ATWINC15x0 Smart Device Kit

Wi-Fi® Smart Device Enablement Kit Developer's Guide

Introduction

This document describes the Wi-Fi Smart Device Enablement Kit development, which is supported by the ATWINC15x0 Wi-Fi Network Controller (WINC) module that enables the developer to demonstrate the functionalities of the ATWINC15x0 module for Internet of Things (IoT) applications.

This kit is pre-programmed with an application for demonstrating connectivity to the Microchip AWS® IoT account with the help of an Amazon Alexa® and a mobile application. The developer can collect the data to control the LEDs on the kit using cloud-based voice service virtual assistant such as Amazon Echo Dot®. The mobile application is used for board registration, network configuration, and controlling and monitoring the Wi-Fi Smart Device Enablement Kit.

For the demonstration shown in this document, the Wi-Fi Smart Device Enablement Kit is pre-configured with the following:

- Customized MCU firmware
- AWS cloud configuration
- Alexa Skill setting
- Mobile application configuration

However, the developers can create and set up their own private AWS account, configure MCU firmware, and build custom Alexa Skill with the help of this document.

Features

- ATWINC15x0, Qualified IEEE® 802.11 b/g/n Network Controller Module
- On-Board Host Microcontroller (SAML21G18B) for Easy Operation and Feature Demonstration
- On-Board CryptoAuthentication™ (ATECC608A) Device for Secure Connection to AWS
- On-Board Li-Ion/Li-Po Charge Management Controller (MCP73833) for Lithium Battery Charging
- On-Board High Power Supply Rejection (PSRR) Low Dropout (LDO) Regulator (MIC5317)
- On-Board Environment Sensor (BME280)
- On-Board Light Sensor (VEML6030)

© 2019 Microchip Technology Inc.

Table of Contents

Introduction....1

Features....1

1. Quick References....4
   - 1.1. Prerequisites....4
   - 1.2. Reference Documentation....4

2. UART Debug Interface....6

3. Boot-Loader Firmware....8
   - 3.1. Programming Procedure....8

4. Application Firmware....11
   - 4.1. Application Firmware Compilation Procedure....11
   - 4.2. Application Firmware Programming Procedure....12
   - 4.3. Operation....15
   - 4.4. Software Customization....19

5. AWS Setup....25
   - 5.1. Create and Administrate AWS account....25
   - 5.2. AWS IoT Just-in-Time Registration Setup....34
   - 5.3. AWS IoT Mobile App Setup....38
   - 5.4. Amazon Cognito Setup....40
   - 5.5. Amazon DynamoDB Setup....45
   - 5.6. AWS Lambda Setup....46

6. AWS Provision Setup....52

7. Alexa Skill Setup....54
   - 7.1. Microchip Wi-Fi Smart Device Smart Home Skill Setup....54
   - 7.2. Microchip Wi-Fi Sensor Board Skill Setup....58

8. Appendix A: Software Installation....69
   - 8.1. Atmel Studio 7....69
   - 8.2. SAM Boot Assistance (SAM-BA)....69
   - 8.3. Python 3.6.x....69
   - 8.4. Python Package Manager....72

9. Appendix B: Mobile Application Configuration....73
   - 9.1. Android Application....73
   - 9.2. iOS Application....74

10. Appendix C: AWS CloudFormation to Setup Cloud....76
    - 10.1. Create IAM User for the Project....76
    - 10.2. Create Lambda Functions for Alexa....81

© 2019 Microchip Technology Inc.

10.3. Create Amazon Cognito for Alexa....83
10.4. Create AWS IoT Policy for Smartphone Application....84
10.5. Create AWS DynamoDB Table....85
11. Document Revision History....87
The Microchip Web Site....88
Customer Change Notification Service....88
Customer Support....88
Microchip Devices Code Protection Feature....88
Legal Notice....89
Trademarks....89
Quality Management System Certified by DNV....90
Worldwide Sales and Service....91

1. Quick References

1.1 Prerequisites

This section describes the prerequisites to develop an IoT application.

Software Prerequisites

- Atmel Studio 7 (for more details, refer to 8.1 Atmel Studio 7)
- SAM-BA® V2.18 (for more details, refer to 8.2 SAM Boot Assistance (SAM-BA))
- Python® 3.6.x (for more details, refer to 8.3 Python 3.6.x)
  Note: Python 3.7.x and Python 2.x are not supported.
- Python Package Manager (for more details, refer to 8.4 Python Package Manager)
- Wi-Fi Smart Device Enablement Kit Smartphone Applications:
  - Android™ – https://play.google.com/store/apps/details?id=com.amazonaws.mchp.awsprovisionkit
  - iOS – https://itunes.apple.com/app/id1460552937
- AWS Service – AWS IoT, AWS Lambda, AWS DynamoDB, AWS IAM
- Alexa Skill Design

Hardware Prerequisites

- Wi-Fi Smart Device Enablement Kit
- Android phone: Android version 6.0 or above
- iOS phone: iOS version 9.3 or above
- Wi-Fi Router
- Echo Dot
- UART Debug Interface

Release Package

The release package is available at https://github.com/MicrochipTech/winc1500-wifi-smart-device-enablement-kit-aws-cloud.

- MCU Firmware – located in `/mcu-firmware` directory.
- Microchip Wi-Fi Smart Device Enablement Kit mobile applications source files are available in `/mobile-app` folder of the release package.
- Python Provision Scripts – located in `/ProvisionScripts` directory.
- Lambda Functions – located in `/lambd-function` directory.
- Hardware PCB Files – located in `/pcb-files` directory.

1.2 Reference Documentation

For further study, refer the following:

- Wi-Fi® Smart Device Enablement Kit User's Guide (DS50002880)
- ATWINC15x0 IEEE® 802.11 b/g/n SmartConnect IoT Module Datasheet (DS70005304)
- SAM L21 Family Datasheet (DS60001477)

- ATECC608A CryptoAuthentication™ Device Summary Datasheet (DS40001977)

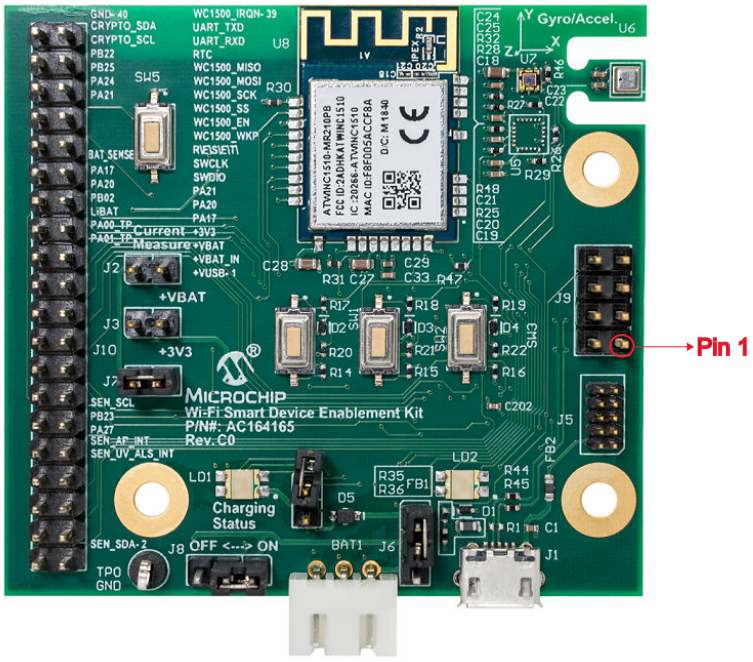
2. UART Debug Interface

The Wi-Fi Smart Device Enablement Kit has debug UART pins for the host MCU (SAML21G18B) and WINC1510 Wi-Fi module. The pins are connected to the J9 connector; the developer can use a USB-to-UART converter to connect a computer and the board Debug UART to capture the MCU or WINC1510 log.

Table 2-1. Pin Details

Pin	Function
3	MCU_DEBUG_UART_RXD
5	MCU_DEBUG_UART_TXD
4	WINC1510_DEBUG_UART_RXD
6	WINC1510_DEBUG_UART_TXD
7	GND

Figure 2-1. Header Description



A terminal emulator can help in diagnosing problems or verify if the device code is running properly. There are a variety of terminal emulators available for Windows®, macOS®, and Linux®. The developer must connect the device to computer before connecting a terminal emulator to the device.

Use these settings for the terminal emulator:

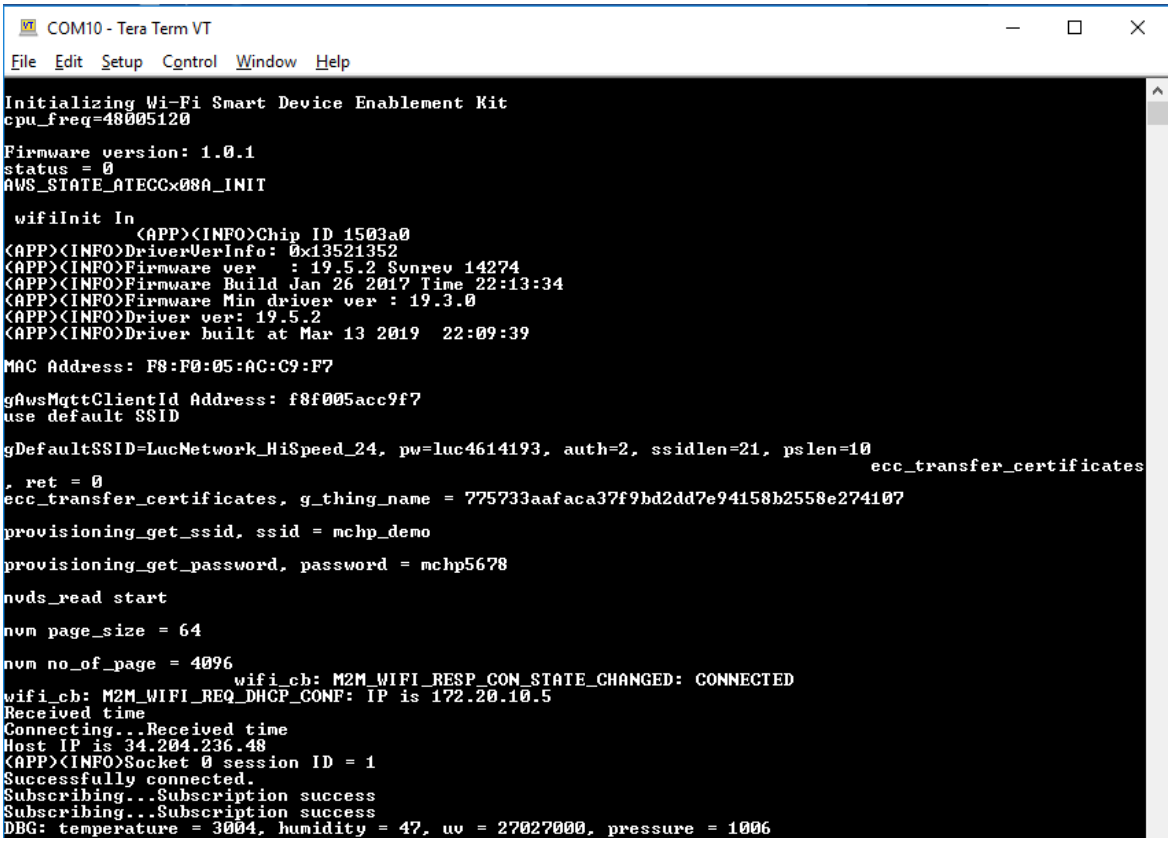
Table 2-2. Terminal Setup

Terminal Settings	Value
Port	Depends on platform and other devices connected to the computer.

.....continued	
Terminal Settings	**Value**
BAUD rate	115200
Data	8 bit
Parity	none
Stop	1 bit
Flow control	none

Below is the application boot-up log:

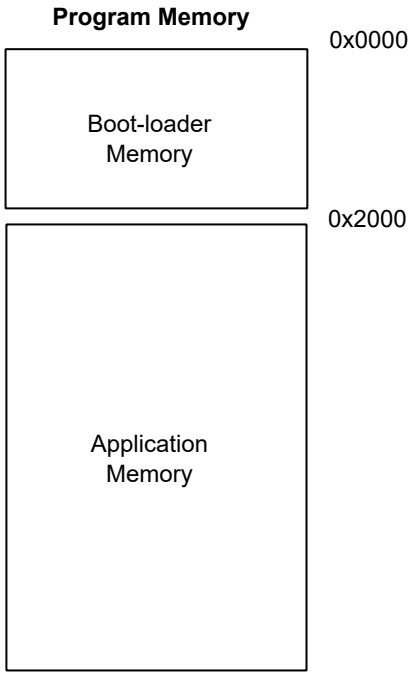
Figure 2-2. Boot-Up Log



3. Boot-Loader Firmware

Boot-loader allows the developer to program MCU firmware through a USB port without Atmel-ICE programmer. Boot-Loader mode is triggered by pressing and holding SW1 switch before powering-up the board. The SAM-BA V2.18 GUI (8.2 SAM Boot Assistance (SAM-BA)) is used for loading a firmware to the board through USB port. The boot-loader is stored in memory address `0x0-0x2000`. The memory address for an application firmware start is `0x2000`.

Figure 3-1. Program Memory



The boot-loader can be re-used even after modifying the application firmware and programming a new application firmware to the kit using the SAM-BA V2.18 tool.

The boot-loader is not required if the developer uses Atmel-ICE Debugger to load firmware to the board instead of the SAM-BA V2.18. In this case, set the starting address of the application firmware to `0x0000`.

Note: By default, the board is pre-programmed with the boot-loader to enable the developer to upgrade the application firmware through a USB port without an Atmel ICE programmer.

3.1 Programming Procedure

This section describes how to program the board with the boot-loader. The boot-loader image is available in the `/boot-loader/` directory of the release package. The user can use the Atmel ICE programming tool to program the boot-loader to the board. The Atmel ICE tool is available at https://www.microchip.com/developmenttools/ProductDetails/atatmel-ice.

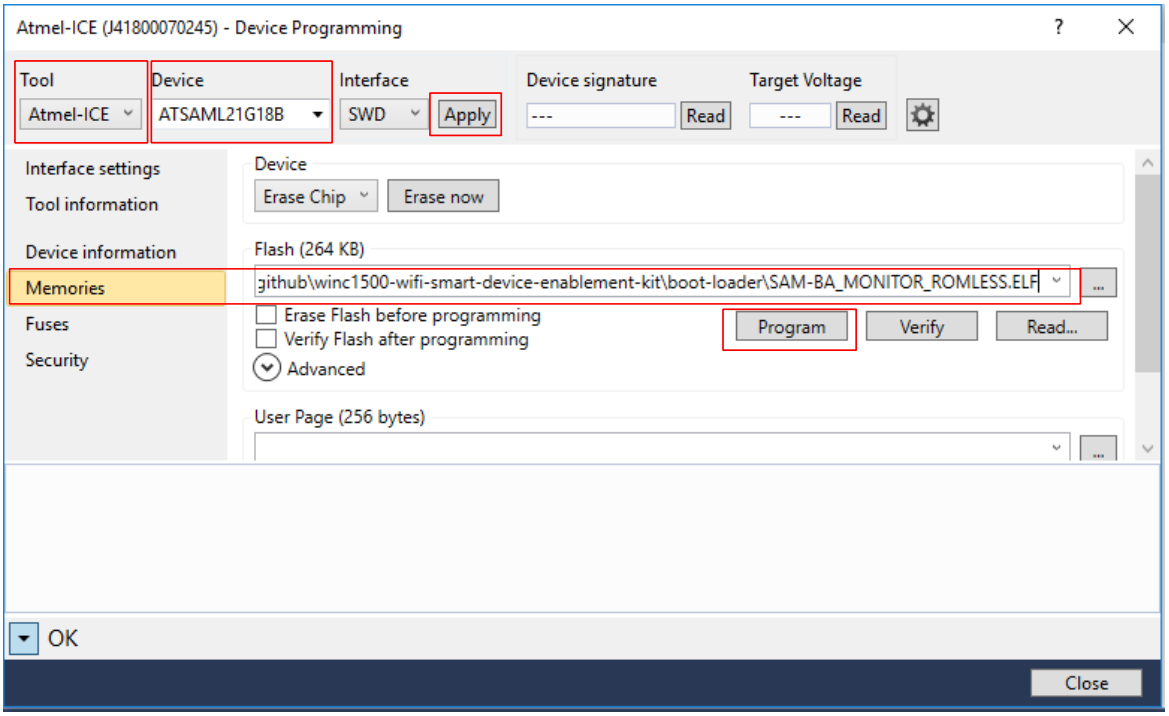
1. Connect Atmel ICE (see Atmel ICE) to the board SWD interface connector (J5).

Figure 3-2. Atmel ICE Connected to the Board SWD Interface Connector (J5)



2. Launch Atmel Studio, select *Tools > Device Programming*.
3. Select “Tool” as *Atmel-ICE* and “Device” as *ATSAML21G18B*.
4. Click **Apply**.
5. In **Memories** tab, fill `SAM-BA_MONITOR_ROMLESS.ELF` in the “Flash” field. The `.ELF` file is available in the `/boot-loader` directory and https://github.com/MicrochipTech/winc1500-wifi-smart-device-enablement-kit-aws-cloud/tree/master/boot-loader.
6. Click **Program**.

Figure 3-3. Boot Loader Firmware



4. Application Firmware

The application firmware is used to demonstrate connectivity to AWS IoT. The firmware connects the board to AWS IoT cloud, updates the sensor data and obtains the LED state to AWS Shadow by publishing and subscribing to the AWS Shadow MQTT topics.

The MCU application firmware source codes are available in the mcu-firmware/ directory of the release package. This directory contains following project solution files.

- `saml21g18b_sensor_board_demo_ECC.atsln` – This project is used for performing AWS account provisioning. A private key is generated for the authentication with the AWS cloud and the AWS account credentials are stored in the ECC608. The developer needs to program this project code to the board to migrate the board and connect to the private AWS account. Python scripts are available in the `ProvisionScripts/` directory to perform AWS account provision. For more details, refer to 6. AWS Provision Setup.
- `saml21g18b_sensor_board_demo_JITR.atsln` – This project is used in the normal application. A board running this firmware can connect to AWS IoT, publish and subscribe AWS IoT Shadow Topics.

For firmware development, at first, the developer needs to program the code of `saml21g18b_sensor_board_demo_ECC.atsln` to provision the board to the private AWS account. Then, customize the application in `saml21g18b_sensor_board_demo_JITR.atsln` and program this project to the board.

4.1 Application Firmware Compilation Procedure

1. Download release package files from https://github.com/MicrochipTech/winc1500-wifi-smart-device-enablement-kit-aws-cloud and copy the source file to the C drive (`C:/`) root folder.
2. Launch Atmel Studio 7.0.
3. Click on *File > Open > Project...*
4. Select project to open `mcu-firmware/saml21g18b_sensor_board_demo_JITR.atsln` or `mcu-firmware/saml21g18b_sensor_board_demo_ECC.atsln` files.
5. By default, the starting address of the application firmware is `0x2000` as the board is pre-programmed with a boot-loader. The starting address of boot-loader is `0x0`.
   - If Atmel ICE is used for programming the firmware to the board; the developer need not use the boot-loader and starting address of the application firmware must be set back to `0`. Skip this step if boot-loader is used for programming the firmware.
   - Perform the following steps on the project solution to set the starting address of the application firmware:
     1. Launch Atmel Studio.
     2. *Open > Right click > Properties > Toolchain > ARM/GNU Linker > Miscellaneous* .
     3. Modify "Wl,--section-start=.text=0x2000" to "Wl,--section-start=.text=0x0000" in the Linker Flag.
6. Click *Build > Rebuild Solution* to compile the source code and generate binary files.

Note: Generated binary files are to be stored in the `C:\github\winc1500-wifi-smart-device-enablement-kit-aws-cloud\mcu-firmware\saml21g18b_sensor_board_demo\Debug` directory.

4.2 Application Firmware Programming Procedure

Use one of the following to program the application firmware to the board:

- Program through USB Port (J1)
- Program through Atmel ICE programming tool

Tools:

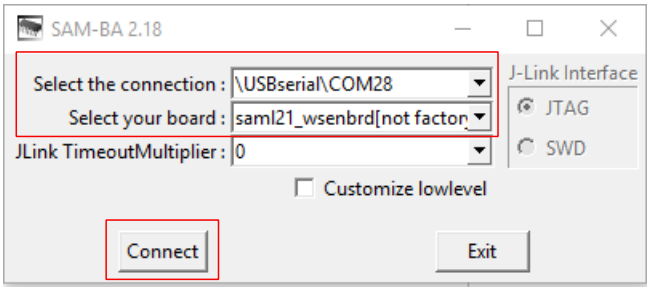
- Atmel-ICE: https://www.microchip.com/developmenttools/ProductDetails/atatmel-ice
- SAM-BA V2.18: 8.2 SAM Boot Assistance (SAM-BA)

Method 1: Programming through USB and J1 connector

Perform the following steps for programming through USB and J1 connector:

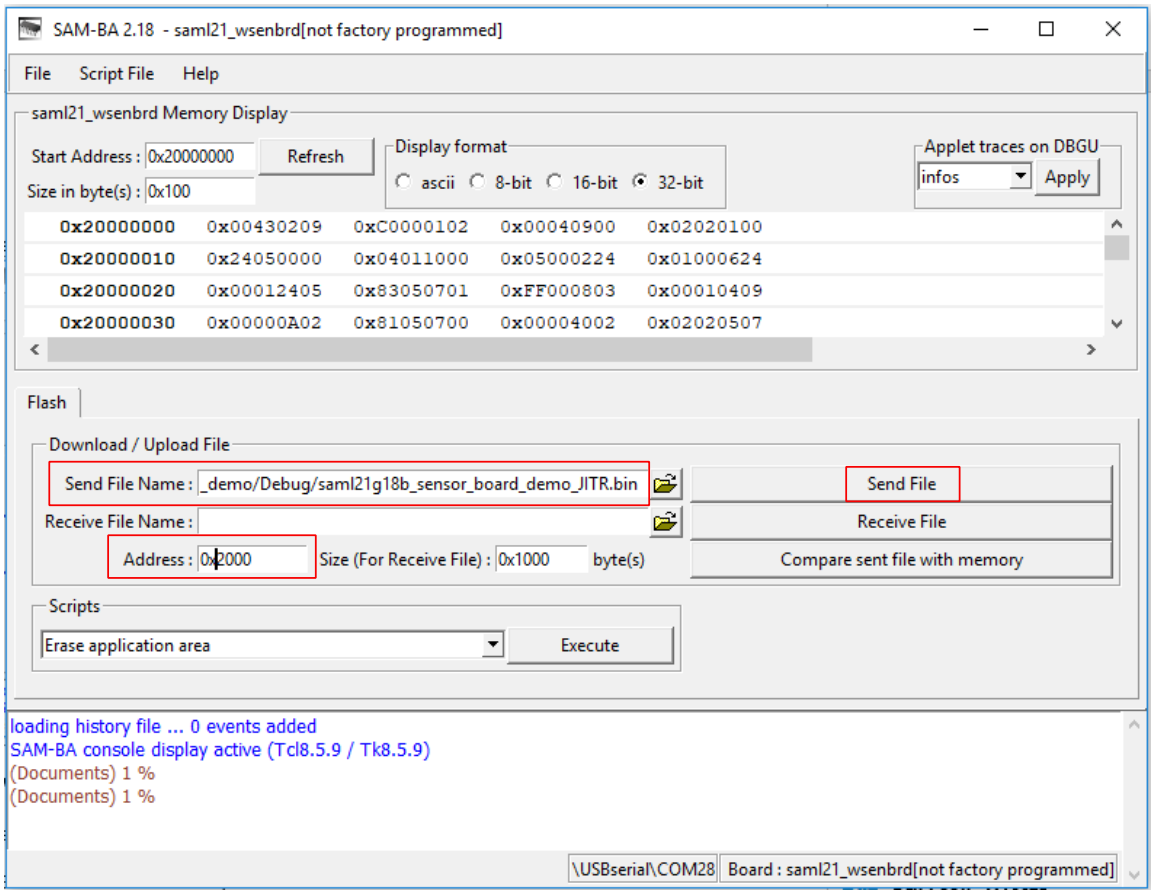
1. Connect USB cable with the PC and the Wi-Fi Smart Device Enablement Kit J1 connector.
2. Press and hold SW1 on the board during power-up. The LD2 LED turns off if the board successfully triggers the boot-loader.
3. Launch SAM-BA V2.18 on PC.
4. Set the connection to `\USBserial\COMx` and the board (saml21_wsenbrd).

   Figure 4-1. SAM-BA 2.18



5. Click **Connect** to add address and firmware bin file.
6. In the **Flash** tab, set the bin file name in “Send File Name” and the “Address.”
7. Click **Send File** to program the bin file to the Wi-Fi Smart Device Enablement Kit.

Figure 4-2. UI Parameters



Method 2: Programming with the Atmel ICE tool

If Atmel ICE is used for programming the application to the board, set the starting address to `0x0000`. For more details, refer to the 4.1 Application Firmware Compilation Procedure.

Perform the following steps for programming through the Atmel ICE tool:

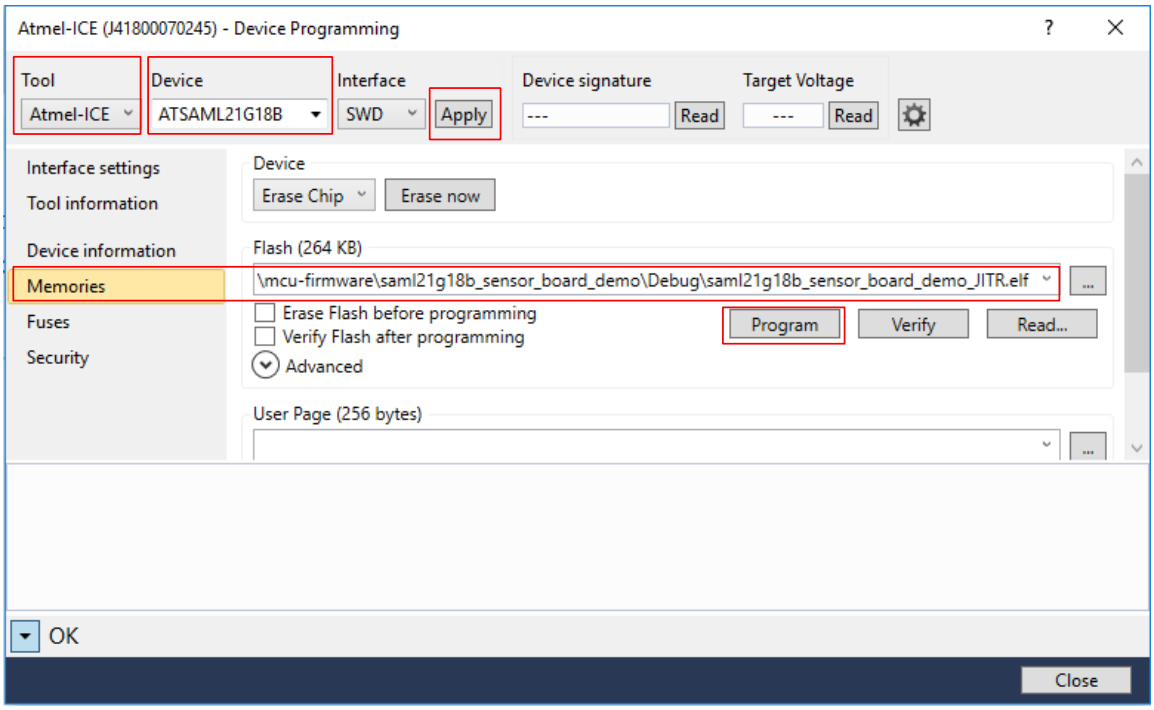
1. Connect Atmel ICE to the board as shown in the following figure.

Figure 4-3. Atmel ACE Connected to Board



2. Launch Atmel Studio, and select *Tools* > *Device Programming*.
3. Select Atmel-ICE as "Tool" and ATSAML21G18B as "Device."
4. Click **Apply**.
5. Select "Memories" and the application firmware in “Flash” as highlighted in the following figure.
6. Click **Program**.

Figure 4-4. Atmel - ICE

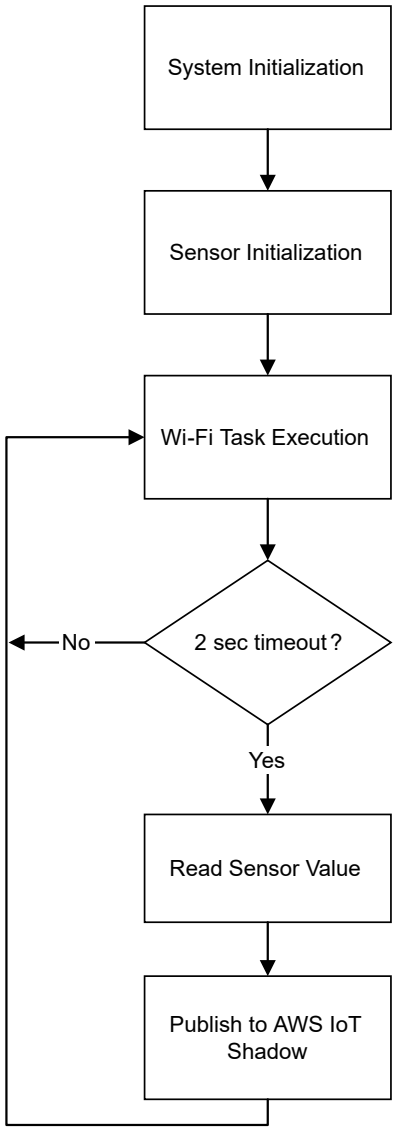


4.3 Operation

4.3.1 Application Flow

When the power is on, the Wi-Fi Smart Device Enablement Kit goes through the initialization phase where the MCU's internal system, WINC15x0 and all the connected sensors are initialized. After this phase, the board attempts to connect to the configured AP. LD2 blinks in blue every 500 ms during this process. When the board successfully connects to the AP and gets the IP address, LED blinks in blue every 100 ms. Then the board attempts to connect to the AWS IoT. After successful connection to the cloud, LD2 remains in steady blue. The board checks the sensor data every two seconds. If the sensor data is changed the data is published to AWS IoT Shadow, and the developer can use a mobile application or speak to Alexa-enabled device (for example, Echo Dot) to check the sensor values, and LED LD2 color.

Figure 4-5. Application Flowchart



4.3.2 Cloud Connection

The host MCU runs AWS IoT SDK to connect the board to AWS IoT cloud. The AWS IoT SDK includes cloud connection API and MQTT pub/sub API. MQTT Client ID is needed for the connection and Subject Key Identifier (SKI) of the device certification is used as the MQTT Client ID. The board connects to AWS IoT cloud with TLS 1.2 RSA cipher suite. Device certificate and private key are stored in ECC608 CryptoAuthentication device. WINC15x0 uses the certificate and key in ECC608 to perform the authentication with the AWS IoT cloud.

4.3.3 Data Exchange

The board uses MQTT protocol to exchange the data with AWS cloud and mobile application. AWS IoT acts as a MQTT broker; mobile application and sensor board act as MQTT Publisher and Subscriber. AWS IoT Device Shadow service is used to store and retrieve the current state information of the board. The board updates sensor and LED data to the AWS IoT by publishing MQTT message to Shadow MQTT */update* topics and get the LED latest state from AWS IoT by subscribing Shadow MQTT */update/*

delta topics. The mobile application controls the LED color by publishing MQTT message to Shadow MQTT */update* topics, and get the sensor data by subscribing Shadow MQTT */update/delta* topics.

The Shadow document of the Wi-Fi Smart Device Enablement Kit is shown as below:

```
{
  "reported": {
    "macAddr": "f8f005accfad",
    "uv": 427000,
    "COUNT": 34,
    "hum": 48,
    "pressure": 1017,
    "temp": 2468,
    "BUTTON_3": 0,
    "BUTTON_1": 1,
    "BUTTON_2": 1,
    "LED_R": 1,
    "LED_G": 0,
    "LED_B": 1,
    "LED_INTENSITY": 21,
    "Light": 1
  }
}
```

4.3.4 Sensor Detection

Firmware reads the environment sensor and light sensor, every two seconds in an infinite loop. The updated sensor data is sent to the AWS IoT cloud.

Environment sensor:

- Detects temperature, humidity and pressure
- I2C address: `0x77`

Light sensor:

- Detects the light intensity
- I2C address: `0x10`

4.3.5 Board Registration and Network Configuration

The board registration and network configuration take place in the following way:

- The developer can perform board registration and network configuration on the mobile application.
- The board connects to the network and allows control/monitoring of the board using the mobile application account.
- During this process, the developer needs to press and hold SW3 button for 5 seconds, LED LD2 blinks in red color when the board enters the board registration and network configuration mode.
- In this mode, the board changes to Wi-Fi AP mode, broadcasted with SSID named “WiFiSmartDevice_F8F005XXXXXX”, where “XXXXXX” is the last hex value of the MAC address. Then, mobile application acts as a Wi-Fi station to connect the board. TCP tunnel with port 8899 is created. Target Network SSID and passphrase are transferred from the mobile application to the board in this tunnel. When the board successfully gets the SSID and passphrase, it changes to Wi-Fi Station mode and connects to the target AP using the Wi-Fi credential.
- In this process, Device Thing ID is also transferred from the board to the mobile application. When the mobile application gets the Device Thing ID, it stores the Device Thing ID with the mobile application account ID and board device name to a table in AWS DynamoDB.
- When the developer signs in to the mobile application, the mobile application scans the table in AWS DynamoDB to display the board information that corresponds to its mobile application account ID.

