Application Note: AWS Remote Control with LTE-M/NB-IoT connectivity using LPC55xxx MCU family

Contents

0. Pre-requisites

Create free user accounts for:

- Amazon Developer Services (https://developer.amazon.com)
- Amazon Web Services - AWS (https://aws.amazon.com)
- Download the latest MCUXpresso IDE for your platform (https://mcuxpresso.nxp.com)
- Download the latest LPC55 SDK (https://mcuxpresso.nxp.com)
  Note: Don't forget to enable AWS component in the SDK.

Important note: Make sure the P4 jumper is set to 3.3V and J4 jumper for DFU is not in place.

1. Objectives

In this Application Note, you will learn how to:

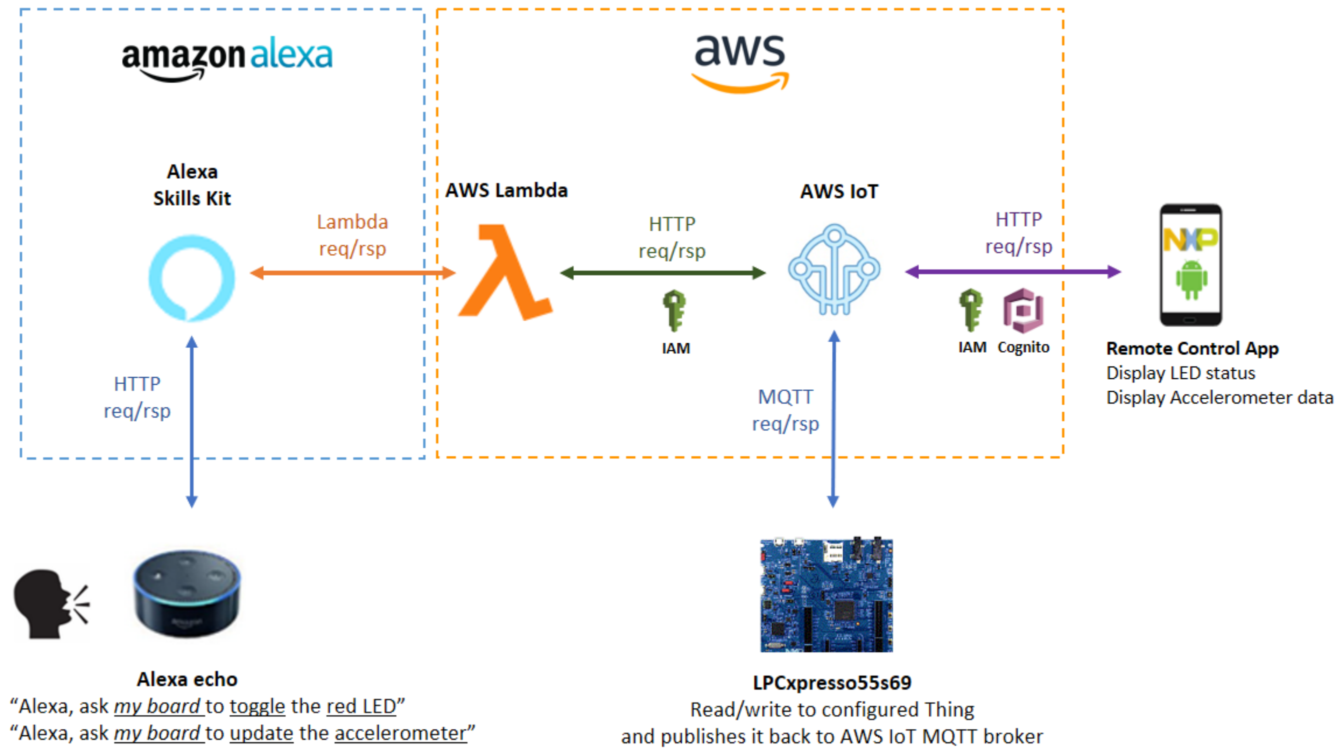
- Create an Alexa skill
- Create an AWS-Lambda function
- Create an AWS Thing with its credentials
- Import an SDK to the MCUXpresso IDE
- Load AWS Thing credentials and download your demo application
- Control LED and update accelerometer data on LPC board using Alexa
- Control LED and update accelerometer data on LPC board using an Android application

2. Hardware

- NXP LPC55S69-EVK, the LPCXpresso55S69 Evaluation Board.
- Alexa Echo (optional, you can also use the Alexa test tool from Alexa Developer Console)
- Android device
- Micro-USB cable

3. Application high level description

1. The user speaks to the Echo device to activate an Alexa skill
2. The Alexa Skill triggers an AWS-Lambda function which sends a message to the AWS-IoT
3. The NXP LPC55S69 is registered on the AWS-IoT and receives the message to perform the targeted action (turn on/off LED, read accelerometer)
4. The NXP LPC55S69 sends a feedback message to the AWS-IoT platform
5. AWS-Lambda retrieves the feedback message and sends it back to the Alexa platform. This makes the Echo device respond to user: "The LED is on/off!" or "The accelerometer data was updated"
6. Instructions to enable demo functionality using Android applications and Alexa Echo.


amazon alexa
aws
Alexa Skills Kit
Lambda req/rsp
AWS Lambda
HTTP req/rsp
IAM
AWS IoT
HTTP req/rsp
IAM
Cognito
Remote Control App
Display LED status
Display Accelerometer data
HTTP req/rsp
MQTT req/rsp
Alexa echo
"Alexa, ask my board to toggle the red LED"
"Alexa, ask my board to update the accelerometer"
LPCxpresso55s69
Read/write to configured Thing
and publishes it back to AWS IoT MQTT broker

4. AWS Configuration (Lambda, Skill and Thing)

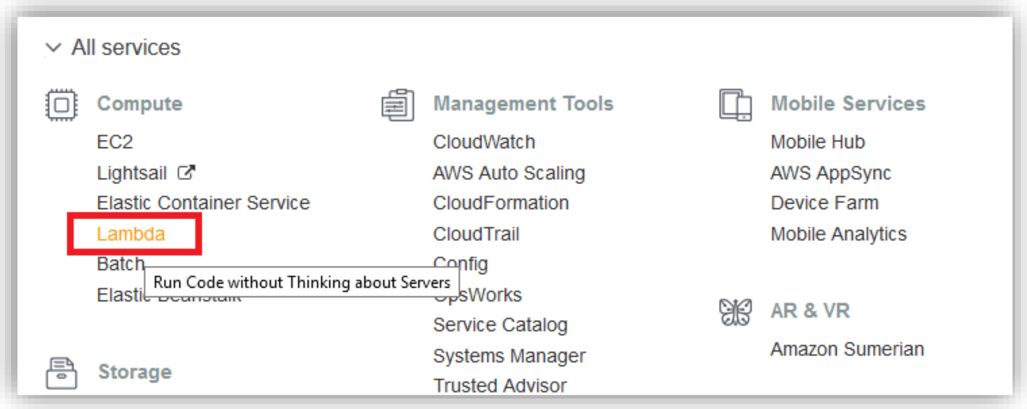
This section will guide you to configure Amazon AWS services such as Lambda, IoT Core, Cognito and IAM), and Amazon Developer (Alexa Skill) to communicate between them.

4.1 Create an AWS Lambda Function

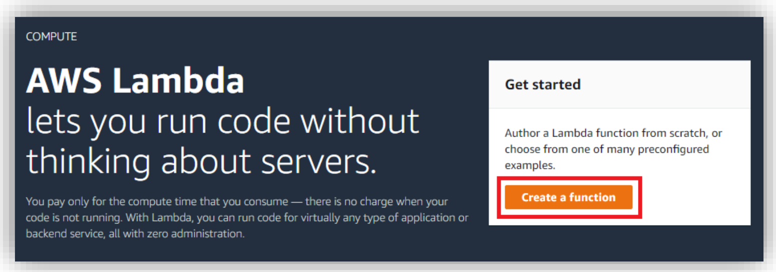
What is AWS Lambda?
AWS Lambda is a compute service that lets you run code without provisioning or managing servers; it executes your code only when needed and scales automatically.

The Lambda function will be linked to the Alexa Skill kit at one side and to AWS IoT at the other. Lambda functions allows us to receive inputs from Alexa and perform action over the AWS IoT network, and vice versa. We need to provide Lambda with an IAM key to allow accessibility to AWS IoT resources.
Follow the steps below to create and configure the Lambda function.

1. Enter the Amazon AWS management console (https://console.aws.amazon.com).
2. Select the US East (N. Virginia) region on the top right (Alexa is available at this region). IoT Things need to be created in same region.
3. Click on 'Services -> Compute -> **Lambda**'



4. Click on “Create function”


COMPUTE

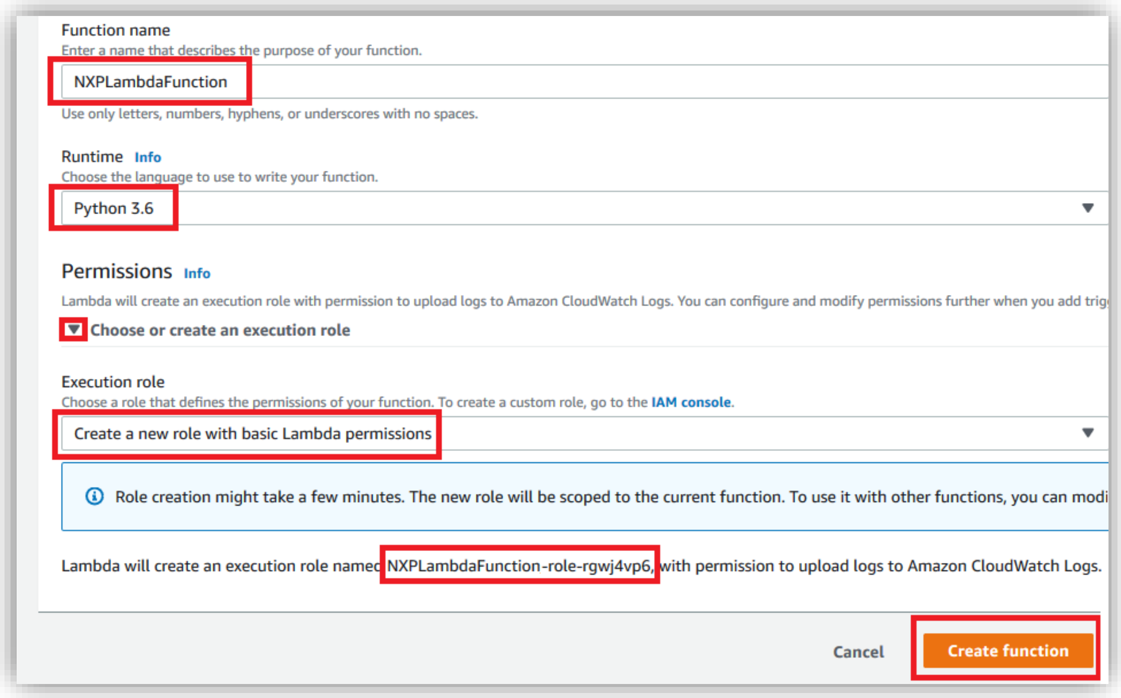
AWS Lambda
lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

5. Select the "Author from Scratch" option and edit the Name you wish to assign to the Lambda Function (for example ‘NXPLambdaFunction’).
6. Select **Python 3.6** in the "Runtime" drop-down menu.
7. Open the "Execution Role" drop-down menu and select "Create a new role with basic Lambda permissions".


Function name
Enter a name that describes the purpose of your function.
NXPLambdaFunction
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Python 3.6

Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add trig

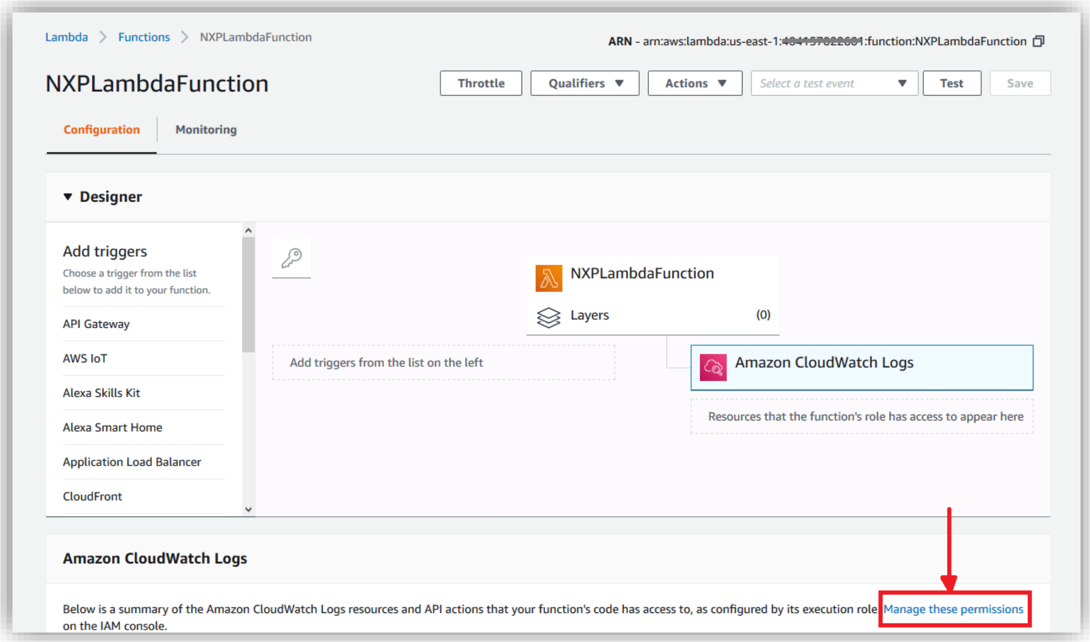
▼ Choose or create an execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions

Role creation might take a few minutes. The new role will be scoped to the current function. To use it with other functions, you can modi

Lambda will create an execution role named NXPLambdaFunction-role-rgwj4vp6, with permission to upload logs to Amazon CloudWatch Logs.

8. Identify the name of the execution role that will be created.
9. Click on the “Create function” button.
10. Verify the Lambda function was created.
11. Click on the “Amazon CloudWatch Logs” icon and scroll down to the “Amazon CloudWatch Logs” section.
12. Click on the “Manage these permissions” link.


Lambda > Functions > NXPLambdaFunction
Throttle
Qualifiers
Actions
Select a test event
Test
Save
Configuration
Monitoring
Designer
Add triggers
Choose a trigger from the list below to add it to your function.
API Gateway
AWS IoT
Alexa Skills Kit
Alexa Smart Home
Application Load Balancer
CloudFront
Layers
(0)
Add triggers from the list on the left
Resources that the function's role has access to appear here
Manage these permissions

ARN - arn:aws:lambda:us-east-1:[redacted]:function:NXPLambdaFunction

NXPLambdaFunction

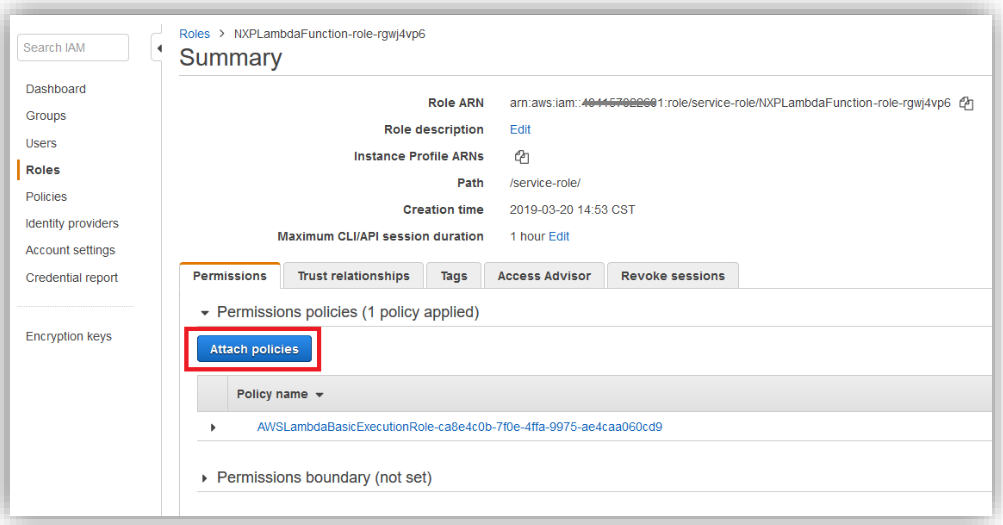
Amazon CloudWatch Logs

Below is a summary of the Amazon CloudWatch Logs resources and API actions that your function's code has access to, as configured by its execution role. Manage these permissions on the IAM console.

Note: On the top right, you can see the ARN (Amazon Resource Number) of the Lambda Function that you've just created. That's the ARN that will allow us to link this function to the Alexa service: we'll get back to this later.

This will take us to the IAM console, where we can assign the permissions to our NXPLambdaFunction role.

13. Click on "Attach policies" button.



Summary

Role ARN	arn:aws:iam::[redacted]:role/service-role/NXPLambdaFunction-role-rgwj4vp6
Role description	Edit
Instance Profile ARNs	
Path	/service-role/
Creation time	2019-03-20 14:53 CST
Maximum CLI/API session duration	1 hour Edit

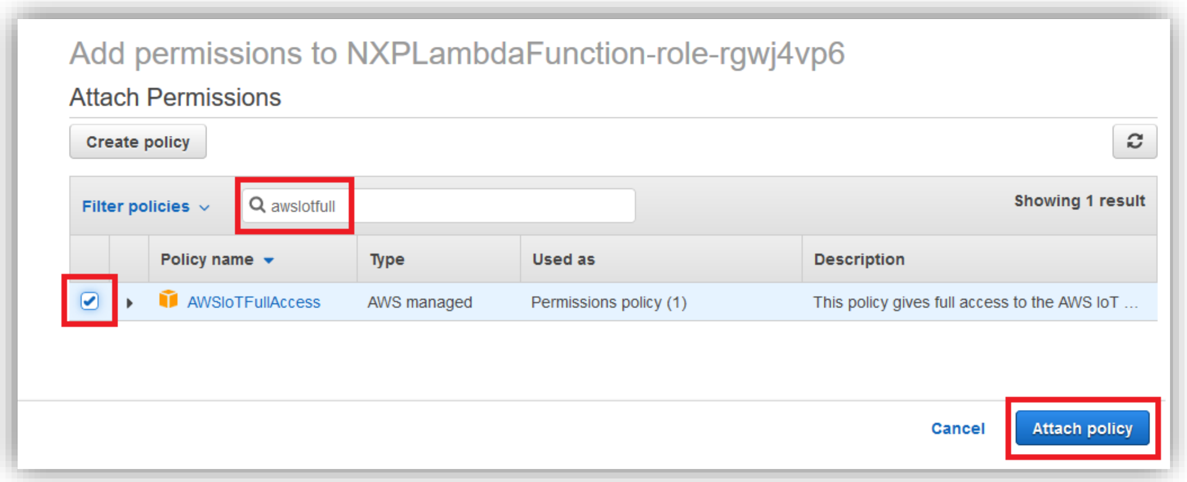
Permissions policies (1 policy applied)

AWSLambdaBasicExecutionRole-ca8e4c0b-7f0e-4ffa-9975-ae4caa060cd9

Permissions boundary (not set)

We will add a predefined policy to our Role that will allow our Lambda function to interact with the AWS IoT Core service.

