedded C] NodeJS

Arduino Yun

First, you will need to create and download security credentials for your device. The following steps will help you to create and download security credentials (a certificate for authentication, and a policy that defines what the device using this certificate is allowed to do).

You can generate a certificate with 1-click. When you generate a certificate, we will also generate a default security policy named mtktest_1-Policy. You can modify this security policy at any time through the 'Resources' panel of this console.

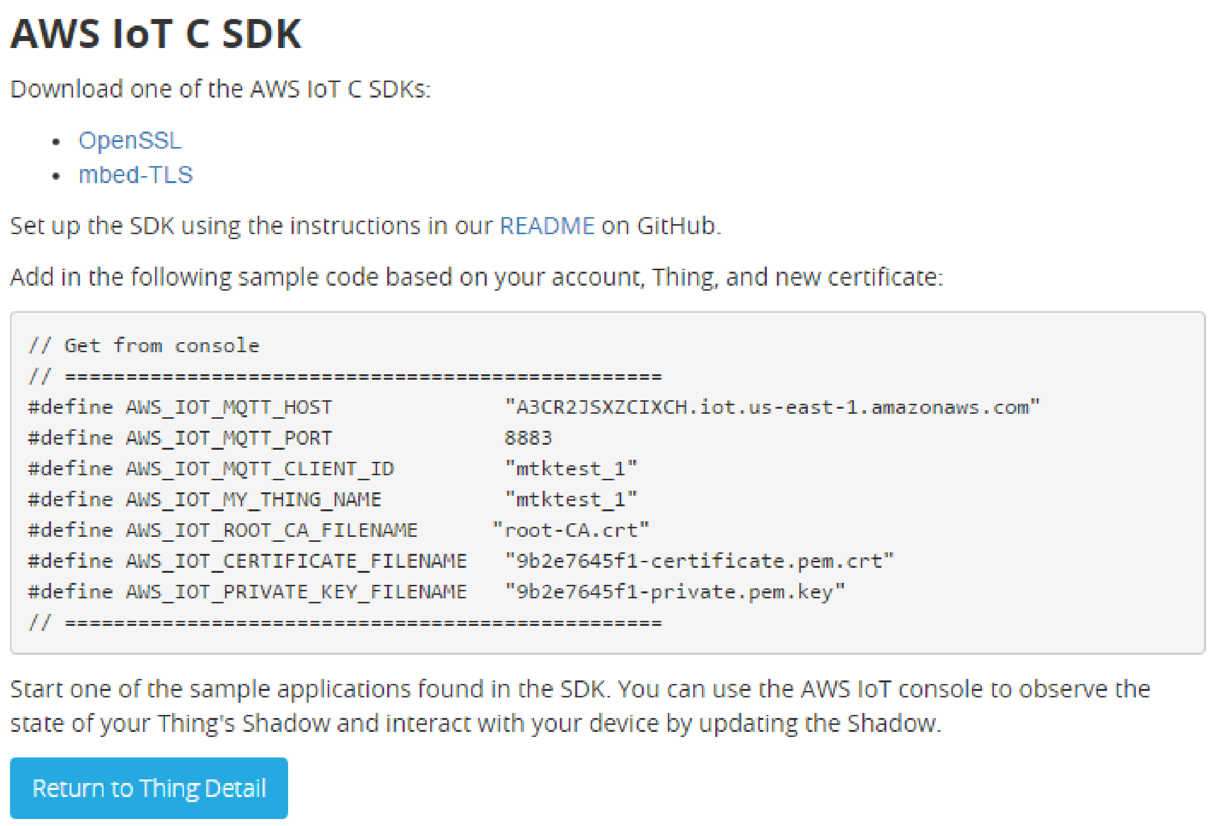
8. Download those 3 files to computer



Please download these files and save them in a safe place. Certificates can be retrieved at any time, but the Private and Public Keys will not be retrievable after closing this form.

- Download Public Key
- Download Private Key
- Download Certificate

9. Please save them and rename it as you want.
10. Change LinkitOne to MS mode and copy those three files to LinkitOne flash memory
11. Switch back to UART mode
12. Click Return to Thing Detail



AWS IoT C SDK

Download one of the AWS IoT C SDKs:

- OpenSSL
- mbed-TLS

Set up the SDK using the instructions in our README on GitHub.