4.3.6 Alexa Voice Control

Voice control feature is added to the board by using Amazon Alexa. The developer can speak to an Alexa-enabled device (for example, Echo Dot) to get the board sensor data or control LED LD2 color. When the developer speaks to the Alexa-enabled device, the voice is streamed to the Alexa cloud for processing and then sent the directives in JSON format to AWS Lambda. There is a Lambda function to handle the directives. The directives include an access token which contains the mobile application account ID. The Lambda function scans the table in AWS DynamoDB to look for the Device Thing ID that belongs to the mobile application account ID. Then, the code in Lambda function updates the value to AWS IoT Shadow or get the AWS IoT Shadow of the device to control the LED LD2 or report the sensor data to Alexa cloud and output to the Alexa-enabled device.

There are two Alexa skills with different features for the Wi-Fi Smart Device Enablement Kit:

1. Alexa Smart Home Skill - the name of this skill is Microchip Wi-Fi Smart Device Smart Home Skill.
2. Alexa Custom Skill - the name of this skill is Microchip Sensor Board Skill.

The Lambda function source files of Microchip Wi-Fi Smart Device Smart Home Skill can be found in the directory `/lambda-function/alexa-smart-home-skill` while the Lambda function source files of Microchip Sensor Board Skill can be found in the directory `/lambda-function/alexa-custom-skill`.

Table 4-1. Microchip Wi-Fi Smart Device Smart Home Skill

Function	Voice command
Turn On/ Off LED LD2	Turn on/ off <DEVICE_NAME>
Adjust light intensity of LED LD2	Set the power to [0 - 100]% on <DEVICE_NAME>

Table 4-2. Microchip Sensor Board Skill

Function	Voice command
Turn on/off LED LD2	Turn light on/off
Turn LED LD2 to different color	Turn light blue/green/red/yellow/white/cyan/magenta
Get LED LD2 color	What is the light color?
Get LED LD2 on/off state	What is the light state?
Get button (SW1, SW2, SW3) status	What are the button states?
Get temperature/humidity from the kit	What is the temperature/humidity?
Assert the Port A 17/ 20 or 21 GPIO to high/low	Set/Clear Port A 17/20/21
Assert the Port B 22/23 to high/low	Set/Clear Port B 22/23

4.3.7 Mobile Application Control

Mobile application controls the board by publishing MQTT message to Shadow MQTT */update* topics, and get the sensor data by subscribing Shadow MQTT */update/delta* topics.

4.4 Software Customization

4.4.1 Architecture

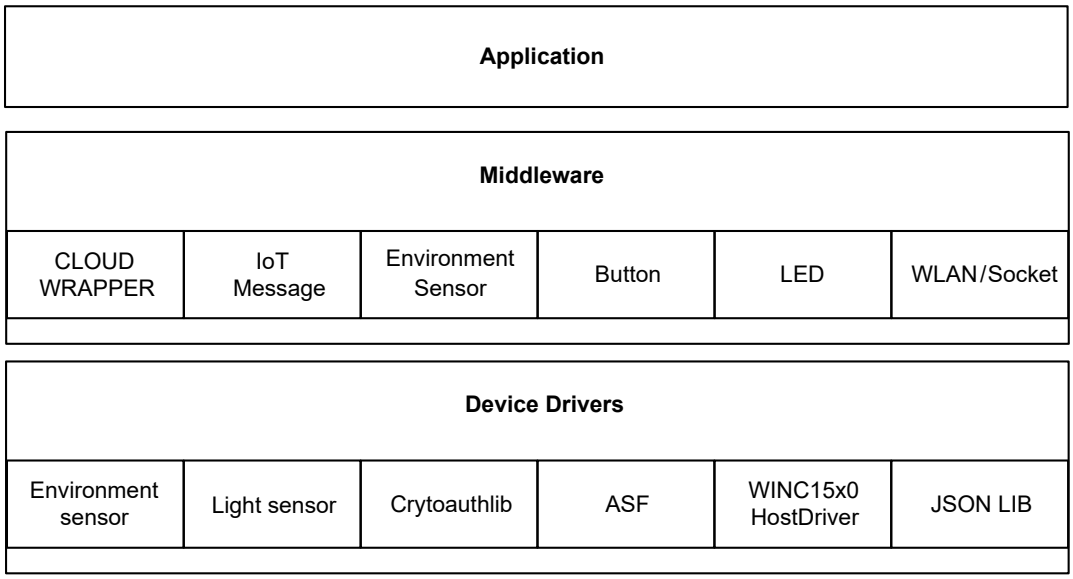
MCU firmware application starts with `main.c` file. It then runs into a while loop after board initialization. The while loop keeps running the following functions:

- wifiTaskExecute() – to handle all the Wi-Fi related functions including the cloud connection, message exchange between the cloud, network configuration, and so on.
- buttonTaskExecute(ms_ticks) – to check the button state and trigger the callback function.
- env_sensor_execute() – to read the sensor value and trigger the callback function.

The following figure shows the firmware architecture. There are middleware modules between the application layer and the device drivers layer. Developers can easily develop the application by using middleware APIs to configure the device drivers. Device drivers layers include:

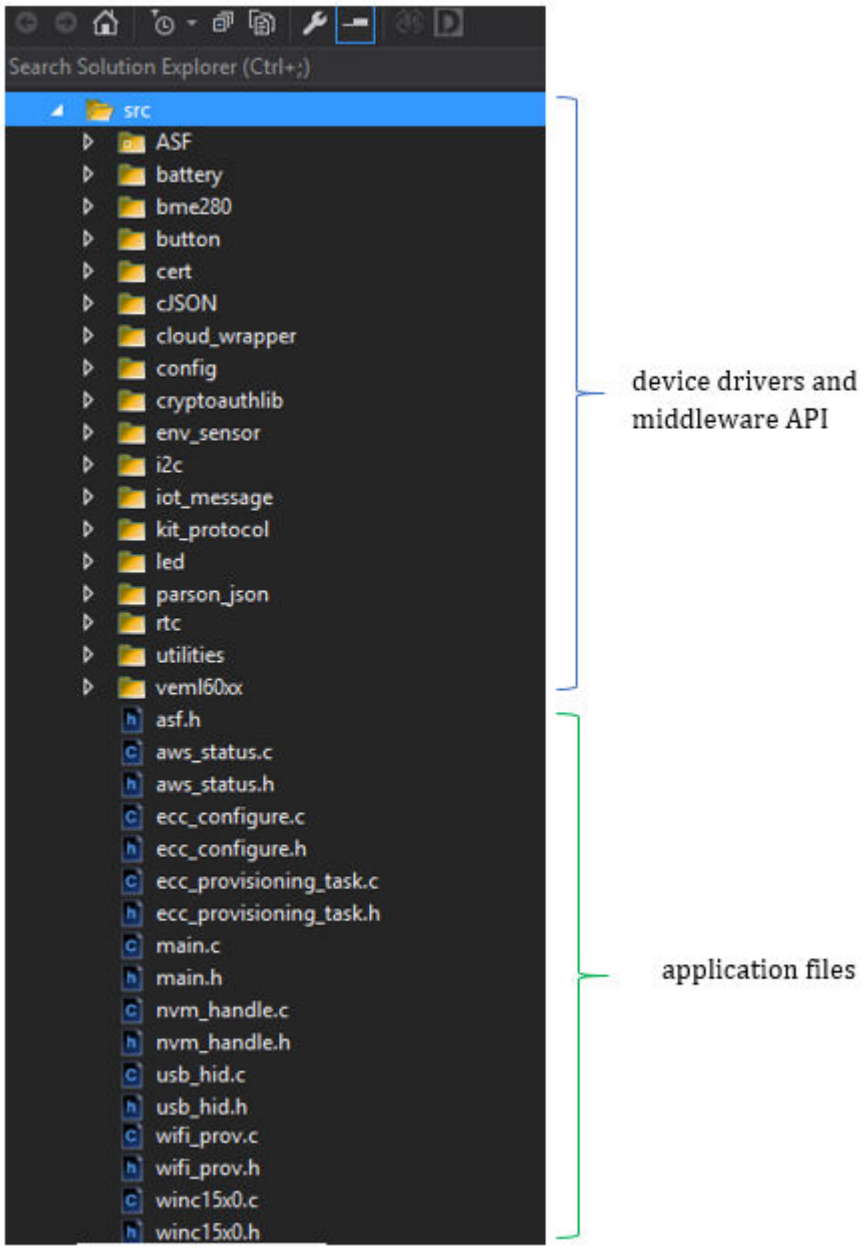
- WINC15x0 Wi-Fi module host driver
- Environment sensor driver
- Light sensor driver
- Crytoautholib for ECC608
- ASF
- JSON library

Figure 4-6. Firmware Architecture



4.4.2 Folder Structure

Figure 4-7. Folder Structure



4.4.3 Middleware APIs

The following table shows the middleware APIs.

Table 4-3. Middleware APIs

Middleware Modules	Files	Function
CLOUD_WRAPPER	`cloud_wrapper.h`	Connect/ disconnect cloud, publish/ subscribe MQTT channel

.....continued		
Middleware Modules	**Files**	**Function**
IOT_MESSAGE	`iot_message.h`	Generate or parson JSON message to/ from the AWS Shadow
BUTTON	`Button.h`	Read button value
LED	`Led.h`	Control LED LD2 color and blinking behavior.
ENV_SENSOR	`env_sensor.h`	Read sensor value
WLAN	`m2m_wifi.h`	WLAN features like AP scanning, AP connection, and so on. This function is available at https://asf.microchip.com/docs/latest/saml21/html/m2m__wifi_8h.html.
SOCKET	`socket.h`	Network socket API. This function is available at https://asf.microchip.com/docs/latest/saml21/html/socket_8h.html.

1. CLOUD_WRAPPER
   - Cloud_RC cloud_connect(char* hostname, char* mqtt_client_id);
     Usage: Connects AWS IoT cloud.

     Parameters:

     - Hostname [input] pointer of AWS IoT Endpoint hostname
     - mqtt_client_id [input]AWS IoT Thing ID

     Return: 0: success

     Other value: fail;
   - Cloud_RC cloud_disconnect();
     Usage: Disconnects AWS IoT cloud.

     Parameters: NA

     Return: 0: success;

     Other value: fail;
   - Cloud_RC cloud_mqtt_publish(char* channel, void* message);
     Usage: Publishes MQTT message to AWS IoT.

     Parameters:

     - Channel [input]MQTT topic that the message published to
     - Message[input]message that publish to the AWS IoT MQTT broker

     Return: 0: success;

     Other value: fail;
   - Cloud_RC cloud_mqtt_subscribe(char* channel, void* cb);
     Usage: Subscribe AWS IoT MQTT topic.

     Parameters:

     - Channel [input]MQTT topic for subscription

- cb [input]callback function that is trigger when MQTT message is received from the subscribe MQTT topic

Return: 0: success;

Other value: fail;

– Cloud_RC cloud_mqtt_yield(int timeout);
Usage: This is used by the MQTT client to manage PING requests to monitor the health of the TCP connection as well as periodically check the socket receive buffer for subscribe messages.

Parameters:

- timeout[input]Maximum number of milliseconds to pass thread execution to the client

Return: 0: success;

Other value: fail;

2. IOT_MESSAGE
   – cJSON* iot_message_reportInfo_shadow(char* device_type, char* mac_addr, int report_data_num, NodeInfo data_info[]);
   Usage: Generates AWS IoT Shadow report message.

   Parameters:

   - device_type[input]Name of the device type
   - mac_addr[input] MAC address of the device
   - report_data_num[input] number report items
   - data_info[input] data of the report items

   Return: JSON message;

   Other value: fail;
3. BUTTON
   – void initialise_button(void);
   Usage: Initializes GPIO for button detection.
   – void buttonTaskExecute(unsigned long tick)
   Usage: Executes button detection, called in a while loop to poll the button every a period.

   Parameters:

   - tick[input]systick value

   – int regButtonShortPressDetectCallback(void* cb, int button);
   Usage: Register callback function for the button short press detect.

   Parameters:

   - cb [input]callback function trigger when short press button is detected
   - button[input] 1: SW1; 2: SW2; 3: SW3

   Return: socket index
   – int unRegButtonShortPressDetectCallback(int sock, int button)
   Usage: Unregister callback function for the button short press detect.

   Parameters:

   - sock [input]socket index that represent the registration

- button[input] 1: SW1; 2: SW2; 3: SW3

Return: 0: success; -1: fail

– int regButtonLongPressDetectCallback(void* cb, int button);
Usage: Register callback function for the button short press detect.

Parameters:

- cb [input]callback function trigger when long press button is detected
- button[input] 1: SW1; 2: SW2; 3: SW3

Return: socket index;

– int unRegButtonLongPressDetectCallback(int sock, int button);
Usage: Unregister callback function for the button long press detect.

Parameters:

- sock [input]socket index that represent the registration
- button[input] 1: SW1; 2: SW2; 3: SW3

Return: 0: success; -1: fail;

4. LED
   – void initialise_led(void);
   Usage: Initializes GPIO for LED control.

   Parameters: N/A

   – void led_ctrl_set_color(Led_Color color, Led_Mode mode);
   Usage: Set LED color and mode.

   Parameters:

   - color [input]LED color options
   - mode[input] LED mode: on/off/ blink normal/blink fast/blink slow

   – void led_ctrl_set_mode(Led_Mode mode);
   Usage: Set LED blink mode.

   Parameters:

   - mode[input] LED mode: on/ off/ blink normal/blink fast/blink slow

   – void toggleLED(void);
   Usage: Toggle LED on/off state.

   Parameters: N/A

   – Led_Color led_ctrl_get_color(void)
   Usage: Gets LED color.

   Parameters: N/A

   Return: Led_Color

5. ENV_SENSOR
   – void env_sensor_data_init(void);
   Usage: Gets the initialize value of the environment sensor and light sensor.

   Parameters: N/A

   – void env_sensor_execute(void)

Usage: Executes sensor value detection, called in a while loop to poll the sensor data every a period, trigger callback function when sensor data need to publish to cloud.

Parameters: N/A

– void register_env_sensor_udpate_callback_handler(void* cb)
Usage: Register callback function for the environment and light sensor data.

Parameters:

- cb [input]callback function

5. AWS Setup

Note: The steps mentioned in this section may vary, based on the updates from Amazon. Refer to the Amazon guides for the latest step-by-step procedure.

Developers can follow 10. Appendix C: AWS CloudFormation to Setup Cloud instead of this section to set up AWS cloud. AWS CloudFormation feature allows easy and convenient way to set up the cloud by uploading template code to cloud rather than creating and configuring the cloud step-by-step.

5.1 Create and Administrate AWS account

5.1.1 Create AWS Account

Perform the following steps to create an AWS account:

1. Go to https://aws.amazon.com/.
2. Click the **Create an AWS Account** button to create an AWS account.
3. After a successful creation, click **Sign In to the Console** and enter the username and password to sign in to AWS account.

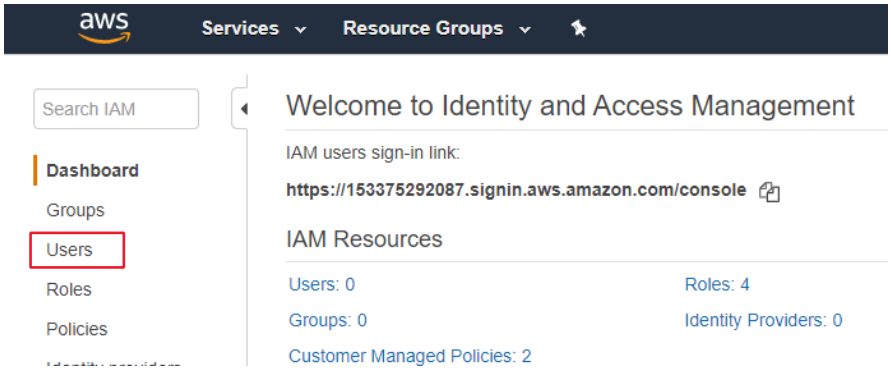
5.1.2 Create IAM User and Group

AWS Identity and Access Management (IAM) is used to manage and control AWS services and resources. The best practice to access AWS resources is to use IAM user account with limited permission instead of using a root user account. As a root user account can access all the AWS resource including the billing information. It is better to protect the root user access key.

Below are the steps to create IAM user account and group for this project. The IAM user account name is *ZTUser* and belongs to *ZTGroup*. This account is used for all the settings in this section.

1. Go to https://console.aws.amazon.com/iam.
2. Select **Users**.

Figure 5-1. User Selection



3. From the "Users" management page, click **Add User** at the top of the page. When the “Add user - Step 1: Details” page displays, enter the following information:
   3.1. Set user details as follows:
   - Username: ZTUser

   3.2. Select AWS access type:
   - Access type

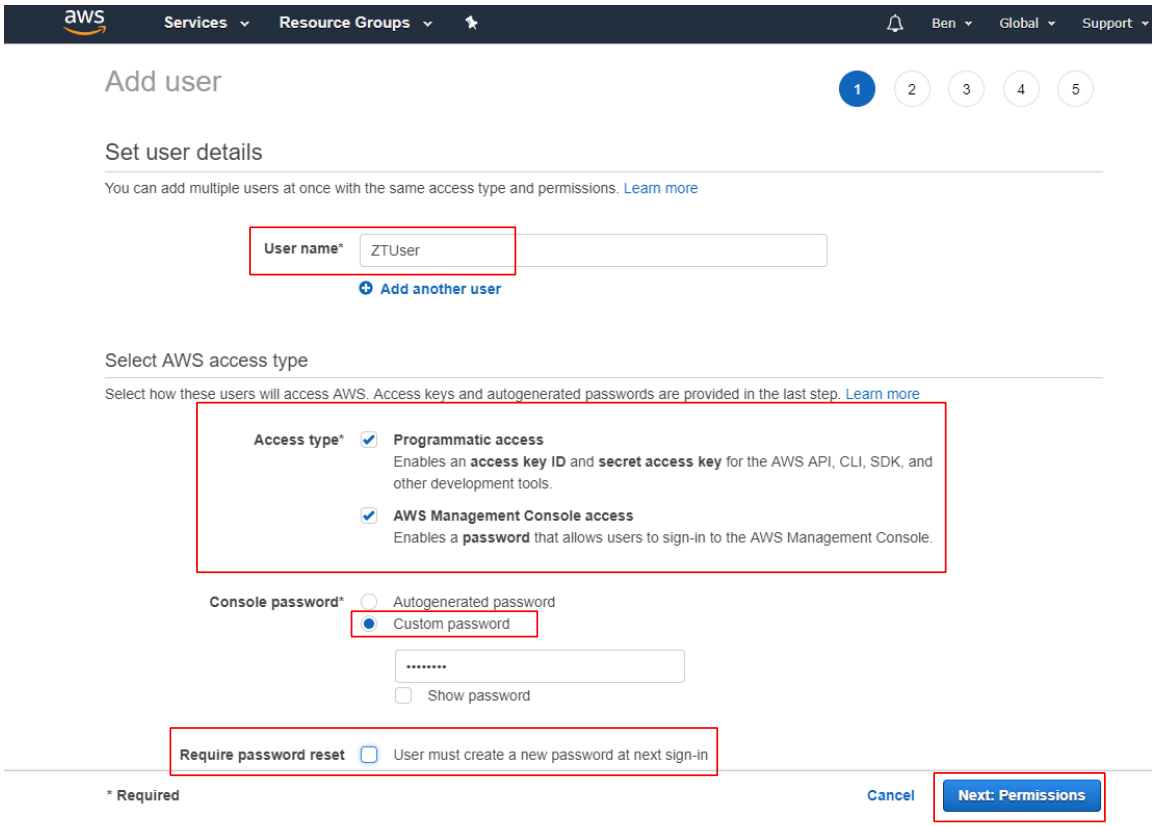
1. Tick “Programmatic access” and “AWS Management Console access” check boxes.

- Console password:
  1. Select “Custom password”.
  2. Enter a password for ZTUser.
  3. Uncheck “Require password reset” check box.
  4. Record the password for logging in to the console later.

Note: The fields with asterisk (*) are mandatory.

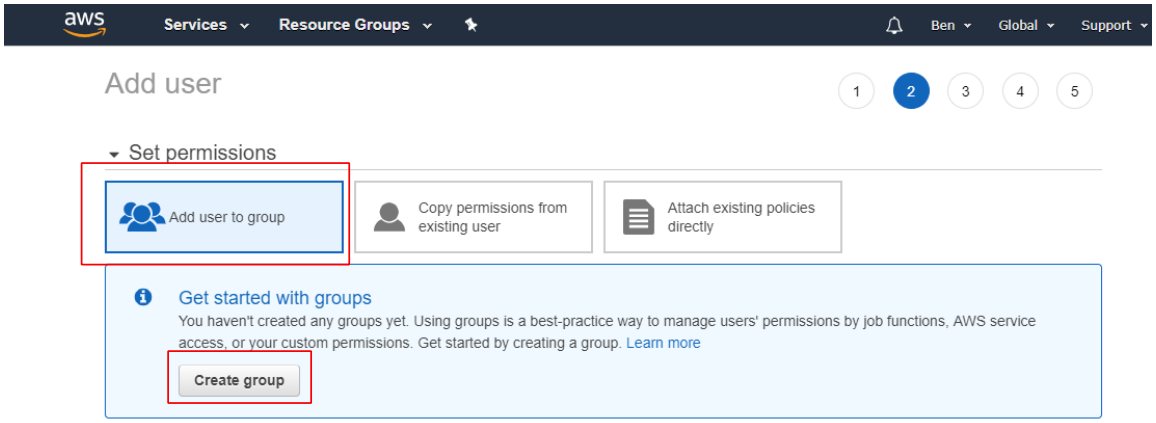
4. Click **Next: Permissions**.

Figure 5-2. Adding a New User and Selecting AWS Access Type



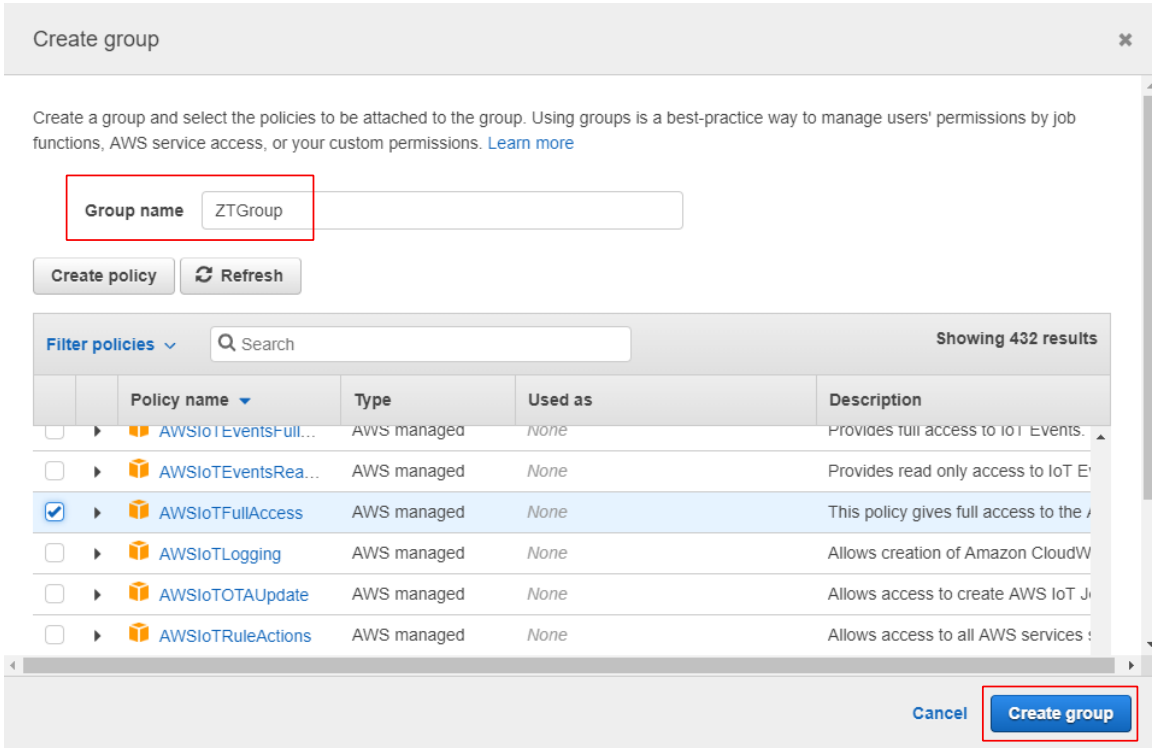
5. Create a new group for the AWS account.
6. Select “Add user to group” for adding a new user.
7. Click **Create group**.

Figure 5-3. Create Group



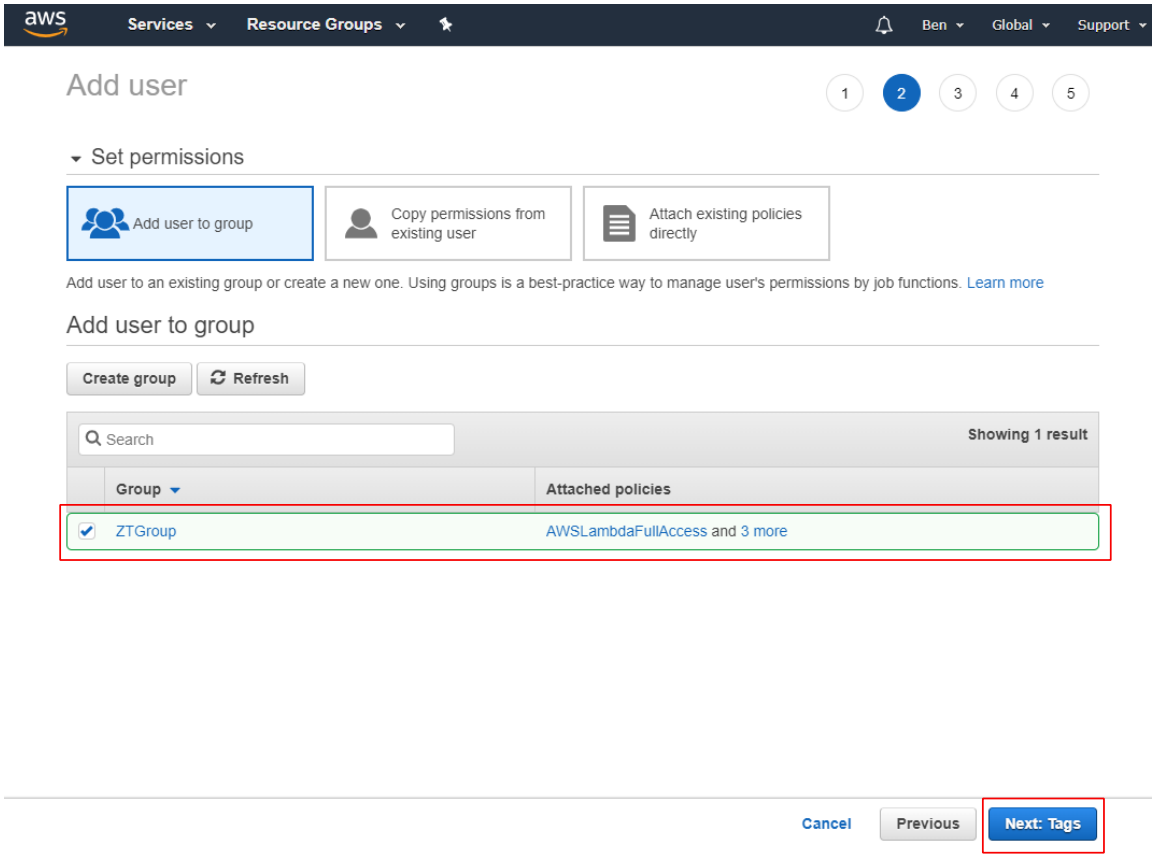
8. From the Create group window, enter the group name: *ZTGroup*.
9. Select the following policy types:
   - "AWSIoTFullAccess"
   - "AWSLambdaFullAccess"
   - "AmazonCognitoPowerUser"
   - "IAMFullAccess"
   - "AmazonDynamoDBFullAccess"
10. After selecting the policies, click **Create Group**.

Figure 5-4. Adding Policies to a Group



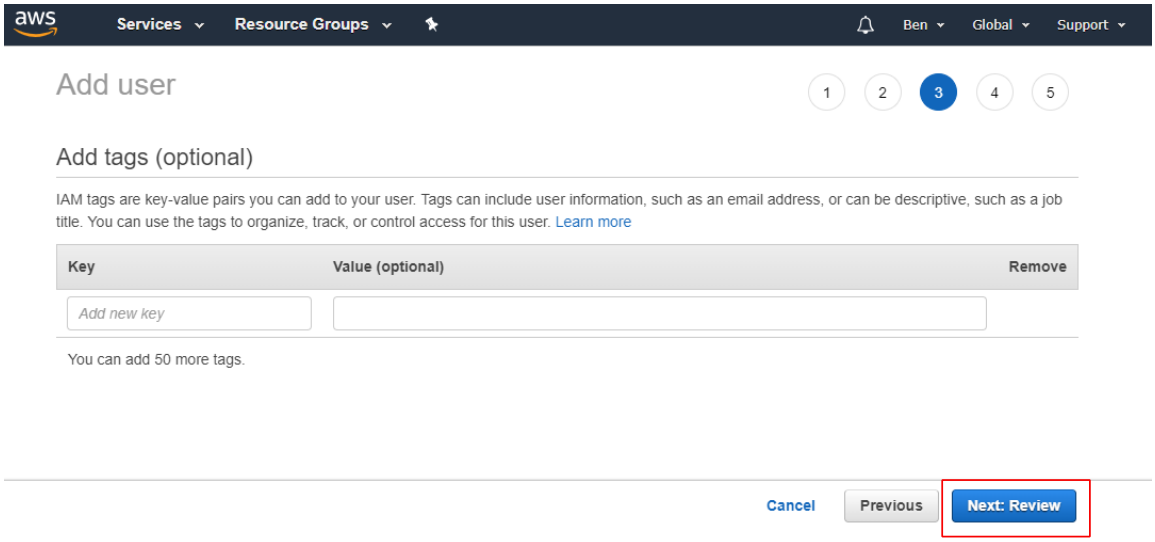
11. In the Set permissions page, the ZTGroup is preselected. This sets permissions for user ZTUser to group policies specified to ZTGroup. Click **Next: Tags**.

Figure 5-5. Setting Permissions



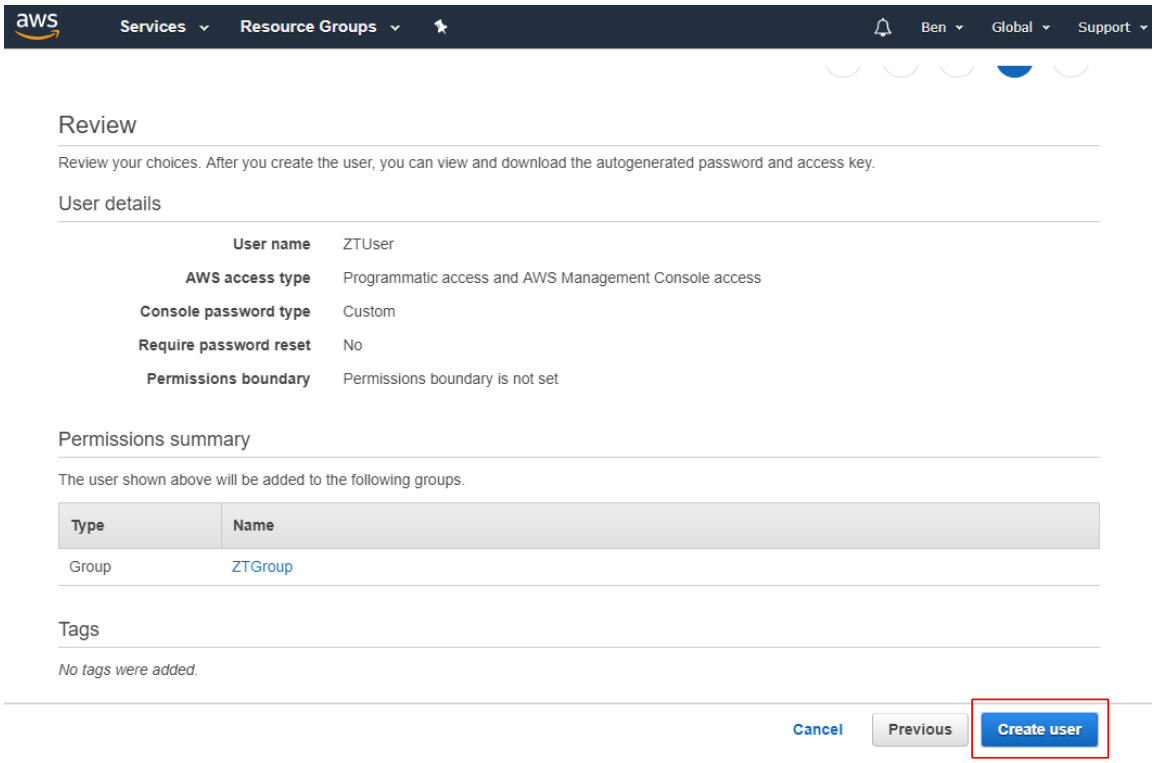
12. This is an optional step and to skip this part, click **Next: Review**.