14. Type "awsiotfull" in the filter.
15. Select the "AWSIoTFullAccess" policy.
16. Click on the "Attach policy" button.

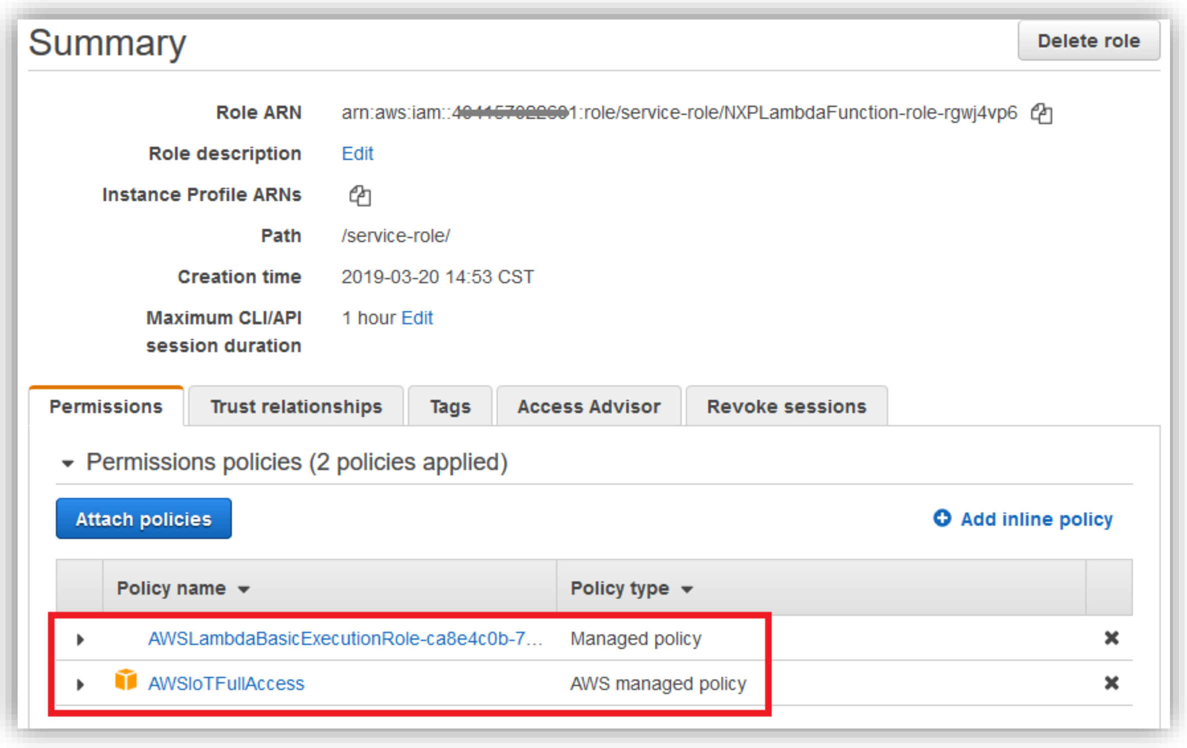


Add permissions to NXPLambdaFunction-role-rgwj4vp6

Attach Permissions

	Policy name	Type	Used as	Description
✓	AWSIoTFullAccess	AWS managed	Permissions policy (1)	This policy gives full access to the AWS IoT …

17. Verify that your role has now two policies: the AWSLambdaBasicExecutionRole and the AWSIoTFullAccess policies.



Summary

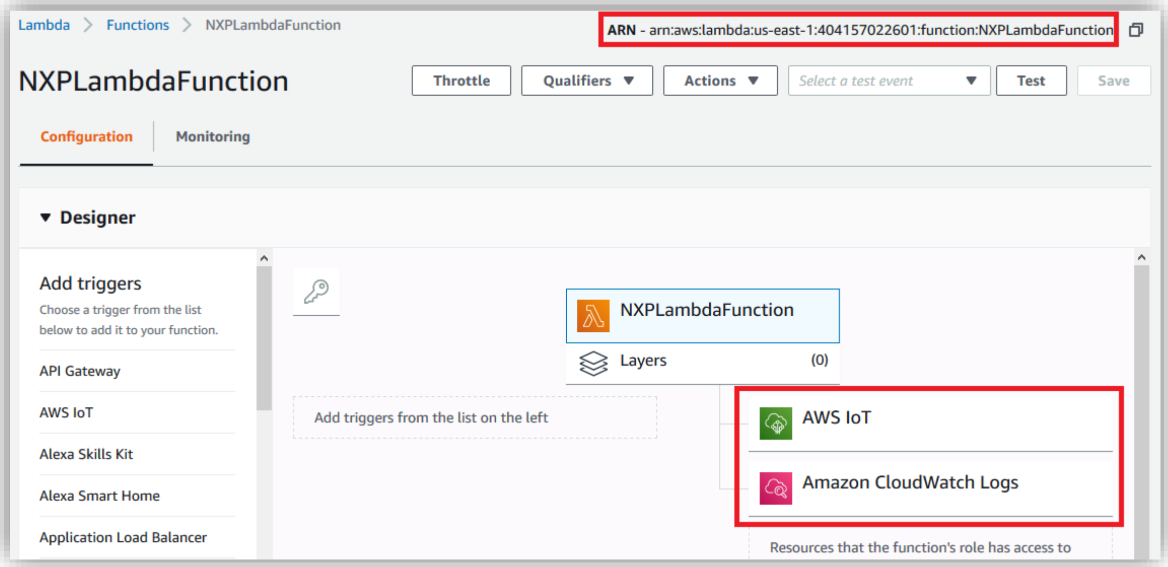
Role ARN	arn:aws:iam::4[redacted]1:role/service-role/NXPLambdaFunction-role-rgwj4vp6
Role description	Edit
Instance Profile ARNs	
Path	/service-role/
Creation time	2019-03-20 14:53 CST
Maximum CLI/API session duration	1 hour Edit

Permissions policies (2 policies applied)

Policy name	Policy type
AWSLambdaBasicExecutionRole-ca8e4c0b-7…	Managed policy
AWSIoTFullAccess	AWS managed policy

You just created and configured the **role** to be assigned to the Lambda function.
A Role is a set of policies that determine what the user or identity using that Role can/cannot do within AWS. (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html).

18. Go back to your NXPLambdaFunction and verify that you now has "AWS IoT" resource available for your Lambda function.

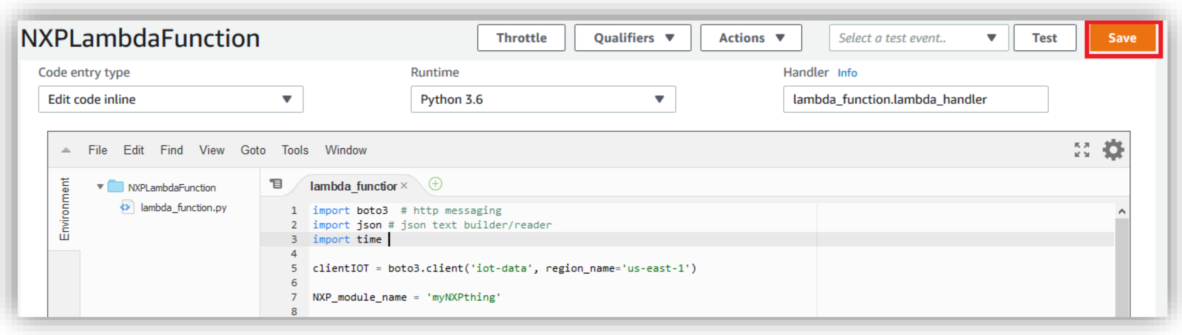

Lambda > Functions > NXPLambdaFunction
ARN - arn:aws:lambda:us-east-1:404157022601:function:NXPLambdaFunction
NXPLambdaFunction
Throttle
Qualifiers
Actions
Select a test event
Test
Save
Configuration
Monitoring
Designer
Add triggers
Choose a trigger from the list below to add it to your function.
API Gateway
AWS IoT
Alexa Skills Kit
Alexa Smart Home
Application Load Balancer
NXPLambdaFunction
Layers (0)
Add triggers from the list on the left
AWS IoT
Amazon CloudWatch Logs
Resources that the function's role has access to

19. Scroll down and locate the editable function code window and erase the preconfigured code.
20. Copy-paste the content of **Alexa_RC_lambda_function.py** code into code window.

Note:
Alexa_RC_lambda_function.py contains functions to interact between Alexa Voice Service and your Thing. Please take some minutes to analyze it and understand how the Lambda function receives and handles a request from Alexa, depending on the Indents and attributes that were received.
Alexa_RC_lambda_function.py file is in the Lab folder.

21. Click on "Save" at top right.

    Note: Notice in the python file that the *NXP_module_name* is the name of the Thing you will use on the your LPCxpresso55S69 project. In this lab, we will use "myNXPthing".



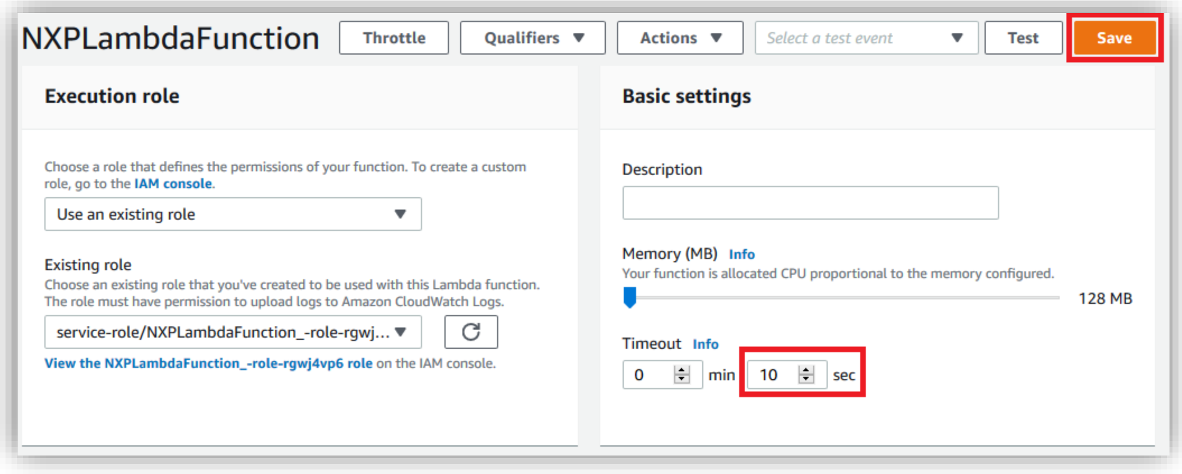
Code entry type: Edit code inline

Runtime: Python 3.6

Handler: lambda_function.lambda_handler

```
import boto3  # http messaging
import json # json text builder/reader
import time

clientIOT = boto3.client('iot-data', region_name='us-east-1')

NXP_module_name = 'myNXPthing'
```

22. Change the timeout settings of the lambda function: Scroll down and find the dedicated section, raise it up to 10 sec, that's enough to allow your lambda function to interact with the other cloud platforms without early stopping. Don't forget to save.



Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

Use an existing role

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/NXPLambdaFunction_-role-rgwj...

View the NXPLambdaFunction_-role-rgwj4vp6 role on the IAM console.

Description

Memory (MB)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout

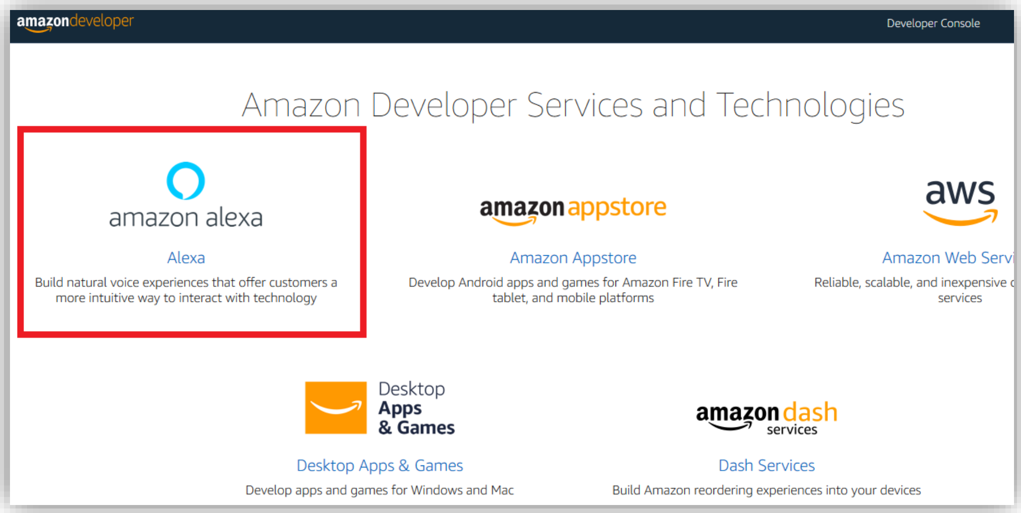
0 min 10 sec

SUCCESS! You've just created and configured the AWS Lambda function. Now let's create the skill for Alexa to interact with our Lambda function.

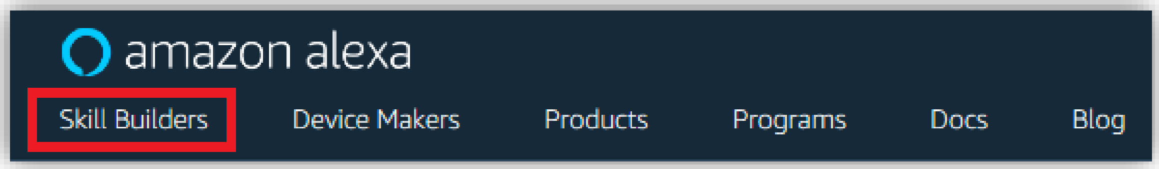
4.2 Create an Alexa skill and link it to the AWS Lambda function

What's a skill? It's a set of expressions and utterances which we want Alexa to be able to recognize. By setting a new skill we will be able to interact with Alexa and trigger our Lambda function by saying those specific expressions to our Echo device.

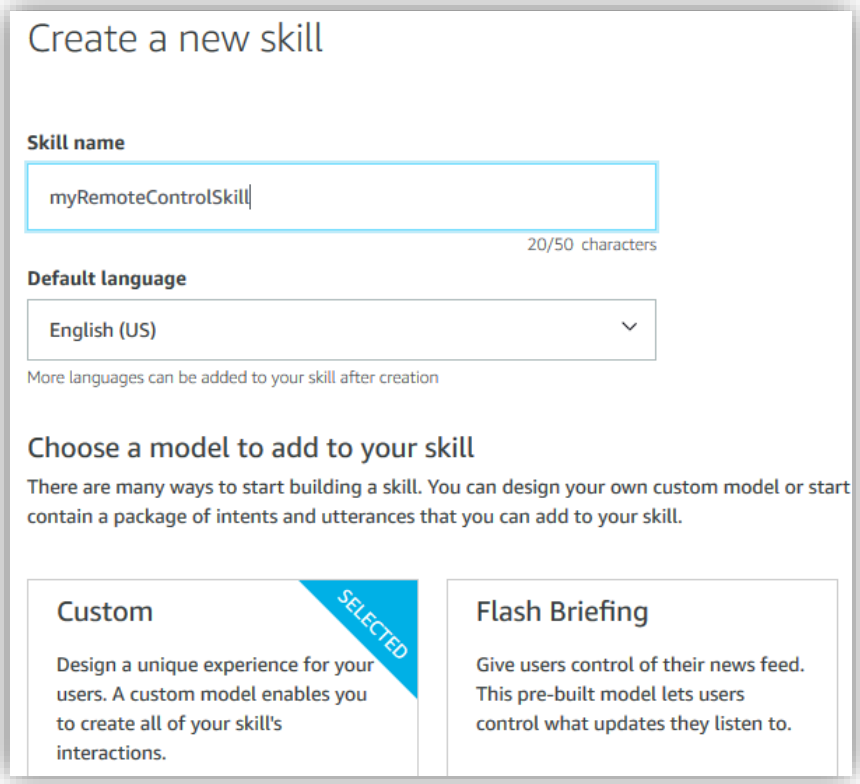
1. Enter the Amazon Developer console (https://developer.amazon.com/)
   Note: Create an Amazon Developer account if you haven't done it yet.
2. Click on the Alexa tab


amazondeveloper
Developer Console
Amazon Developer Services and Technologies
amazon alexa
Alexa
Build natural voice experiences that offer customers a more intuitive way to interact with technology
amazon appstore
Amazon Appstore
Develop Android apps and games for Amazon Fire TV, Fire tablet, and mobile platforms
aws
Amazon Web Serv
Reliable, scalable, and inexpensive c services
Desktop Apps & Games
Desktop Apps & Games
Develop apps and games for Windows and Mac
amazon dash services
Dash Services
Build Amazon reordering experiences into your devices

3. Click on Skill Builders



4. Click on “Start a Skill”
5. Click on “Create Skill”
6. Set a name to your Skill, for example “myRemoteControlSkill”, select ‘English (US)’ language and Custom Model. Then, click “Create Skill”


Create a new skill

Skill name

myRemoteControlSkill

20/50 characters

Default language

English (US)

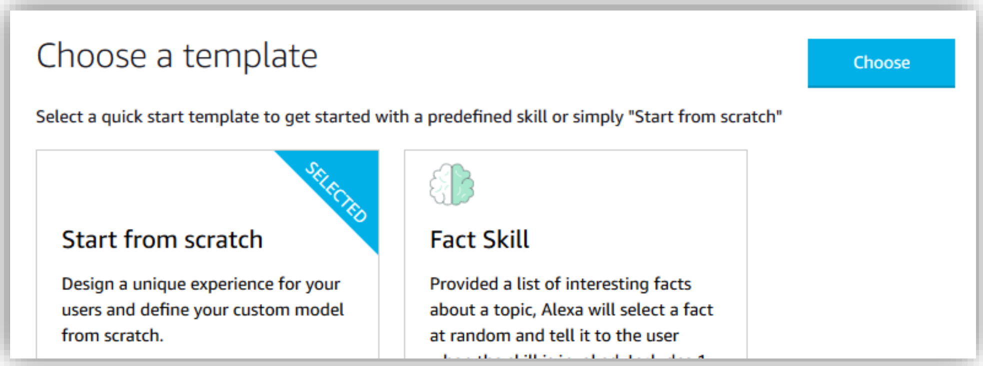
More languages can be added to your skill after creation

Choose a model to add to your skill

There are many ways to start building a skill. You can design your own custom model or start contain a package of intents and utterances that you can add to your skill.