Add in the following sample code based on your account, Thing, and new certificate:

```
// Get from console
// =================================================
#define AWS_IOT_MQTT_HOST              "A3CR2JSXZCIXCH.iot.us-east-1.amazonaws.com"
#define AWS_IOT_MQTT_PORT              8883
#define AWS_IOT_MQTT_CLIENT_ID         "mtktest_1"
#define AWS_IOT_MY_THING_NAME          "mtktest_1"
#define AWS_IOT_ROOT_CA_FILENAME      "root-CA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME   "9b2e7645f1-certificate.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "9b2e7645f1-private.pem.key"
// =================================================
```

Start one of the sample applications found in the SDK. You can use the AWS IoT console to observe the state of your Thing's Shadow and interact with your device by updating the Shadow.

13. Download the LinkitOne source code from github:

https://github.com/MediaTek-Labs/aws_mbedtls_mqtt

14. Open aws_paho_shadow/aws_paho_shadow.ino with Arduino

15. In aws_mtk_iot_config.h, change the settings for wifi, your 3 certification files' name and your AWS thing name. Currently, LinkitOne only supports the ip address to connect to the AWS server, please rewrite your ip address for your host name (default is "data.iot.us-east-1.amazonaws.com", you could get the ip address by pint this host)

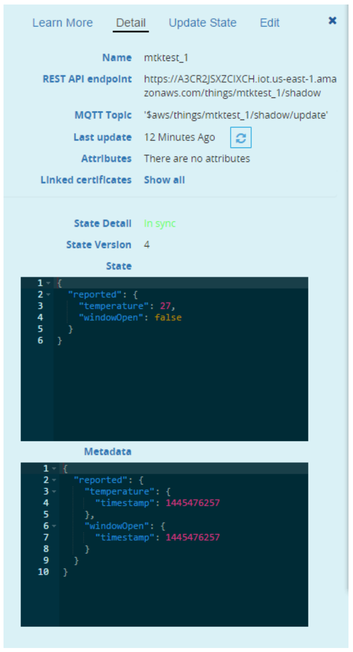
```
#ifndef SRC_SHADOW_IOT_SHADOW_CONFIG_H_
#define SRC_SHADOW_IOT_SHADOW_CONFIG_H_

//Change the IP address to the HOST and modify the certification file names
VMSTR IP_ADDRESS = "54.86.88.20"; //currently only support IP address
char cafileName[] = "G5.pem";
char clientCRTName[] = "cert.pem";
char clientKeyName[] = "privatekey.pem";

// Get from console
// =================================================
#define AWS_IOT_MQTT_HOST              "data.iot.us-east-1.amazonaws.com" ///< Customer specific MQTT HOST. The same will be used for Thing Shadow
#define AWS_IOT_MQTT_PORT              8883 ///< default port for MQTT/S
#define AWS_IOT_MQTT_CLIENT_ID         "LinkitOne" ///< MQTT client ID should be unique for every device
#define AWS_IOT_MY_THING_NAME          "mtktest_1" ///< Thing Name of the Shadow this device is associated with
#define AWS_IOT_ROOT_CA_FILENAME       "G5.pem" ///< Root CA file name
#define AWS_IOT_CERTIFICATE_FILENAME   "cert.pem" ///< device signed certificate file name
#define AWS_IOT_PRIVATE_KEY_FILENAME   "privatekey.pem" ///< Device private key filename
// =================================================

#endif /* SRC_SHADOW_IOT_SHADOW_CONFIG_H_ */
```

16. You will see in the AWS console, the data will be updated every time LinkitOne push the data to it.



Name mtktest_1
REST API endpoint https://A3CR2JSXZCIXCH.iot.us-east-1.amazonaws.com/things/mtktest_1/shadow
MQTT Topic '$aws/things/mtktest_1/shadow/update'
Last update 12 Minutes Ago
Attributes There are no attributes
Linked certificates Show all