Figure 5-6. Adding Tags



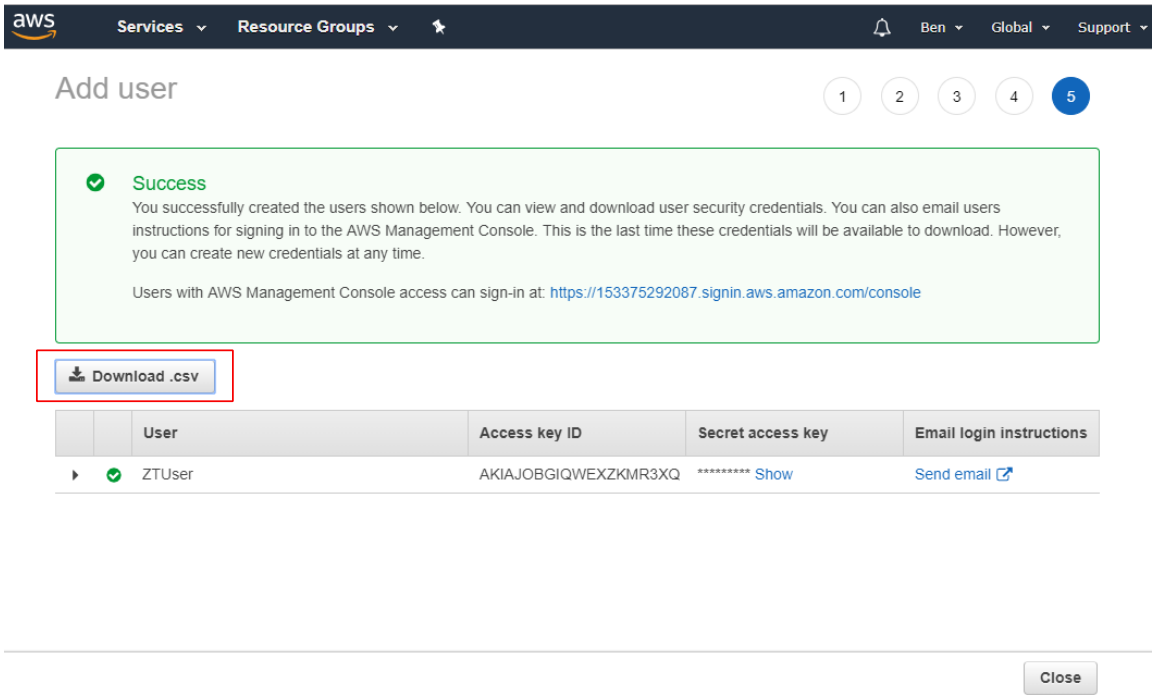
13. Verify the details and click **Create user** to create the user account.

Figure 5-7. Reviewing User Details and Creating a User



14. AWS creates a unique account sign-in URL and access credentials (Access key ID and Secret access key). To download and save the account information file, click **Download .csv**. These credentials are used to configure the settings under ZTUser account later.

Figure 5-8. Successful User Creation Window



5.1.3 Create Policy and Role for JITR

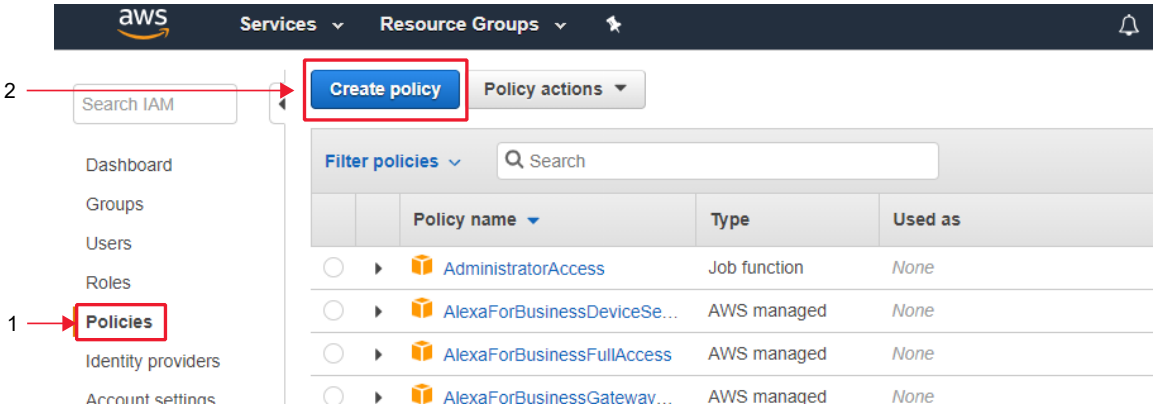
Just-In-Time Registration (JITR) allows the user to register a device at the time of connection. JITR reduces the manufacturing burden of registering a device with AWS before it is connected. In later steps, create a Lambda function that is responsible for registering new devices. The following are the steps to create a custom policy and role that is used by the JITR Lambda function:

Create Policy

To create policy, perform the following steps:

1. Go to https://console.aws.amazon.com/iam.
2. Click “Policies”.
3. From the Policies page, click **Create Policy**.

Figure 5-9. Policy Creation

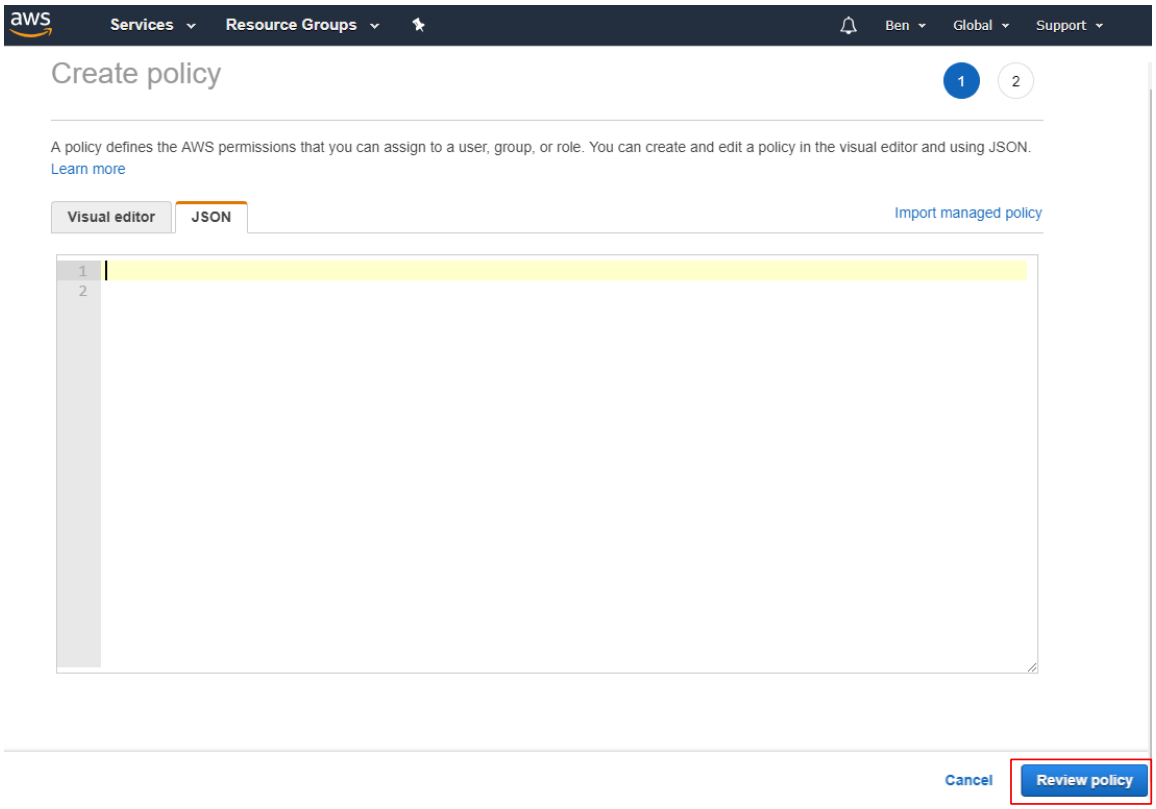


4. Select **JSON**. Copy and paste the following code and click **Review Policy**.

Figure 5-10. Policy Review

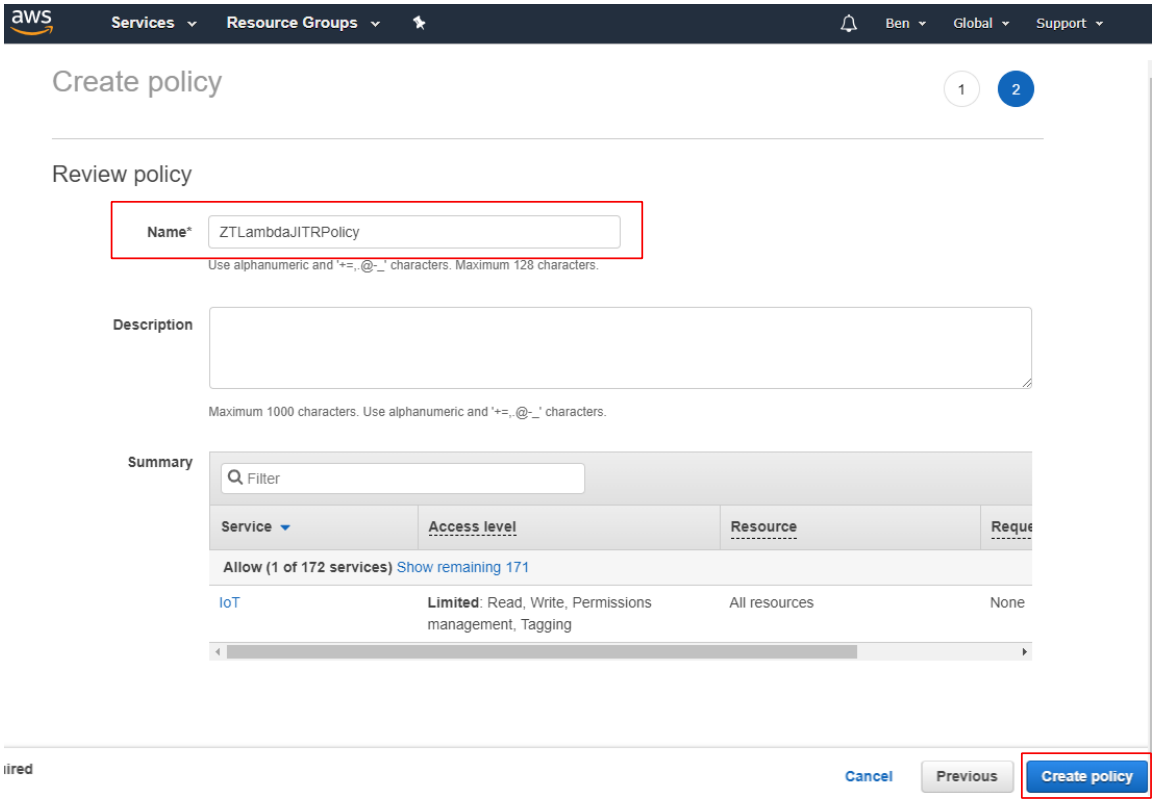
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCertificate",
                "iot:CreatePolicy",
                "iot:AttachPrincipalPolicy",
                "iot:CreateThing",
                "iot:CreateThingType",
                "iot:DescribeCertificate",
                "iot:DescribeCaCertificate",
                "iot:DescribeThing",
                "iot:DescribeThingType",
                "iot:GetPolicy"
                "iot:CreateThingGroup",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

Figure 5-11. Policy Review



5. Enter the policy name as *ZTLambdaJITRPolicy* in the “Name” field. Click **Create policy**.

Figure 5-12. Create Policy

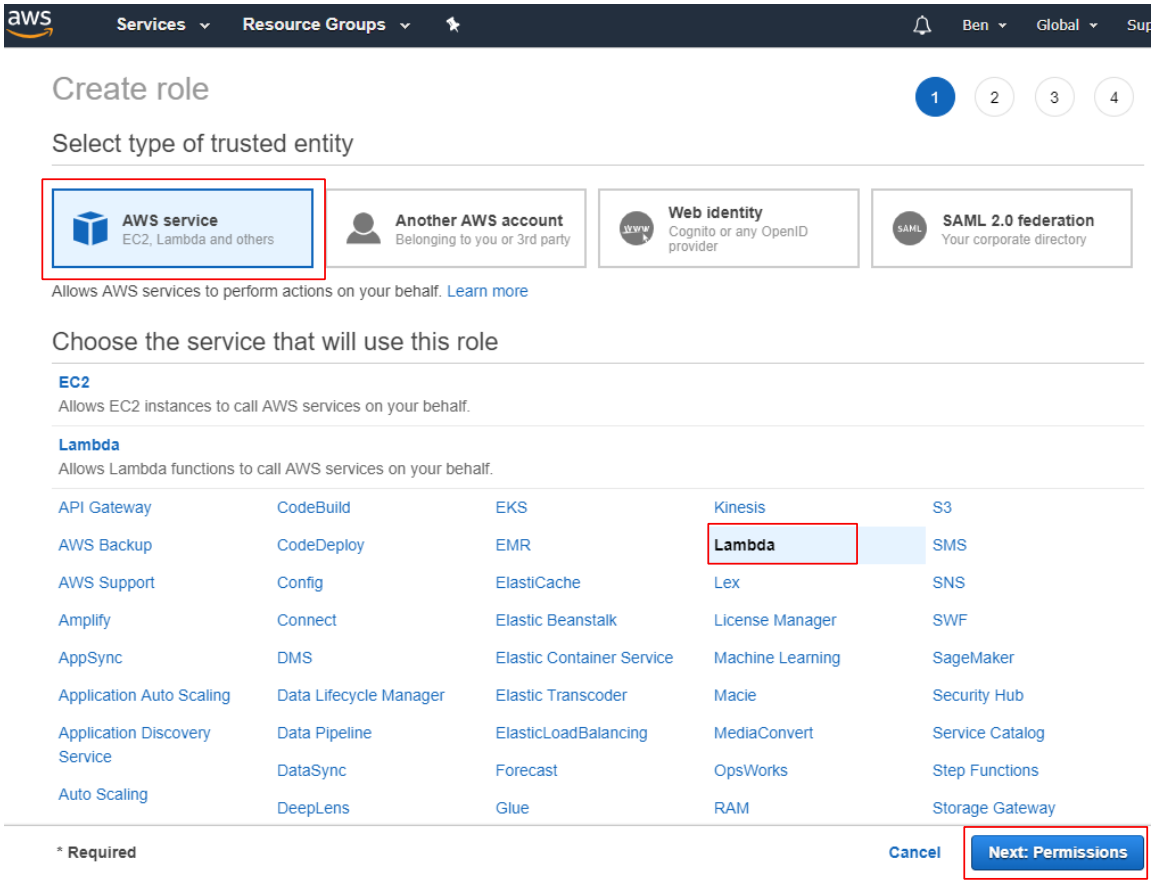


Create Roles

To create roles, perform the following steps:

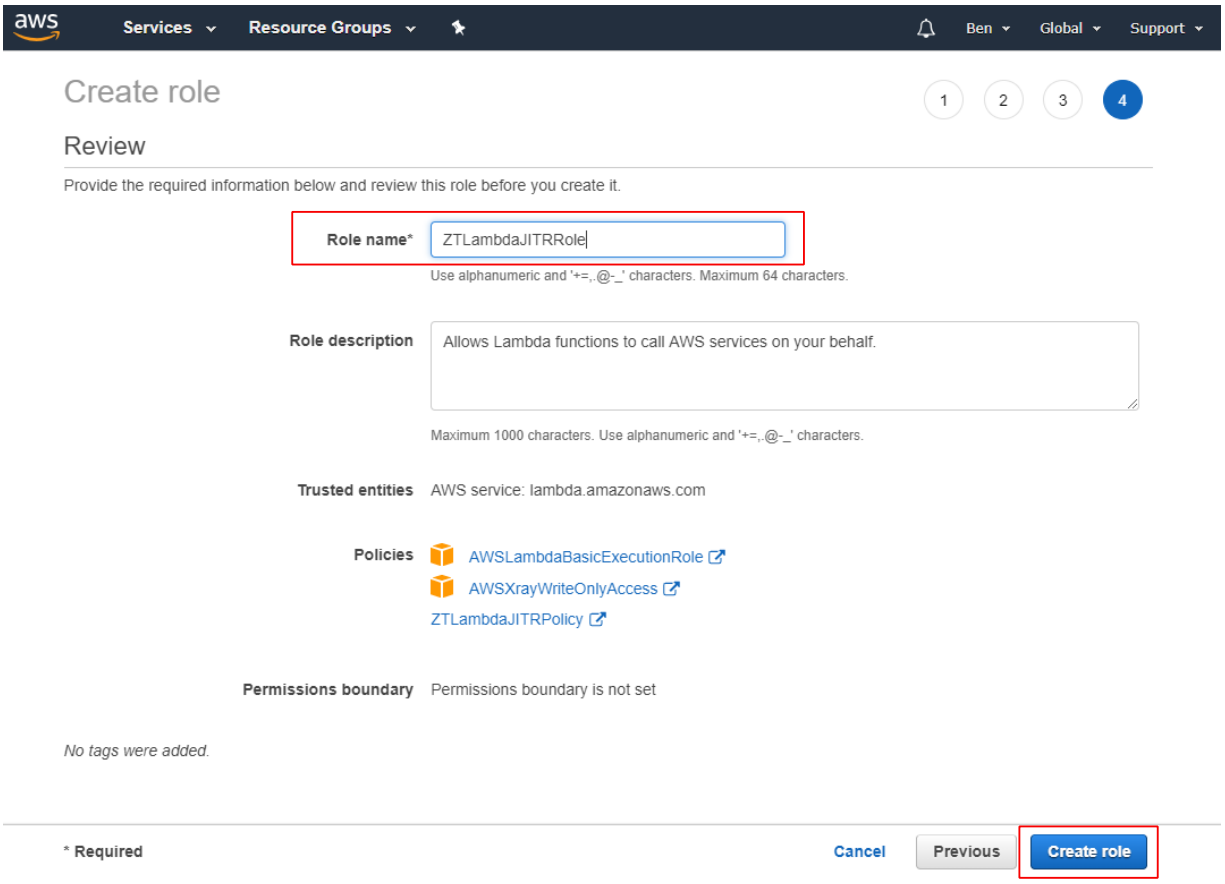
1. Go to https://console.aws.amazon.com/iam.
2. Click **Roles**.
3. In the Roles page, click **Create Role**.
4. Under “Select type of trusted entity”, select **AWS Service** and select “Lambda” service. Click **Next: Permissions**.

Figure 5-13. Select Type of Trusted Entity



5. Attach the following policies:
   - AWSLambdaBasicExecutionRole
   - AWSXrayWriteOnlyAccess
   - ZTLambdaJITRPolicy
6. Click **Next Step**.
7. Set Role name as *ZTLambdaJITRRole*.
8. Click **Create role**.

Figure 5-14. Creating Role



5.1.4 Configure AWS CLI

In the following section, use ZTUser account to configure the cloud settings and provision the board to AWS User account. While performing the AWS account provision, the user needs to configure AWS Command Line Interface (CLI) before running Python scripts.

The following are the steps to configure AWS CLI with ZTUser credentials:

1. Download and install the AWS CLI tool from https://aws.amazon.com/cli/.
2. Run the following command: `aws configure --profile ZTUser` in the Command Prompt.
3. Enter the Access Key ID and Secret Access Key of ZTUser account when prompted. Copy and paste the credentials to avoid any typing mistakes.
4. Observe the following results on the Command Prompt window:

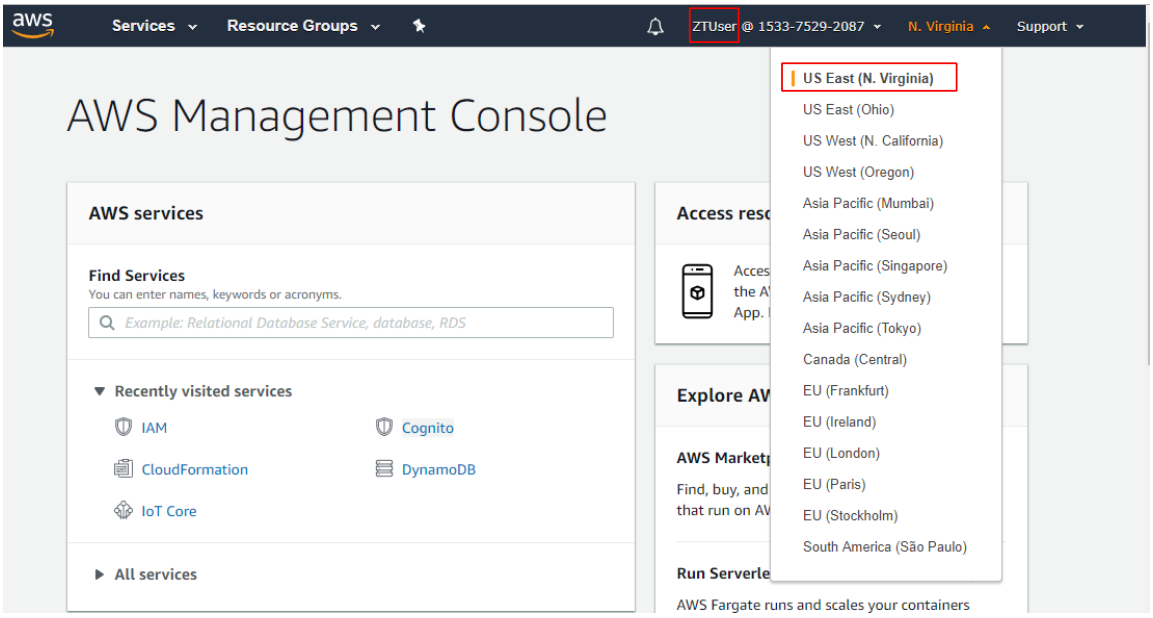
```
>aws configure --profile ZTUser
AWS Access Key ID [None]: ACCESSKEYID
AWS Secret Access Key [None]: SECRETACCESSKEY
Default region name [None]: us-east-1 ( <-- Enter the region that you selected )
Default output format [None]:
```

5.2 AWS IoT Just-in-Time Registration Setup

Use the ZTUser account to configure AWS IoT settings.

1. Open a web browser and go to the user sign-in URL that was assigned when you created ZTUser. The URL will have the following format:
   – https://xxxxxxxxxxxx.signin.aws.amazon.com/console, where xxxxxxxxxxxx is the account ID
   – Enter the User Name *ZTUser*.
   – Enter the Password set when creating the user account.
2. Once logged in, change your region to the one closest to you by selecting the region menu (upper-right, left of support menu). In the following steps **US East (Virginia)** is selected. There are only 4 regions that can support the Lambda function for Alexa skills. Make sure to select the following four regions:
   – Asia Pacific (Tokyo)
   – EU (Ireland)
   – US East (N. Virginia)
   – US West (Oregon)

Figure 5-15. List of Countries

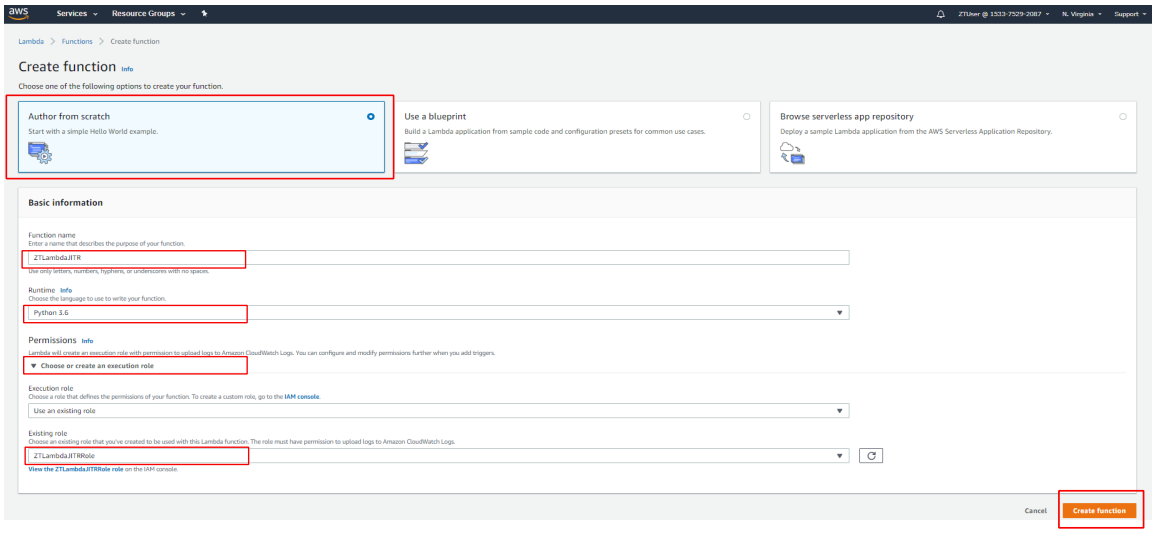


Create Lambda function

Perform the following steps to create the JITR Lambda function.

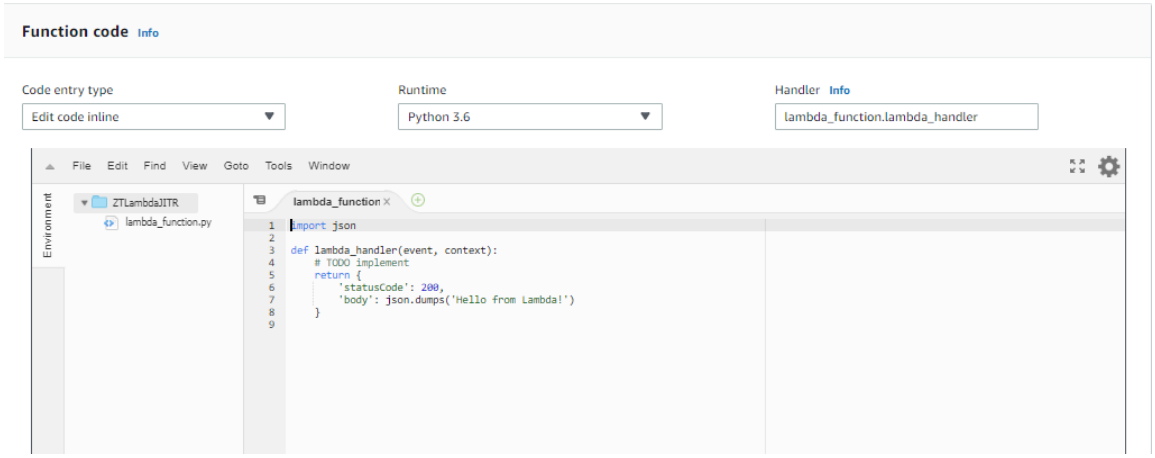
1. Click https://console.aws.amazon.com/lambda. The JITR Lambda function is code that is called from AWS IoT when a new device attempts to connect but is not registered yet. It is the function's responsibility to perform the actual registration of the device with AWS IoT.
2. Click **Create Function**.
3. Select “Author from scratch”. Type the name of the new function in the “Name” field “ZTLambdaJITR”.
4. Select “Python 3.6” under the “Runtime” field, select “Choose an existing role” under the 'Role' field, and select the previously created “ZTLambdaJITRRole” under the 'Existing role' field.
5. Click **Create Function**.

Figure 5-16. Creating a Function



6. Copy and paste the code from `lambda-function/zt-lambda-jitr/ZTLambdaJITR.py` to the code entry area of the Lambda function.

Figure 5-17. Adding a Functional Code



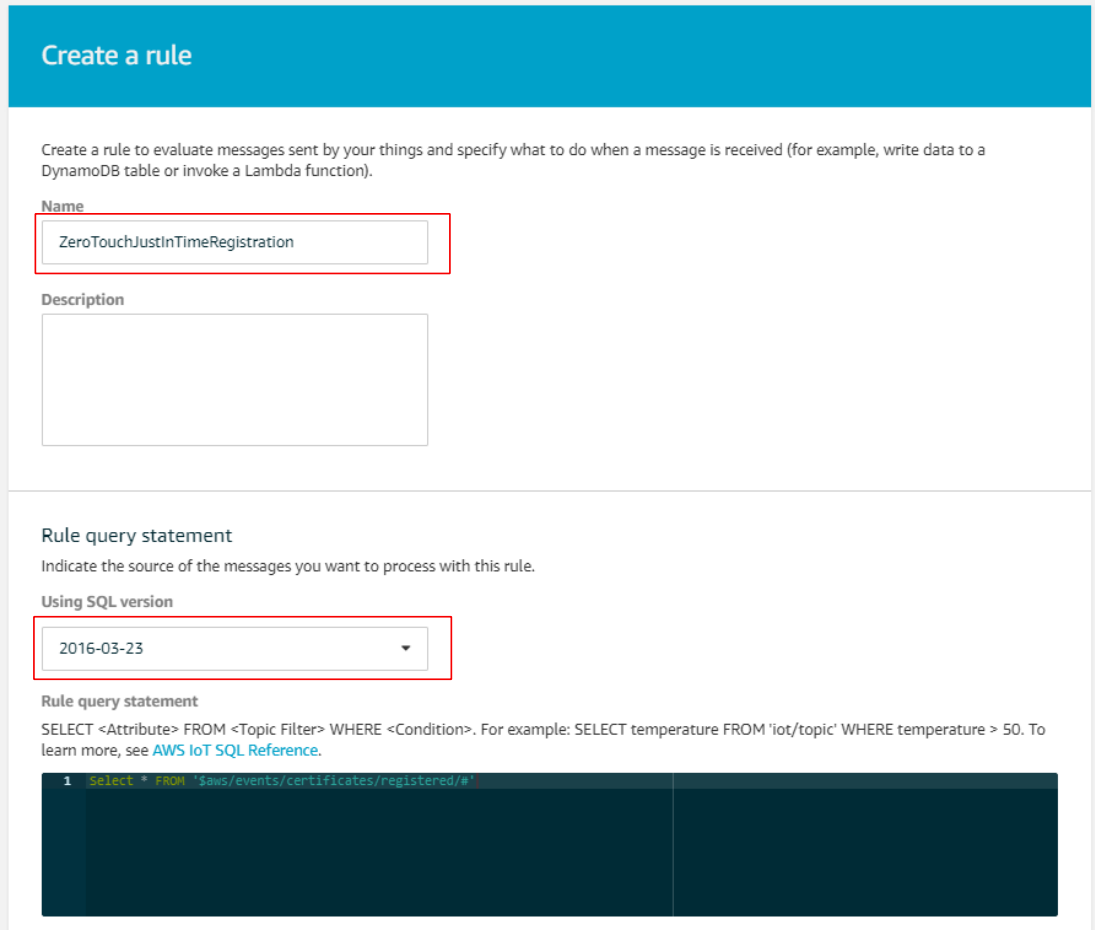
7. Save changes to the Lambda function code.

Create IoT Rules Engine

While the Lambda function performs the registration, it needs to be triggered by an event. The following instructions creates a rule, which runs the Lambda function when a device connects for the first time.

1. Click https://console.aws.amazon.com/iot.
2. Go to the 'Act' section from the menu.
3. Click the 'Create a rule' button.
4. Fill in the following fields:
   - Name: ZeroTouchJustInTimeRegistration
   - SQL version: 2016-03-23
   - Rule query statement: Select * FROM '$aws/events/certificates/registered/#'
   - Condition:

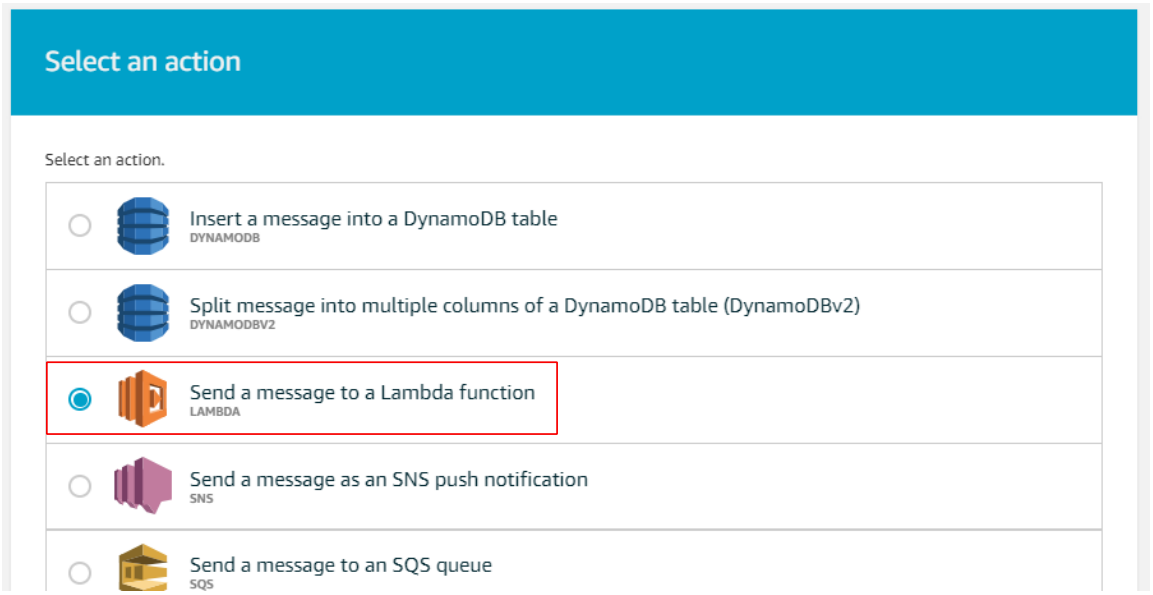
Figure 5-18. Creating a Rule



$aws/events/certificates/registered/# is a special administrative MQTT topic that AWS IoT will publish to when a device connects with a certificate that hasn't been seen before but has been signed by a CA that was registered in the account. The **#** at the end indicates we want to trigger this rule for any CA registered with the account.