Custom
SELECTED
Design a unique experience for your users. A custom model enables you to create all of your skill's interactions.
Flash Briefing
Give users control of their news feed. This pre-built model lets users control what updates they listen to.

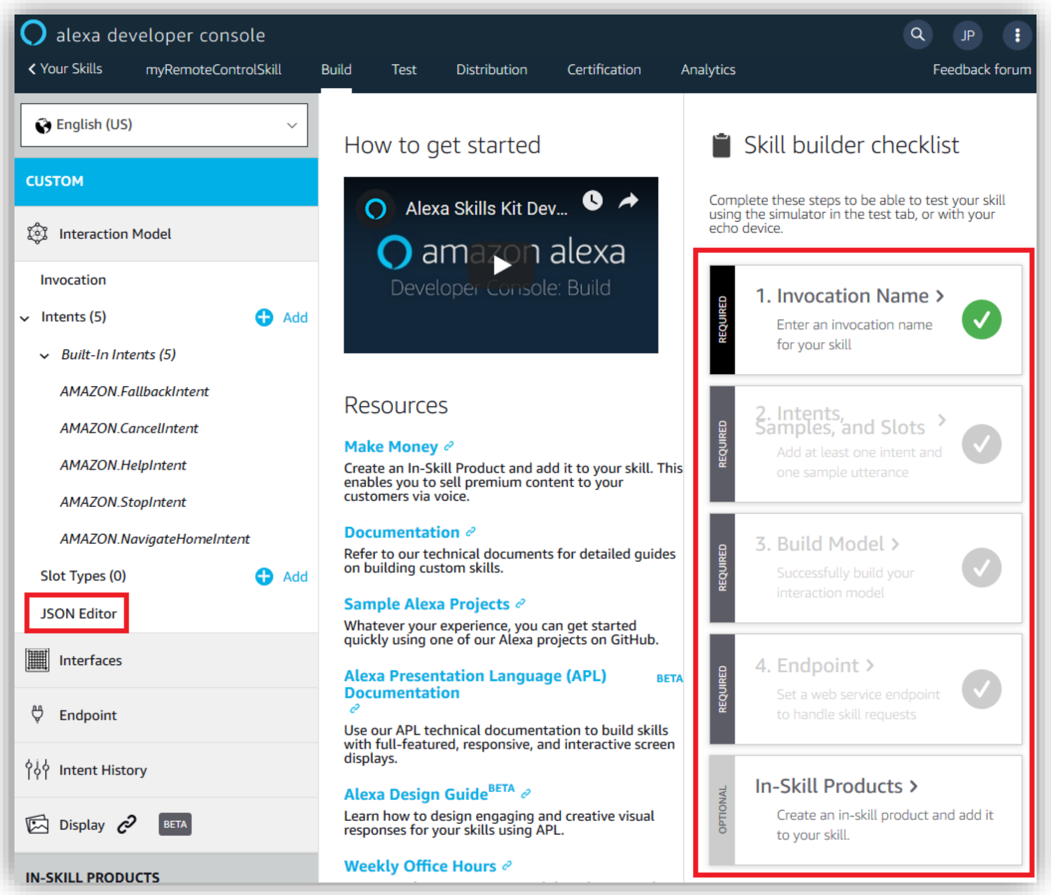
7. Select to "Start from scratch" as a template and click on the choose button.


Choose a template

Select a quick start template to get started with a predefined skill or simply "Start from scratch"

Choose
SELECTED
Start from scratch
Design a unique experience for your users and define your custom model from scratch.
Fact Skill
Provided a list of interesting facts about a topic, Alexa will select a fact at random and tell it to the user

8. Notice on the right, the Skill builder checklist that needs to be fulfilled for the Skill to be ready. You can watch the Alexa skills kit developer video to learn how you might do this step. We will use the provided script for this lab.



Make Money
Create an In-Skill Product and add it to your skill. This enables you to sell premium content to your customers via voice.

Documentation
Refer to our technical documents for detailed guides on building custom skills.

Sample Alexa Projects
Whatever your experience, you can get started quickly using one of our Alexa projects on GitHub.

Alexa Presentation Language (APL) Documentation BETA
Use our APL technical documentation to build skills with full-featured, responsive, and interactive screen displays.

Alexa Design Guide BETA
Learn how to design engaging and creative visual responses for your skills using APL.

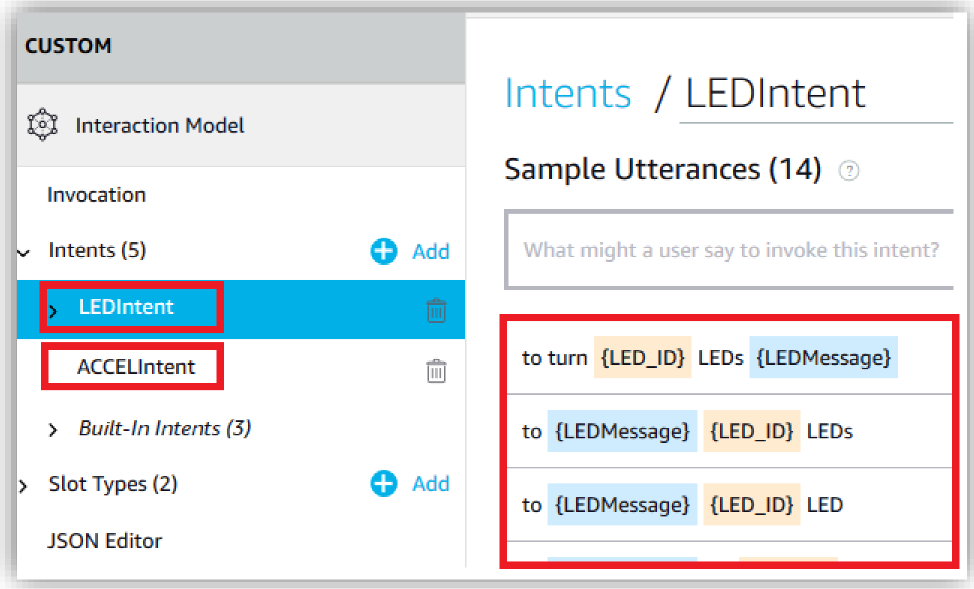
Weekly Office Hours

Complete these steps to be able to test your skill using the simulator in the test tab, or with your echo device.

9. Click on the JSON Editor.
10. Delete the default JSON data and copy-paste the content of **Alexa_RC_json_skill.json** file located in this lab folder.
11. Click on **Save Model.**

The JSON file is the translation of what you can manually add using the user-interface menu on the left. Notice that there is now are a **LEDIntent** and **ACCELIntent** on your Intent's list, and the **invocation name** is set to "*my board*". You can manually add or delete Intents for your application using the Graphical Interface or writing it on the JSON Editor.

Navigate through your intents and identify the different Utterances your Alexa will be able to recognize to manipulate the LED and Accelerometer.


CUSTOM

Interaction Model

Invocation

Intents (5) Add

LEDIntent

ACCELIntent

Built-In Intents (3)

Slot Types (2) Add

JSON Editor

Intents / LEDIntent

Sample Utterances (14)

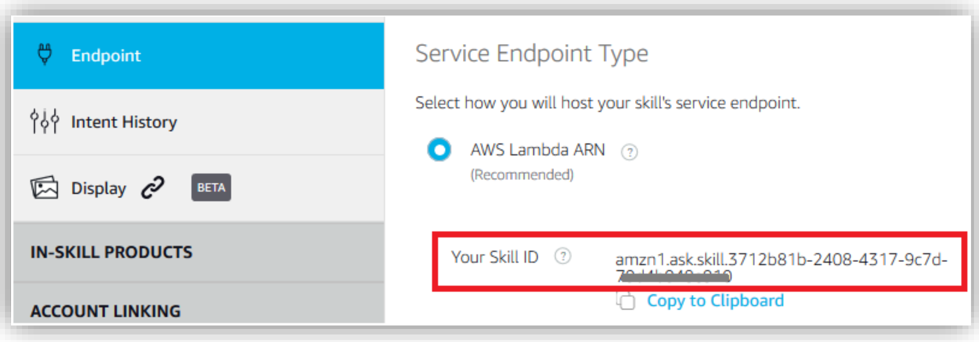
What might a user say to invoke this intent?

to turn {LED_ID} LEDs {LEDMessage}

to {LEDMessage} {LED_ID} LEDs

to {LEDMessage} {LED_ID} LED

12. Click on **Build Model**.
13. Select the "Endpoint" section and select the AWS Lambda ARN, then copy the **Alexa Skill ID**, it's something like "amzn1.ask.skill[...]"


Endpoint

Intent History

Display BETA

IN-SKILL PRODUCTS

ACCOUNT LINKING

Service Endpoint Type

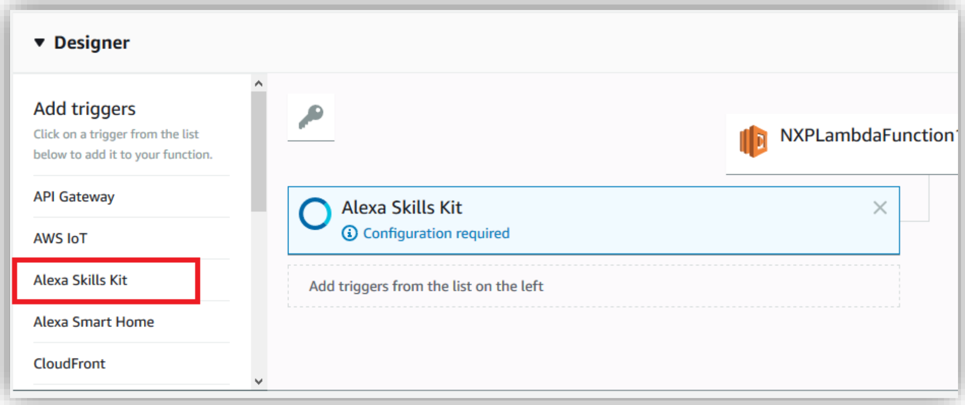
Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

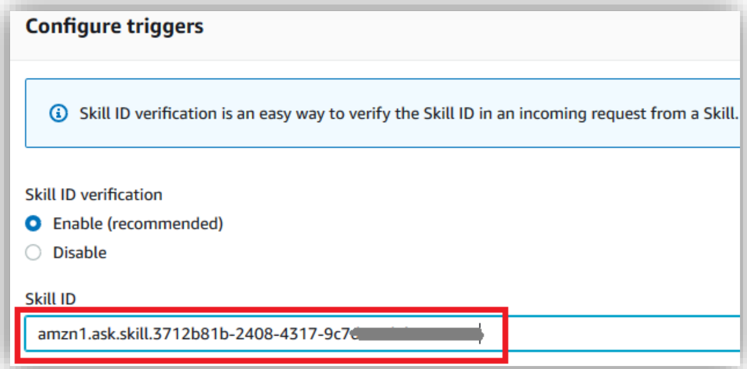
Your Skill ID amzn1.ask.skill.3712b81b-2408-4317-9c7d-[redacted]

Copy to Clipboard

14. Leaving the Skills tab opened to come back to in a minute. Open your NXP Lambda function (https://console.aws.amazon.com/lambda/home).
15. Click on your Lambda function name to display the "Add triggers" section.
16. On the "Add triggers" section, add an **Alexa Skills Kit** trigger



17. Scroll down to "Configure triggers" and paste the Alexa Skill ID that you just copied from your Skill (step 13).

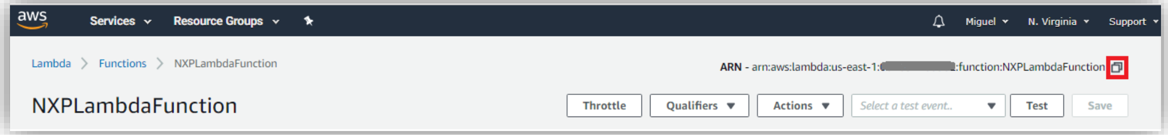

Configure triggers

Skill ID verification is an easy way to verify the Skill ID in an incoming request from a Skill.

Skill ID verification
[✓ Enable (recommended)]
Disable

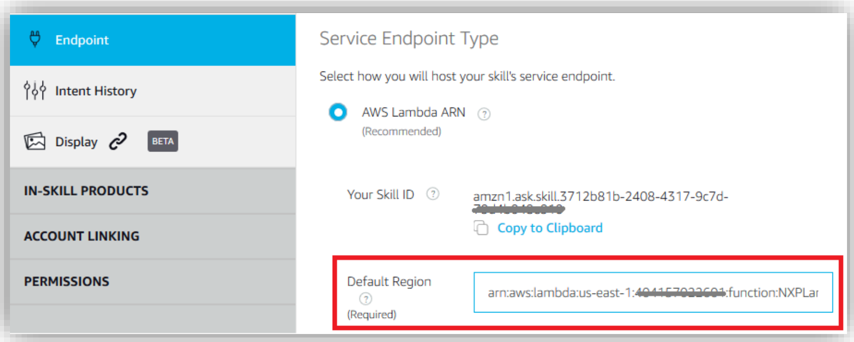
Skill ID
amzn1.ask.skill.3712b81b-2408-4317-9c7[redacted]

18. Click on **Add** at the bottom right and then **Save** the NXPLambdaFunction
19. Copy the ARN (Amazon Resource Name) of your Lambda function, it's located on the top right of the Lambda function (Just click on the icon next to it). You will need this on the Skill.



ARN - arn:aws:lambda:us-east-1:[redacted]:function:NXPLambdaFunction

20. Back to your skill (https://developer.amazon.com), set the Endpoint of the Alexa service on the Default Region and paste the copied skill ID in the Default Region window.



Service Endpoint Type

Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

Your Skill ID amzn1.ask.skill.3712b81b-2408-4317-9c7d-[redacted]

Copy to Clipboard

Default Region (Required) arn:aws:lambda:us-east-1:[redacted]:function:NXPLar

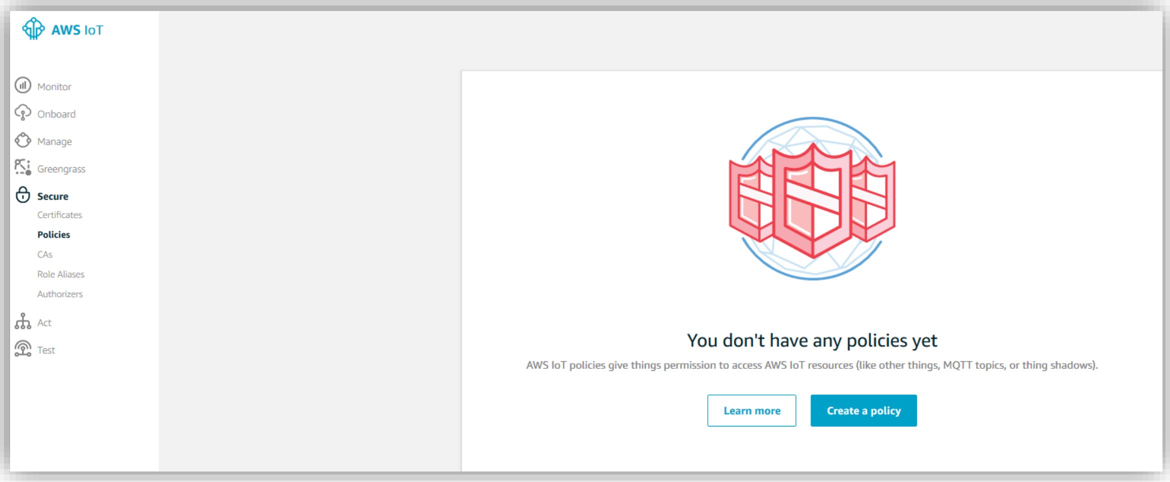
21. Click on **Save Endpoints.**

SUCCESS! We have just created and configured the Alexa Skill and the AWS Lambda function. Now let's create the 'Thing' that your NXP board will use.

4.3 Create an IoT thing, policy, private key and certificates for your device

AWS IoT policies grant or deny access to AWS IoT resources such as things, thing shadows, and MQTT topics. We need to grant access to our Thing by attaching an AWS IoT policy to the certificate associate with our thing.

1. Open the AWS IoT console website (https://console.aws.amazon.com/iot/home)
2. In the left navigation pane, choose Secure, and then choose Policies.
   If you do not have any policies registered in your account, the "**You don't have any policies yet**" page is displayed. If you see this page, choose to **Create a policy**.

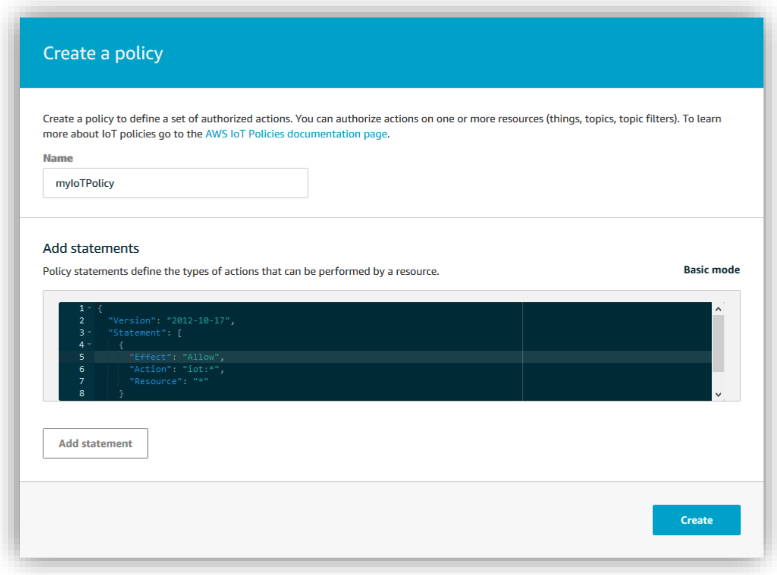


You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows).