State Detail In sync
State Version 4
State

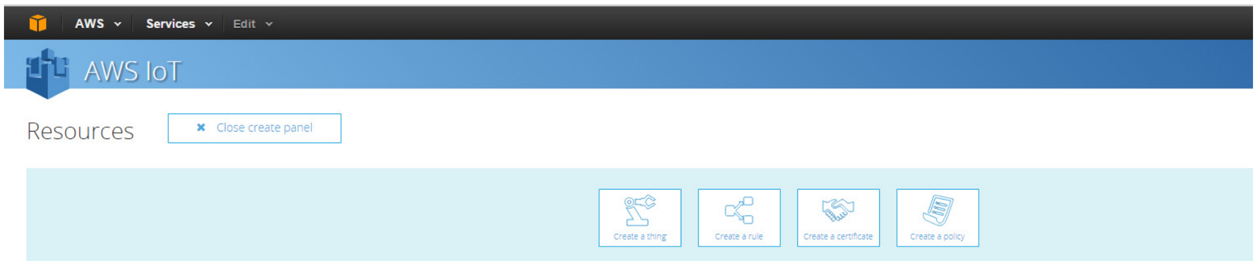
```
{
  "reported": {
    "temperature": 27,
    "windowOpen": false
  }
}
```

Metadata

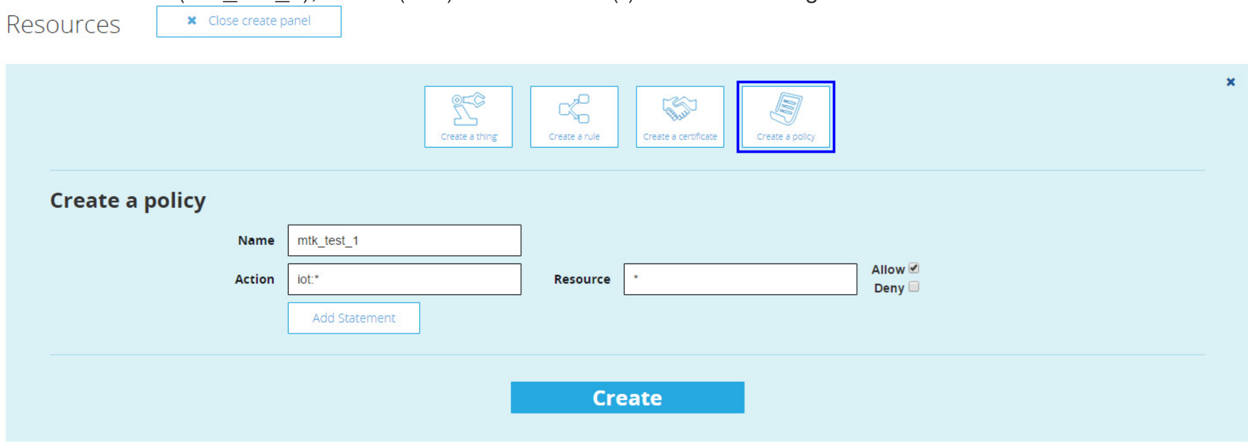
```
{
  "reported": {
    "temperature": {
      "timestamp": 1445476257
    },
    "windowOpen": {
      "timestamp": 1445476257
    }
  }
}
```

MQTT message pub/sub example

1. Go to AWS IoT and open up the AWS IoT Dashboard



2. Choose “Create policy”
3. Fill out the Name(mtk_test_1), Action (iot:*) and Resource(*) like the following screen shot:



Create a policy

Name: mtk_test_1

Action: iot:*

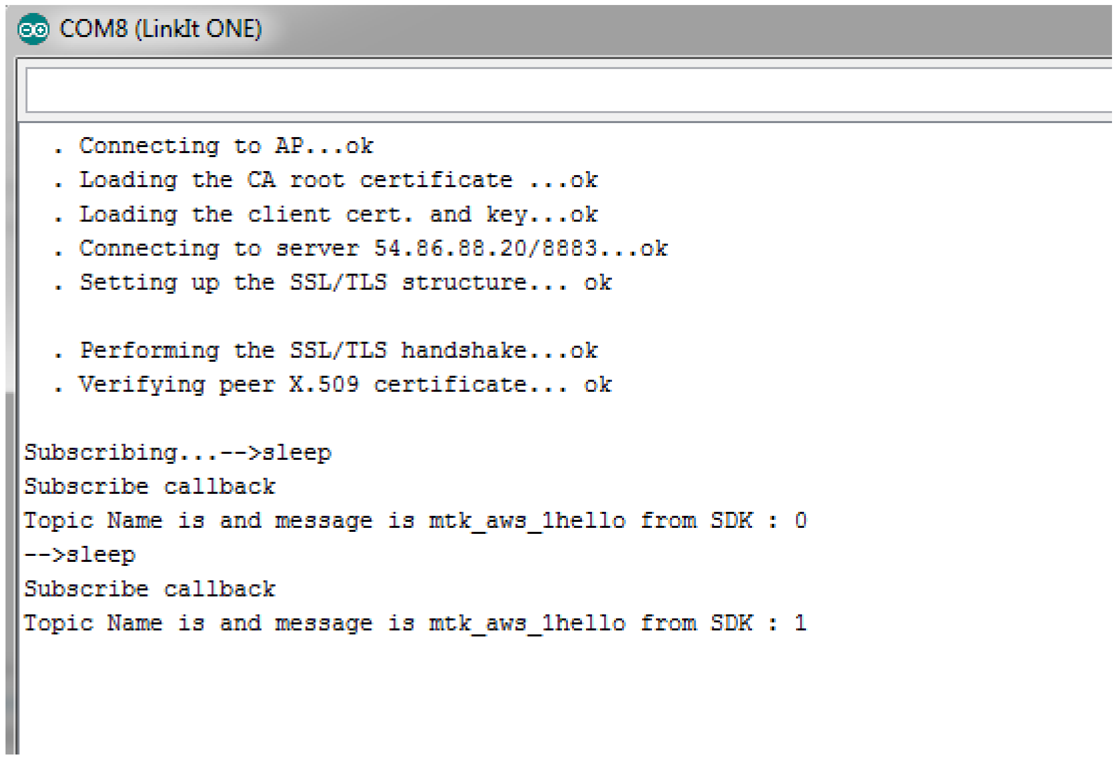
Resource: *

Allow ✓ Deny

4. Click the “Create” button
5. Open aws_paho_mqtt.ino with Arduino
6. Change the settings in aws_mtk_iot_config.h:

```
// Get from console
// =================================================
#define AWS_IOT_MQTT_HOST              "data.iot.us-east-1.amazonaws.com" ///< Customer specific MQTT HOST. The same will be used for Thing Shadow
#define AWS_IOT_MQTT_PORT              8883 ///< default port for MQTT/S
#define AWS_IOT_MQTT_CLIENT_ID         "LinkitOne" ///< MQTT client ID should be unique for every device
#define AWS_IOT_MY_THING_NAME          "mtk_aws_1" ///< Thing Name of the Shadow this device is associated with
#define AWS_IOT_ROOT_CA_FILENAME       "G5.pem" ///< Root CA file name
#define AWS_IOT_CERTIFICATE_FILENAME   "cert.pem" ///< device signed certificate file name
#define AWS_IOT_PRIVATE_KEY_FILENAME   "privatekey.pem" ///< Device private key filename
#define AWS_IOT_TOPIC_NAME             "mtk_aws_1" //AWS policy name
// =================================================
```

7. Like shadow example, change those certification files and also the AWS_IOT_TOPIC_NAME to the policy name you just created.
8. Run the sketch and see the logs from monitor:


COM8 (LinkIt ONE)

```
  . Connecting to AP...ok
  . Loading the CA root certificate ...ok
  . Loading the client cert. and key...ok
  . Connecting to server 54.86.88.20/8883...ok
  . Setting up the SSL/TLS structure... ok

  . Performing the SSL/TLS handshake...ok
  . Verifying peer X.509 certificate... ok

Subscribing...-->sleep
Subscribe callback
Topic Name is and message is mtk_aws_1hello from SDK : 0
-->sleep
Subscribe callback
Topic Name is and message is mtk_aws_1hello from SDK : 1
```

9. If you have another terminal like Mac, you could send a message through MQTT to it like following:

```
Topic Name is and message is mtk_aws_1hello from SDK : 5
-->sleep
Subscribe callback
Topic Name is and message is mtk_aws_1hello from SDK : 6
Subscribe callback
Topic Name is and message is mtk_aws_1   Hello from Mac, can you hear me???
-->sleep
Subscribe callback
```