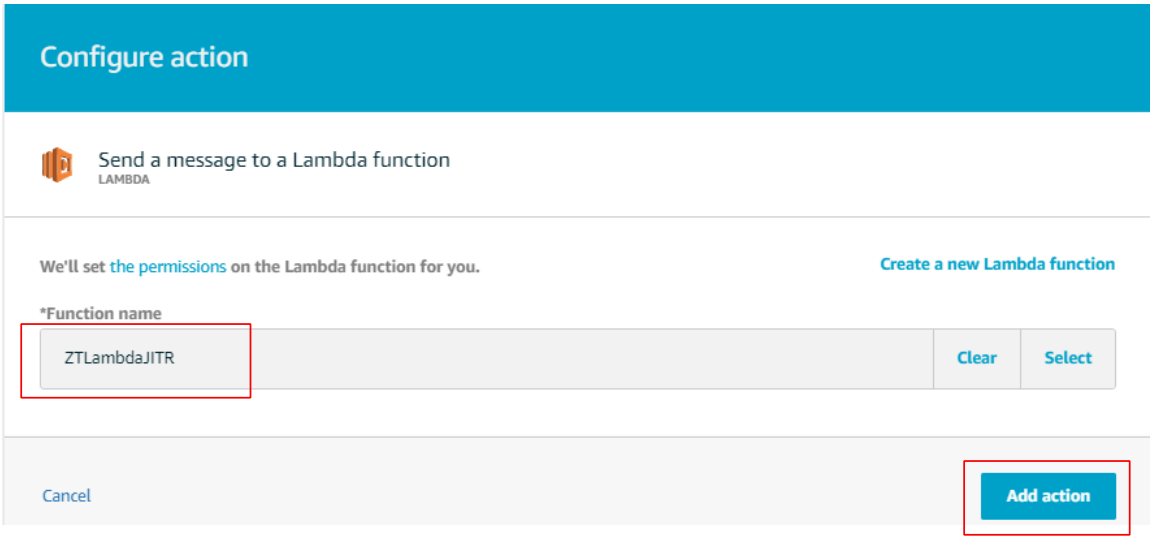
5. Click **Add action**.
6. Select **Send a message to a Lambda Function**.

Figure 5-19. Selecting an Action for a Rule



7. Click **Configure action**.
8. Select the “ZTLambdaJITR” function.

Figure 5-20. Configure Action



9. Click **Add action** and **Create rule** to finish the action.

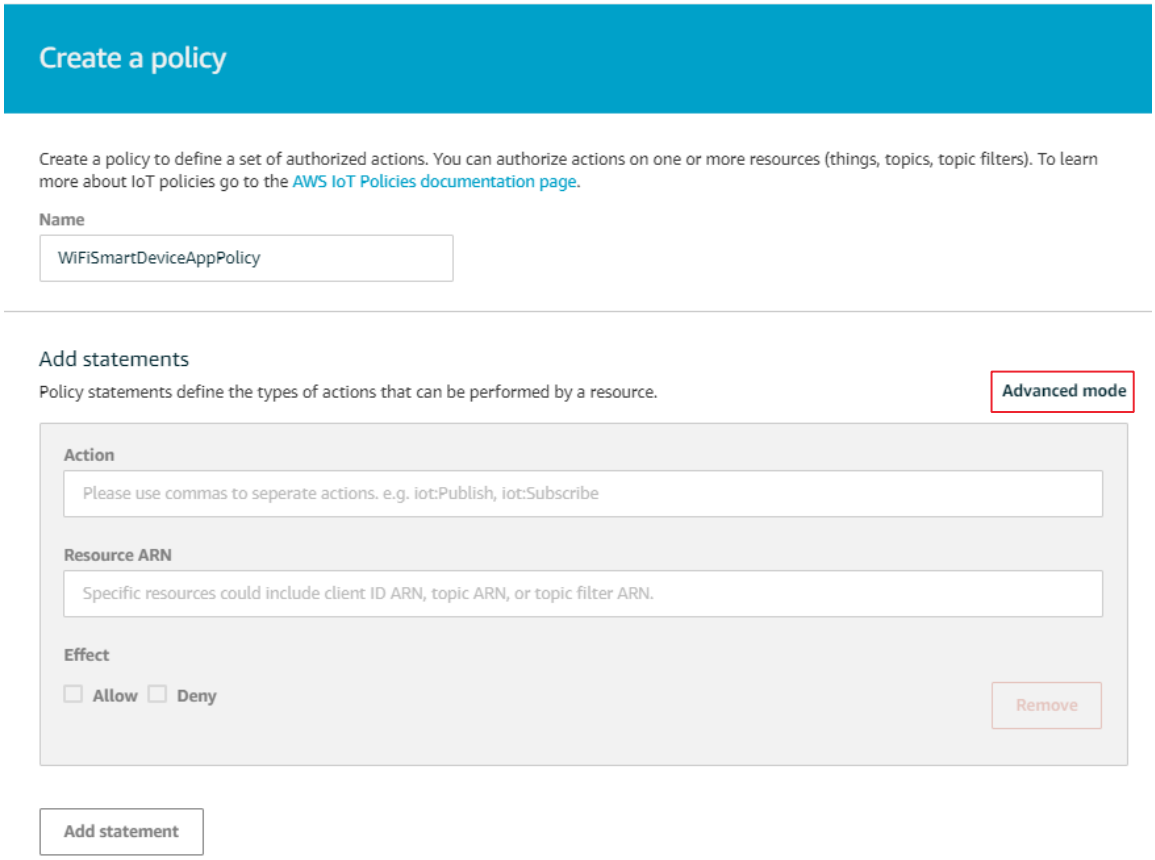
5.3 AWS IoT Mobile App Setup

1. Click https://console.aws.amazon.com/iot/.
2. Click **Secure**, and **policies** in left panel.
3. Click **Create**.
4. Type in policy name (example: WiFiSmartDeviceAppPolicy). Note down this policy name, as this string will be used in the demo mobile app source code, when making the attach policy API call.

The following is an example for the policy. This policy allows access to all topics under your AWS IoT account. Copy/paste the following text to policy statement.

5. Select **Advance mode** to add statements.

Figure 5-21. Adding Statements using Advanced Mode Option



6. Add the following example policy to the policy statement to allow access to all topics under the specific AWS IoT account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "*"
    }
```