3. Type **myIoTPolicy** in the Name text box to identify your policy. In the **Add statements** section, click **Advanced mode.** Modify lines 5, 6, and 7 with the following content. If you receive an error fix your typing to make it exactly like this.

   ```
   {
     "Effect": "Allow",
     "Action": "iot:*",
     "Resource": "*"
   }
   ```
4. Choose **Create**



Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name

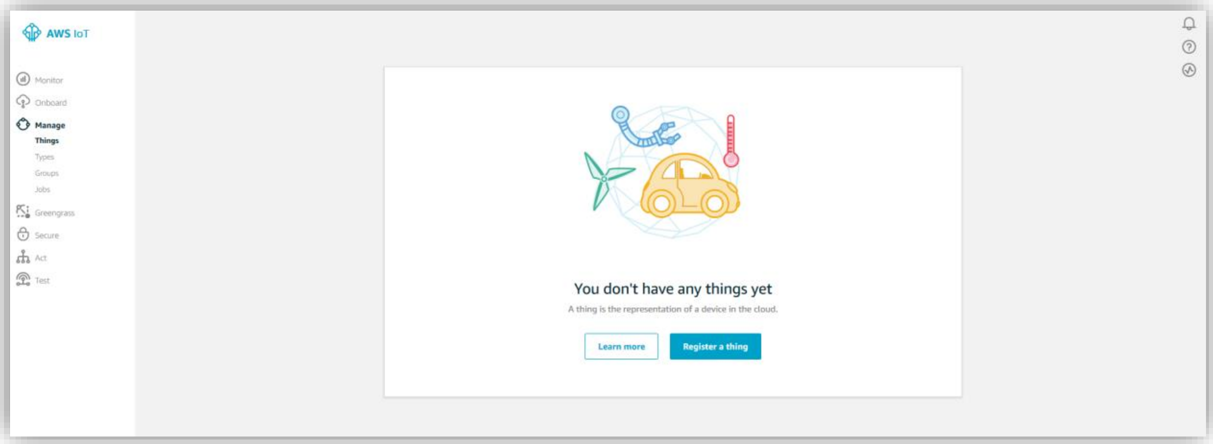
myIoTPolicy

Add statements

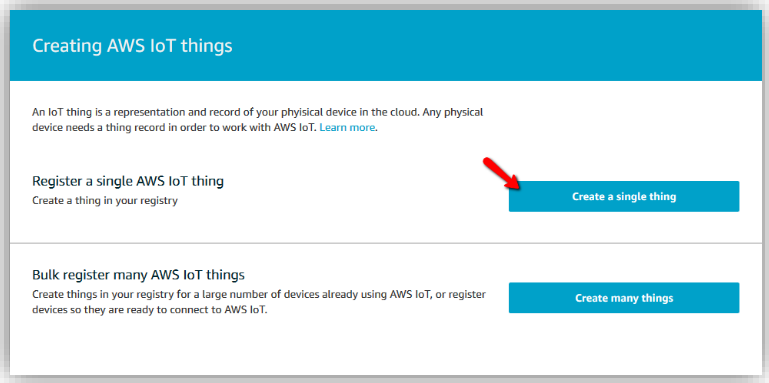
Policy statements define the types of actions that can be performed by a resource.

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

5. In the left navigation pane, choose Manage, and then choose Things. If you do not have any IoT things registered in your account, the **You don't have any things yet** page is displayed. If you see this page, choose **Register a thing**. Otherwise, choose to **Create**.



6. On the *Creating AWS IoT things* page, choose to **Create a single thing**



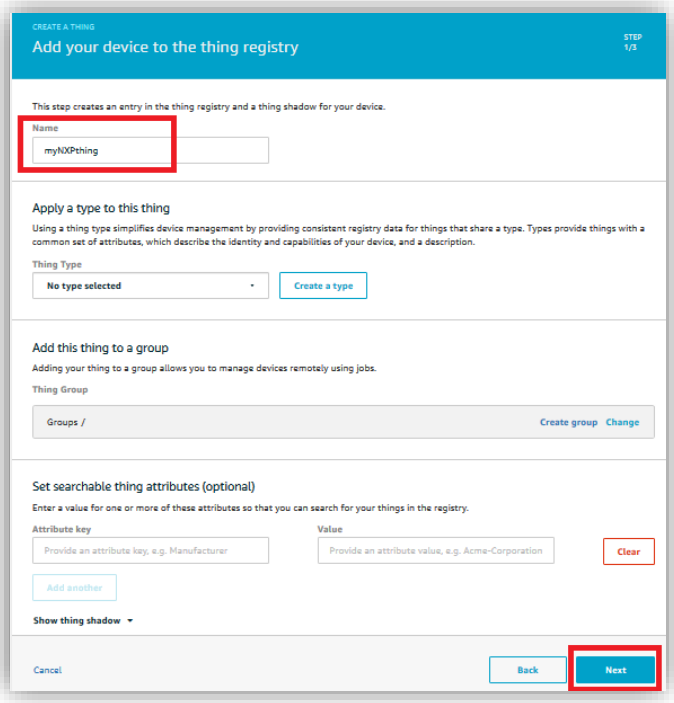
An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

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

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

7. On the *Add your device to the thing registry* page, type **myNXPthing** in the name text box, then click **Next.**

Note: This will be the **name of our "thing"**, this is the name we will use on the **Lambda function.**


CREATE A THING

Add your device to the thing registry

STEP 1/3

This step creates an entry in the thing registry and a thing shadow for your device.

Name: myNXPthing

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

Add this thing to a group

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

Thing Group: Groups /

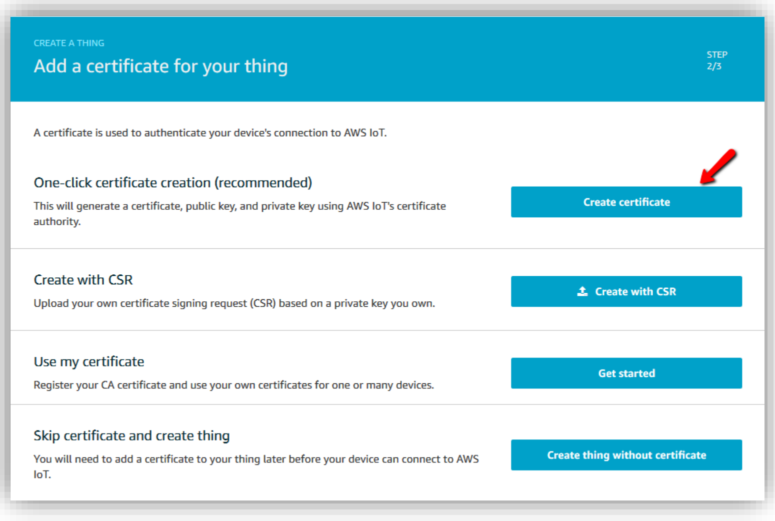
Set searchable thing attributes (optional)

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

8. On the ‘*Add a certificate for your thing*’ page, under **One-click certificate creation**, choose to **Create certificate**


CREATE A THING

Add a certificate for your thing

STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)

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

Create with CSR

Upload your own certificate signing request (CSR) based on a private key you own.

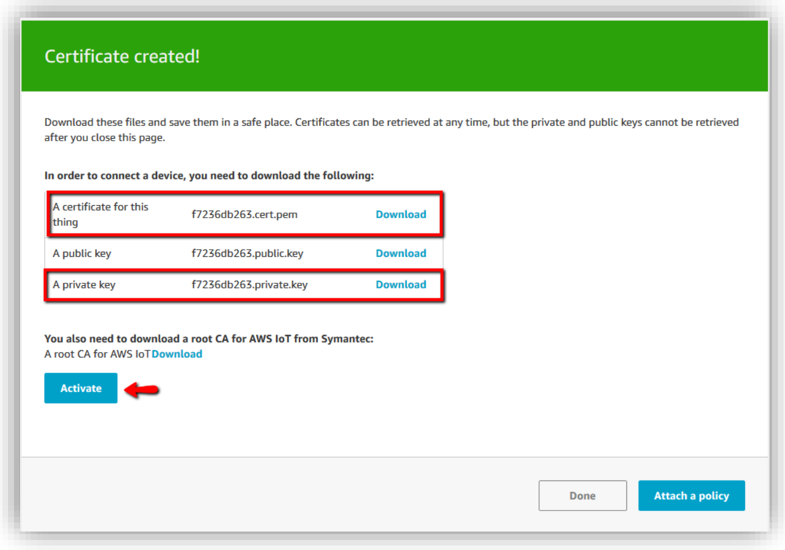
Use my certificate

Register your CA certificate and use your own certificates for one or many devices.

Skip certificate and create thing

You will need to add a certificate to your thing later before your device can connect to AWS IoT.

9. Download your **private key** and **certificate** by choosing the Download links for each. Choose **Activate** to switch on your certificate. Then click Attach a policy


Certificate created!

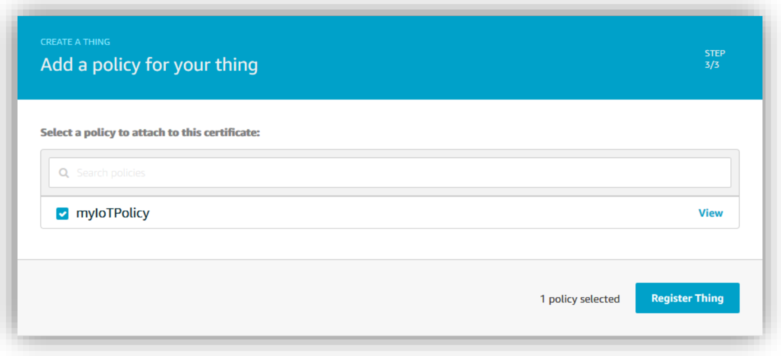
Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	f7236db263.cert.pem	Download
A public key	f7236db263.public.key	Download
A private key	f7236db263.private.key	Download

You also need to download a root CA for AWS IoT from Symantec:
A root CA for AWS IoT Download

10. Select the checkbox next to myIoTPolicy (that we created before) and choose Register Thing


CREATE A THING
Add a policy for your thing
STEP 3/3

Select a policy to attach to this certificate:

SUCCESS! Now, we have a Thing with its credentials.

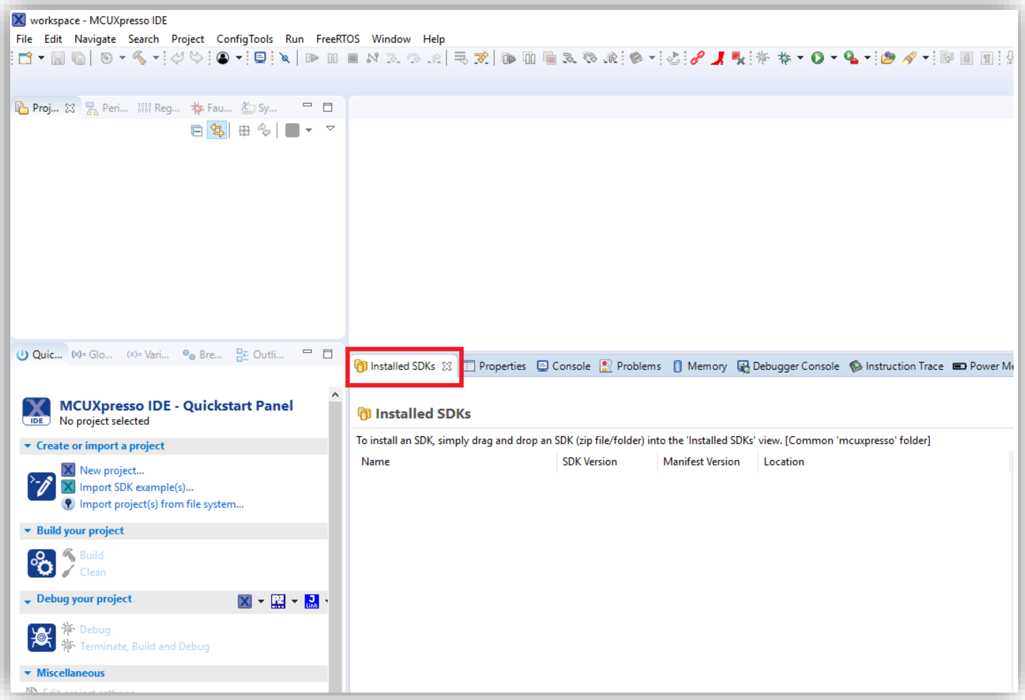
5. LPCxpresso55S69 configuration

This section includes instructions to load and configure the AWS project that will be loaded to your LPCXpresso55S69.

5.1 Import the MCUXpresso SDK

1. Open the MCUXpresso IDE

2. Select "Installed SDKs" tab within the MCUXpresso IDE windows


workspace - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run FreeRTOS Window Help
Proj... Peri... Reg... Fau... Sy...
Quic... Glo... Vari... Bre... Outli...
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug
Miscellaneous
Installed SDKs
Properties
Console
Problems
Memory
Debugger Console
Instruction Trace
Power M

Installed SDKs

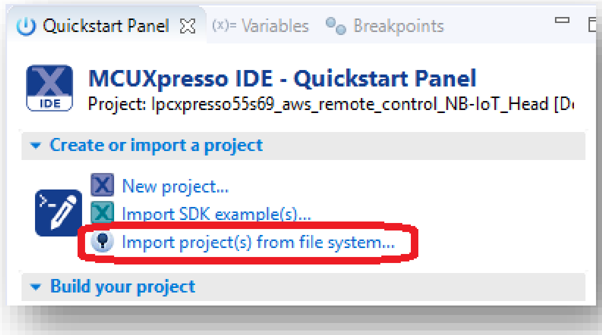
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

Name	SDK Version	Manifest Version	Location

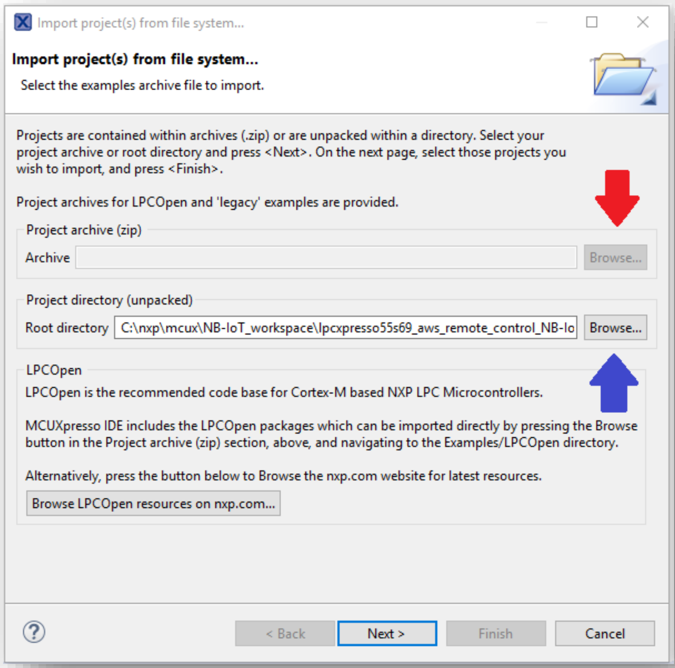
3. Open Windows Explorer, and drag and drop the previously downloaded "**SDK_X.X.X_LPCXpresso55S69.zip**" SDK file into the **Installed SDKs** view.
4. Click OK to the confirmation window.
5. The installed SDK will appear in the **Installed SDKs** tab.

5.2 Import and configure the aws_remote_control_celullar example project

1. On the **Quickstart Panel**, select “Import project(s) from file system…”


Quickstart Panel
(x)= Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: lpcxpresso55s69_aws_remote_control_NB-IoT_Head [D
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project

2. Click on the **Browse…** button (red one if you got an archive, blue one if the project is already unpacked) , go to the archive or repository of the project and click **next.**


Import project(s) from file system...
Import project(s) from file system...
Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archives for LPCOpen and 'legacy' examples are provided.

Project archive (zip)

Archive

Project directory (unpacked)

Root directory C:\nxp\mcux\NB-IoT_workspace\lpcxpresso55s69_aws_remote_control_NB-Io

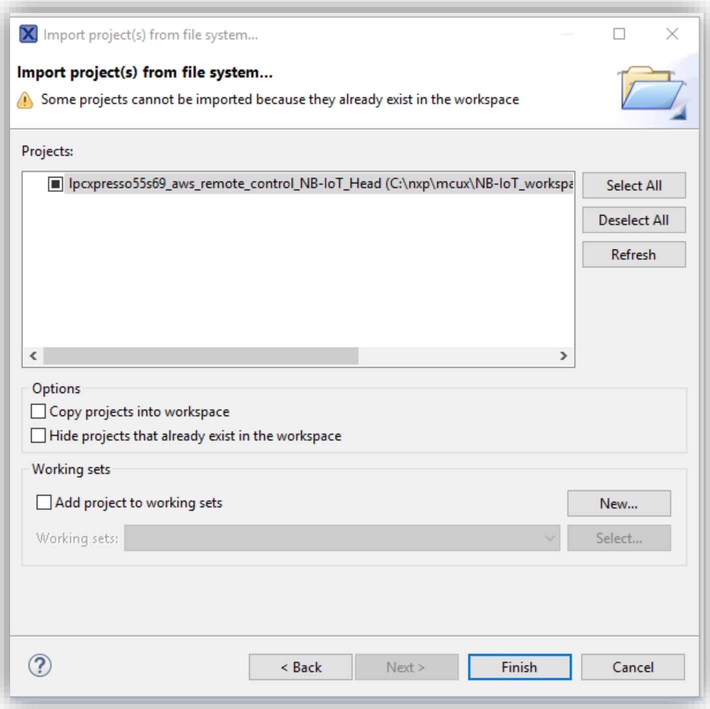
LPCOpen

LPCOpen is the recommended code base for Cortex-M based NXP LPC Microcontrollers.