```
    ]
}
```

7. Click **Create**.

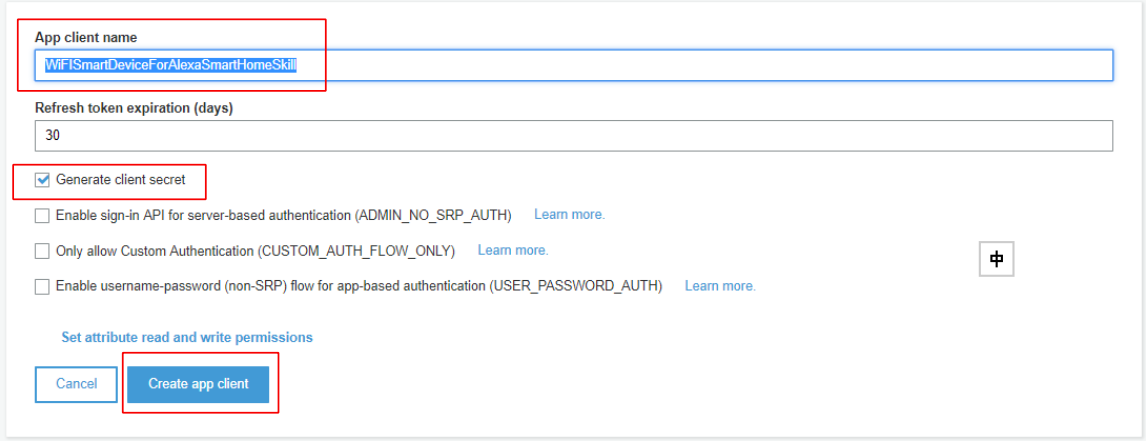
5.4 Amazon Cognito Setup

Amazon Cognito is used to provide user identity for the product end users, including sign-up, sign-in features. With Amazon Cognito, user can sign-up and sign-in the account using mobile application and control the boards.

Perform the following steps to set up Amazon Cognito:

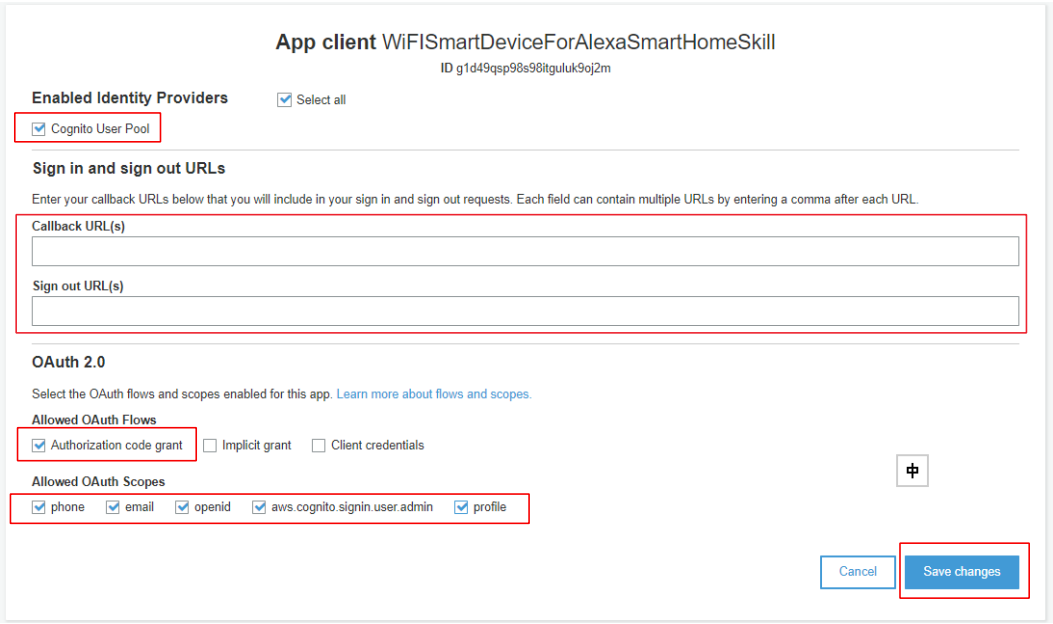
1. Go to Amazon Cognito Console in AWS cloud https://console.aws.amazon.com/cognito/.
2. Click **Manage User Pools** to open **Your User Pools** browser.
3. Click **Create a user pool**.
4. Enter pool name as *WiFiSmartDeviceUserPool* and select *Review defaults*.
5. Click **Create pool** to create the user pool.
6. Select "App clients" in the left panel and click **Add an app client** to create App Client “WiFISmartDeviceForAlexaSmartHomeSkill" in the user pool just created.
7. Tick the "Generate client secret" check box.
8. Click **Create app client**.

Figure 5-22. Creating Smart Home App Client



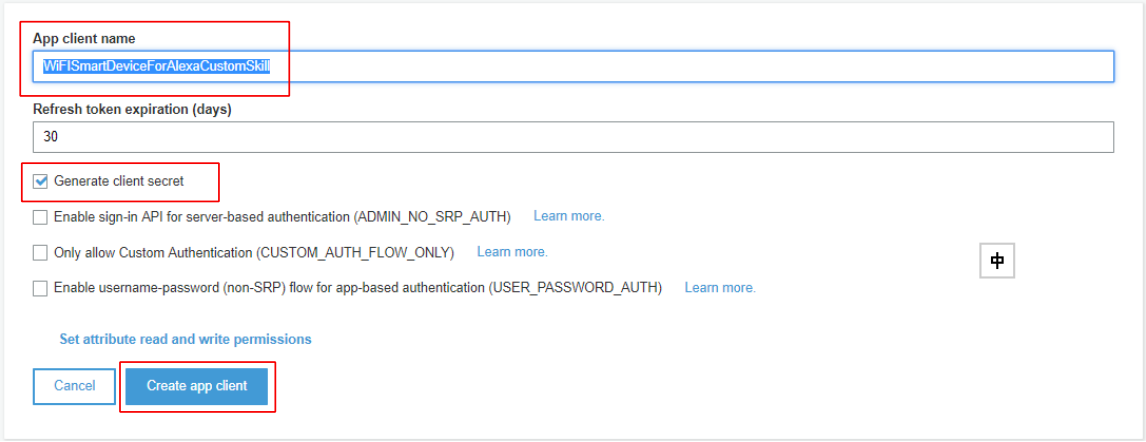
9. Select "App clients settings".
10. In App Client setting of "WiFISmartDeviceForAlexaSmartHomeSkill", tick the "Cognito User Pool" check box.
11. Additionally tick the "Authorization code grant", "phone", "email", "openid", "aws.cognito.signin.user.admin", and "profile" check boxes.
12. Add the Callback URL with all the Alexa Smart Home Skill Re-direct URL. This URL can be found when you create Alexa Smart Home Skill in 7.1 Microchip Wi-Fi Smart Device Smart Home Skill Setup .
13. Add https://www.google.com in the “Sign out URL” field.
14. Click **Save changes**.

Figure 5-23. App Client Configuration for Smart Home Skill



15. Select "App clients" and click **Add an app client** to create App Client " WiFISmartDeviceForAlexaCustomSkill" in the User Pool just created.
16. Select "Generate client secret" check-box.
17. Click **Create app client**.

Figure 5-24. Creating Custom App Client

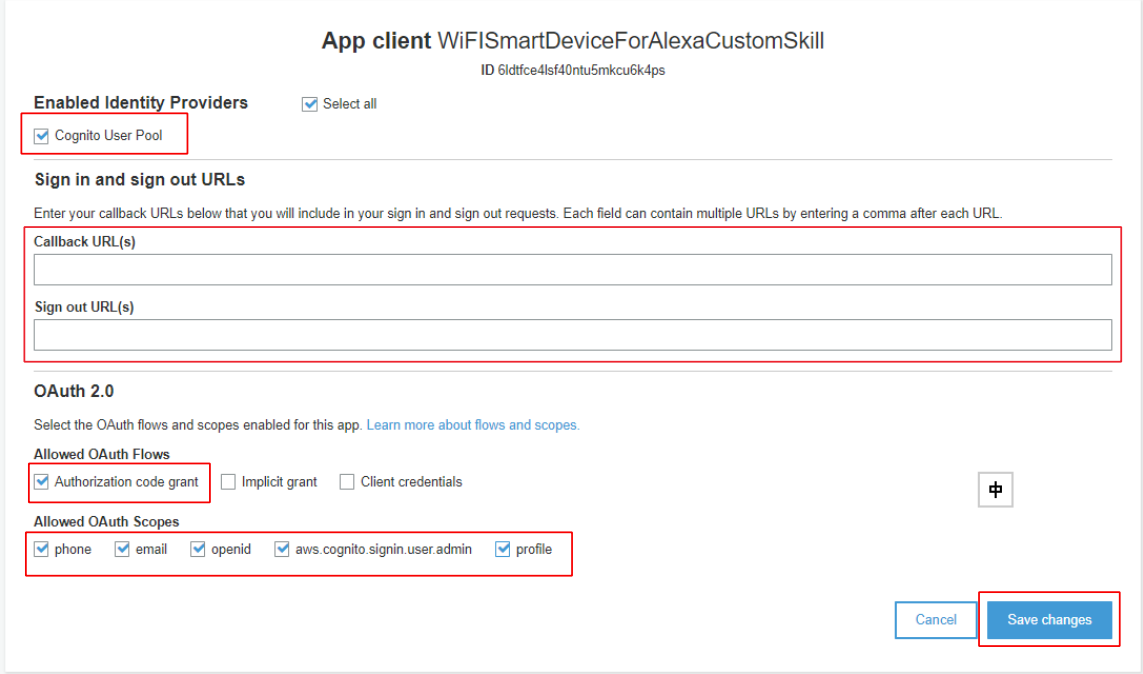


18. Select "App clients settings".
19. In App Client setting of " WiFISmartDeviceForAlexaCustomSkill ", select the following check-boxes:
    - "Cognito User Pool"
    - "Authorization code grant"
    - "phone"
    - "email"
    - "openid"
    - "aws.cognito.signin.user.admin"

– "profile"

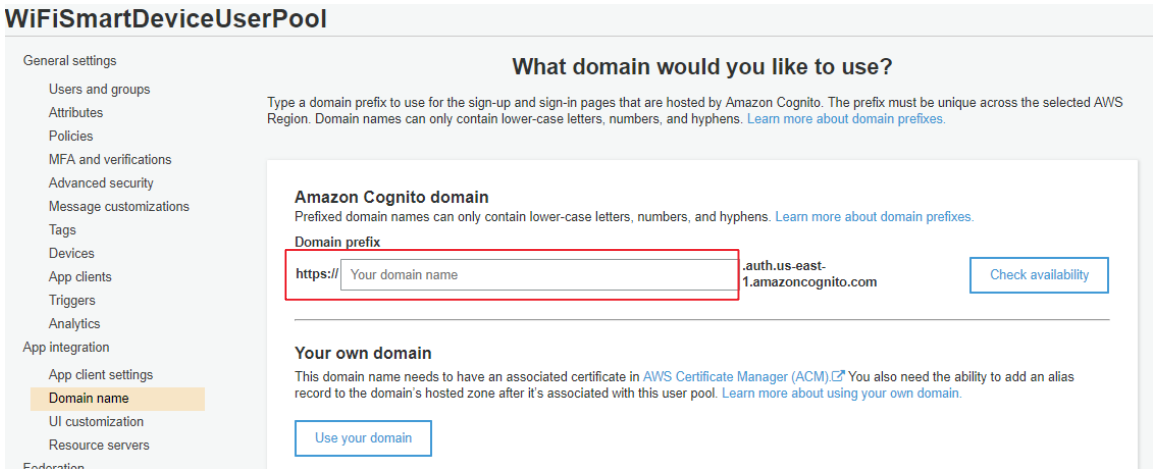
20. Fill in Callback URL with all the Alexa Custom Skill Re-direct URL. This URL can be found when you create Alexa Custom Skill in 7.2 Microchip Wi-Fi Sensor Board Skill Setup . Fill in “Sign out URL”, fill in https://www.google.com.
21. Click **Save changes**.

Figure 5-25. App Client Configuration for Custom Skill



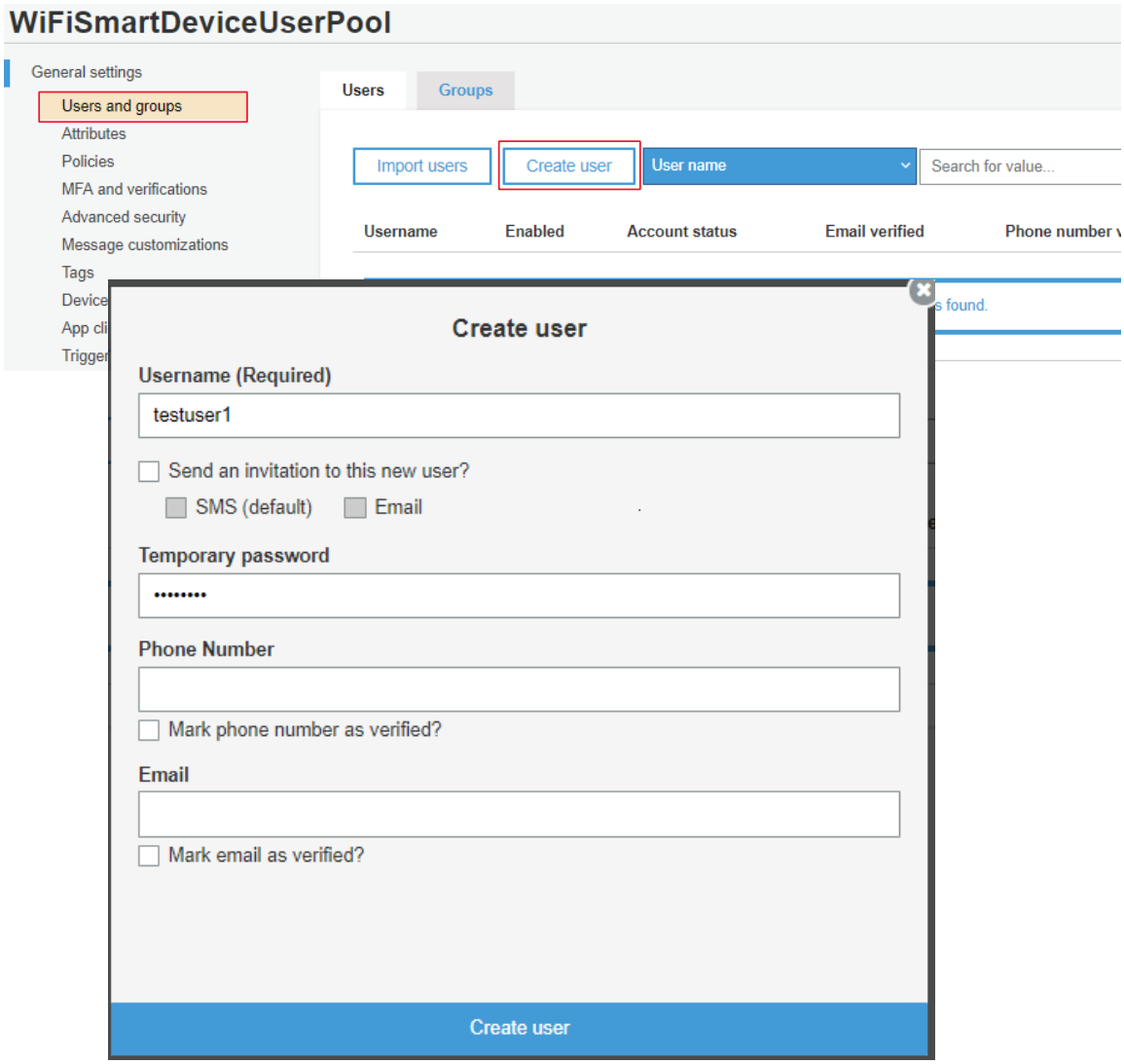
22. Select “Domain Name”, type the domain name. Click **Save Changes**. Change the domain name if the domain name is already used by another.

Figure 5-26. Adding a Cognito Domain Name



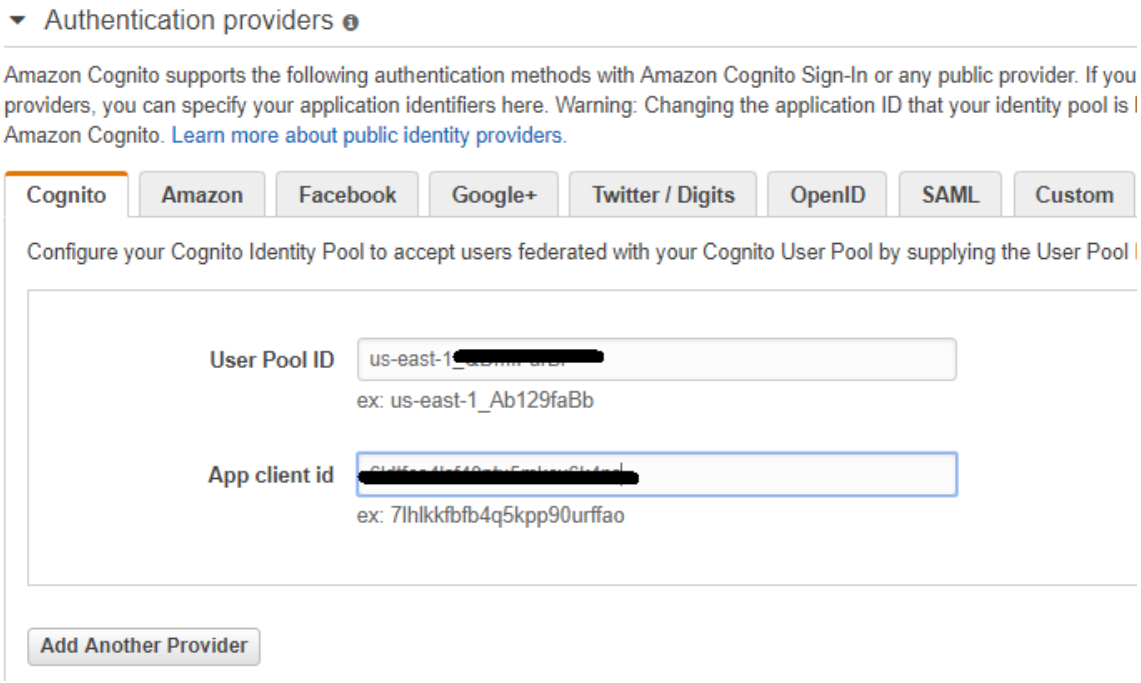
23. Select “Users and groups” and click **Create user**.

Figure 5-27. Selecting a Users and Groups and Creating a User Creadentials



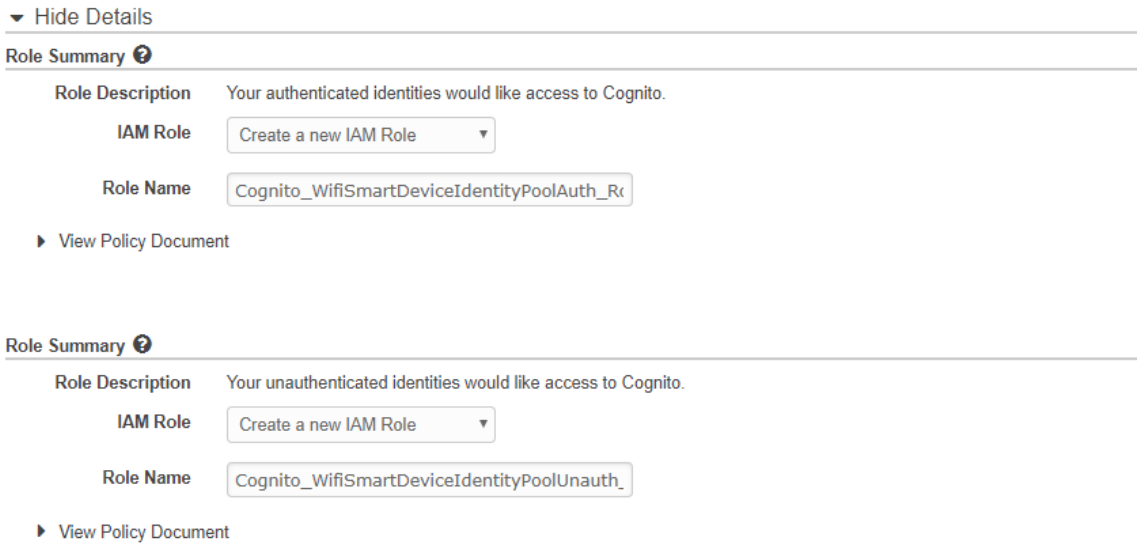
24. Fill in the user information as follows, and click **Create user**.
25. Click https://console.aws.amazon.com/cognito/ to visit Amazon Cognito Console in AWS cloud.
26. Click **Manage Identity Pools**.
27. Click **Create new identity pool**.
28. Type Identity pool name as “WifiSmartDeviceIdentityPool”.
29. In "Authentication providers" of the Identity pool setting, select “Cognito”. Type the “user pool ID” and the “App client ID” of "WiFISmartDeviceForAlexaCustomSkill " (The user pool ID and App client ID is found in the user pool setting page).

Figure 5-28. Authentication Providers



30. Click **Create Pool**. While creating the identity pool, Cognito helps setup two roles in Identity and Access Management (IAM). The example format for the names are: Cognito_<Identity_Pool_Name>Auth_Role and Cognito_< Identity_Pool_Name >Unauth_Role. Click **View Details** to see details on the console.

Figure 5-29. Roles Setup



31. Click **Allow** to create the roles.

32. The user needs to attach a policy to the authenticated role to setup permissions to access the required AWS IoT APIs.
    32.1. Create the IAM Policy shown as follows in the IAM Console and attach it to the authenticated role.
    32.2. Search for the pool name in the IAM console which was created, and click the link for the auth role.
    32.3. Click **Attach Policies** and add the following policy using the JSON tab.
    32.4. Click **Review Policy** and give the policy a descriptive name.
    32.5. Click **Create Policy**. This policy allows the sample app to create a new certificate (including private key) and attach a policy to the certificate.

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

Add “AmazonDynamoDBFullAccess” policy to authenticated role.

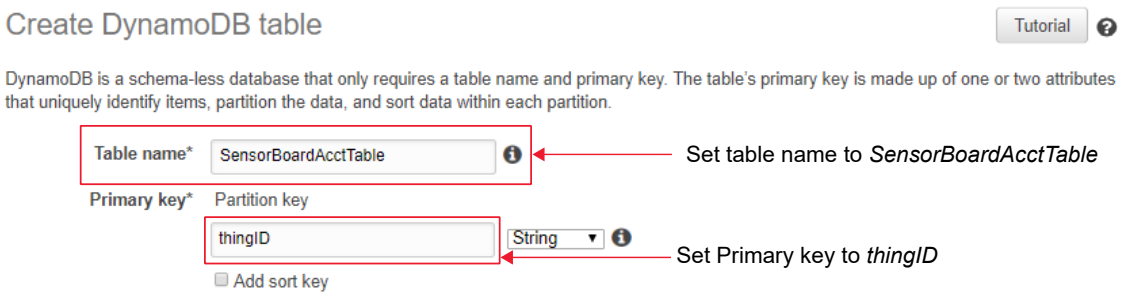
5.5 Amazon DynamoDB Setup

DynamoDB Table is used to store the user account ID, Device Thing ID and the device name. The Mobile App saves this information to the DynamoDB table during the process of board registration. The information in this table is for the Alexa voice control feature. When the user speaks to an Alexa Enabled device (for example, Echo dot), Alexa Cloud sends directives to the Lambda function in the AWS account, and the Lambda function scans this table to control the corresponding Thing Shadow.

Steps:

1. Click https://console.aws.amazon.com/dynamodb.
2. Click **Create Table**.
3. Enter *SensorBoardAcctTable* in the “Table name” field, and *thingID* (String) in the “Primary key” field.

Figure 5-30. Creating a DynamoDB Table



5.6 AWS Lambda Setup

Lambda functions are needed to process the directives received from Alexa Skills. There are two Alexa Skills for the Wi-Fi Smart Device Enablement Kit (Microchip Wi-Fi Smart Device Smart Home Skill and Microchip Sensor Board Skill). Each Alexa Skill needs one Lambda function for processing the directives. Details of the skills are described in 4.3.6 Alexa Voice Control.

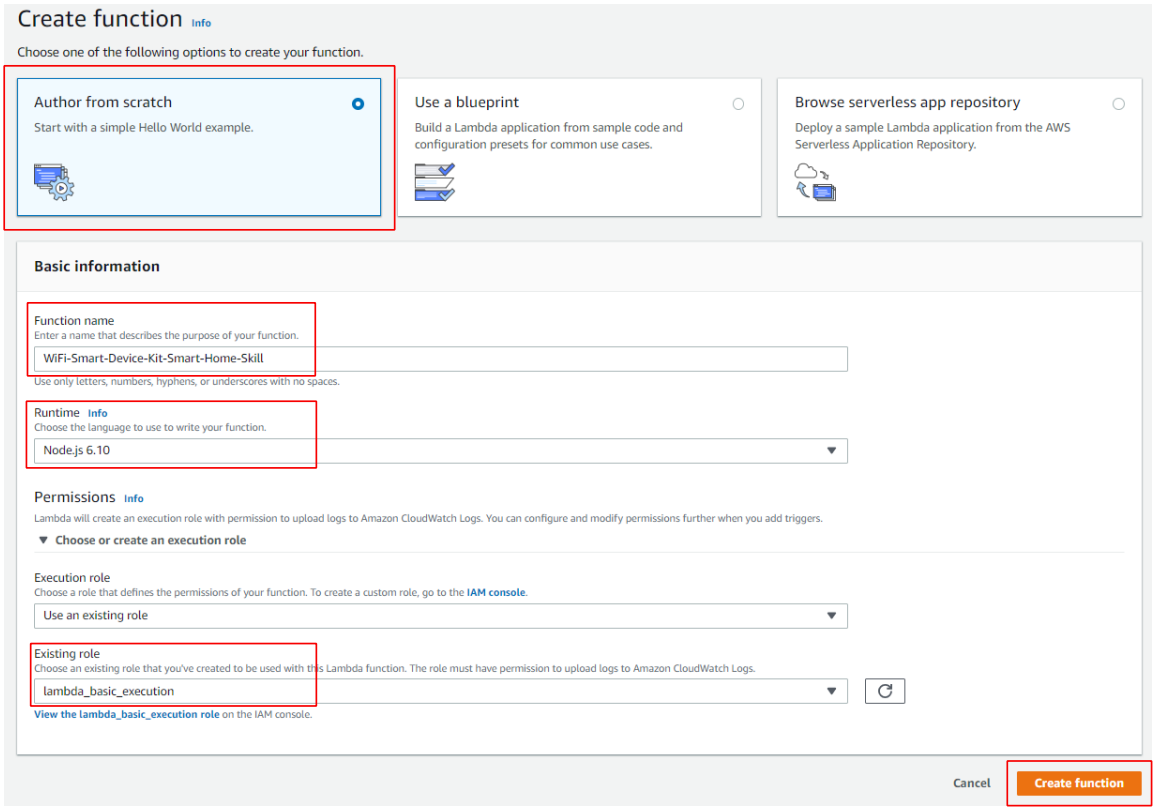
5.6.1 Lambda Setup for Microchip Wi-Fi Smart Device Smart Home Skill

Lambda function code for this Smart Home Skill is available in the `\lambda-function\alexa-smart-home-skill` directory.

Perform the following steps to set up Lambda for Microchip Wi-Fi Smart Device Smart Home Skill:

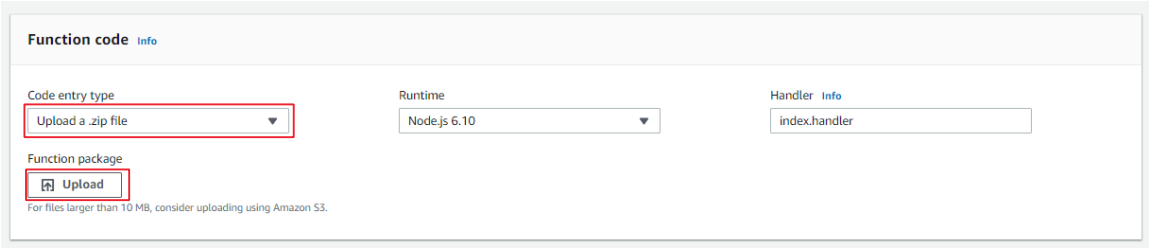
1. Go to AWS Lambda Console in AWS cloud https://console.aws.amazon.com/lambda.
2. Click **Create Function**.
3. Select "Author from scratch" and fill in the following:
   3.1. A name for the Lambda function. For example: “WiFi-Smart-Device-Kit-Smart-Home-Skill”.
   3.2. Select "Node.js 6.10" from the "Runtime" drop down.
4. Select a role for the Lambda function from the “Roles” field (user can create a Role in IAM console, the role must have "AWSLambdaFullAccess" and "CloudWatchLogsFullAccess" policy). The following figure shows the “lambda_basic_execution” role.
5. Go to IAM console, select “Roles” and select “lambda_basic_execution” role. Attach policy "AWSLambdaFullAccess" and "CloudWatchLogsFullAccess".

Figure 5-31. Creating a Lambda Function



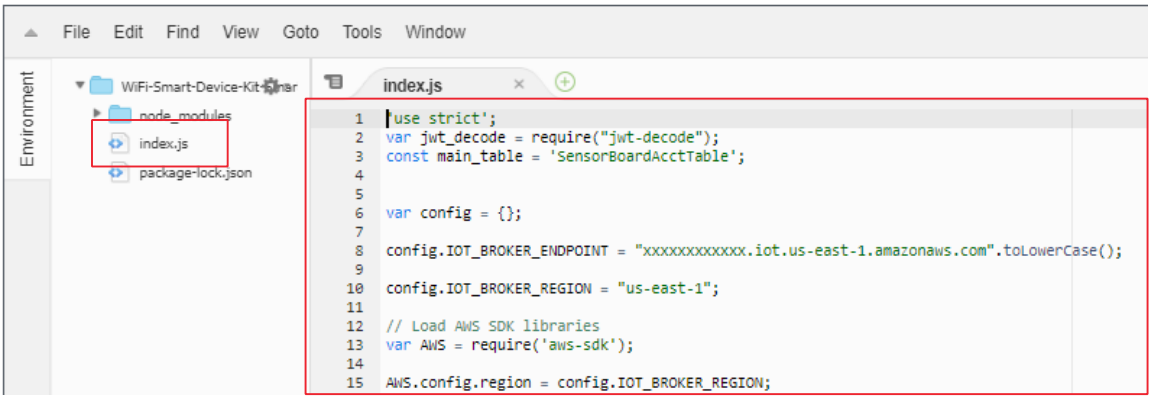
6. After creating the Lambda function, upload the function code to the Lambda function. As the function code includes a number of files, the user can upload the code as a zip file as explained in the following steps.
   - Select "Upload a .ZIP file" at "Code entry type".
   - Select `\lambda-function\alexa-smart-home-skill\alexa-smart-home-skill.zip` and click **Save**.

Figure 5-32. Uploading a Lambda Function Code



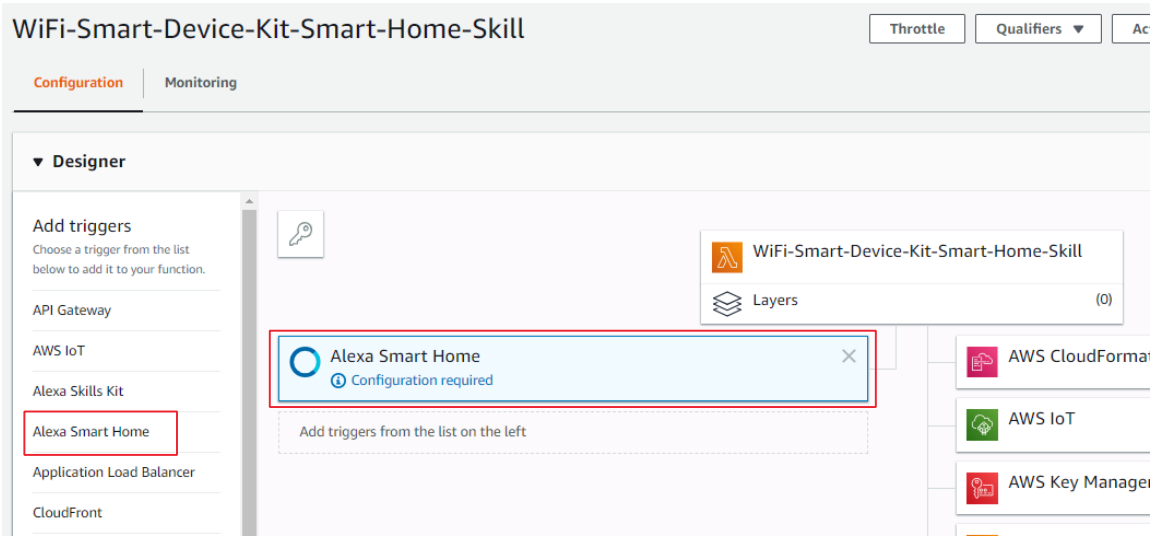
7. In “Function code”, edit `index.js` to add the AWS IoT Endpoint and the Region. The AWS IoT Endpoint can be found from the Settings page of AWS IoT Console.

Figure 5-33. Editing index.js



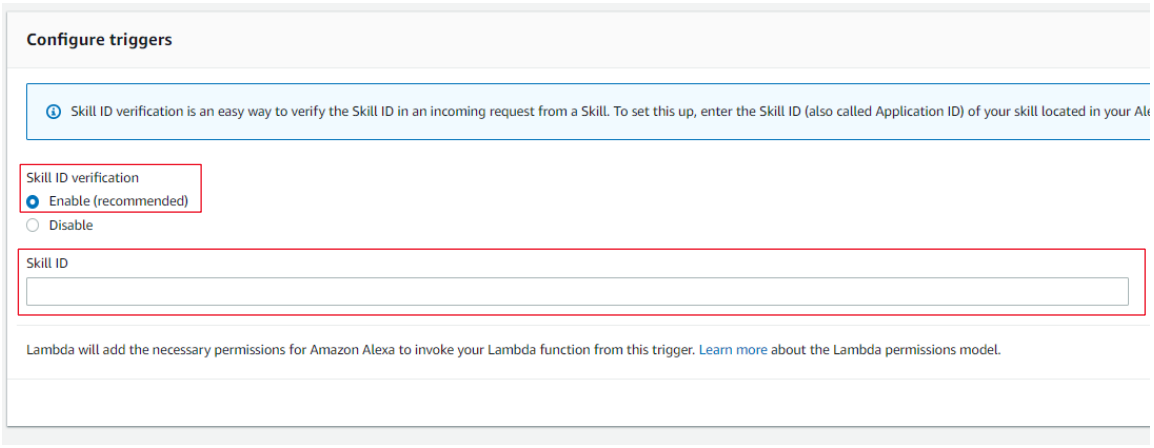
8. Add triggers "Alexa Smart Home", as shown in the following figure:

Figure 5-34. Adding Triggers



9. Add the skill ID in “Skill ID” field, and select “Enable” in “Skill ID verification”. Alexa Skill ID can be found in the Alexa Skill configuration page, refer to 7. Alexa Skill Setup. User can fill this information later after creating the Alexa skill.

Figure 5-35. Configuring Triggers



10. Click **Save**.

5.6.2 Lambda Setup for Microchip Wi-Fi Sensor Board Skill

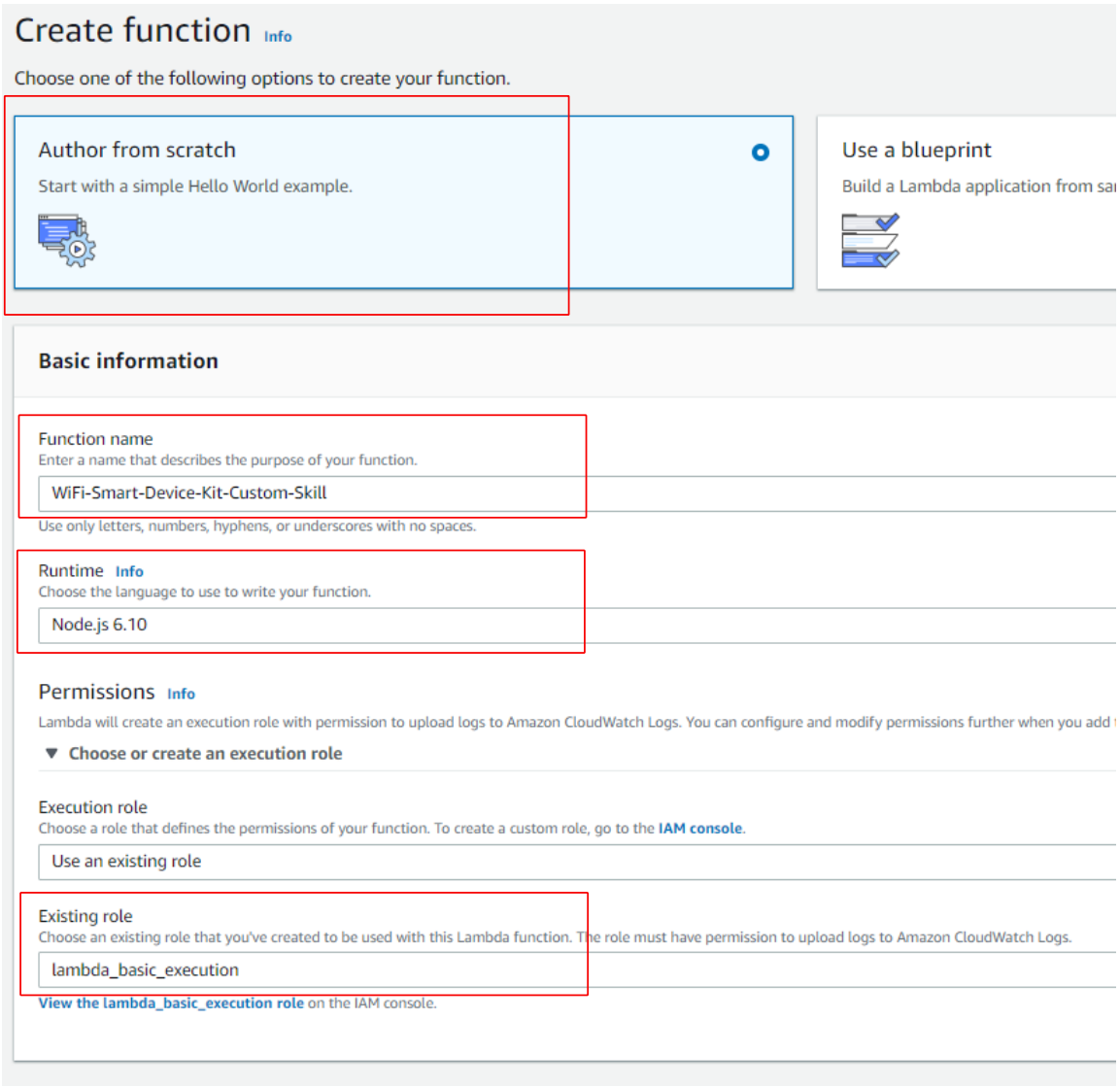
Lambda function code for Custom Skill is available in the `\lambda-function\alexa-custom-skill` directory.

Perform the following steps to setup Lambda for Microchip Wi-Fi sensor board skill:

1. Go to AWS Lambda Console in AWS cloud https://console.aws.amazon.com/lambda.
2. Click **Create Function**.
3. Select "Author from scratch" and add a name for the Lambda function. The name can be “WiFi-Smart-Device-Kit-Custom-Skill”.
4. Select "Node.js 6.10" for "Runtime".

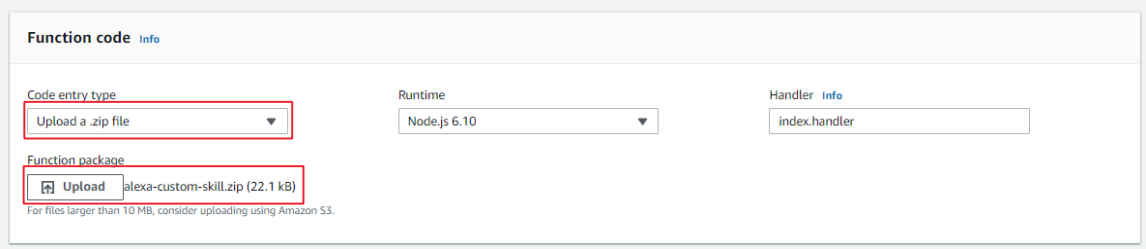
5. Select a Role for the Lambda function (user can create a Role in IAM, the role must have "AWSLambdaFullAccess" and "CloudWatchLogsFullAccess" policy). In the following figure, “lambda_basic_execution” role is created.
6. Go to IAM console, select “Roles”, select “lambda_basic_execution” role. Attach policy "AWSLambdaFullAccess" and "CloudWatchLogsFullAccess" to this role.

Figure 5-36. Creating a Lambda Function for Custom Skill



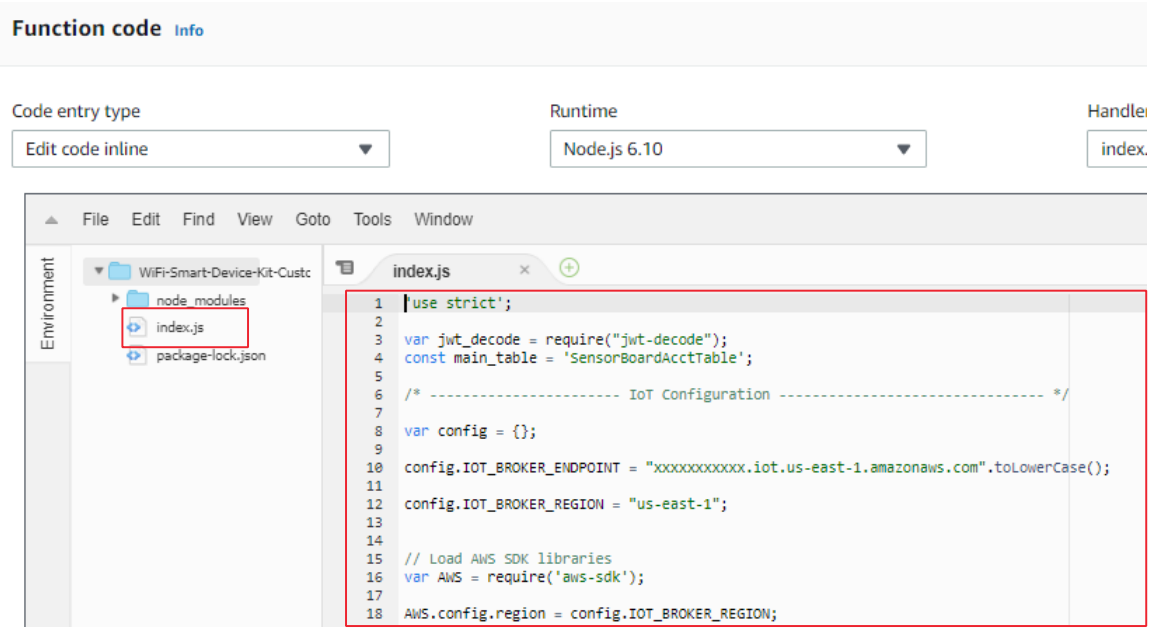
7. After creating the Lambda function, upload the function code to the Lambda function. As the function code includes a number of files, you can upload the code as a zip file.
   - Select "Upload a .ZIP file" at "Code entry type".
   - Select `\lambda-function\alexa-custom-skill\alexa-custom-skill.zip` and click **Save**.

Figure 5-37. Uploading a Lambda Function Code



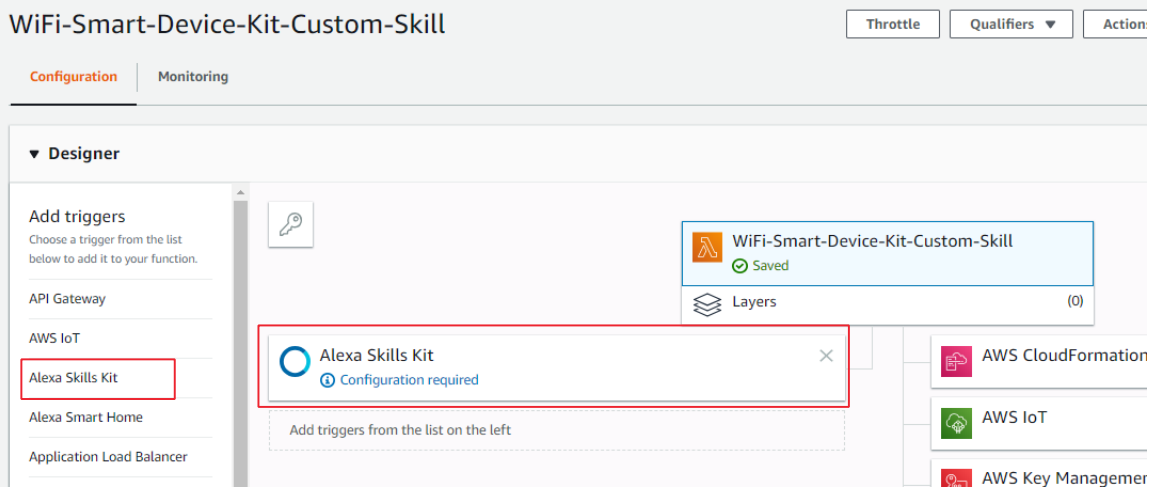
8. In “Function code”, edit `index.js` to add the AWS IoT Endpoint and the Region. The AWS IoT Endpoint can be found from the Settings page of AWS IoT Console.

Figure 5-38. Editing index.js



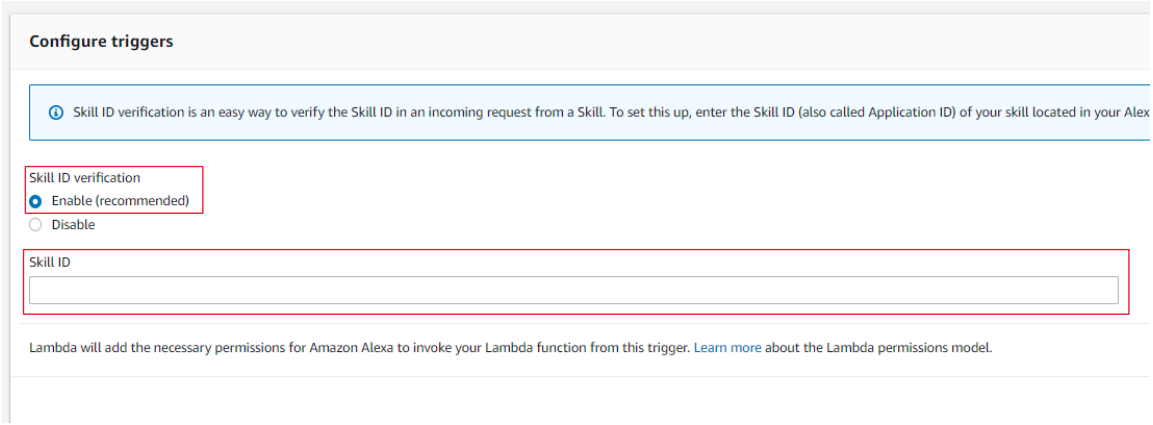
9. Add the trigger "Alexa Skill Kit" as shown in the following figure:

Figure 5-39. Adding Triggers



10. Add the Skill ID in the “Skill ID” field, and enable the skill in skill configuration. Alexa Skill ID can be found in Alexa Skill configuration page (refer to 7. Alexa Skill Setup). The user can fill this information later after creating the Alexa skill.

Figure 5-40. Configuring Triggers



11. Click **Save**.

6. AWS Provision Setup

The Wi-Fi Smart Device Enablement Kit is pre-programmed with the credentials required to connect and communicate with the Microchip AWS IoT account. The pre-programmed AWS account credentials are lost after re-programming the board. The user needs to migrate the board to connect to the private AWS account for customization. This section shows the steps to provision the board to connect to the private AWS account created in 5. AWS Setup.

These are the following steps for the AWS provisioning procedure:

1. Create and register the signer CA to AWS IoT account.
2. Request a Certificate Signing Request (CSR) from the board.
3. Create a device certificate using the CSR and signer CA.
4. Send the device certificate, signer certificate and AWS connection information to the board.

The Python scripts in the /ProvisionScripts directory are used to perform the above tasks.

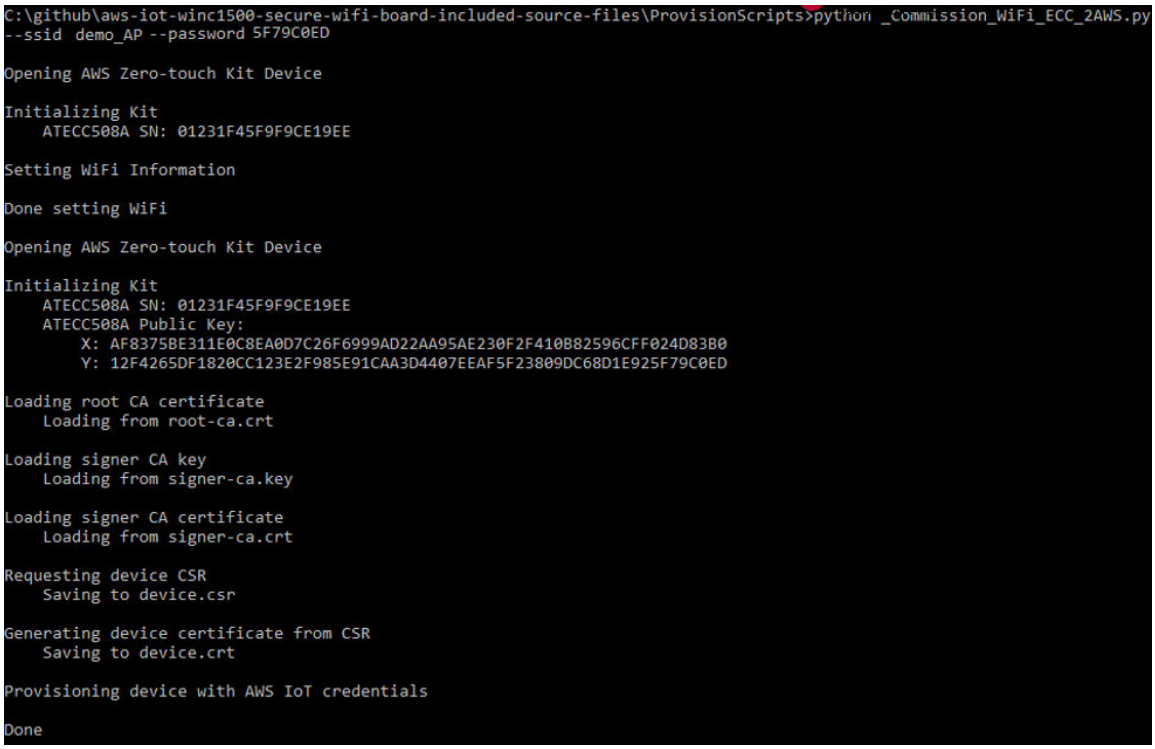
Perform the following steps:

1. Open Command Prompt, go to /ProvisionScripts folder and input the Python command: `python _CreateCertsAndRegister2AWS.py --profile <your-aws-cli-profile name>`
   Note: Make sure to install and configure AWS CLI tool with your AWS account (ZTUser). The profile name of your account can be found in `~/.aws/`credentials. The steps to configure AWS CLI can be found in 5.1.4 Configure AWS CLI.

   This command performs the following tasks:

   1.1. Creates a certCA (`rootCA.crt`) with its private key (`root-ca.key`, if this one already exists it reuses it)
   1.2. Creates a signing certificate (`signer-ca.csr`) and its private key (`signer-ca.key`, if this one already exists it reuses it)
   1.3. The root-ca signs the `signer-ca.csr` and creates `signer-ca.crt`.
   1.4. This `signer-ca.crt` is then uploaded to the AWS IoT account.
2. Program the firmware of the project `saml21g18b_sensor_board_demo_ECC.atsln` to the board. Refer to 4.1 Application Firmware Compilation Procedure for programming the firmware procedure. Project `saml21g18b_sensor_board_demo_ECC.atsln` is used to perform AWS account provision. The developer needs to program this project code to the board to perform AWS account provision to migrate the board to connect with the private AWS account.
3. LED LD2 on board will blink in yellow color.
4. Open Command Prompt, go to /ProvisionScripts folder and input the `_Commission_WiFi_ECC_2AWS.py` Python command, enter SSID and password. This command performs the following tasks:
   4.1. Script commission the ECC608 with SSID and password for the Wi-Fi connection.
   4.2. Script request the ECC608 to generate a signing certificate (with its private key remaining private in the ECC).
   4.3. The script receives the CSR and signs it with the signer-ca private key.
   4.4. Script sends it back to the ECC608 that stores it.
5. Program the firmware of the project `saml21g18b_sensor_board_demo_JITR.atsln` to the board.

Figure 6-1. Console Log



6. LED LD2 blinks in normal speed at the beginning; this indicates that the board is trying to connect to AP. Then, LED LD2 blinks faster in blue color, this indicates that the board is successfully connected to the AP and gets the IP address. Lastly, LED LD2 turns steady blue, which indicates that the board is successfully connected to the AWS cloud.
7. If commissioning other Sensor Board, follow the above steps (2-6). There is no need to create a new CERT.

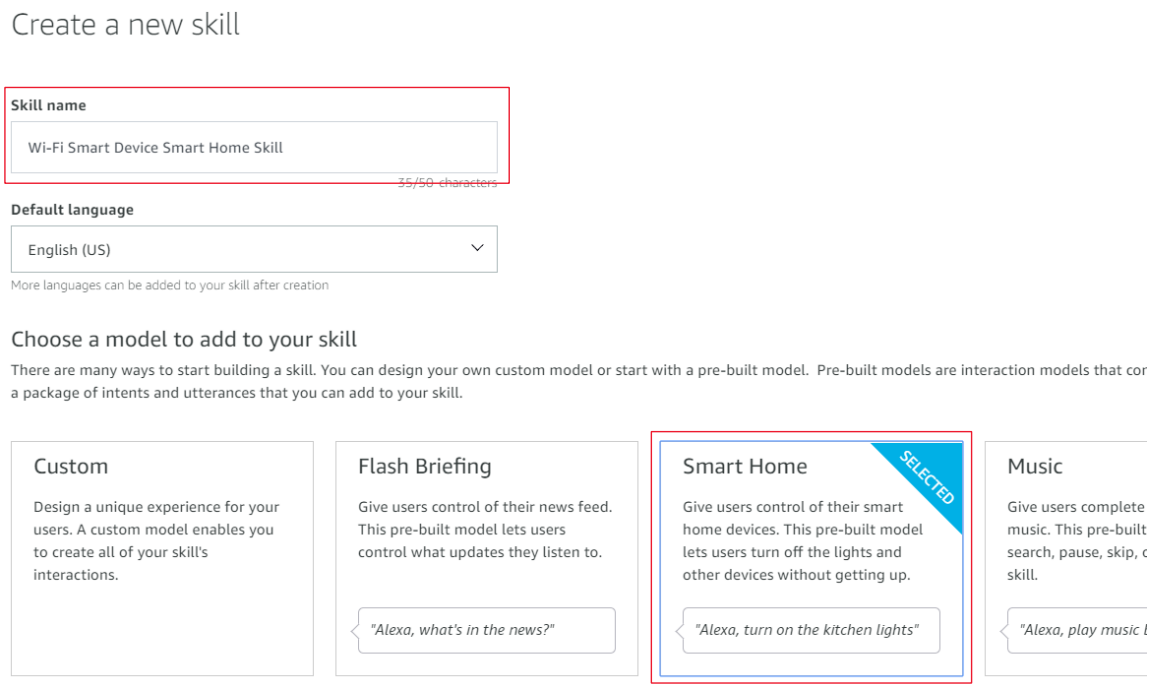
7. Alexa Skill Setup

The user needs to create Alexa skills at https://developer.amazon.com to use this project. This project supports Smart Home Skill (Microchip Wi-Fi Smart Device Smart Home Skill) and Custom Skill (Microchip Wi-Fi Sensor Board Skill). The following sections show the different settings of these skills.

7.1 Microchip Wi-Fi Smart Device Smart Home Skill Setup

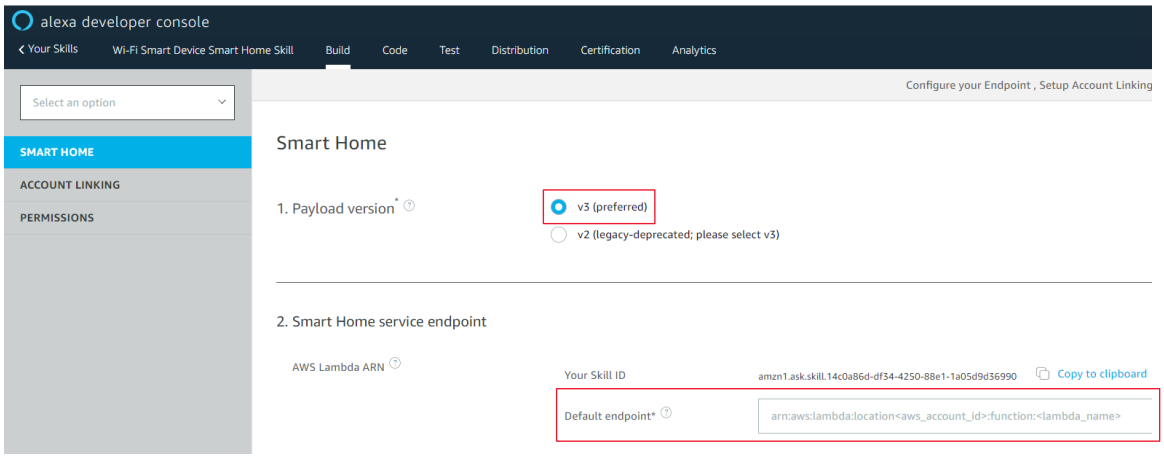
1. To create Alexa Skill, go to Amazon developer website https://developer.amazon.com/home.html.
2. Log in with the Amazon developer credentials. If not registered, register and set the credentials.
3. Go to *Alexa > Alexa Skills Kit > Create Skill*.
4. Add “Skill name” as *Wi-Fi Smart Device Smart Home Skill*, choose model as “Smart Home” and click the **Create skill** button.

Figure 7-1. Creating a New Smart Home Skill in Alexa



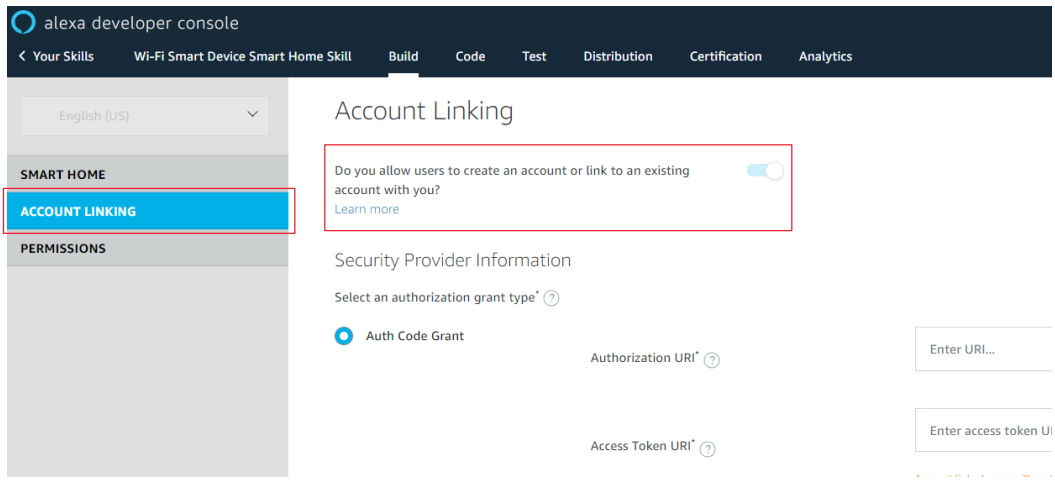
5. In the Smart Home page, choose the following settings:
   - In “Payload version.”, select *v3*.
   - Fill in AWS Lambda ARN to “Default endpoint*”.
     - The Lambda ARN can be found from the AWS Lambda function settings page (refer to 5.6.1 Lambda Setup for Microchip Wi-Fi Smart Device Smart Home Skill).
     - Click the **Save** button in the upper right corner.
     - **Note:** User needs to perform step 8 and step 9 in 5.6.1 Lambda Setup for Microchip Wi-Fi Smart Device Smart Home Skill before clicking the **Save** button, otherwise, it fails to save the endpoints. The Skill ID can be found in the following screen)

Figure 7-2. Payload Selection and Adding a Default Endpoint



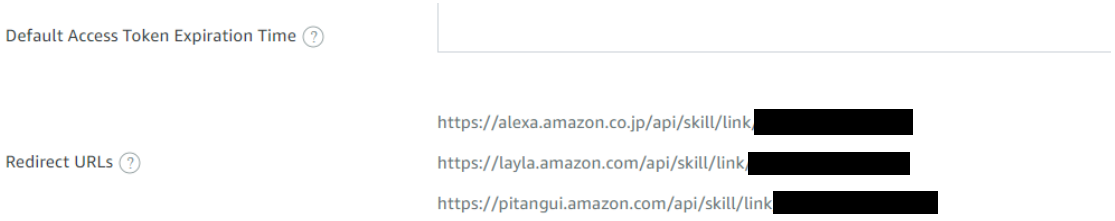
6. Select Account Linking from the left panel. The skill is needed to be linked to the Cognito user pool app client "WiFiSmartDeviceForAlexaSmartHomeSkill" that you set in 5.4 Amazon Cognito Setup.
   – To link the account, use the following settings:
     - Authorization URI: https://<Domain_Name_Of_Cognito_User_Pool>/login
     - Access Token URI: https://<Domain_Name_Of_Cognito_User_Pool>/token
     - Client ID: <User_Pool_App_Client_ID>
     - Client Secret: <User_Pool_App_Client_Secret>
     - Client Authentication Scheme: HTTP Basic
     - Scope: profile

     Figure 7-3. Linking an Account



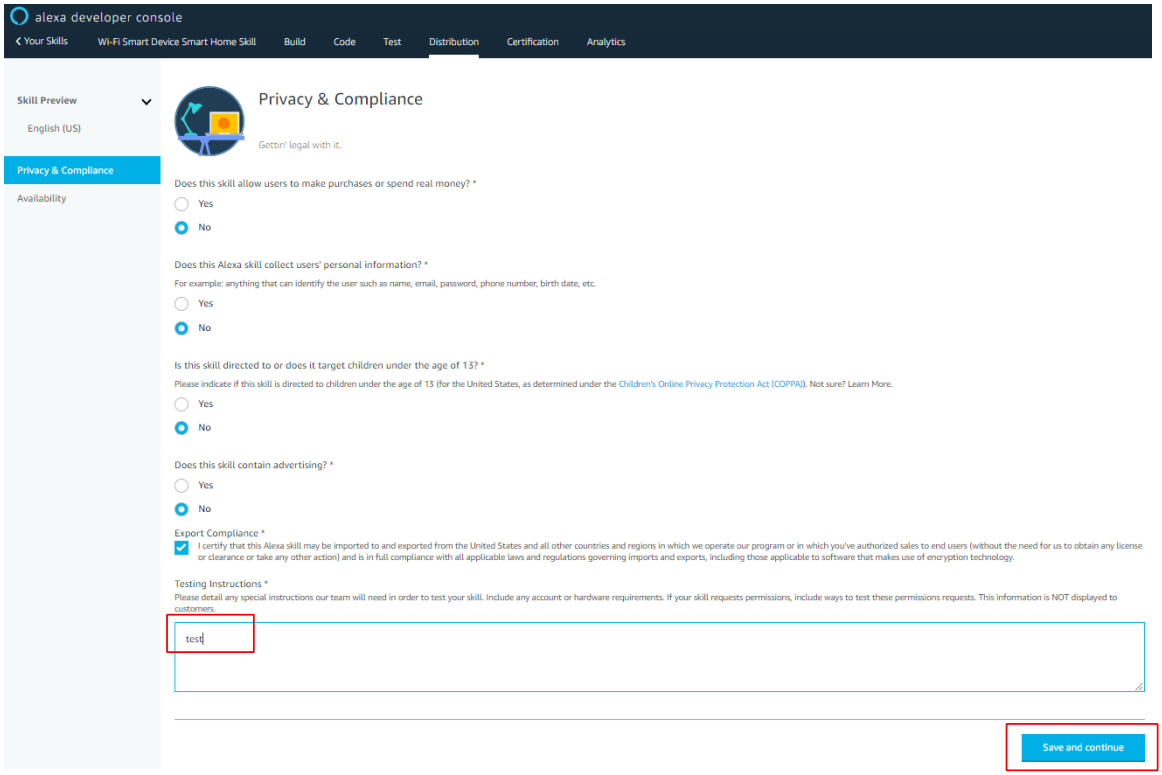
7. Redirect URLs in the Account Linking page are used as Callback URLs. These URLs must be filled in the Callback URLs of the Cognito user pool app client "WiFiSmartDeviceForAlexaSmartHomeSkill". For more details, refer to 5.4 Amazon Cognito Setup.

Figure 7-4. Redirect URLs



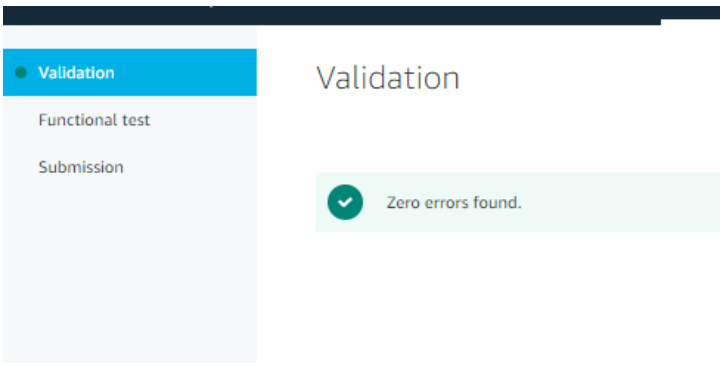
8. Click the **Save** button.
9. Go to **Distribution** tab, enter the following information:
   - “Public Name”: <Your Skill Name>
   - “One Sentence Description”: test
   - “Detailed Description”: test
   - “Example Phrases”: Alexa, set the power to 60% on device
   - Upload Small Skill Icon (108 x 108 pixel)
   - Upload Large Skill Icon (512x512 pixel)
   - “Category”: Smart Home
   - “Keywords”: Sensor
   - “Privacy Policy URL”: http://microchip.com
   - “Terms of Use URL”: http://microchip.com
10. Click the **Save and continue** button.
11. Go to Privacy & Compliance page and fill the following values:
    - Does this skill allow users to make purchases or spend real money? * **- No**
    - Does this Alexa skill collect users' personal information? * - **No**
    - Is this skill directed to or does it target children under the age of 13? * - **No**
    - Does this skill contain advertising? * - **No**
    - Export Compliance * - tick the check box
    - Testing Instructions * - type **Test**
12. Click the **Save and Continue** button.

Figure 7-5. Privacy & Compliance Page



13. In the Availability page, click the **Save and continue** button.
14. With the correct settings and values, Validation page displays “Zero errors found” message.

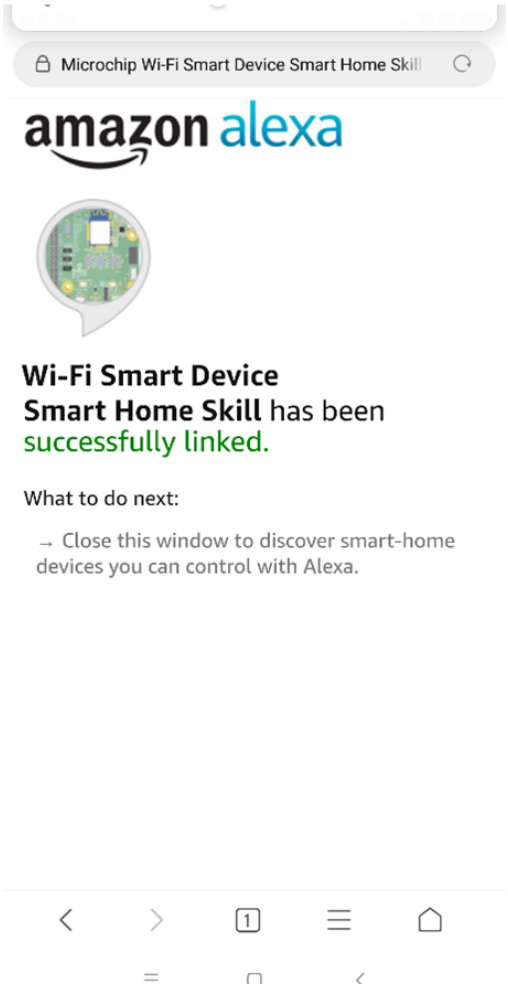
Figure 7-6. Validation Page



15. Login to the Alexa mobile application with the Alexa account.
    – Alexa mobile application can be download from below link:
        • For Android devices: https://play.google.com/store/apps/details?id=com.amazon.dee.app&hl=zh_HK
        • For iOS devices: https://itunes.apple.com/us/app/amazon-alexa/id944011620?mt=8
16. On the Alexa mobile application, select *Skill -> Your Skills*. The Alexa Skill that is created can be found on the application.
17. Enable the skill by logging in with Cognito user account.

18. The user can see following message on the application if the skill is successfully enabled and linked to Amazon Cognito.

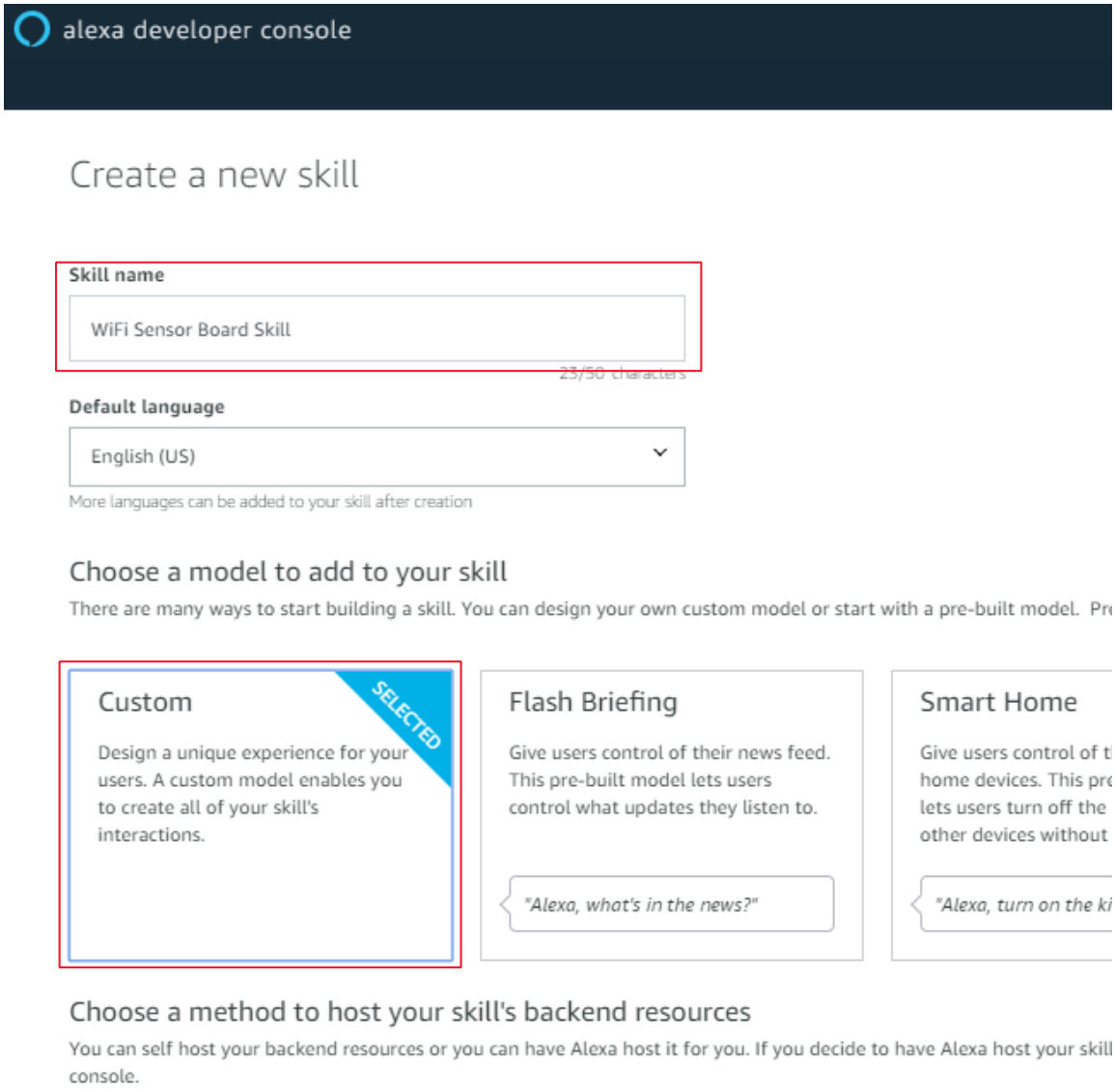
Figure 7-7. Mobile App Window



7.2 Microchip Wi-Fi Sensor Board Skill Setup

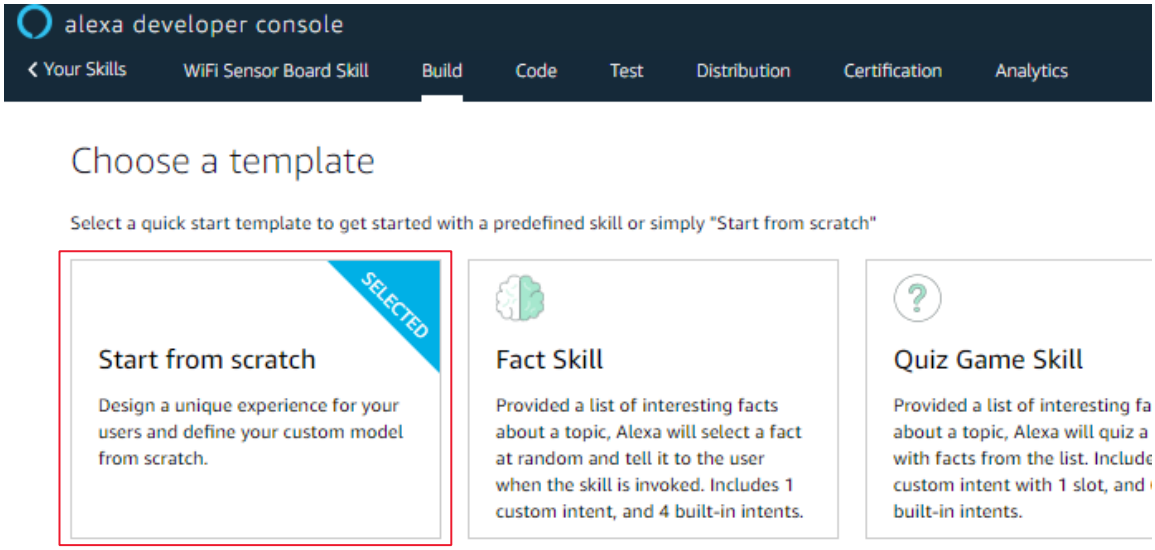
1. To create Alexa Skill, go to Amazon developer website https://developer.amazon.com/home.html.
2. Log in with the Amazon developer credentials. If not registered, register and set the credentials.
3. Select *Alexa > Alexa Skills Kit > Create Skill*.
4. Add “Skill name” as *WiFi Sensor Board Skill*, choose model as “Custom” and click the **Create skill** button.

Figure 7-8. Creating a New Custom Skill in Alexa



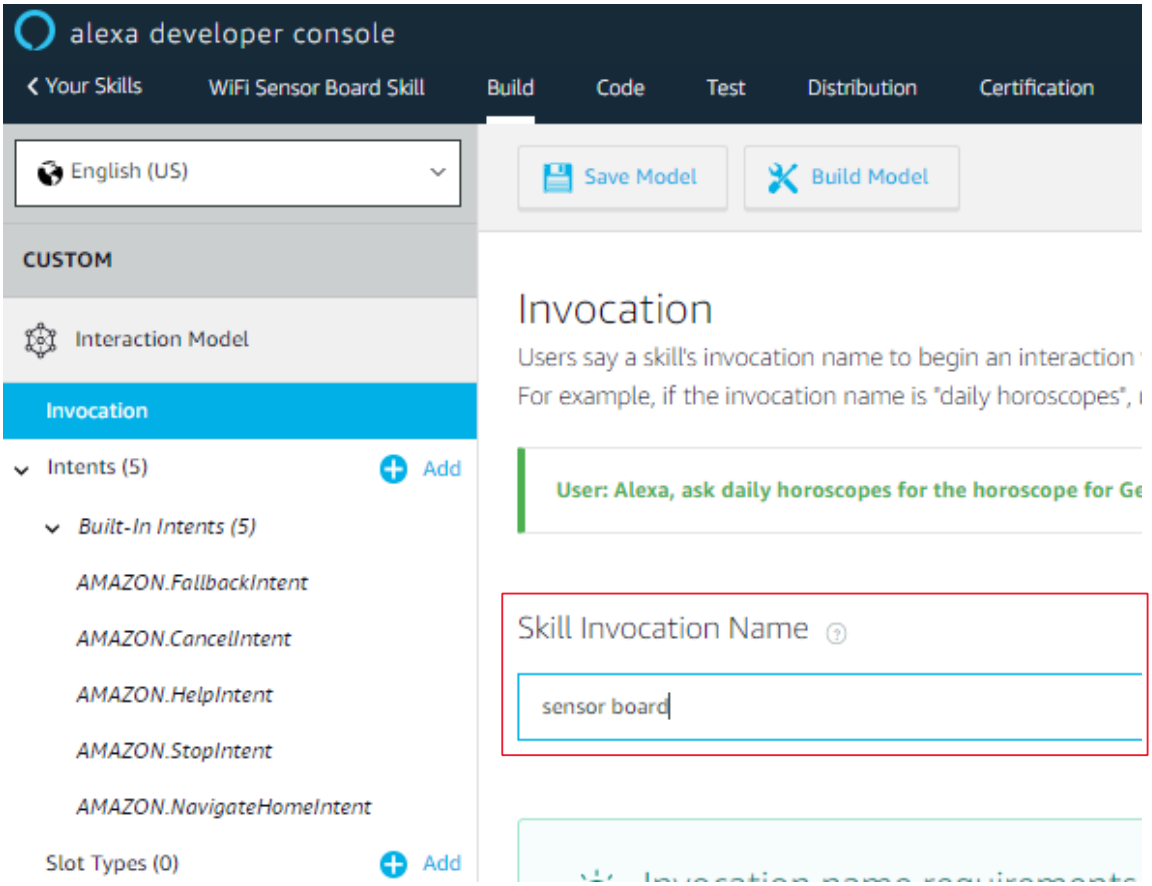
5. In the Custom page, choose the template as “Start from scratch”.

Figure 7-9. Choosing Start from Scratch Template



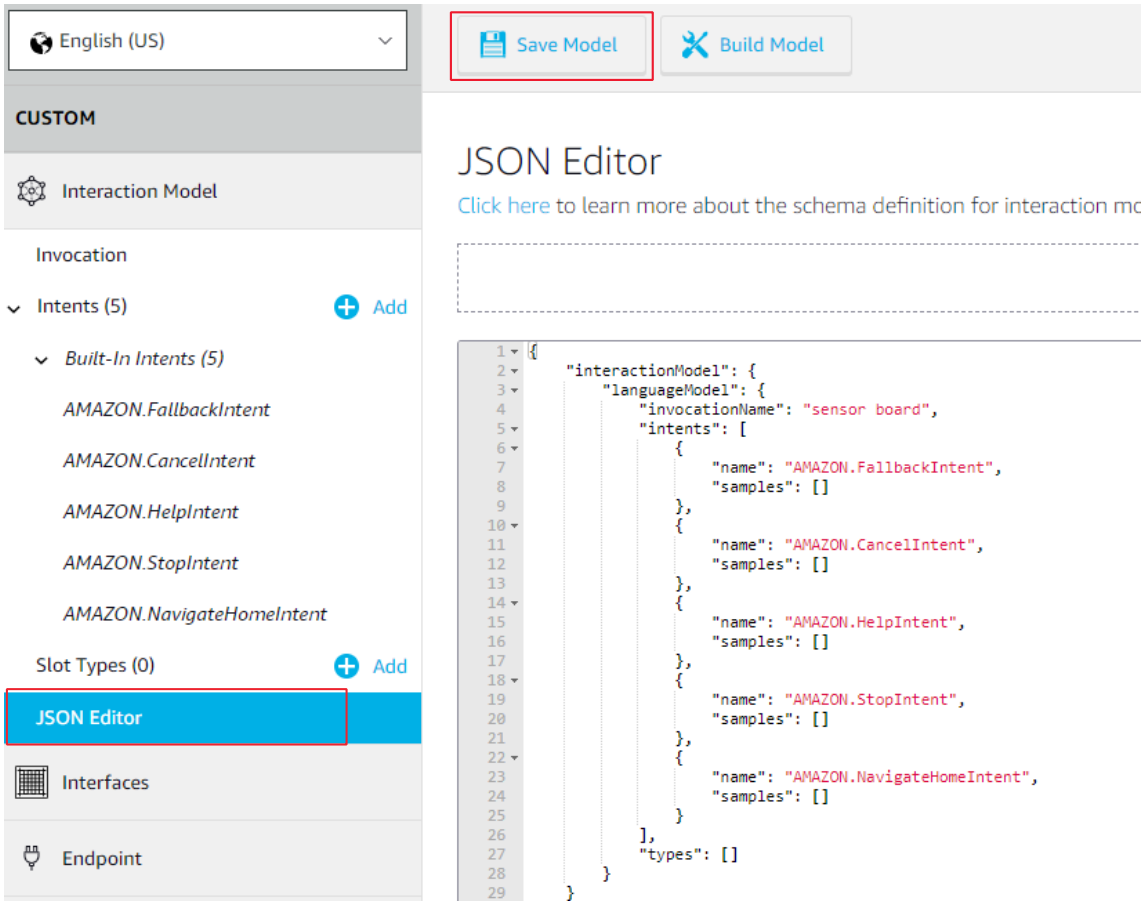
6. Add the “Skill Invocation Name”.

Figure 7-10. Skill Invocation Name



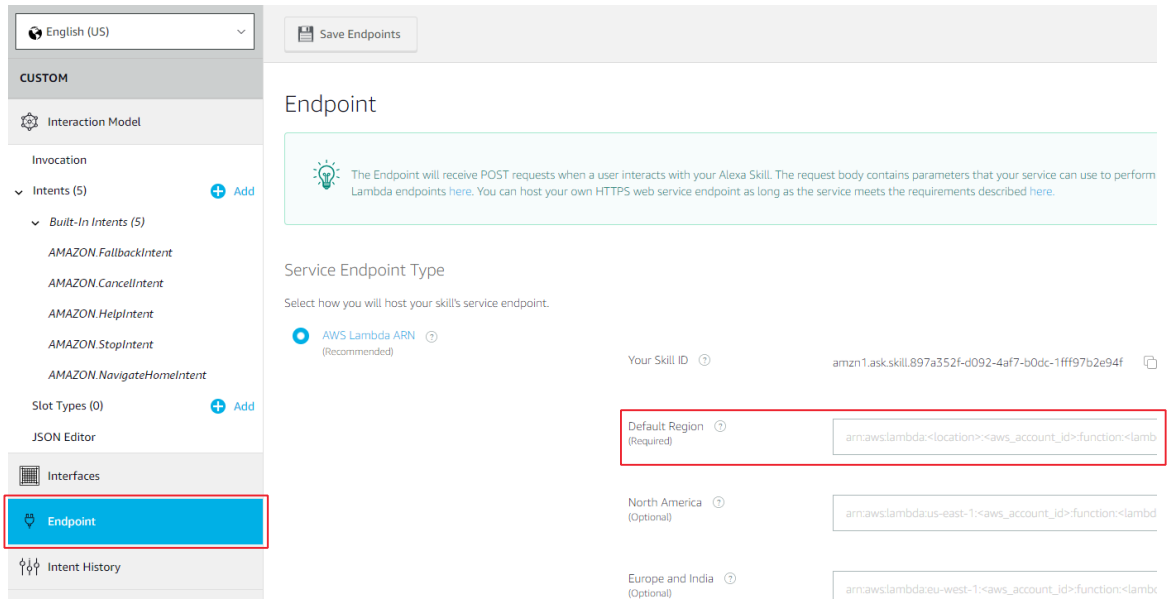
7. Go to JSON Editor page, drag and drop the custom skill JSON file from the `alexa-skill/` folder to JSON Editor.
8. Click **Save Model**.

Figure 7-11. Saving the Model



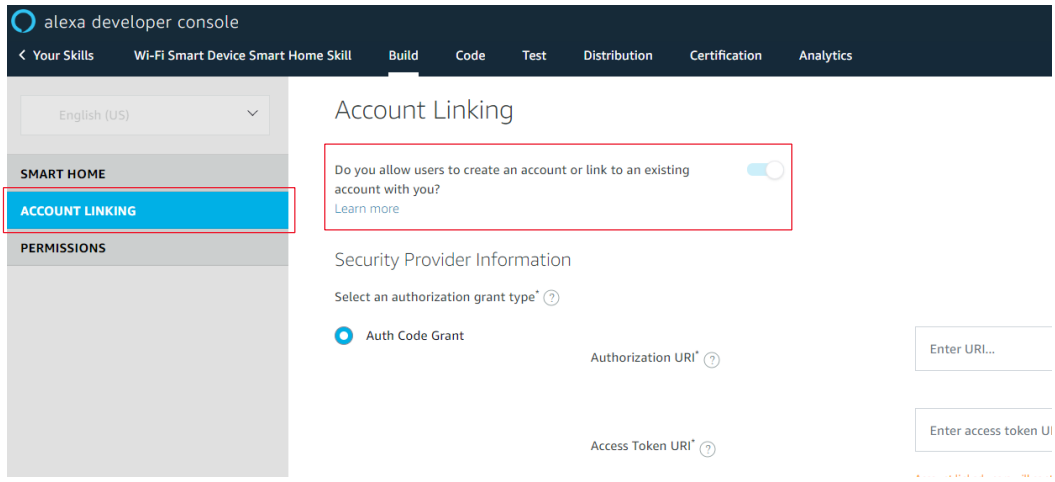
9. Go to Endpoint page, fill the following Default Region:
   – AWS Lambda ARN – the Lambda ARN can be found from the AWS Lambda function setting page in 5.6.1 Lambda Setup for Microchip Wi-Fi Smart Device Smart Home Skill.
10. Default Region
11. Click **Save Endpoints**.

Figure 7-12. Saving Endpoints



12. Go to Account Linking page and enable the Account Linking. The Skill must be linked to the Cognito user pool app client `WiFiSmartDeviceForAlexaSmartHomeSkill`. For more details on the Amazon Cognito user pool app client, refer to 5.4 Amazon Cognito Setup.
    - To link the account, use the following settings:
        - "Authorization URL": https://<Domain_Name_Of_Cognito_User_Pool>/login
        - "Access Token URL": https://<Domain_Name_Of_Cognito_User_Pool>/token
        - "Client ID": <User_Pool_App_Client_ID>
        - "Client Secret": <User_Pool_App_Client_Secret>
        - "Client Authentication Scheme": HTTP Basic
        - "Scope": profile

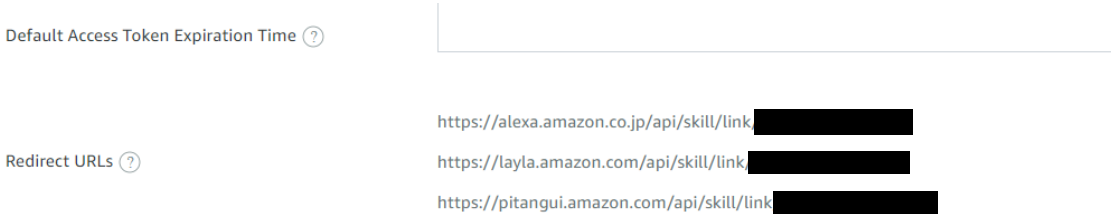
        Figure 7-13. Linking an Account



13. Redirect URLs in the Account Linking page are used as Callback URLs. These URLs must be filled in the Callback URLs of the Cognito user pool app client

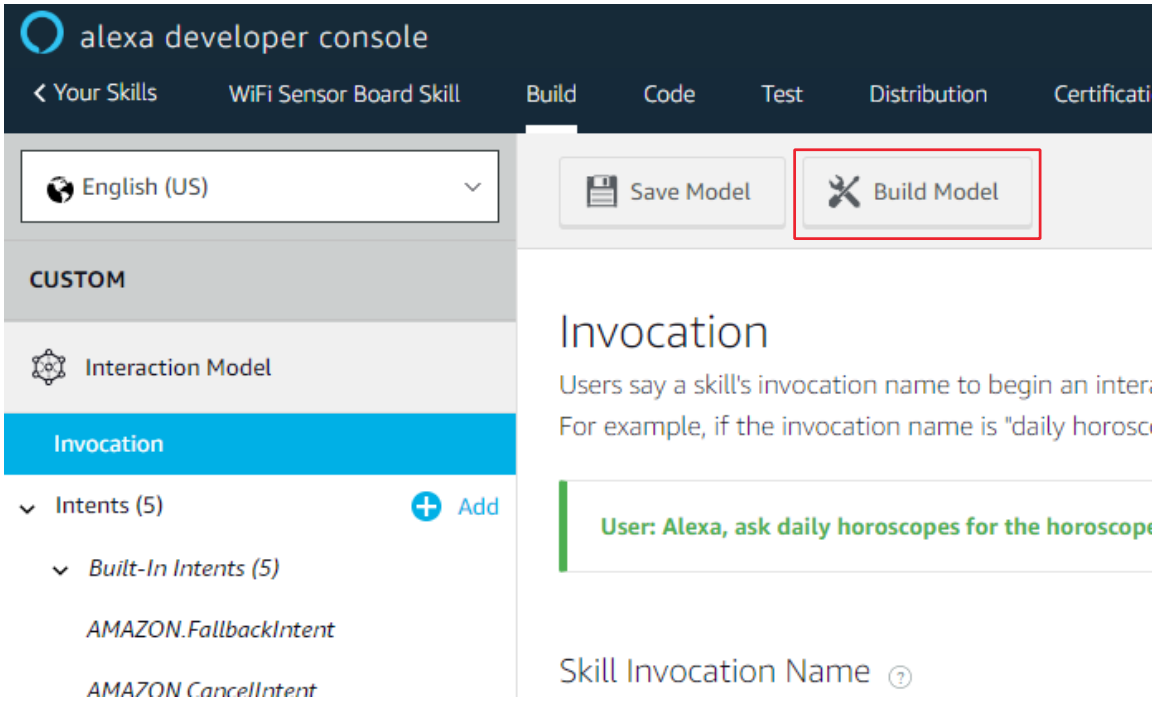
`WiFiSmartDeviceForAlexaSmartHomeSkill`. For more details, refer to 5.4 Amazon Cognito Setup.

Figure 7-14. Redirect URLs



14. Click the **Save** button.
15. Go to **Build** tab, select “Invocation” in the left panel, click **Build Model**.

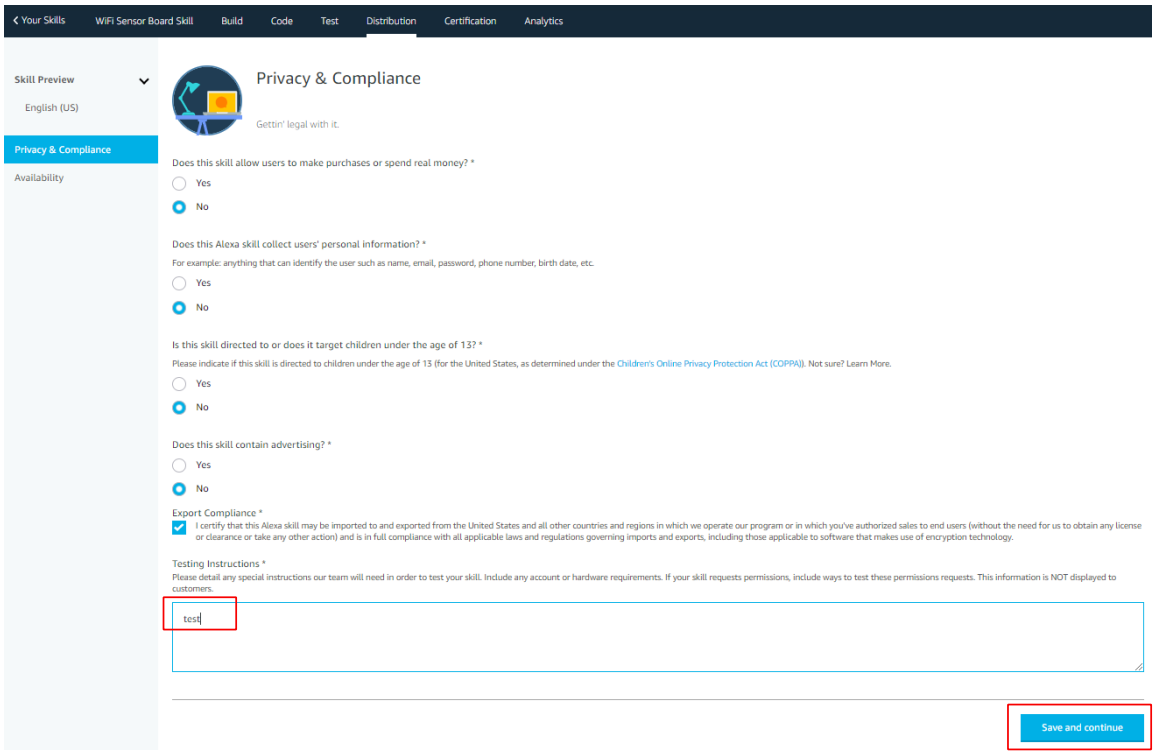
Figure 7-15. Build Model



16. Go to **Distribution** tab, enter the following information:
17. Click the **Save and continue** button.
    - “Public Name”: <Your Skill Name>
    - “One Sentence Description”: test
    - “Detailed Description”: test
    - “Example Phrases”: Alexa, set the power to 60% on device
    - Upload Small Skill Icon (108 x 108 pixel)
    - Upload Large Skill Icon (512x512 pixel)
    - “Category”: Education & Reference
    - “Keywords”: Sensor
    - “Privacy Policy URL”:http://microchip.com
    - “Terms of Use URL”:http://microchip.com

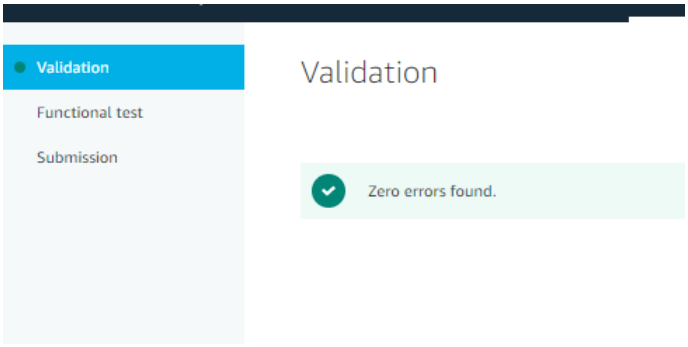
18. Go to Privacy & Compliance page and fill the following values:
    - Does this skill allow users to make purchases or spend real money? * **- No**
    - Does this Alexa skill collect users' personal information? * - **No**
    - Is this skill directed to or does it target children under the age of 13? * - **No**
    - Does this skill contain advertising? * - **No**
    - Export Compliance * - tick the check box
    - Testing Instructions * - type **Test**
19. Click the **Save and continue** button.

Figure 7-16. Privacy & Compliance Page



20. In the Availability page, click the **Save and continue** button.
21. With the correct settings and values, Validation page displays “Zero errors found” message.

Figure 7-17. Validation Page

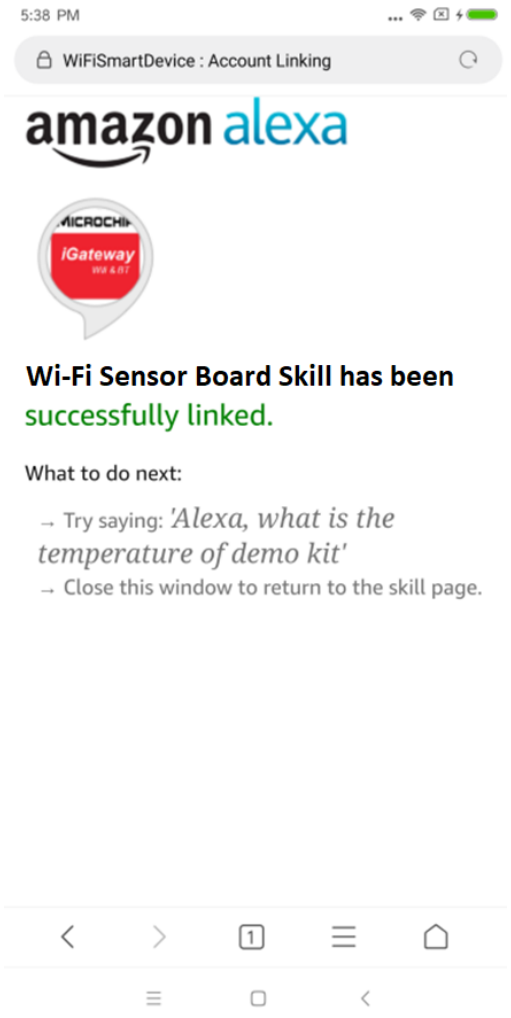


22. Log in to your Alexa mobile application with your Alexa account.
    - Alexa mobile application can be downloaded from the link below:

- For Android devices: https://play.google.com/store/apps/details?id=com.amazon.dee.app&hl=zh_HK.
- For iOS devices: https://itunes.apple.com/us/app/amazon-alexa/id944011620?mt=8

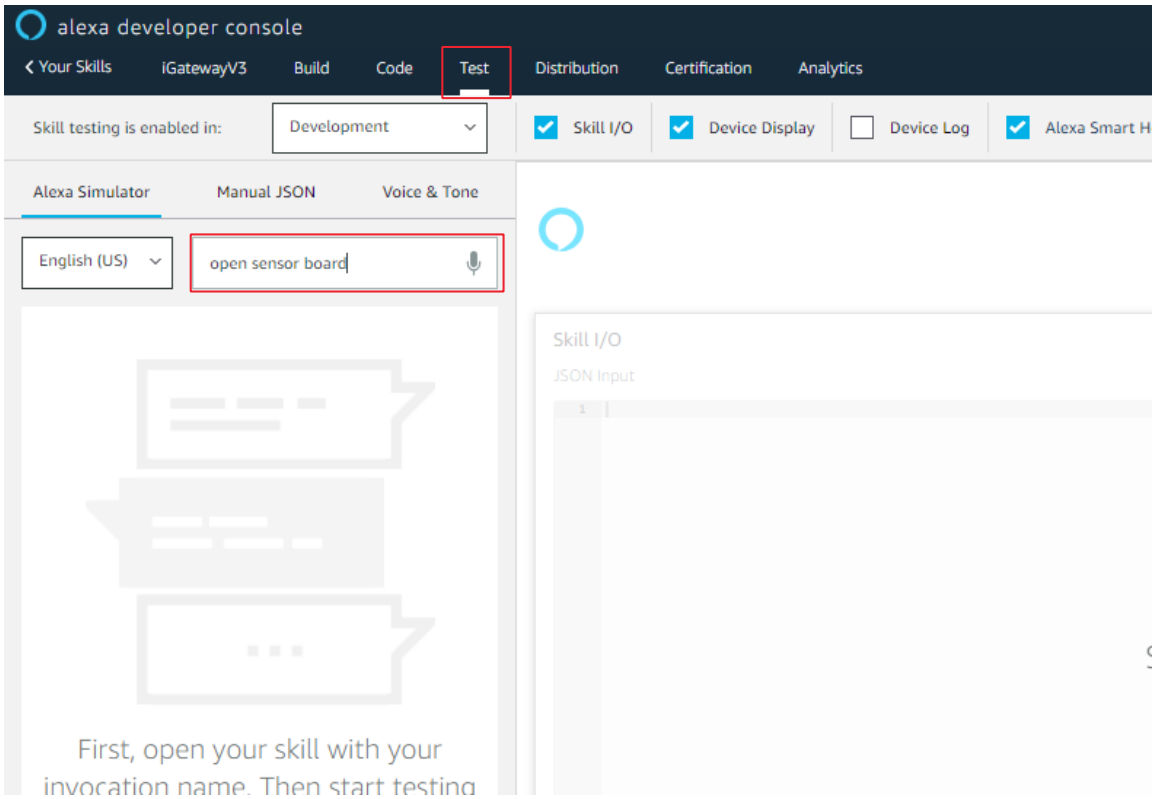
23. On the Alexa mobile phone application, select *Skill -> Your Skills*.
24. The Alexa Skill that is created can be found on the application.
25. Enable the Skill by logging in with Cognito user account.
26. The user can see following message on the application if the skill is successfully enabled and linked to Amazon Cognito.

Figure 7-18. Mobile App Window



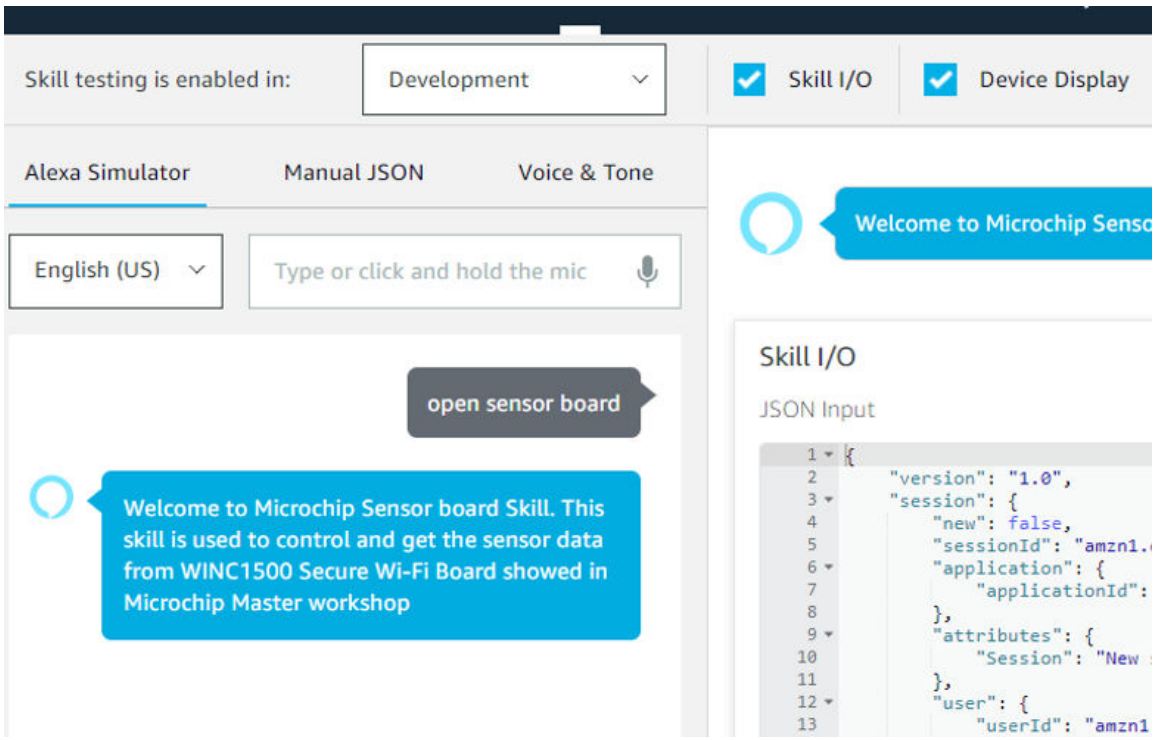
27. Select “Test” in the upper panel and type “open sensor board” to test the skill.

Figure 7-19. Testing the Skill



28. The welcome response sentence, “Welcome to Microchip Sensor board Skill…” is set in the Lambda function. It indicates that the Alexa skill can successfully connect to the Lambda function.

Figure 7-20. Simulator Response



8. Appendix A: Software Installation

The developer needs to install these software/tools for Wi-Fi Smart Device Enablement Kit firmware development or cloud configuration:

8.1 Atmel Studio 7

Install Atmel Studio 7 to compile the firmware project files in the `mcu-firmware/` folder . Download the installation file from: http://www.microchip.com/mplab/avr-support/atmel-studio-7

8.2 SAM Boot Assistance (SAM-BA)

Install SAM-BA V2.18 to program the firmware with boot-loader over the USB port. Download the installation file from: https://www.microchip.com/developmenttools/ProductDetails/atmel%20sam-ba%20in-system%20programmer

After installation, the developer needs to configure the GUI with the following steps to make it work with the ATWINC15x0 Secure Wi-Fi Board:

1. Copy the folder `saml21_wsenbrd\` to `C:\Program Files (x86)\Atmel\sam-ba_2.18\tcl_lib` from the release package.
2. Rename `boards.tcl` to `boards_old.tcl` in `C:\Program Files (x86)\Atmel\sam-ba_2.18\tcl_lib`.
3. Copy `boards.tcl` in sam-ba folder to `C:\Program Files (x86)\Atmel\sam-ba_2.18\tcl_lib`.

8.3 Python 3.6.x

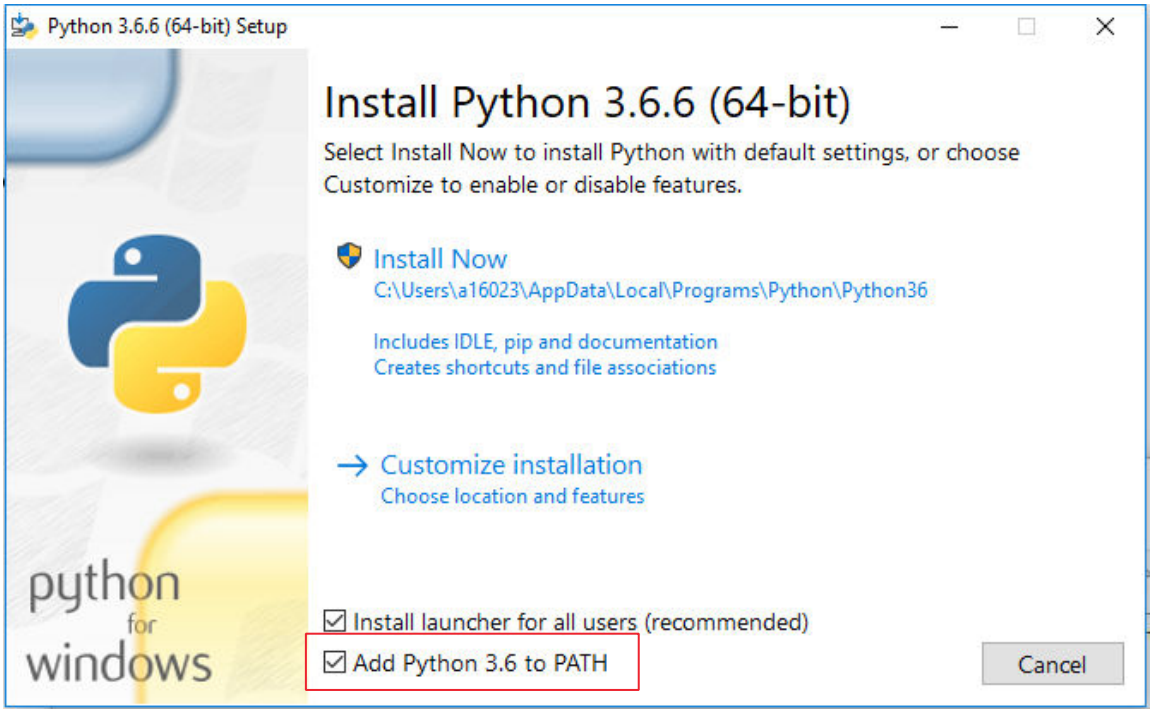
The user needs to use Python scripts to provision the Secure Wi-Fi Board to the network and the board to the user account. The user can view the Python scripts to see the detailed steps involved. Download the installation file from:https://www.python.org/downloads/release/python-366/

Note: Python 3.7.x and Python 2.x are not supported.

Perform the following steps for installation:

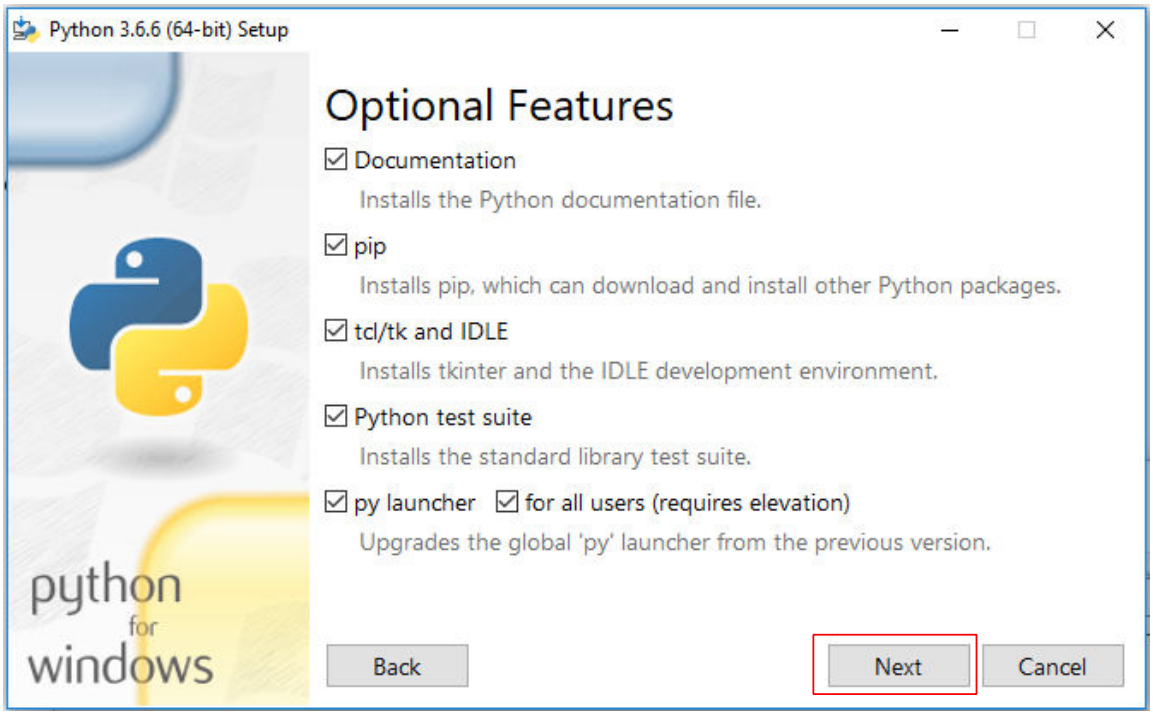
1. When installing Python 3.6.x, select 'Add Python 3.6 to PATH'.

Figure 8-1. Adding the Path



2. Choose 'Customize Installation' and make sure everything is selected.
3. Click **Next**.

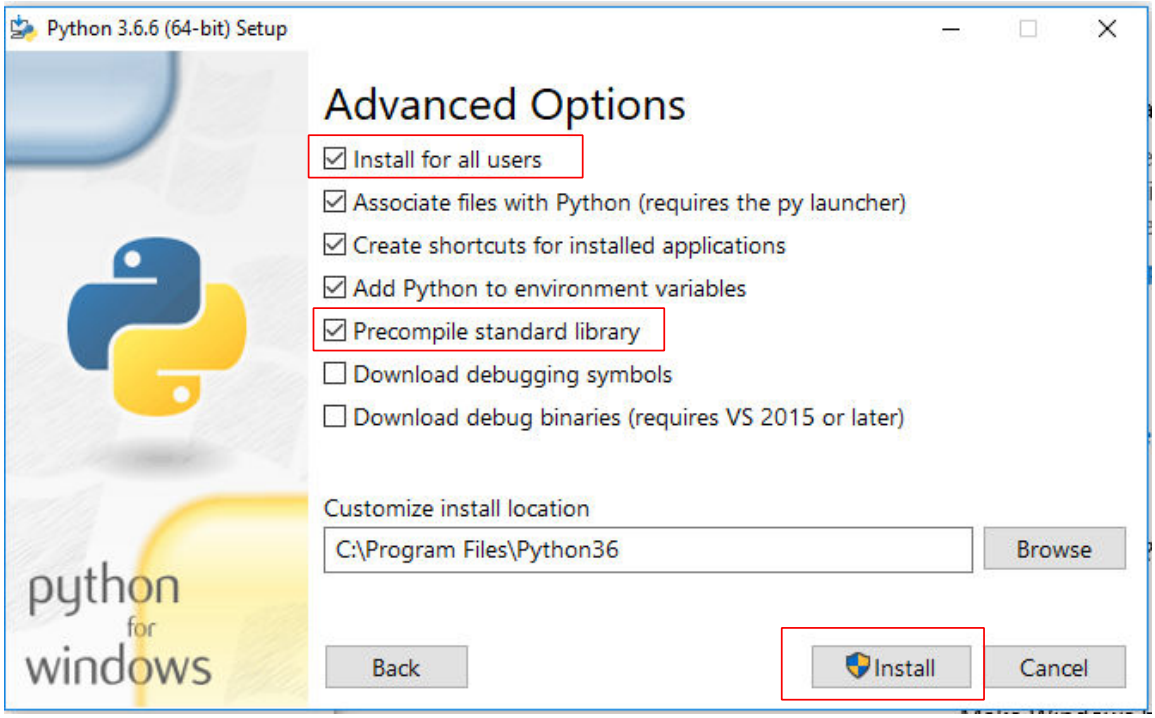
Figure 8-2. Customizing Installation



4. Select “Install for all users” and “Precompile standard library”.

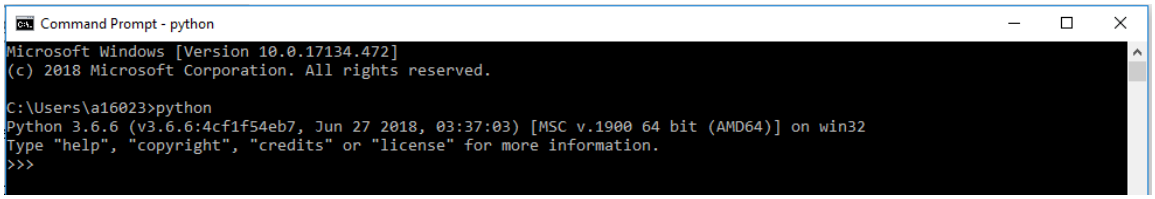
5. Click **Install**.

Figure 8-3. Installation Setup



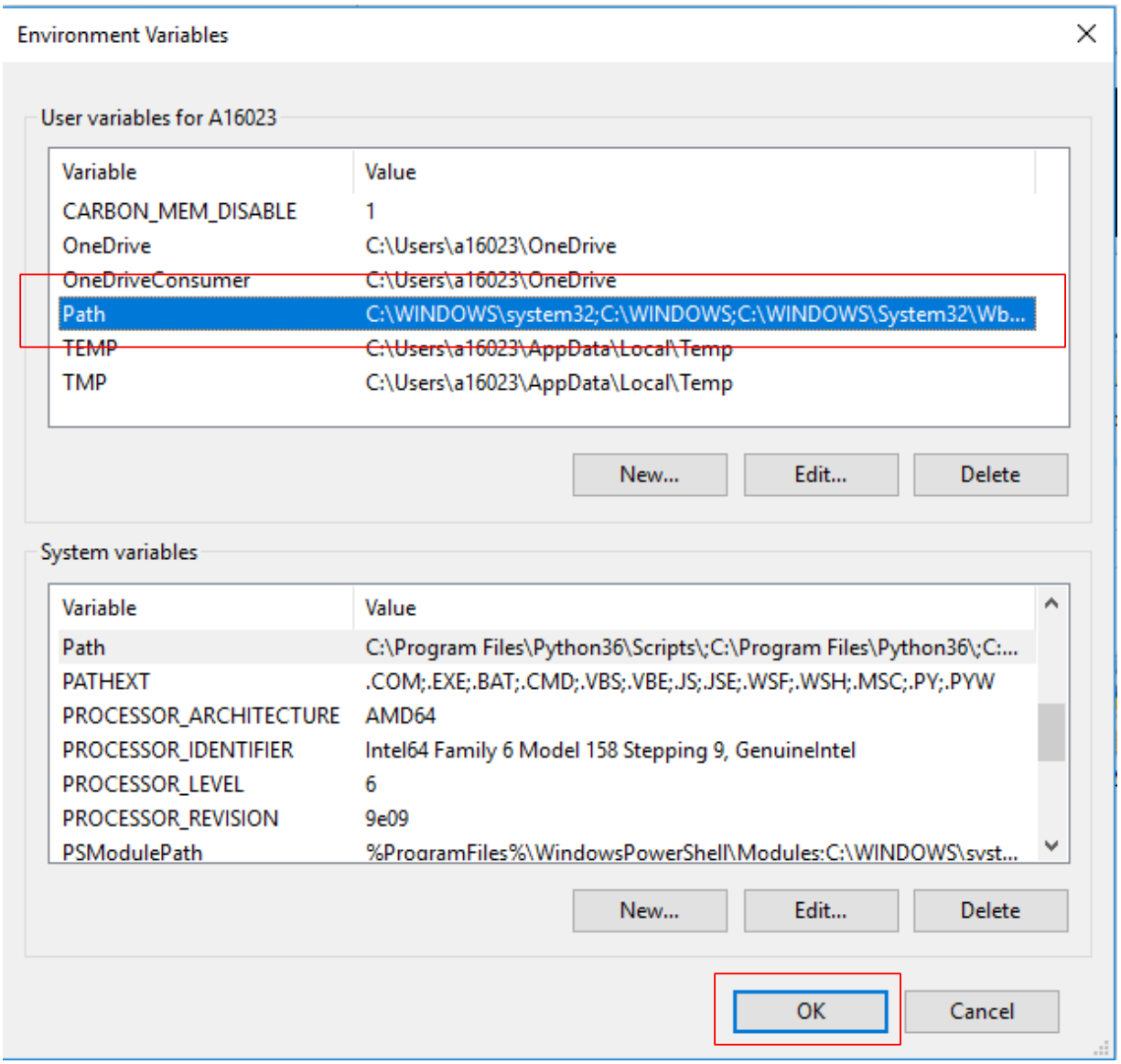
6. After the successful installation, the user needs to ensure that the PC is using the correct Python version. Check it by typing “python” on Windows Command Prompt.

Figure 8-4. Python Version on the Command Prompt



7. For Windows PC, if the Python version is not correct, the user can check if only Python 3.6 is added in the Environment Variable Path. Remove other Python tools which are added to the Environment Variable Path.
8. To check the Environment Variable, press the <Windows key>.
9. Go to *System* > *Advanced system settings*.
10. In the **Advanced** tab, click **Environment Variables...** and select the Path under “User variables” and “System variables.”
11. Click **OK**.

Figure 8-5. Setting Up the Path Variable



8.4 Python Package Manager

Perform the following steps to use Python Package Manager (PyPM) to install the required packages:

1. Locate `requirements.txt` in the /ProvisionScripts directory.
2. Open the Start menu and search for 'cmd.'
3. Right-click on 'Command Prompt' and select 'Run as Administrator.'
4. In command prompt, navigate to the directory and run the following command: `pip install -r requirements.txt`.

9. Appendix B: Mobile Application Configuration

9.1 Android Application

The mobile application source files are available in the `/mobile-app/android` directory of the release package.

The developer needs to configure the mobile application to connect to the AWS account. For this purpose, change the default value in `/src/com/amazonaws/mchp/awsprovisionkit/utils/ConfigFileConstant.java` directly in the mobile application source files and compile to build the apk file.

The file contains the fields below. The developer can get all these settings in the AWS account after the AWS cloud setup.