MCUXpresso IDE includes the LPCOpen packages which can be imported directly by pressing the Browse button in the Project archive (zip) section, above, and navigating to the Examples/LPCOpen directory.

Alternatively, press the button below to Browse the nxp.com website for latest resources.

3. Select the AWS Remote Control demo project and click on **Finish.** If you cannot select the project, this is because it is already existing in your workspace. To workaround this, import the project from **File→ Import… → General → Existing Projects into Workspace**.


Import project(s) from file system...

Import project(s) from file system...

Some projects cannot be imported because they already exist in the workspace

Projects:

- [■] lpcxpresso55s69_aws_remote_control_NB-IoT_Head (C:\nxp\mcux\NB-IoT_workspa

Options

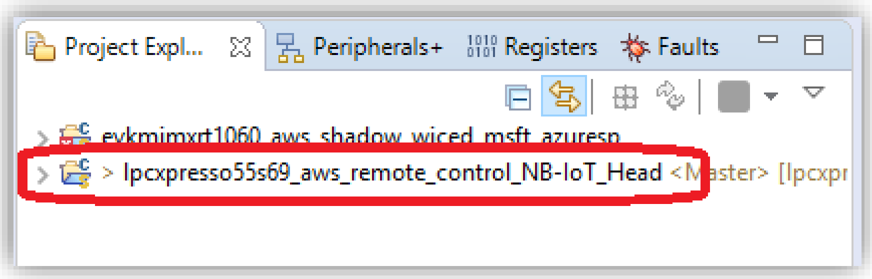
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

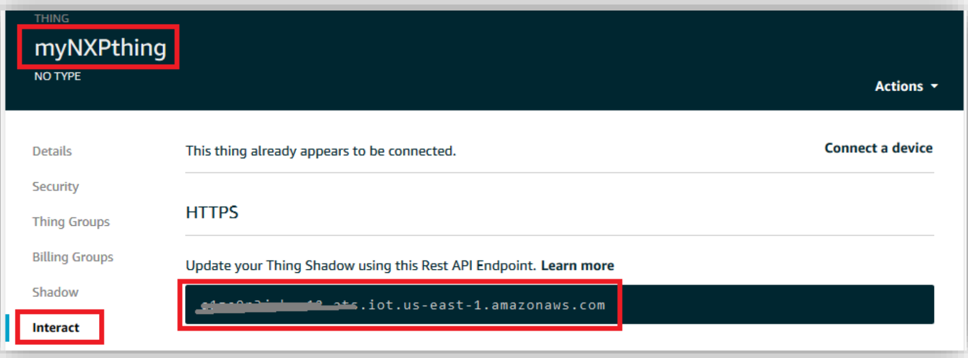
4. You should now see the project loaded into the workspace as shown below



evkmimxrt1060_aws_shadow_wiced_msft_azuresp

> lpcxpresso55s69_aws_remote_control_NB-IoT_Head <Master> [lpcxpr

5. Configuring the aws 'thing' credentials.
   Locate the file .../amazon-freertos/demos/**aws_clientcredential.h** and configure:
   - Broker **endpoint** –
     - Copy the Rest API endpoint URL from:
       AWS Console → IoT Core → Manage → Things → 'myNXPthing' → Interact
       https://console.aws.amazon.com/iot/home?region=us-east-1



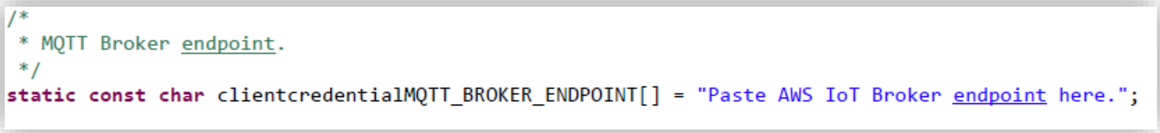
This thing already appears to be connected.

HTTPS

Update your Thing Shadow using this Rest API Endpoint. Learn more

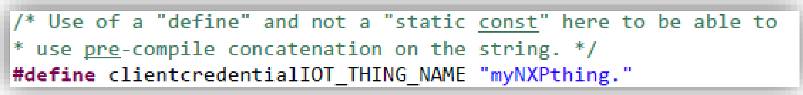
```
[redacted].iot.us-east-1.amazonaws.com
```

- Paste the AWS Rest API endpoint URL here, replacing everything between the quotation marks



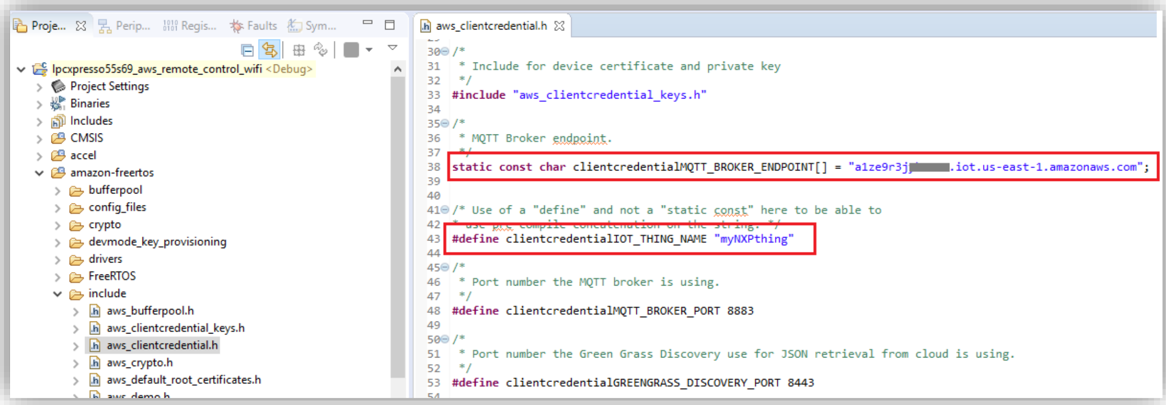
```
/*
 * MQTT Broker endpoint.
 */
static const char clientcredentialMQTT_BROKER_ENDPOINT[] = "Paste AWS IoT Broker endpoint here.";
```

- **Thing name** – “myNXPthing”



```
/* Use of a "define" and not a "static const" here to be able to
* use pre-compile concatenation on the string. */
#define clientcredentialIOT_THING_NAME "myNXPthing."
```

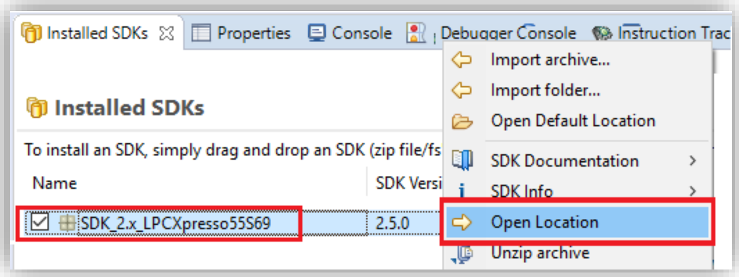
It should look like this:



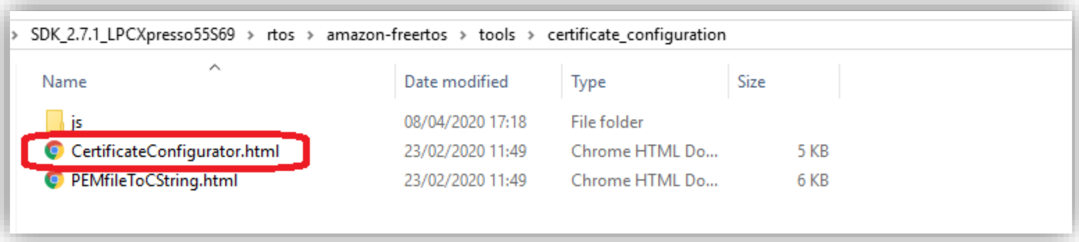
```
/*
 * Include for device certificate and private key
 */
#include "aws_clientcredential_keys.h"

/*
 * MQTT Broker endpoint.
 */
static const char clientcredentialMQTT_BROKER_ENDPOINT[] = "a1ze9r3j[redacted].iot.us-east-1.amazonaws.com";


/* Use of a "define" and not a "static const" here to be able to
 * use pre-compile concatenation on the string. */
#define clientcredentialIOT_THING_NAME "myNXPthing"

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

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

6. Unzip the SDK file and locate the **Certificate Configurator** in the following path: …/SDK_x.x.x_LPCXpresso55S69/rtos/amazon-freertos/tools/certificate_configuration/ **CertificateConfigurator.html**

   Note: You can locate your *SDK_x.x.x_LPCXpresso55S69.zip* file by right-clicking on your **Installed SDK** item and click on "Open Location."

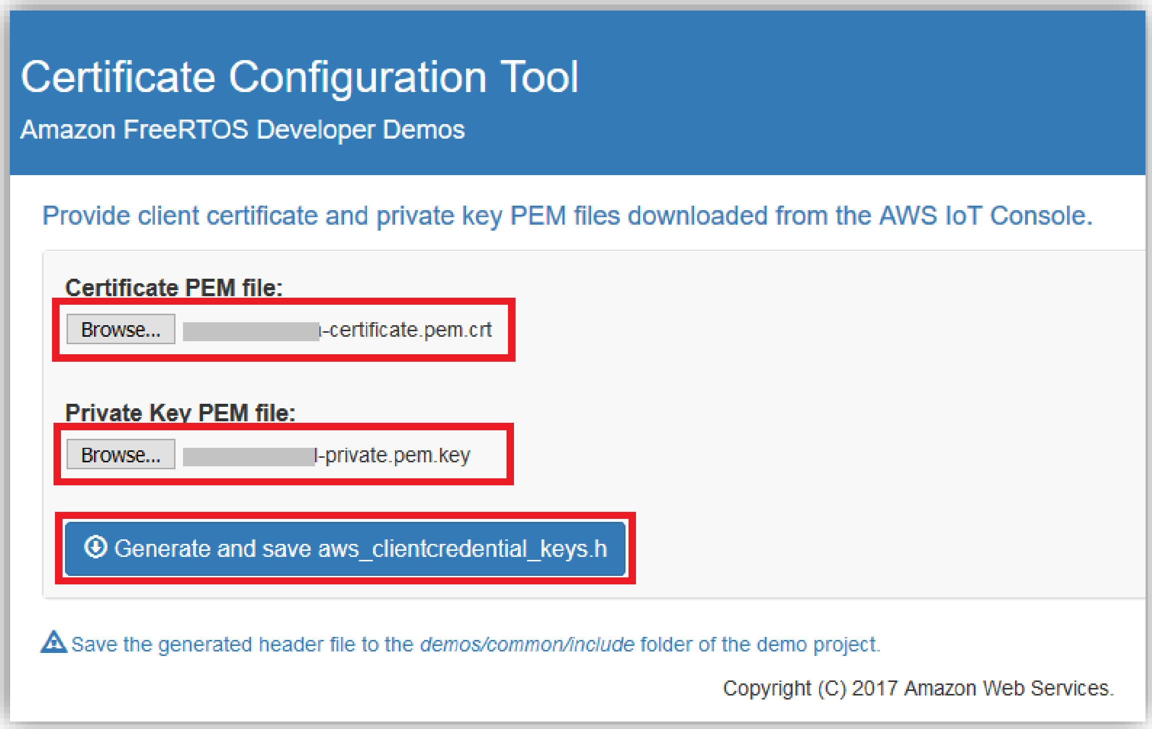


7. Open the **CertificateConfigurator** file; this will generate a "aws_clientcredential_keys.h" header file, based on the certificate files you previously downloaded. It's located it at: <SDK>\rtos\amazon-freertos\tools\certificate_configuration


SDK_2.7.1_LPCXpresso55S69 › rtos › amazon-freertos › tools › certificate_configuration

Name	Date modified	Type	Size
js	08/04/2020 17:18	File folder	
CertificateConfigurator.html	23/02/2020 11:49	Chrome HTML Do...	5 KB
PEMfileToCString.html	23/02/2020 11:49	Chrome HTML Do...	6 KB

8. Browse to the **Certificate** and **Key** files you previously downloaded from your Thing (myNXPthing) and click on "Generate and save "aws_clientcredential_keys.h"



Certificate Configuration Tool

Amazon FreeRTOS Developer Demos

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

Certificate PEM file:

Browse... [redacted]-certificate.pem.crt

Private Key PEM file:

Browse... [redacted]-private.pem.key

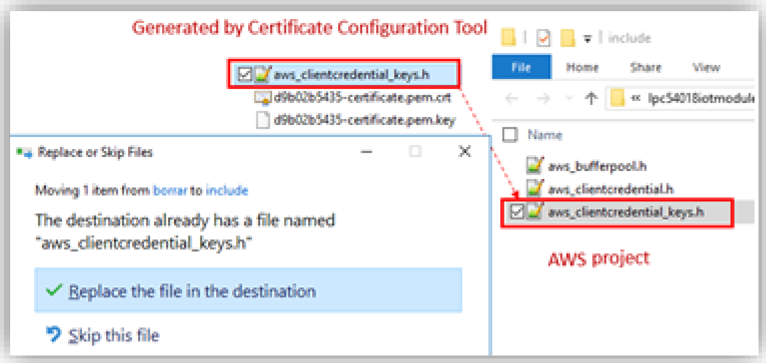
Generate and save aws_clientcredential_keys.h

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

Copyright (C) 2017 Amazon Web Services.

Note: If no file is downloaded, allow blocked content in the explorer.

9. Copy the newly generated "**aws_clientcredential_keys.h**" and replace the one from your aws_remote_control_cellular project. The file is located at: .../lpcxpresso55s69_aws_remote_control_wifi\amazon-freertos\demos



Generated by Certificate Configuration Tool

aws_clientcredential_keys.h
d9b02b5435-certificate.pem.crt
d9b02b5435-certificate.pem.key

Replace or Skip Files

Moving 1 item from borrar to include

The destination already has a file named "aws_clientcredential_keys.h"

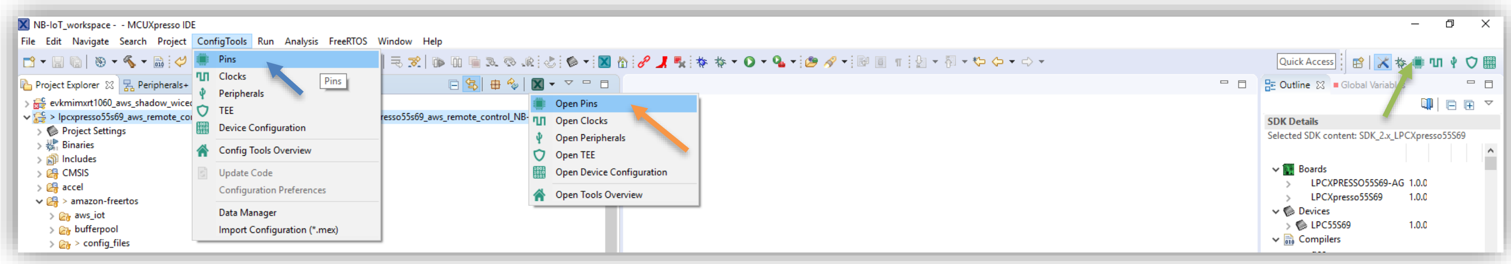
Replace the file in the destination

Skip this file

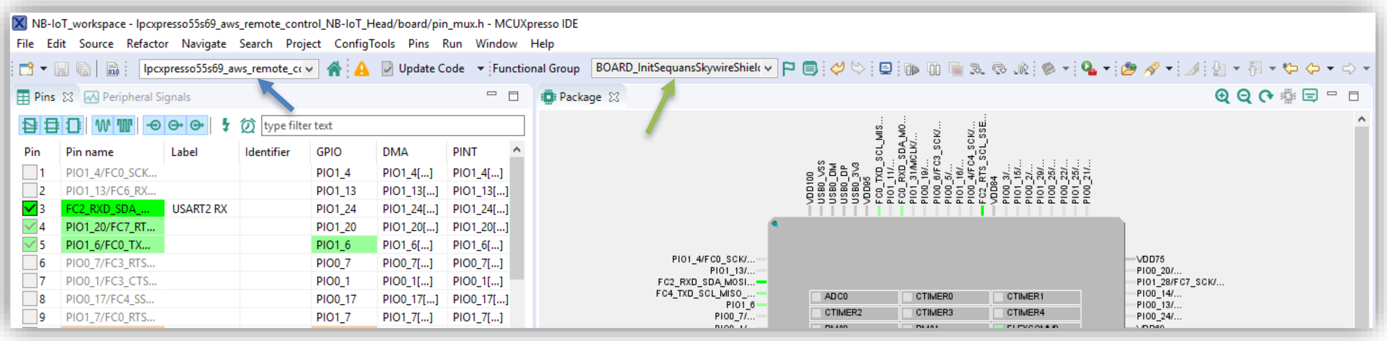
That's it! Your thing is now ready to communicate with AWS through the IoT Core.

5.3 Pin configuration for the Cellular module

The Cellular module is using 6 pins for the communication (excluding the power supply).
To check the I/O configuration, you can use the **MCUXpresso Pin tool configurator** by clicking one of the icon shown below

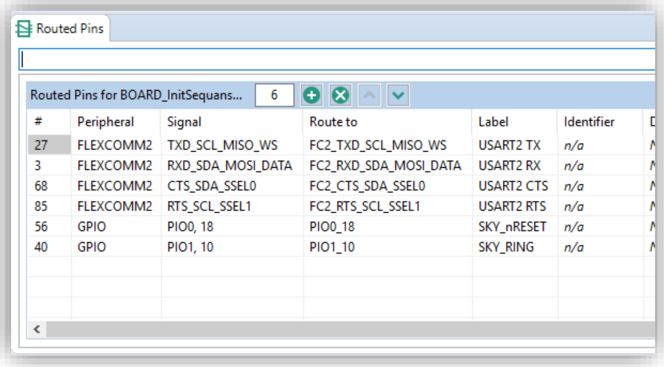


Once the Pin tool opened, make sure the correct project is selected (Blue arrow) and choose the **BOARD_InitSequansSkywireShield** functional group (Green arrow)


NB-IoT_workspace - lpcxpresso55s69_aws_remote_control_NB-IoT_Head/board/pin_mux.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Pins Run Window Help
lpcxpresso55s69_aws_remote_co
Update Code
Functional Group
BOARD_InitSequansSkywireShield
Pins
Peripheral Signals
Package
type filter text

	Pin	Pin name	Label	Identifier	GPIO	DMA	PINT
	1	PIO1_4/FC0_SCK...			PIO1_4	PIO1_4[...]	PIO1_4[...]
	2	PIO1_13/FC6_RX...			PIO1_13	PIO1_13[...]	PIO1_13[...]
✓	3	FC2_RXD_SDA_...	USART2 RX		PIO1_24	PIO1_24[...]	PIO1_24[...]
✓	4	PIO1_20/FC7_RT...			PIO1_20	PIO1_20[...]	PIO1_20[...]
✓	5	PIO1_6/FC0_TX...			PIO1_6	PIO1_6[...]	PIO1_6[...]
	6	PIO0_7/FC3_RTS...			PIO0_7	PIO0_7[...]	PIO0_7[...]
	7	PIO0_1/FC3_CTS...			PIO0_1	PIO0_1[...]	PIO0_1[...]
	8	PIO0_17/FC4_SS...			PIO0_17	PIO0_17[...]	PIO0_17[...]
	9	PIO1_7/FC0_RTS...			PIO1_7	PIO1_7[...]	PIO1_7[...]

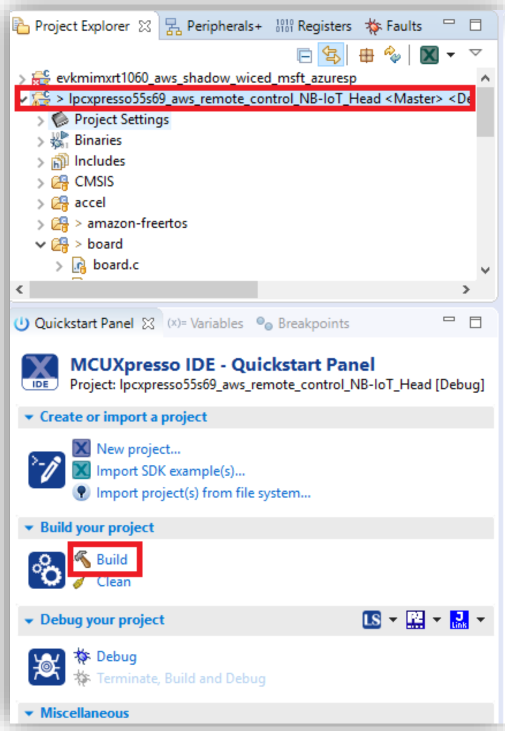
At the bottom, you can now see the routed pins showing you which peripheral is in use, which signal and which I/O.



#	Peripheral	Signal	Route to	Label	Identifier
27	FLEXCOMM2	TXD_SCL_MISO_WS	FC2_TXD_SCL_MISO_WS	USART2 TX	n/a
3	FLEXCOMM2	RXD_SDA_MOSI_DATA	FC2_RXD_SDA_MOSI_DATA	USART2 RX	n/a
68	FLEXCOMM2	CTS_SDA_SSEL0	FC2_CTS_SDA_SSEL0	USART2 CTS	n/a
85	FLEXCOMM2	RTS_SCL_SSEL1	FC2_RTS_SCL_SSEL1	USART2 RTS	n/a
56	GPIO	PIO0, 18	PIO0_18	SKY_nRESET	n/a
40	GPIO	PIO1, 10	PIO1_10	SKY_RING	n/a

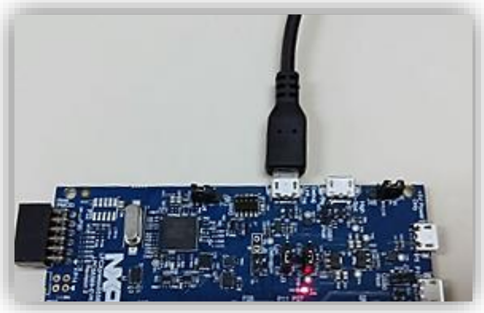
5.4 Compile and download the aws_remote_control_cellular project

1. Select your project folder and click on Build.

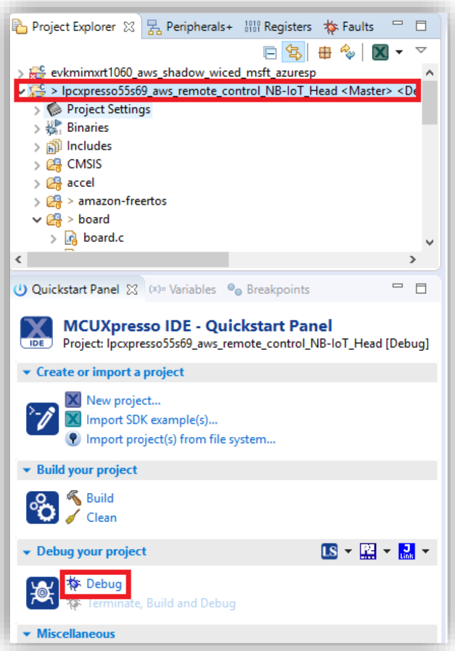

Project Explorer
Peripherals+
Registers
Faults
evkmimxrt1060_aws_shadow_wiced_msft_azuresp
> lpcxpresso55s69_aws_remote_control_NB-IoT_Head <Master> <De
Project Settings
Binaries
Includes
CMSIS
accel
> amazon-freertos
> board
board.c
Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: lpcxpresso55s69_aws_remote_control_NB-IoT_Head [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug
Miscellaneous

2. Wait until the project is built.
3. Verify the build finished without any errors.
4. Mount the Avnet shield with the Monarch Go module on the LPCXpresso55S69-EVK

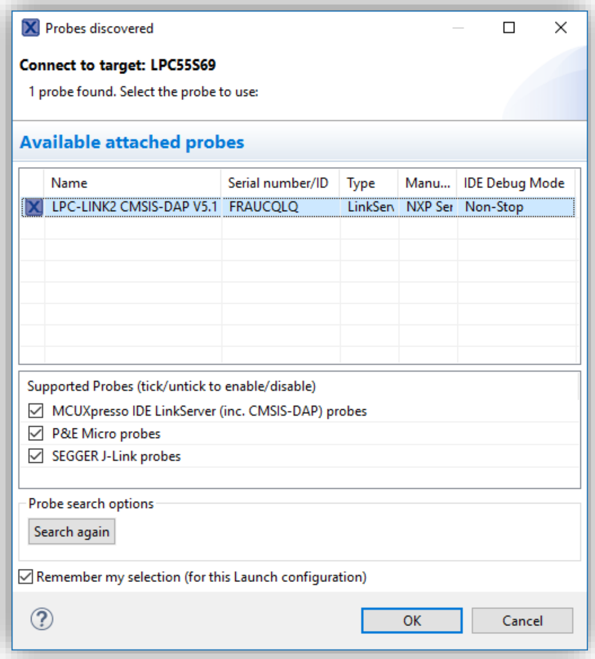
5. Connect the Debug Link port from your LPCXpresso55S69 to your PC using a micro-USB cable.



6. Program your LPCXpresso55S69 by clicking on “**Debug**” from the ‘Quickstart Panel.’


Project Explorer
Peripherals+
Registers
Faults
evkmimxrt1060_aws_shadow_wiced_msft_azuresp
> lpcxpresso55s69_aws_remote_control_NB-IoT_Head <Master> <De
Project Settings
Binaries
Includes
CMSIS
accel
> amazon-freertos
> board
board.c
Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: lpcxpresso55s69_aws_remote_control_NB-IoT_Head [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug
Miscellaneous

7. Select your probe and click “OK” to start programming your board.



Connect to target: LPC55S69

1 probe found. Select the probe to use:

Available attached probes

Name	Serial number/ID	Type	Manu...	IDE Debug Mode
LPC-LINK2 CMSIS-DAP V5.1	FRAUCQLQ	LinkSer	NXP Se	Non-Stop

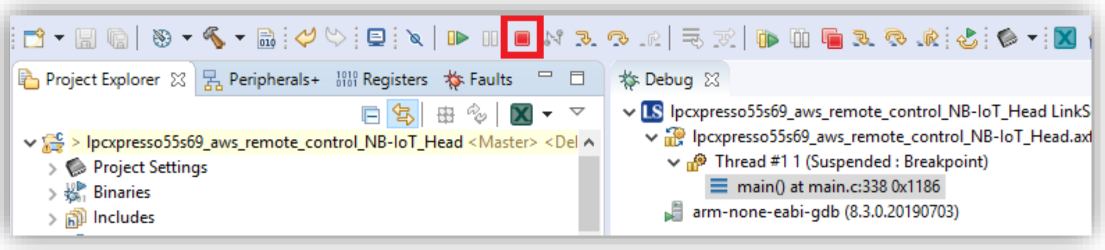
Supported Probes (tick/untick to enable/disable)

- [x] MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- [x] P&E Micro probes
- [x] SEGGER J-Link probes

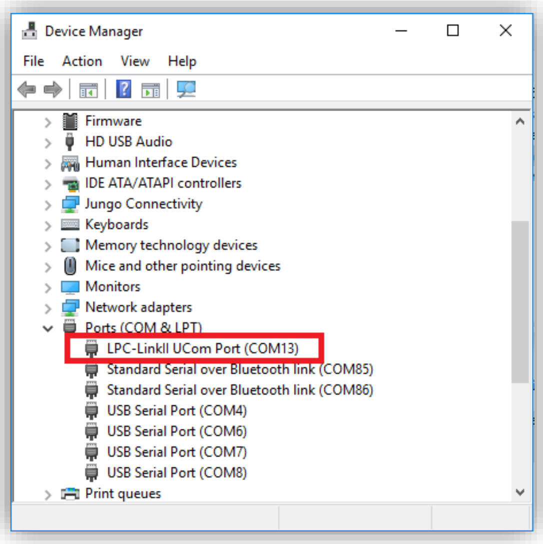
Probe search options

- [x] Remember my selection (for this Launch configuration)

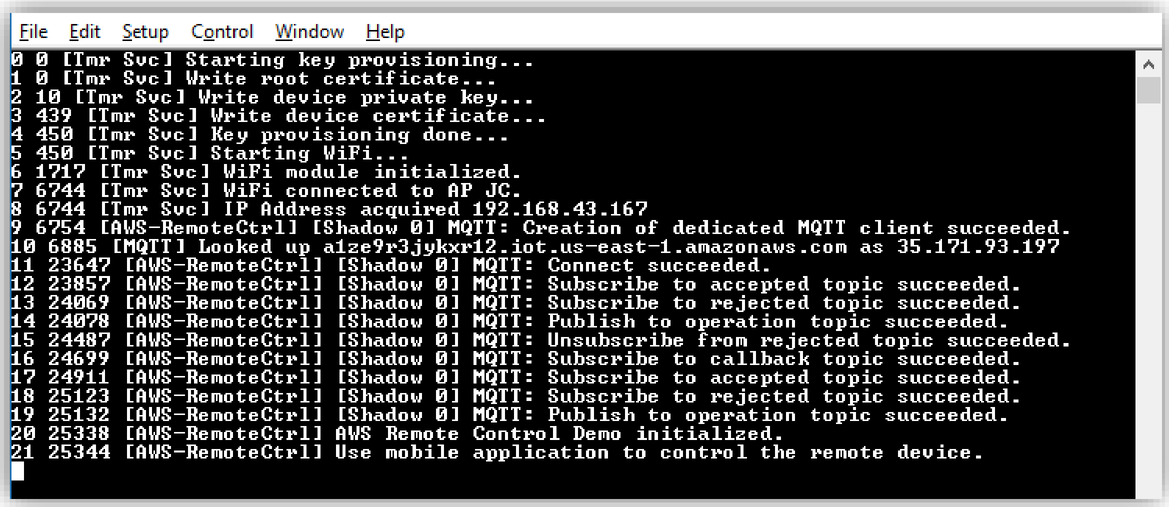
8. Wait until the program is successfully downloaded.
9. Stop the debugging session


Project Explorer
Peripherals+
Registers
Faults
Debug
lpcxpresso55s69_aws_remote_control_NB-IoT_Head <Master> <Del
Project Settings
Binaries
Includes
lpcxpresso55s69_aws_remote_control_NB-IoT_Head LinkS
lpcxpresso55s69_aws_remote_control_NB-IoT_Head.axf
Thread #1 1 (Suspended : Breakpoint)
main() at main.c:338 0x1186
arm-none-eabi-gdb (8.3.0.20190703)

10. Open the “Device Manager” on your Windows PC and identify the COM port of your board.



11. Open a Terminal emulator software (TeraTerm, PuTTY) and connect to the COM port using the following settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit,
    - No flow control
12. Press the reset button on your board.



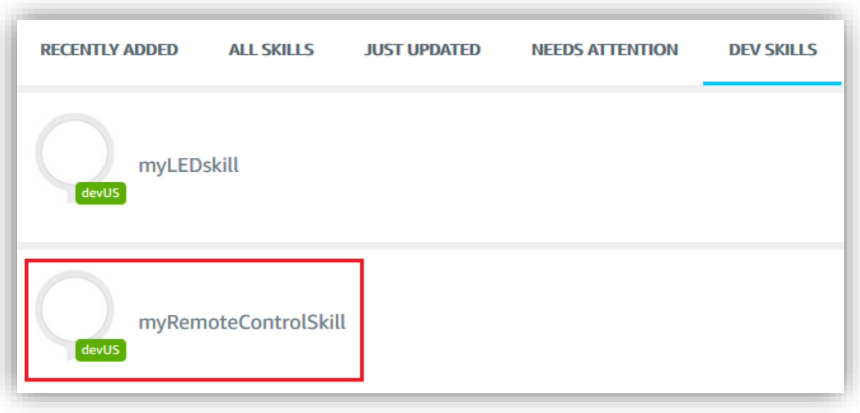
```
0 0 [Tmr Svc] Starting key provisioning...
1 0 [Tmr Svc] Write root certificate...
2 10 [Tmr Svc] Write device private key...
3 439 [Tmr Svc] Write device certificate...
4 450 [Tmr Svc] Key provisioning done...
5 450 [Tmr Svc] Starting WiFi...
6 1717 [Tmr Svc] WiFi module initialized.
7 6744 [Tmr Svc] WiFi connected to AP JC.
8 6744 [Tmr Svc] IP Address acquired 192.168.43.167
9 6754 [AWS-RemoteCtrl] [Shadow 0] MQTT: Creation of dedicated MQTT client succeeded.
10 6885 [MQTT] Looked up a1ze9r3jykxr12.iot.us-east-1.amazonaws.com as 35.171.93.197
11 23647 [AWS-RemoteCtrl] [Shadow 0] MQTT: Connect succeeded.
12 23857 [AWS-RemoteCtrl] [Shadow 0] MQTT: Subscribe to accepted topic succeeded.
13 24069 [AWS-RemoteCtrl] [Shadow 0] MQTT: Subscribe to rejected topic succeeded.
14 24078 [AWS-RemoteCtrl] [Shadow 0] MQTT: Publish to operation topic succeeded.
15 24487 [AWS-RemoteCtrl] [Shadow 0] MQTT: Unsubscribe from rejected topic succeeded.
16 24699 [AWS-RemoteCtrl] [Shadow 0] MQTT: Subscribe to callback topic succeeded.
17 24911 [AWS-RemoteCtrl] [Shadow 0] MQTT: Subscribe to accepted topic succeeded.
18 25123 [AWS-RemoteCtrl] [Shadow 0] MQTT: Subscribe to rejected topic succeeded.
19 25132 [AWS-RemoteCtrl] [Shadow 0] MQTT: Publish to operation topic succeeded.
20 25338 [AWS-RemoteCtrl] AWS Remote Control Demo initialized.
21 25344 [AWS-RemoteCtrl] Use mobile application to control the remote device.
```

13. Your Thing is ready and accessible through the NXP ***AWS Remote Control*** Android application and ***Alexa*** voice control.

6. Alexa Echo configuration

This section provides steps to enable your newly created skill in your Alexa environment.
This is how you can Configure Alexa Application to enable your new Skill:

1. Download the “Amazon Alexa app” from Google Play Store
2. Open the **Alexa App** with your Amazon Developer credentials (https://alexa.amazon.com).
3. Go to "*Skills*" from the left-side menu and click on "*Your Skills*" on the **top right.**
4. Select "*DEV SKILLS*" and you should find the "myRemoteControl*skill*" skill that you created.



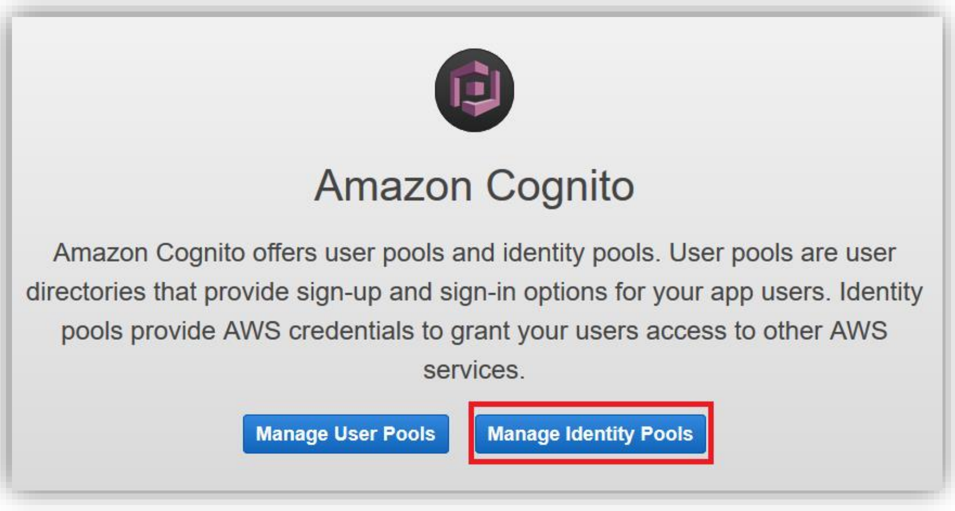
7. Enable and configure Android application

This section describes the procedure to enable your android application to communicate and control your AWS thing.

7.1 Enable Android application to communicate with our Thing

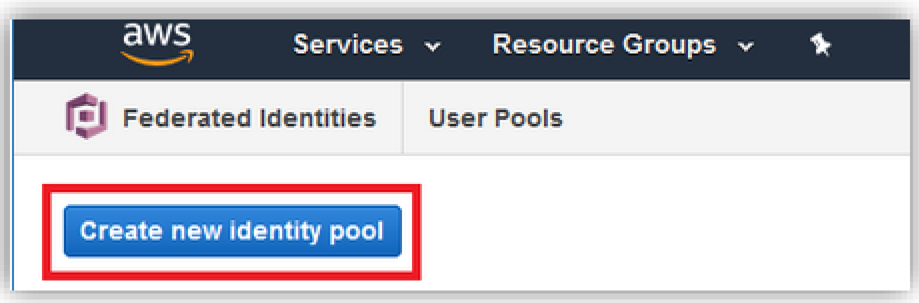
Cognito configuration will allow your Android application to communicate with your thing. Steps are also described in the project under ../doc/readme.txt.

1. In the Amazon Cognito Console https://console.aws.amazon.com/cognito/ select "Manage Identity Pools" and "Create new identity pool".



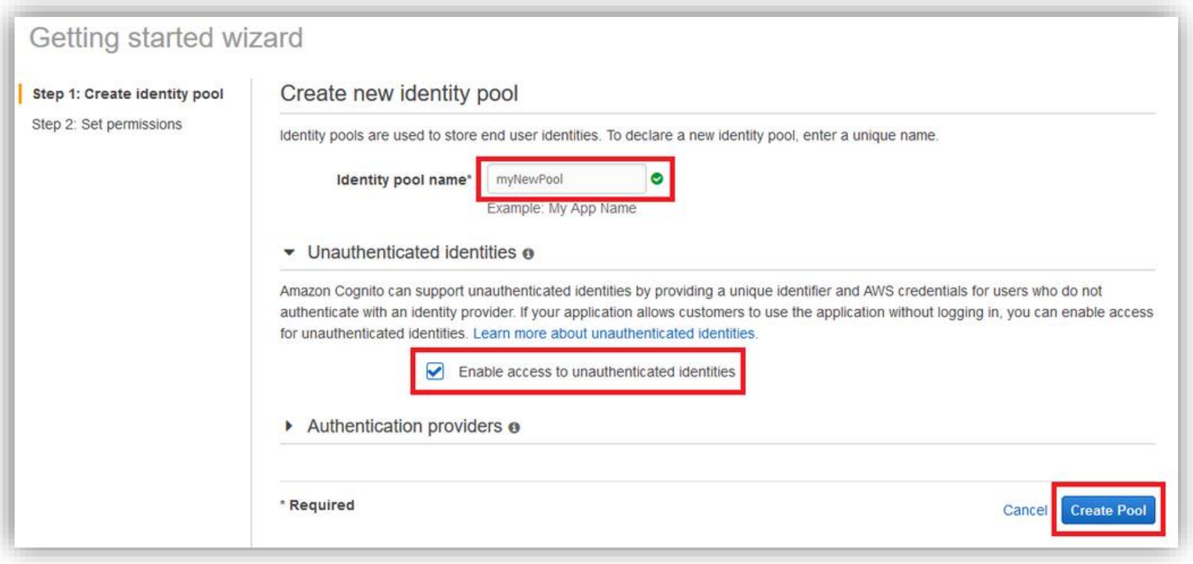
Amazon Cognito

Amazon Cognito offers user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools provide AWS credentials to grant your users access to other AWS services.



2. Ensure Enable access to unauthenticated identities is checked. This allows the sample application to assume the unauthenticated role associated with this identity pool.

   Note: to keep this example simple it makes use of unauthenticated users in the identity pool. This can be used for getting started and prototypes but unauthenticated users should typically only be given read-only permissions in production applications.


Getting started wizard

Step 1: Create identity pool

Step 2: Set permissions

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* myNewPool

Example: My App Name

Unauthenticated identities

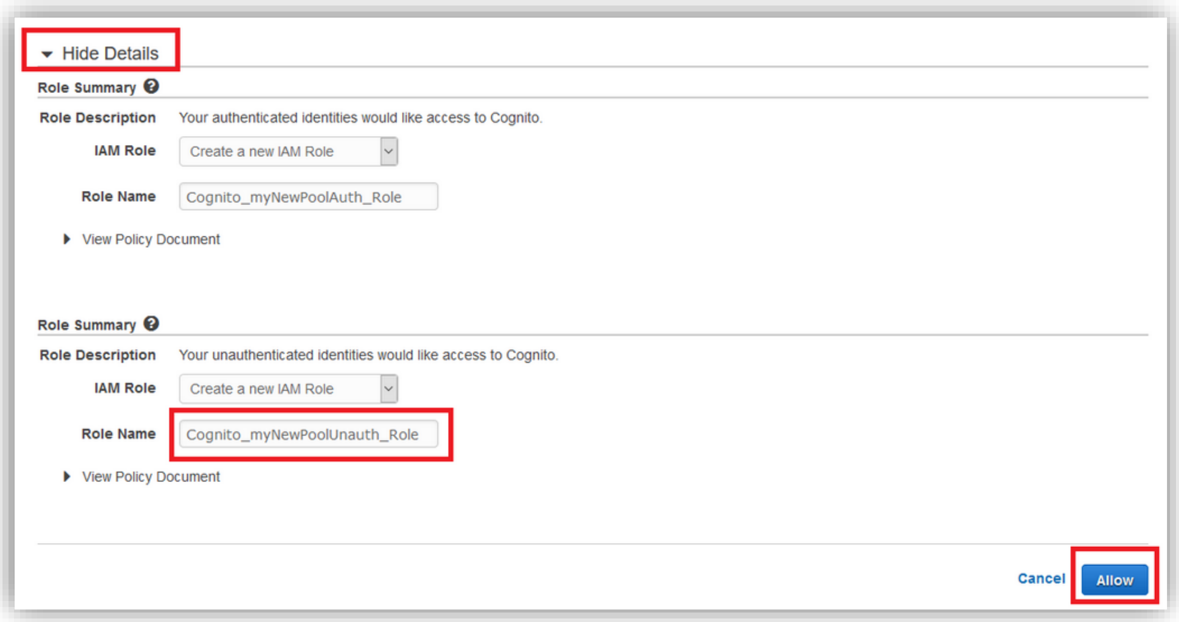
Amazon Cognito can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider. If your application allows customers to use the application without logging in, you can enable access for unauthenticated identities. Learn more about unauthenticated identities.

[✓] Enable access to unauthenticated identities

Authentication providers

* Required

(verify the name of the Unauthenticated pool):



Role Summary

Role Description Your authenticated identities would like access to Cognito.

IAM Role Create a new IAM Role

Role Name Cognito_myNewPoolAuth_Role

View Policy Document

Role Summary

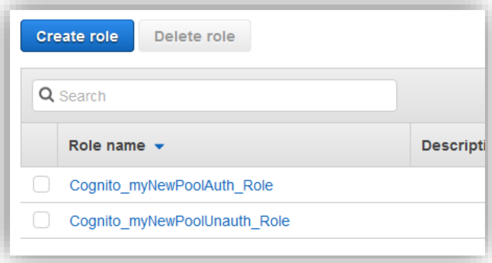
Role Description Your unauthenticated identities would like access to Cognito.

IAM Role Create a new IAM Role

Role Name Cognito_myNewPoolUnauth_Role

View Policy Document

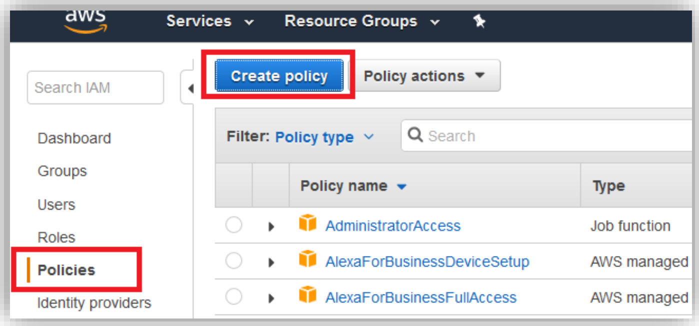
3. As part of creating the identity pool, Cognito will setup two roles in Identity and Access Management (IAM) https://console.aws.amazon.com/iam/home#roles. These will be named something similar to: "Cognito_PoolNameAuth_Role" and "Cognito_PoolNameUnauth_Role". We will use Unauthanticated role.



	Role name	Descripti
☐	Cognito_myNewPoolAuth_Role	
☐	Cognito_myNewPoolUnauth_Role	

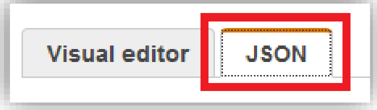
4. Select the Policies tab in the left list

5. Create a policy to be attached into "Cognito_PoolNameUnauth_Role" through "Policies" menu, selecting "Create policy".



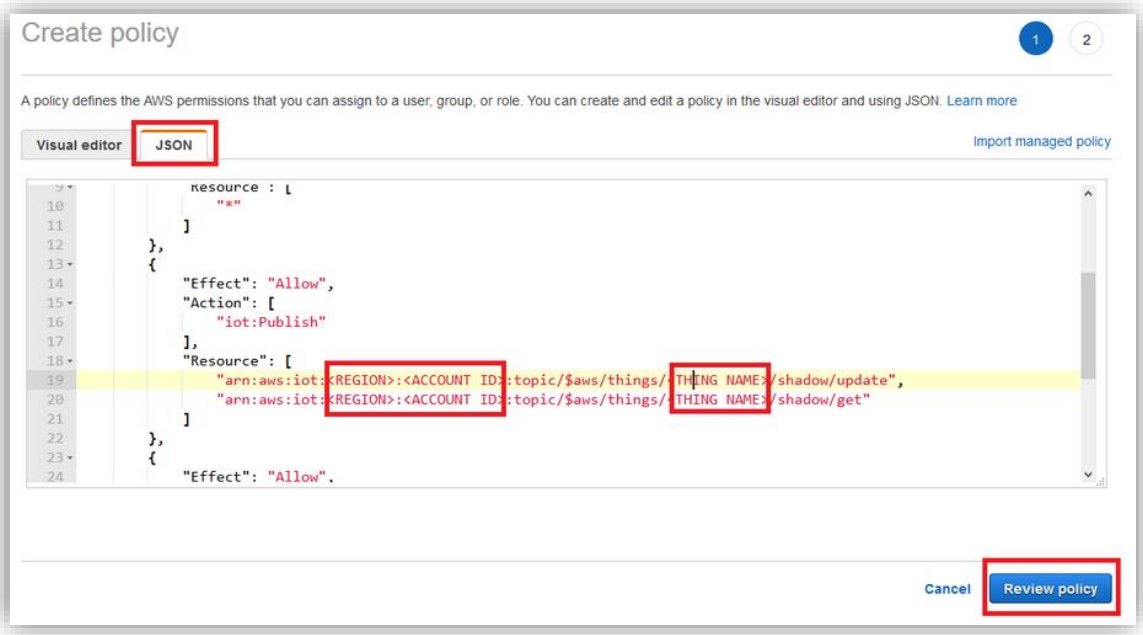
		Policy name	Type
○	▸	AdministratorAccess	Job function
○	▸	AlexaForBusinessDeviceSetup	AWS managed
○	▸	AlexaForBusinessFullAccess	AWS managed

6. Select the JSON tab,



7. Delete the existing content and copy the example policy below into "Policy Document" JSON field and name it for example "<THING NAME>Policy". Replace <REGION>, <ACCOUNT ID> and <THING NAME> with your respective values (don't forget to remove the < > symbols). This policy allows the application to get and update the shadow used in this sample.
   The note below shows you where to find the required values to put in here.

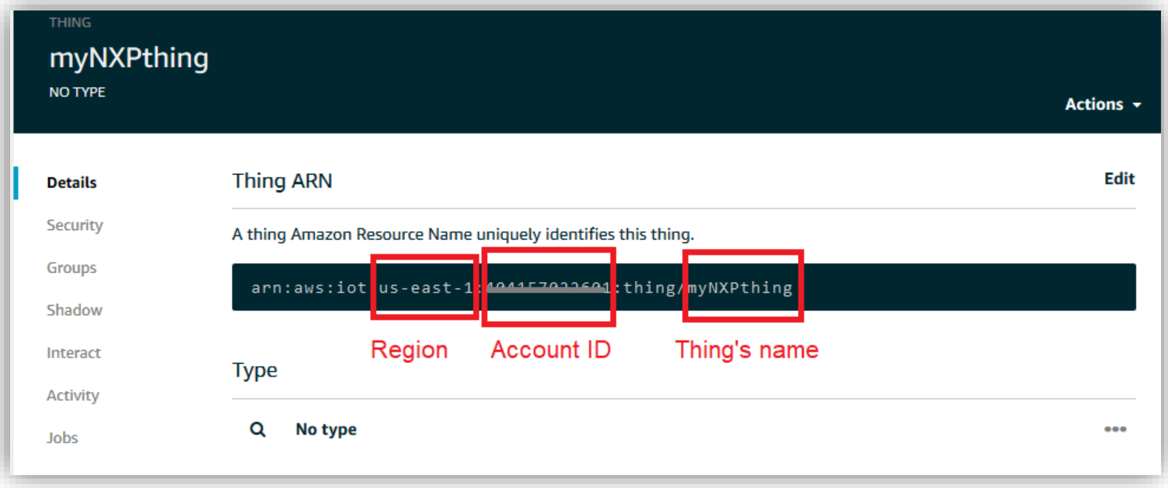
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:<REGION>:<ACCOUNT ID>:topic/$aws/things/<THING NAME>/shadow/update",
                "arn:aws:iot:<REGION>:<ACCOUNT ID>:topic/$aws/things/<THING NAME>/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```


Create policy

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