```
    //===============================================================================
    // User need to set below parameters to your AWS account credentials
    //===============================================================================

    public static final String CUSTOMER_SPECIFIC_ENDPOINT_DEFAULT_VAL = "xxxxxx.us-
east-1.amazonaws.com";
    public static final String COGNITO_POOL_ID_DEFAULT_VAL = "us-east-1:xxx-xxxxx-xxxx-xx-
xxxxxxxxxx";
public static final String AWS_IOT_POLICY_NAME_DEFAULT_VAL = "xxx"
// e.g. "AWS_IOT_POLICY_NAME";
    public static final String AWS_IOT_REGION_DEFAULT_VAL ="xxx"     // e.g.  "us-east-1";
    public static final String COGNITO_USER_POOL_ID_DEFAULT_VAL ="xxx"     //  "us-
east-1_xxxxxxx";
    public static final String COGNITO_REGION_DEFAULT_VAL = "xxx"     // e.g. "us-east-1";
    public static final String COGNITO_CLIENT_ID_DEFAULT_VAL = "xxxxxxxxxxxxxxxxxxxxxxxx";
    public static final String COGNITO_CLIENT_SECRET_DEFAULT_VAL = "xxxxxxxxxxxxxxxx";
    public static final String DB_TABLE_NAME_DEFAULT_VAL = "SensorBoardAcctTable";
```

Instead of modifying the `ConfigFileConstant.java` file and building the apk, the developer can modify the `config.txt` file in the smartphone after installing the mobile application. The file is located in `/SDCARD/Android/data/com.amazonaws.mchp.awsprovisionkit/files/MyFileStorage/` and contains the fields below.

```
CUSTOMER_SPECIFIC_ENDPOINT=xxx     // e.g. xxxx.us-east-1.amazonaws.com
COGNITO_POOL_ID=xxx        // e.g. us-east-1:xxxxxx-xxxxxxxxx-xxxx-xxxxxxxxx
AWS_IOT_POLICY_NAME=xxx     // e.g. "AWS_IOT_POLICY_NAME";
AWS_IOT_REGION=xxxx        // e.g. us-east-1
COGNITO_USER_POOL_ID=xxx     // e.g. us-east-1_xxxxxxxxxxxxx
COGNITO_REGION= xxx        // e.g. us-east-1
COGNITO_CLIENT_ID=xxxxxxxxxxxxxxxx
COGNITO_CLIENT_SECRET=xxxxxxxxxxxxxxxxx
```

- CUSTOMER_SPECIFIC_ENDPOINT
  This is the AWS IoT EndPoint, which can be found in the setting page of AWS IoT console.
- COGNITO_POOL_ID
  This is the AWS Cognito Identity pool ID after creating the Cognito Identity pool.
- AWS_IOT_POLICY
  This is the AWS IoT policy name used for the mobile application to access AWS IoT resources.
- AWS_IOT_REGION
  This is the AWS IoT Region.
- COGNITO_USER_POOL_ID

This is the Cognito user pool ID after creating the Cognito user pool.

- COGNITO_REGION
  This is the Cognito region.
- COGNITO_CLIENT_ID
  This is the App client ID configured in the Cognito user pool.
- COGNITO_CLIENT_SECRET
  This is the ”WiFISmartDeviceForAlexaSmartHomeSkill” or “WiFISmartDeviceForAlexaCustomSkill” App client secret configured in the Cognito user pool.

9.2 iOS Application

The mobile application source files are available in the `/mobile-app/ios/` directory of the release package.

The developer needs to configure the mobile application to connect to the AWS account. For this purpose, the developer can change the default value in `mobile-app/ios/CognitoYourUserPoolsSample/Constants.swift` directly in the mobile application source files and compile the source code.

The file contains the fields below. The developer can get all these settings in the AWS account after the AWS cloud setup.

```
    //================================================================================
    // User need to set below parameters to your AWS account credentials
    //================================================================================

let CognitoIdentityUserPoolId = "xxxxx" // e.g. "us-east-1_xxxxxx"
let CognitoIdentityUserPoolAppClientId = "xxxxx"
let CognitoIdentityUserPoolAppClientSecret = "xxxxx"
let CognitoRegion = "xxxxx" // e.g. "us-east-1"
let PoolId = "xxxxx" // e.g. "us-east-1:c4f36b6a-84c7-4acd-b143-0665f9fbd9f9"