```
                "Resource": [
                    "*"
                ]
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iot:Publish"
                ],
                "Resource": [
                    "arn:aws:iot:<REGION>:<ACCOUNT ID>:topic/$aws/things/<THING NAME>/shadow/update",
                    "arn:aws:iot:<REGION>:<ACCOUNT ID>:topic/$aws/things/<THING NAME>/shadow/get"
                ]
            },
            {
                "Effect": "Allow",
```

Note: You can locate these parameters under your Thing's ARN (https://console.aws.amazon.com/iot/home). Select Manage from the left column and then click on 'myNXPthing'.


THING

myNXPthing

NO TYPE

Thing ARN

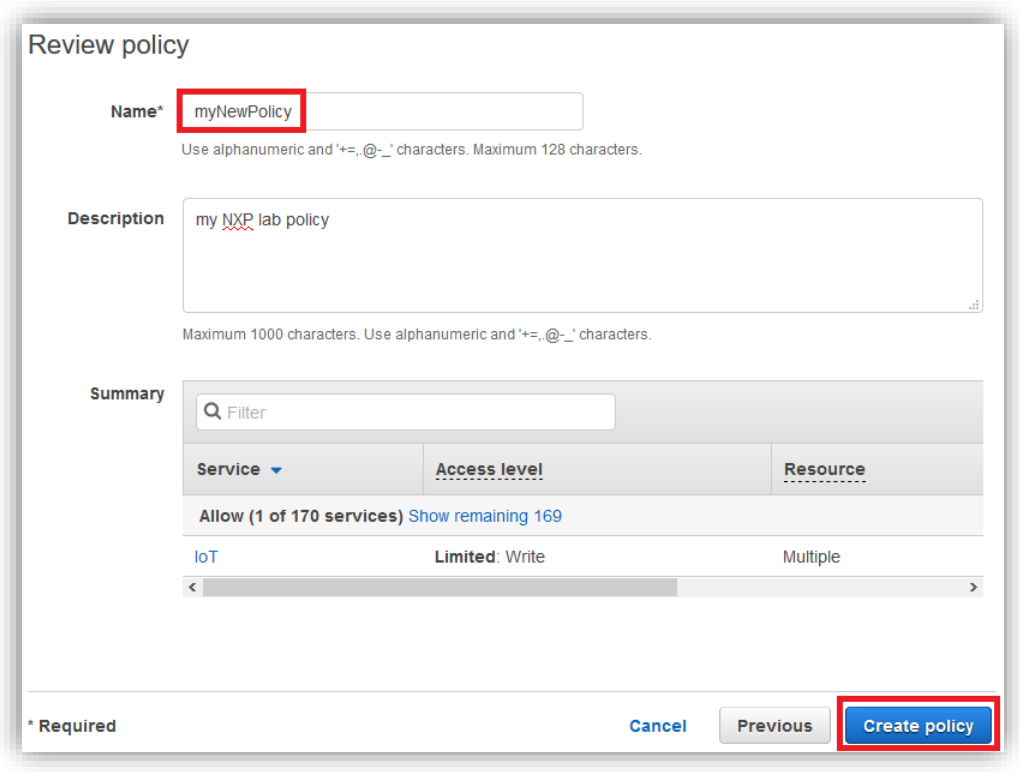
A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:us-east-1:[redacted]:thing/myNXPthing
```

Type

No type

8. **Set a name** to your new policy and click on **Create policy.**

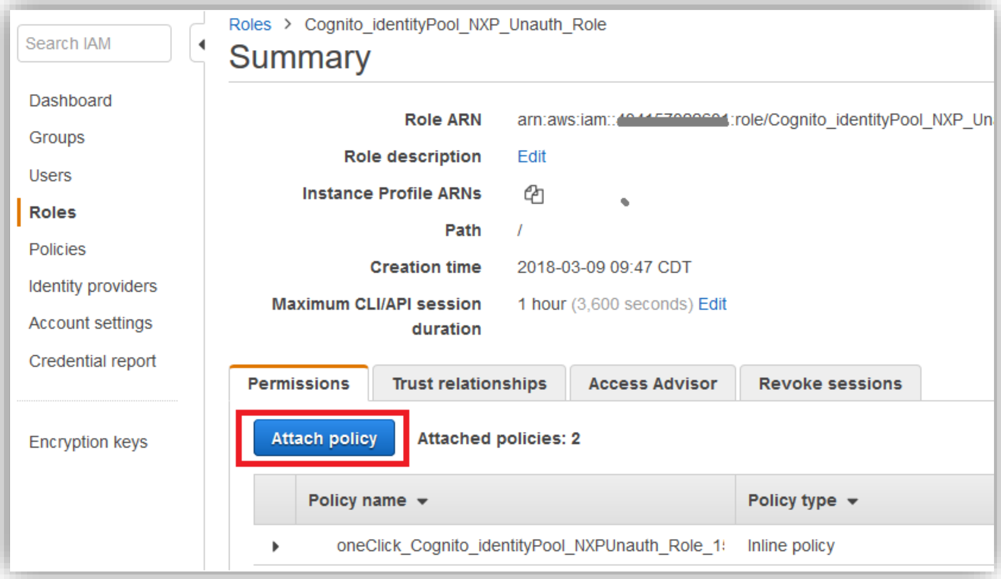

Review policy

Name*	myNewPolicy
	Use alphanumeric and '+=,.@-_' characters. Maximum 128 characters.
Description	my NXP lab policy
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Summary

Service	Access level	Resource
Allow (1 of 170 services) Show remaining 169		
IoT	**Limited**: Write	Multiple

9. Newly created policy now needs to be attached to the unauthenticated role which has permissions to access the required AWS IoT APIs:
    a. Open "**Cognito_PoolNameUnauth_Role**" under "Roles" menu.
    b. In the "Permissions" tab, select "Attach policies" to view list of all AWS policies.
    c. Type in the Filter policies window “myNewPolicy” and check the small box to the left of the policy
    d. Click on "Attach policy" button.

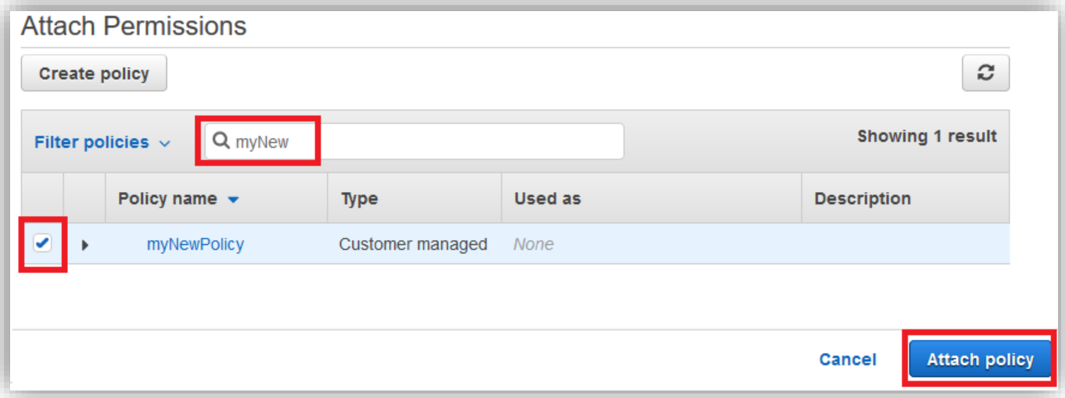

Roles > Cognito_identityPool_NXP_Unauth_Role

Summary

Role ARN	arn:aws:iam::[redacted]:role/Cognito_identityPool_NXP_Un
Role description	Edit
Instance Profile ARNs	
Path	/
Creation time	2018-03-09 09:47 CDT
Maximum CLI/API session duration	1 hour (3,600 seconds) Edit

Attached policies: 2

Policy name	Policy type
oneClick_Cognito_identityPool_NXPUnauth_Role_1	Inline policy


Attach Permissions

myNew

Showing 1 result

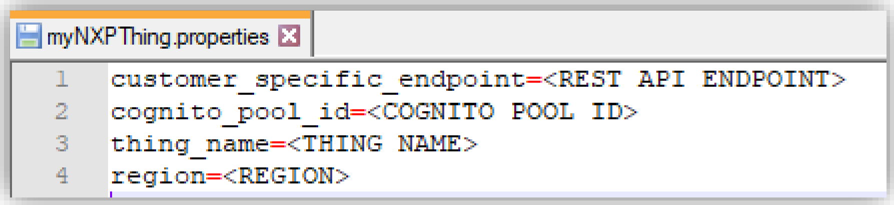
	Policy name	Type	Used as	Description
✓	myNewPolicy	Customer managed	None	

More information on AWS IAM roles and policies can be found here: http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage.html

More information on AWS IoT policies can be found here: http://docs.aws.amazon.com/iot/latest/developerguide/authorization.html

7.2 Configure and install Android application

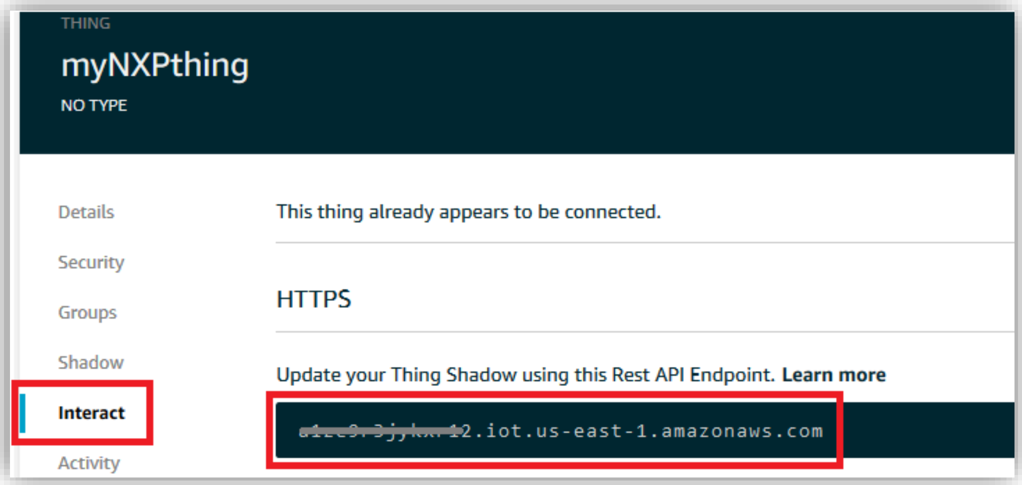
1. Prepare "AwsRemoteControlPreferences.properties" file with **your AWS credentials.** File is located at <SDK Folder>\boards\lpcxpresso55s69\aws_examples\remote_control_android\ **AwsRemoteControlPreferences.properties**


myNXPThing.properties

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

Where do I obtain these parameters? See below:

<REST API ENDPOINT>
Your Thing's Rest API Endpoint is located under your Thing's "Interact" section:

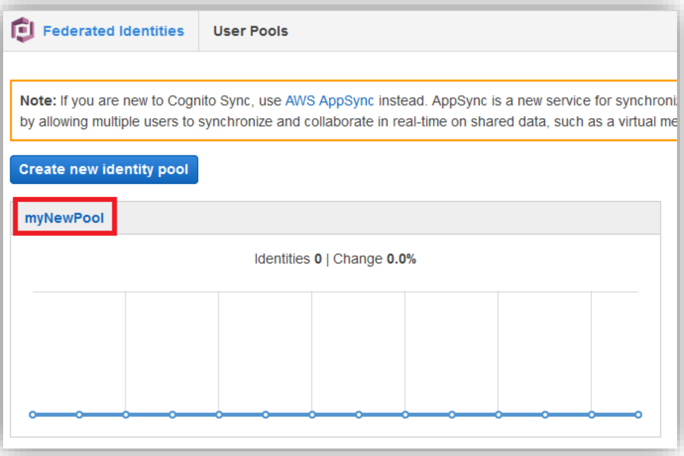


This thing already appears to be connected.

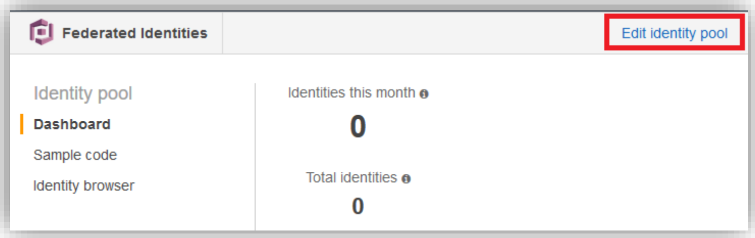
Update your Thing Shadow using this Rest API Endpoint.

```
[redacted]2.iot.us-east-1.amazonaws.com
```

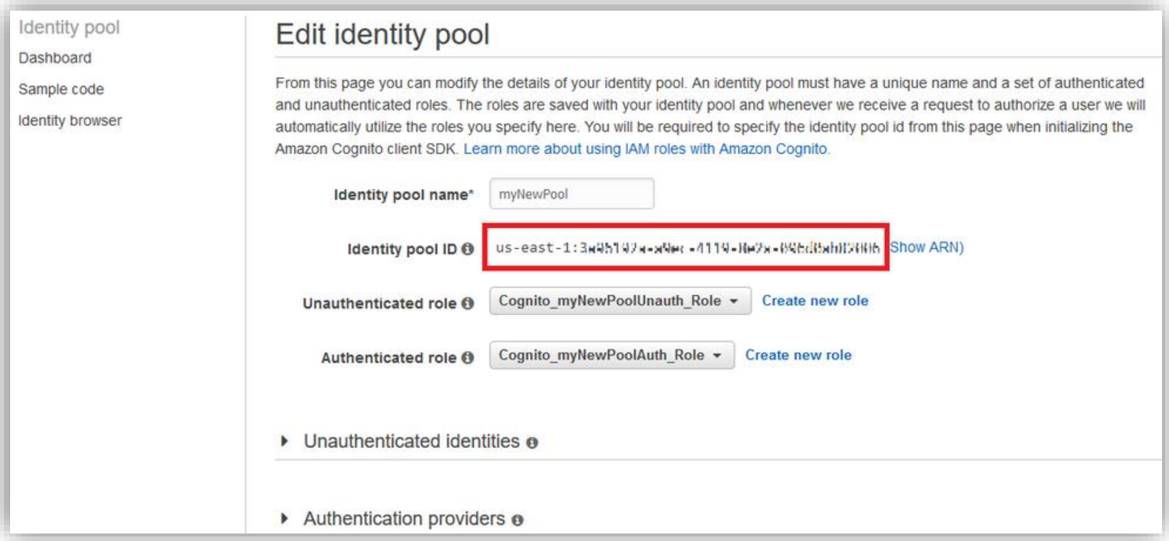
<COGNITO POOL ID>
To obtain the Pool ID constant, select your Federated Entity Pool (https://console.aws.amazon.com/cognito/federated) and select "*Edit identity pool*" in the top right corner. Copy **Identity pool ID** (it will look like <REGION>:<ID>). See below.



Note: If you are new to Cognito Sync, use AWS AppSync instead. AppSync is a new service for synchroni
by allowing multiple users to synchronize and collaborate in real-time on shared data, such as a virtual me



Identities this month	0
Total identities	0



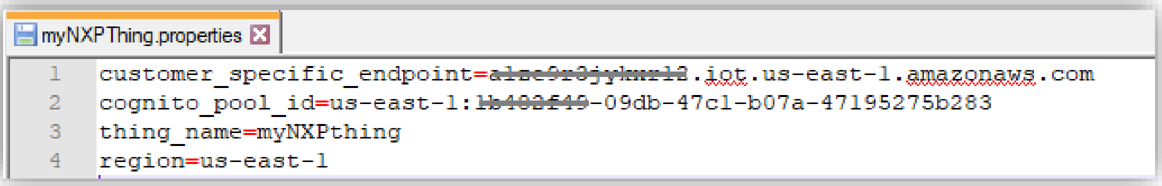
Edit identity pool

From this page you can modify the details of your identity pool. An identity pool must have a unique name and a set of authenticated and unauthenticated roles. The roles are saved with your identity pool and whenever we receive a request to authorize a user we will automatically utilize the roles you specify here. You will be required to specify the identity pool id from this page when initializing the Amazon Cognito client SDK. Learn more about using IAM roles with Amazon Cognito.

Identity pool name*	myNewPool
Identity pool ID	us-east-1:[redacted] (Show ARN)
Unauthenticated role	Cognito_myNewPoolUnauth_Role — Create new role
Authenticated role	Cognito_myNewPoolAuth_Role — Create new role

Make sure you select the whole ID, including the region.

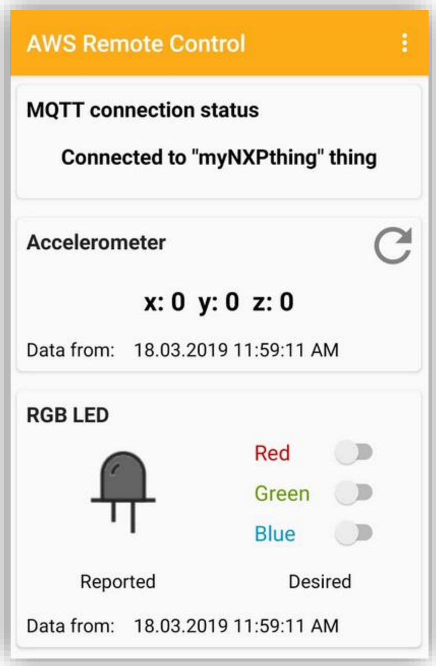
The file should look like this:



myNXPThing.properties

```
customer_specific_endpoint=[redacted].iot.us-east-1.amazonaws.com
cognito_pool_id=us-east-1:[redacted]-09db-47c1-b07a-47195275b283
thing_name=myNXPthing
region=us-east-1
```

2. Locate the AWS Remote Control Android application (**AwsRemoteControl.apk**) under same SDK folder as "properties" file in
   <SDK>\boards\lpcxpresso55s69\aws_examples\remote_control_android\AwsRemoteControl.apk
   Connect your Android device to your PC. Make sure you select to "File Transfer" instead of just charging in your android device.
3. Drag & drop the *AwsRemoteControlPreferences.properties* and *AwsRemoteControl.apk* files to a known location in your Android device.
4. Install the *AwsRemoteControl.apk* on the Android device. Application requires at least Android version 5.1 (Android SDK 22).
5. Run the application. You will be asked to select a properties file with AWS IoT preferences. Browse to the dropped file (*AwsRemoteControlPreferences.properties*) and select it. Then application will establish MQTT connection to AWS server, download last state of thing's shadow and it will be ready for user input.

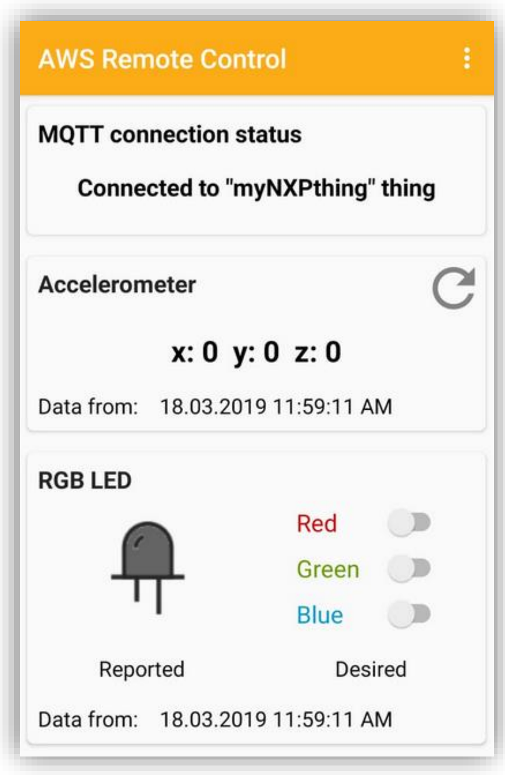


8. Test your new Skill and Android application

Now that your setup is complete, you can test it with your LPCXpresso55S69 using the Android application and Alexa voice commands.

8.1 Control the evaluation board using Android Application

Verify that you can control the LED status and read Accelerometer data using the AWS Remote Control Android application



AWS Remote Control

MQTT connection status

Connected to "myNXPthing" thing

Accelerometer

x: 0 y: 0 z: 0

Data from: 18.03.2019 11:59:11 AM

RGB LED

Red
Green
Blue

Reported Desired

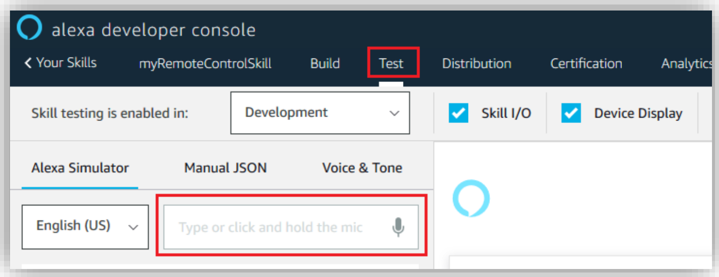
Data from: 18.03.2019 11:59:11 AM

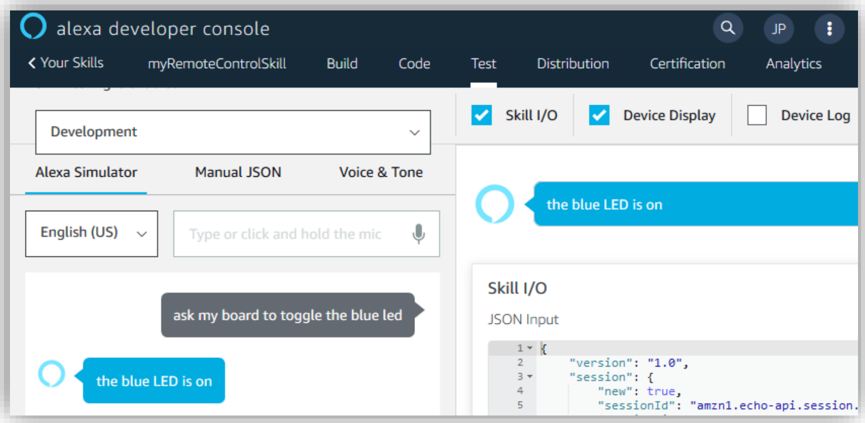
8.2 Control the evaluation board using Alexa voice commands

Verify that you can control the LED status and read Accelerometer using Alexa voice commands:

> "Alexa, ask my board to update the accelerometer"
> "Alexa, ask my board to turn the red LED on"
> "Alexa, ask my board to toggle the blue LED"

Note: If you don't have an Echo device you can run your test using the Alexa application or the Alexa developer console -> Test (Skills -> myRemoteControlSkill -> Test).
https://developer.amazon.com/alexa/console/ask





ask my board to toggle the blue led

the blue LED is on

```
{
    "version": "1.0",
    "session": {
        "new": true,
        "sessionId": "amzn1.echo-api.session.
```

Note: If you ask the board to update the accelerometer data, it should be reflected in the android application.

Now you can develop your own Skills and Lambda functions!

9. LPC55S69 hardware acceleration for AWS demo

The LPC55S69 MCU improves the mbedtls generic driver with some of the security features supported by the SoC and the SDK, refer to the /mbedtls/port/ksdk for further analysis of these features.

HASHCRYPT module

- AES (Advanced Encryption Standard)

CASPER module

- TLS ECP (Elliptic Curves over GF(P))
- RSA operations with public key
- NIST P-256 operations
- ECDSA sign/verify