let AWSRegion = AWSRegionType.USEast1 // e.g. AWSRegionType.USEast1
let IOT_ENDPOINT = "xxxxx" // e.g. "https://xxxxxxx.iot.us-east-1.amazonaws.com"
let PolicyName = "xxxxx" // e.g. "secure-wifi-board-mobile-app-policy"
```

- IOT_ENDPOINT
  This is the AWS IoT EndPoint, which can be found in the setting page of AWS IoT console.
- PoolId
  This is the AWS Cognito Identity pool id, which can be retrieved when you create the Cognito Identity pool.
- PolicyName
  This is the AWS IoT policy name used for the mobile app to access AWS IoT resources.
- AWSRegion
  This is the AWS IoT Region.
- CognitoIdentityUserPoolId
  This is the Cognito user pool id after creating the Cognito user pool.
- CognitoRegion
  This is the Cognito region.
- CognitoIdentityUserPoolAppClientId

This is the “WiFISmartDeviceForAlexaSmartHomeSkill” or “WiFISmartDeviceForAlexaCustomSkill” App client ID configured in the Cognito user pool.

- CognitoIdentityUserPoolAppClientSecret
  This is the ”WiFISmartDeviceForAlexaSmartHomeSkill” or “WiFISmartDeviceForAlexaCustomSkill” App client secret configured in the Cognito user pool.

10. Appendix C: AWS CloudFormation to Setup Cloud

AWS CloudFormation gives developers an easy way to create and manage a collection of related AWS resources by a template code in either YAML or JSON format. Developers can use AWS CloudFormation via browser console or AWS Command Line Interface (CLI). This service allows easy cloud configuration by uploading the template code to cloud rather than creating and configuring the cloud step by step.

In this project, developers can use AWS CloudFormation to create and configure all the AWS resources that are needed for the Wi-Fi Smart Device Enablement Kit. The AWS resources created in this chapter are the same as the resources created in 5. AWS Setup.

All the template codes used for AWS CloudFormation can be found in the `cloud-formation-templates\` directory and the template codes are coded in YAML format.

After finishing all the steps in this chapter, developers can successfully set up the AWS Cloud and refer to 6. AWS Provision Setup to provision the Wi-Fi Smart Device Enablement Kit to the AWS Cloud account setup in this chapter. Refer to 7. Alexa Skill Setup to finish the remaining steps to create the Alexa Skills.

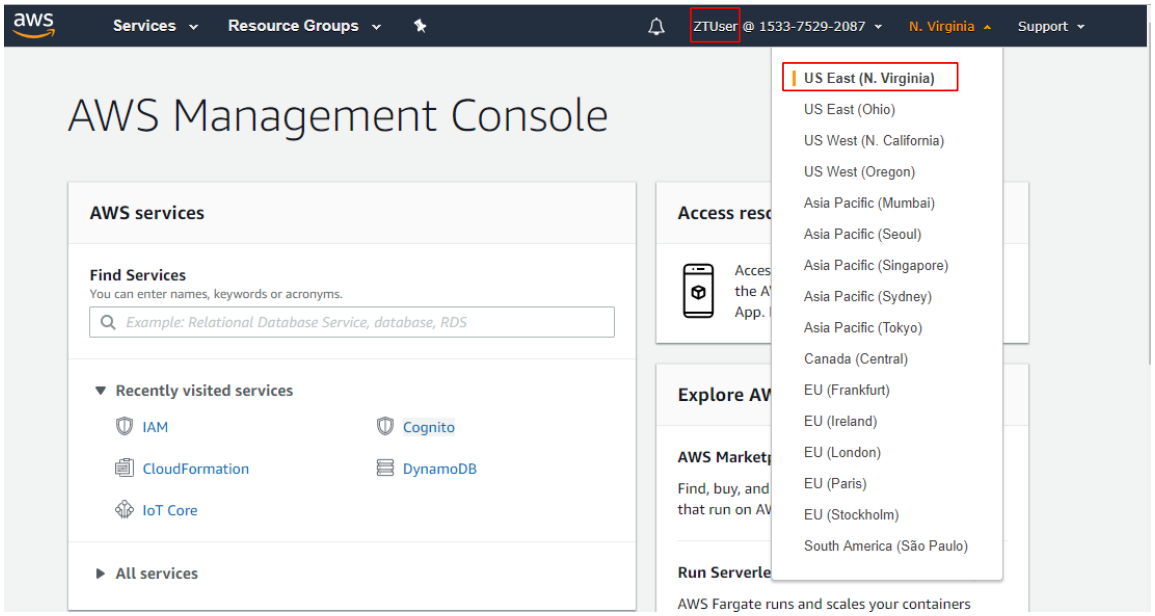
Note:

1. To get the Smart Home Skill ID, perform the steps mentioned in 7.1 Microchip Wi-Fi Smart Device Smart Home Skill Setup and refer to the Figure 7-2.
2. To get the Custom Skill ID, perform the steps mentioned in 7.2 Microchip Wi-Fi Sensor Board Skill Setup and refer to the Figure 7-2.

10.1 Create IAM User for the Project

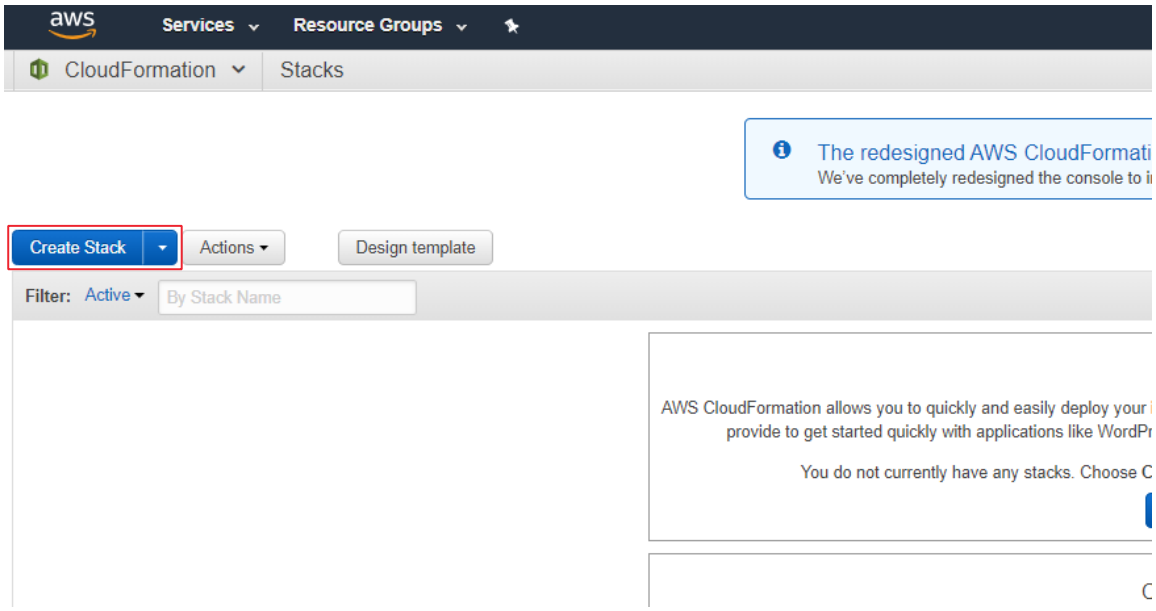
1. After creating the AWS account in 5. AWS Setup, go to https://aws.amazon.com/ and log in to the AWS account with root account credentials.
2. Once logged in, change your region to the one closest to you by selecting the region menu (for example, US East Virginia). Alexa skills support only the following regions:
   - Asia Pacific (Tokyo)
   - EU (Ireland)
   - US East (N. Virginia)
   - US West (Oregon)

Figure 10-1. Selecting the Region



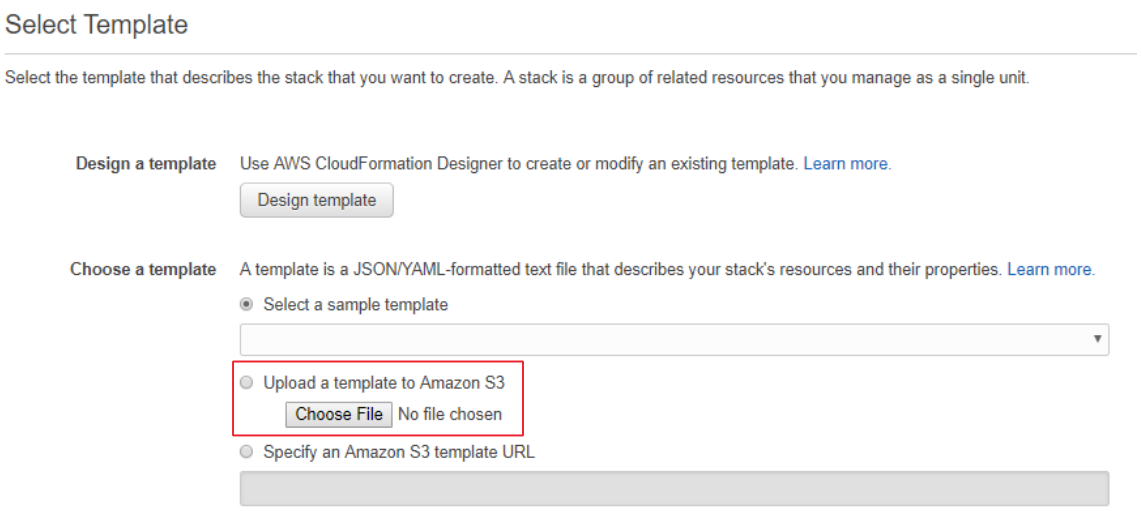
3. Go to https://console.aws.amazon.com/cloudformation and click the **Create Stack** drop-down.

Figure 10-2. Creating a Stack



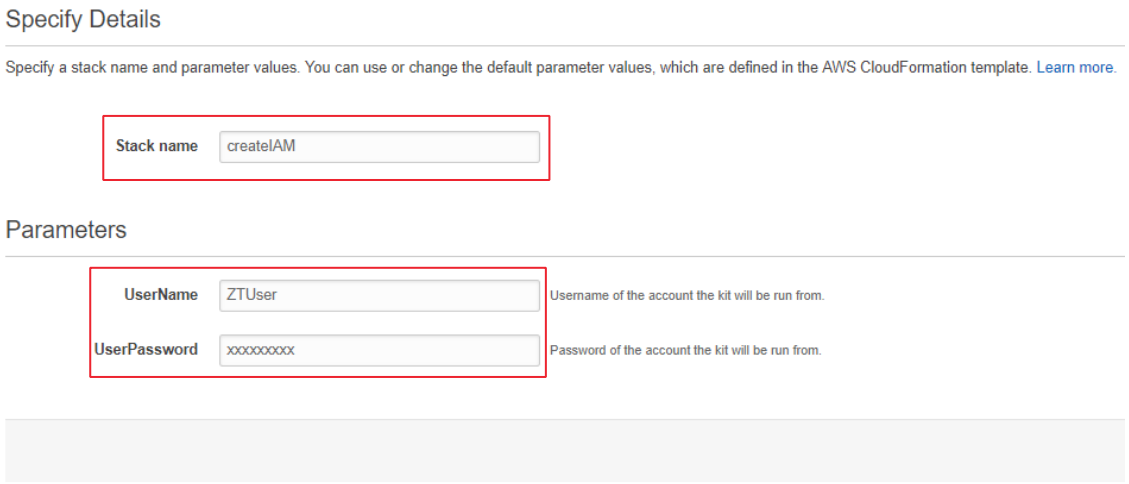
4. Select “Upload a template to Amazon S3” and choose the `\cloud-formation-templates \create_IAM.template.yaml`.

Figure 10-3. Uploading a Template



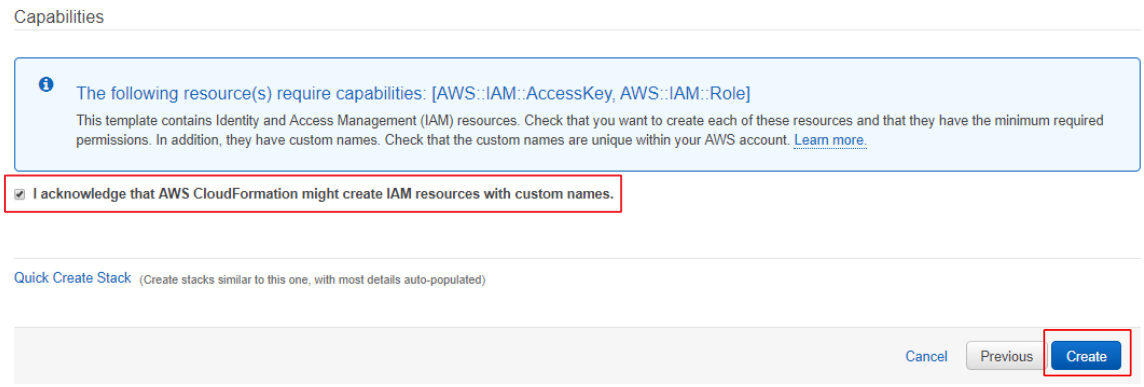
5. Click the **Next** button.
6. Enter “Stack name” as *createIAM*.
7. Enter “UserPassword” as *ZTUser*.
8. Click the **Next** button to continue.

Figure 10-4. Stack Name and Parameters



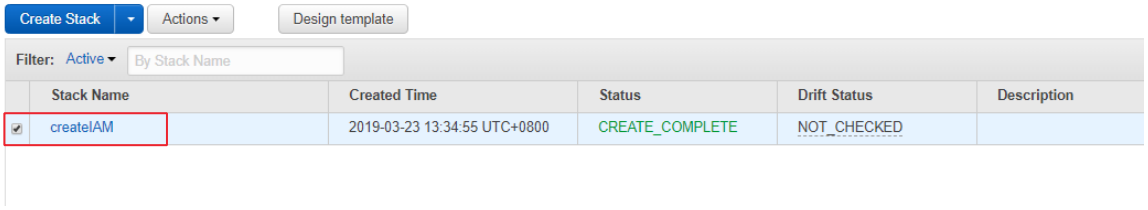
9. Tick the highlighted check box and click the **Create** button.

Figure 10-5. Acknowledgment



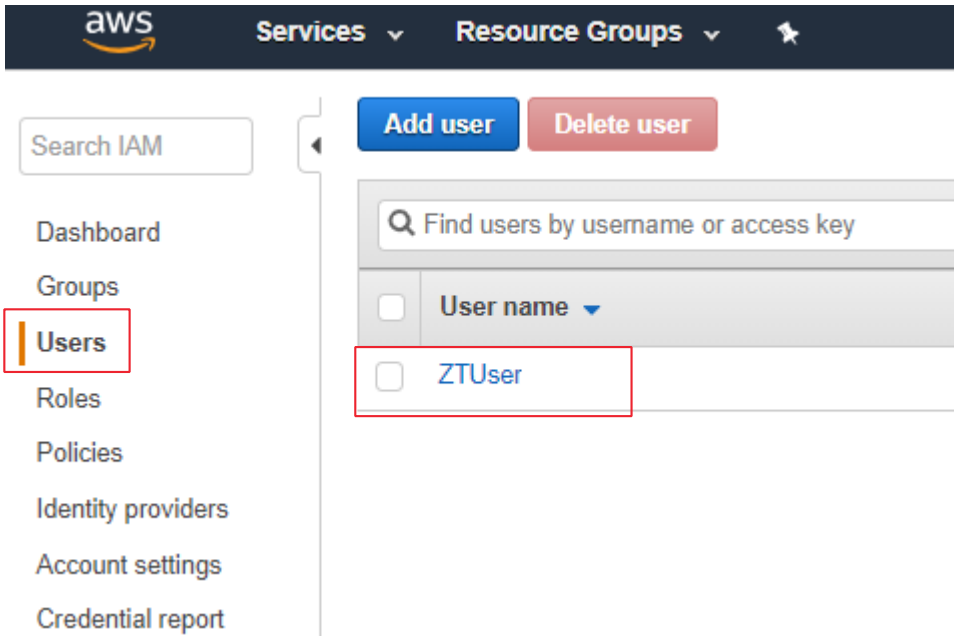
10. After sometime, the status of “createIAM” stack changes to “CREATE_COMPLETE”.

Figure 10-6. Stack Status



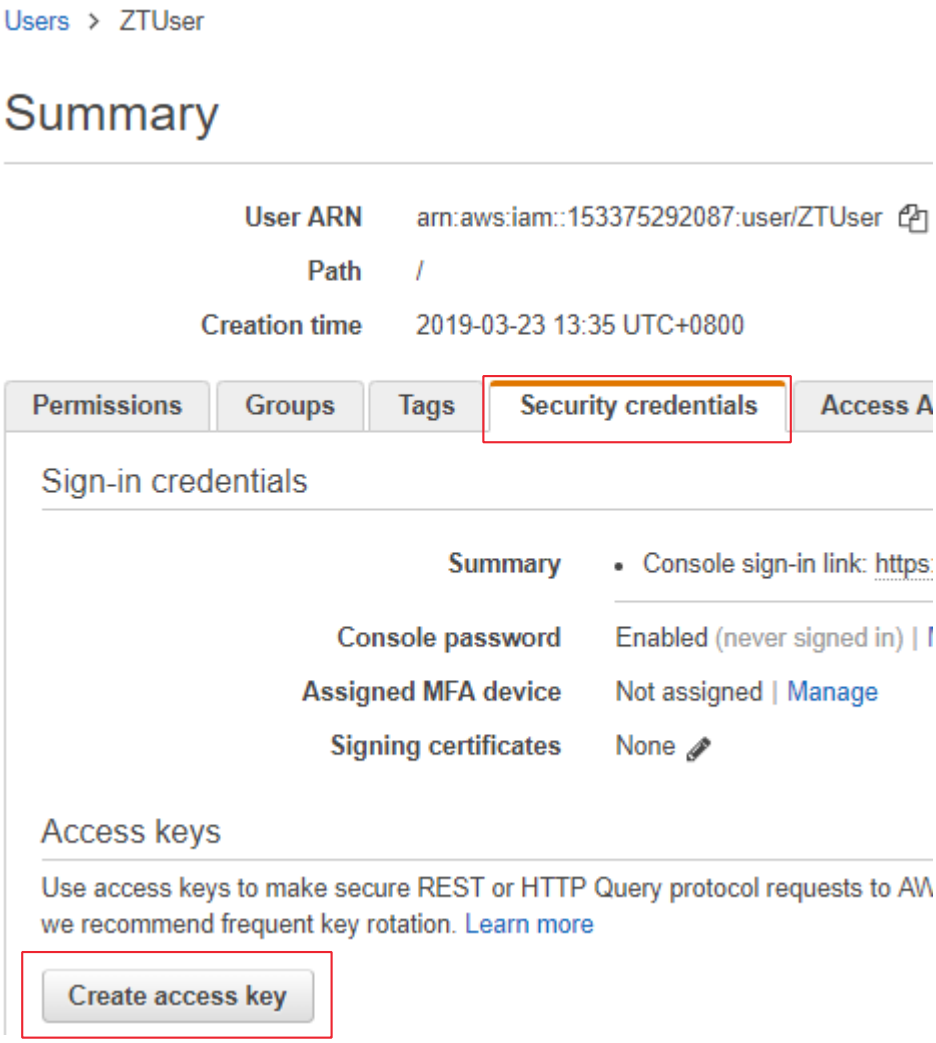
11. Go to IAM Console https://console.aws.amazon.com/iam/.
12. Select “ZTUser” from the “Users” in the left panel under the User name.

Figure 10-7. User Selection



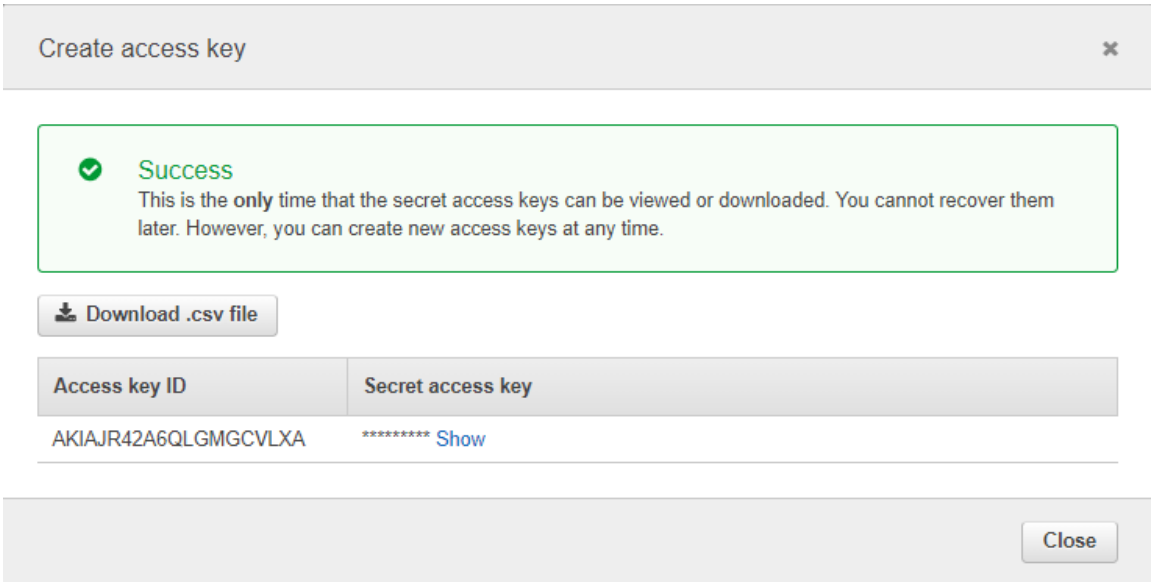
13. Click *ZTUser > Security Credentials*. Click the **Create access key** button.

Figure 10-8. ZTUser Security Credentials



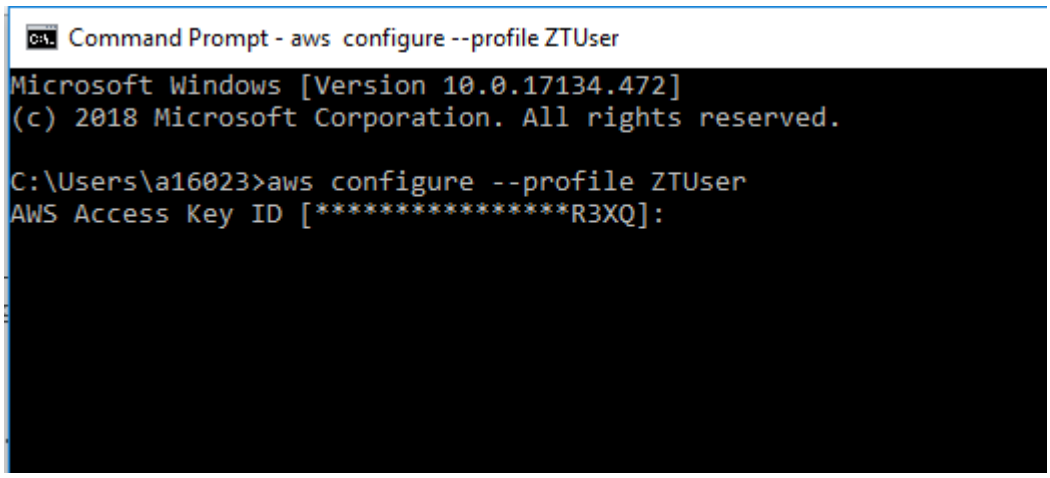
14. “Access key ID” and “Secret access key” are created. Copy the Access key ID and Secret access key for the following configurations.

Figure 10-9. Access Key ID and Secret Access Key



15. Install AWS CLI to your PC by following the steps in 5.1.4 Configure AWS CLI.
16. Open Command Prompt and run the following command: `aws configure --profile ZTUser`.
17. Enter the Access Key ID and Secret Access Key of ZTUser account when prompted.

Figure 10-10. Configuring Access Key ID and Secret Access Key



18. Observe the following results.

```
>aws configure --profile ZTUser
AWS Access Key ID [None]: <ACCESSKEYID>
AWS Secret Access Key [None]: <SECRETACCESSKEY>
Default region name [None]: us-east-1 ( <-- Enter the region that you selected )
```

10.2 Create Lambda Functions for Alexa

1. Edit the `\cloud-formation-templates\ createlambdaforalexa.template.yaml` file.

2. Add the Alexa Smart Home Skill ID (Wi-Fi Smart Device Smart Home Skill) from 7.1 Microchip Wi-Fi Smart Device Smart Home Skill Setup in the argument *EventSourceToken*.

```
CloudFormationLambdaTriggerSmartHomeSkill:
  Type: 'AWS::Lambda::Permission'
  Properties:
    Action: 'lambda:InvokeFunction'
    FunctionName: !Ref CloudFormationLambdaforAlexaSmartHome
    Principal: 'alexa-connectedhome.amazon.com'
    EventSourceToken: 'amzn1.ask.skill.1d6f1f85-bf71-4340-8930-cdexxxxxxxxx'
```

3. Add the Alexa Custom Skill ID (Wi-Fi Sensor Board Skill) from 7.2 Microchip Wi-Fi Sensor Board Skill Setup in the argument *EventSourceToken*.

```
CloudFormationLambdaTriggerCustomSkill:
  Type: 'AWS::Lambda::Permission'
  Properties:
    Action: 'lambda:InvokeFunction'
    FunctionName: !Ref CloudFormationLambdaforAlexaCustom
    Principal: 'alexa-appkit.amazon.com'
    EventSourceToken: 'amzn1.ask.skill.1d6f1f85-bf71-4340-8930-cxxxxxxxxxxx'
```

4. Save the file after modification.
5. Open Command Prompt and go to the `\cloud-formation-templates\` directory. Run the command: `aws s3 mb s3://<YOUR BUCKET NAME> --profile ZTUser`.

   Note: Ensure that the bucket name must be unique across all existing bucket names in Amazon S3 and must not contain uppercase or space characters.

   Figure 10-11. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws s3 mb s3://test0323 --profile ZTUser
make_bucket: test0323
```

6. Run command: `aws cloudformation package --template-file createLambdaForAlexa.template.yaml --s3-bucket <YOUR BUCKET NAME> --output-template-file createLambdaForAlexa.package.yaml`.

   Figure 10-12. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation package --template-file
 createLambdaForAlexa.template.yaml --s3-bucket test0323 --output-template-file createLambdaForAlexa.package.yaml
Uploading to 99a72403601ce6c1edb628851652af39  21947 / 21947.0  (100.00%)
Successfully packaged artifacts and wrote output template to file createLambdaForAlexa.package.yaml.
Execute the following command to deploy the packaged template
aws cloudformation deploy --template-file C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates\
createLambdaForAlexa.package.yaml --stack-name <YOUR STACK NAME>
```

7. Run command: `aws cloudformation deploy --template-file createLambdaForAlexa.package.yaml --stack-name createLambdaForAlexa --capabilities CAPABILITY_NAMED_IAM --profile ZTUser`.

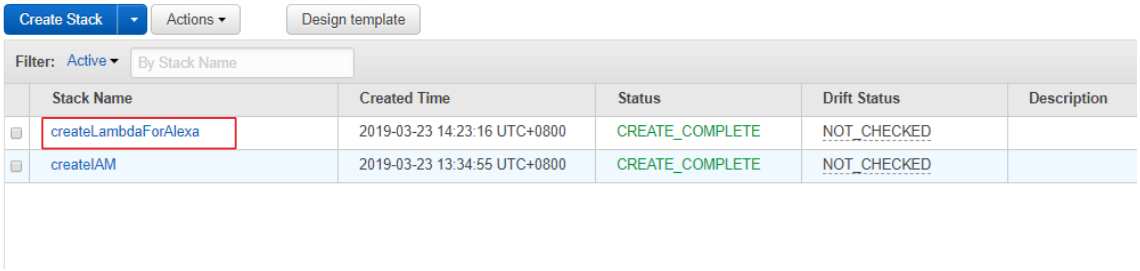
   Figure 10-13. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation deploy --template-file
createLambdaForAlexa.package.yaml --stack-name createLambdaForAlexa --capabilities CAPABILITY_NAMED_IAM --profile ZTUser


Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - createLambdaForAlexa
```

8. Go to https://console.aws.amazon.com/cloudformation to find the Stack “createLambdaForAlexa”.

Figure 10-14. Status of createLambdaForAlexa



9. In the AWS Lambda console, “WiFi-Smart-Device-Kit-Custom-Skill” and “WiFi-Smart-Device-Kit-Smart-Home-Skill” are created. Go to AWS Lambda console, edit “WiFi-Smart-Device-Kit-Custom-Skill”. In “Function code”, edit `index.js` to input the AWS IoT Endpoint and the Region. The AWS IoT Endpoint can be found from the Settings page of AWS IoT console.

Figure 10-15. AWS Lambda Console

```
6  var config = {};
7
8  config.IOT_BROKER_ENDPOINT = "xxxxxxxxxxx.iot.us-east-1.amazonaws.com".toLowerCase();
9
10 config.IOT_BROKER_REGION = "us-east-1";
11
12 // Load AWS SDK libraries
13 var AWS = require('aws-sdk');
14
```

10.3 Create Amazon Cognito for Alexa

1. Edit the `\cloud-formation-templates\ createCognito.template.yaml` file. Add “CallbackURL” of “Wi-Fi Smart Device For Alexa Smart Home Skill” Cognito User Pool App Client from 7.1 Microchip Wi-Fi Smart Device Smart Home Skill Setup .

```
  WiFISmartDeviceForAlexaSmartHomeSkillClientSettings:
    Type: 'Custom::CognitoUserPoolClientSettings'
    Properties:
      ServiceToken: !GetAtt CloudFormationCognitoUserPoolClientSettings.Arn
      UserPoolId: !Ref UserPool
      UserPoolClientId: !Ref WiFISmartDeviceForAlexaSmartHomeSkillClient
      SupportedIdentityProviders:
        - COGNITO
      CallbackURL: 'https://layla.amazon.com/api/skill/link/M2TDOWMJ3J2LNM,https://
alexa.amazon.co.jp/api/skill/link/M2TDOWMJ3J2LNM,https://pitangui.amazon.com/api/skill/
link/M2TDOWMJ3J2LN'
      LogoutURL: 'https://www.google.com'
      AllowedOAuthFlowsUserPoolClient: true
      AllowedOAuthFlows:
```

2. Add “CallbackURL” of “Wi-Fi Smart Device For Alexa Custom Skill” App Client from 7.2 Microchip Wi-Fi Sensor Board Skill Setup .

```
  WiFISmartDeviceForAlexaCustomSkillClientSettings:
    Type: 'Custom::CognitoUserPoolClientSettings'
    Properties:
      ServiceToken: !GetAtt CloudFormationCognitoUserPoolClientSettings.Arn
      UserPoolId: !Ref UserPool
      UserPoolClientId: !Ref WiFISmartDeviceForAlexaCustomSkillClient
      SupportedIdentityProviders:
        - COGNITO
      CallbackURL: 'https://layla.amazon.com/api/skill/link/M2TDOWMJ3J2LNM,https://
alexa.amazon.co.jp/api/skill/link/M2TDOWMJ3J2LNM,https://pitangui.amazon.com/api/skill/
link/M2TDOWMJ3J2LN'
      LogoutURL: 'https://www.google.com'
```

```
      AllowedOAuthFlowsUserPoolClient: true
      AllowedOAuthFlows:
```

3. Add a unique name in the “Domain”of the AWS Cognito User Pool for successful cloud formation.

```
    UserPoolDomain:
    Type: 'Custom::CognitoUserPoolDomain'
    Properties:
      ServiceToken: !GetAtt CloudFormationCognitoUserPoolDomain.Arn
      UserPoolId: !Ref UserPool
      Domain: 'userpool-test-018903'
```

4. In the Command Prompt, run the command: `aws cloudformation package --template-file createCognito.template.yaml --s3-bucket <YOUR BUCKET NAME> --output-template-file createCognito.package.yaml`.

Figure 10-16. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation package --template-file
 createCognito.template.yaml --s3-bucket test0323 --output-template-file createCognito.package.yaml
Uploading to d665b52fe7099d6eb2172b0a35e65a66  903 / 903.0  (100.00%)
Successfully packaged artifacts and wrote output template to file createCognito.package.yaml.
Execute the following command to deploy the packaged template
aws cloudformation deploy --template-file C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates\
createCognito.package.yaml --stack-name <YOUR STACK NAME>
```

5. Run command: `aws cloudformation deploy --template-file createCognito.package.yaml --stack-name createCognito --capabilities CAPABILITY_NAMED_IAM --profile ZTUser`.

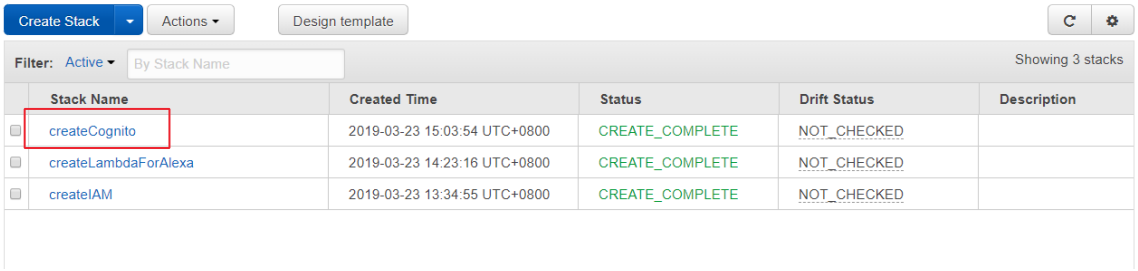
Figure 10-17. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation deploy --template-file
createCognito.package.yaml --stack-name createCognito --capabilities CAPABILITY_NAMED_IAM --profile ZTUser

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - createCognito
```

6. Go to https://console.aws.amazon.com/cloudformation and find the stack “createCognito”. A Cognito User Pool “WiFiSmartDeviceUserPool” and a Cognito Identity Pool “WiFiSmartDeviceIdentity” are successfully created. The developer can check this in the Amazon Cognito console.

Figure 10-18. Status of createCognito



10.4 Create AWS IoT Policy for Smartphone Application

This section provides the steps to create AWS IoT policy used for the mobile application to access AWS IoT resources. The policy name is WiFiSmartDeviceAppPolicy by default but it can be changed in the `create_iot_policy_mobile_app.template.yaml` file.

1. In the Command Prompt, run command: `aws cloudformation package --template-file create_iot_policy_mobile_app.template.yaml --s3-bucket <YOUR BUCKET NAME> --output-template-file create_iot_policy_mobile_app.package.yaml`.

Figure 10-19. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation package --template-file
 create_iot_policy_mobile_app.template.yaml --s3-bucket test0323 --output-template-file create_iot_policy_mobile_app.pac
kage.yaml

Successfully packaged artifacts and wrote output template to file create_iot_policy_mobile_app.package.yaml.
Execute the following command to deploy the packaged template
aws cloudformation deploy --template-file C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates\
create_iot_policy_mobile_app.package.yaml --stack-name <YOUR STACK NAME>
```

2. Run command: `aws cloudformation deploy --template-file create_iot_policy_mobile_app.package.yaml --stack-name createIoTPolicy --capabilities CAPABILITY_NAMED_IAM --profile ZTUser`.

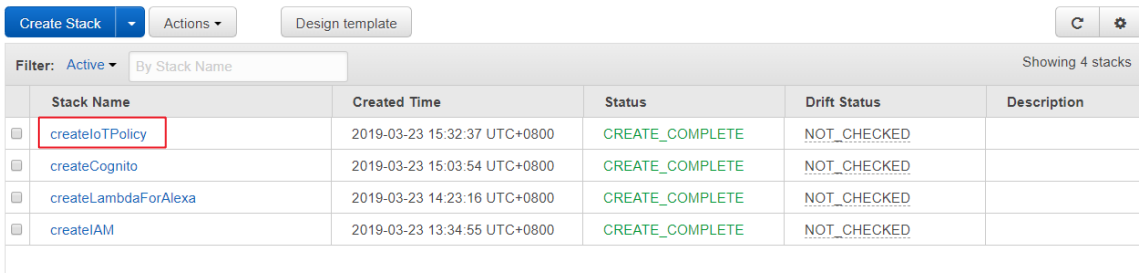
Figure 10-20. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation deploy --template-file
create_iot_policy_mobile_app.package.yaml --stack-name createIoTPolicy --capabilities CAPABILITY_NAMED_IAM --profile ZTU
ser

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - createIoTPolicy
```

3. Go to https://console.aws.amazon.com/cloudformation and find the new stack created using “createIoTPolicy”. An AWS IoT Policy “WiFiSmartDeviceAppPolicy” is successfully created. The developer can check this in the AWS IoT console.

Figure 10-21. Status of createIoTPolicy



10.5 Create AWS DynamoDB Table

Perform the following steps for creating AWS DynamoDB table:

1. In the Command Prompt, run command: `aws cloudformation package --template-file create_dynamodb.template.yaml --s3-bucket <YOUR BUCKET NAME> --output-template-file create_dynamodb.package.yaml`.

Figure 10-22. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation package --template-file
 create_dynamodb.template.yaml --s3-bucket test0323 --output-template-file create_dynamodb.package.yaml

Successfully packaged artifacts and wrote output template to file create_dynamodb.package.yaml.
Execute the following command to deploy the packaged template
aws cloudformation deploy --template-file C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates\
create_dynamodb.package.yaml --stack-name <YOUR STACK NAME>
```

2. Run command: `aws cloudformation deploy --template-file create_dynamodb.package.yaml --stack-name createDynamoDB --capabilities CAPABILITY_NAMED_IAM --profile ZTUser.`

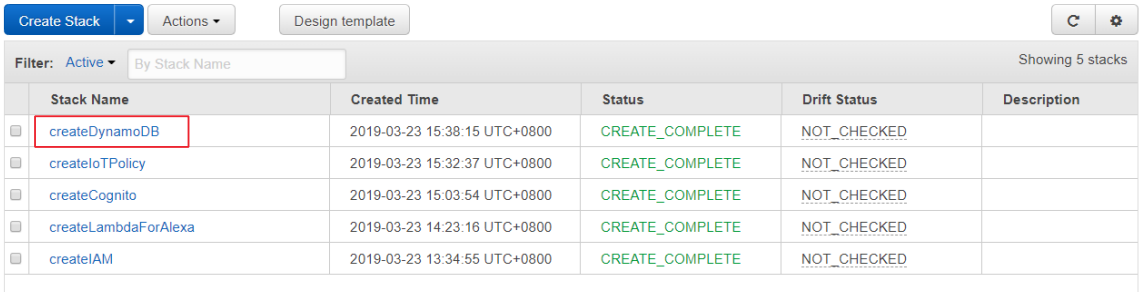
Figure 10-23. Console Log

```
C:\github\winc1500-wifi-smart-device-enablement-kit\cloud-formation-templates>aws cloudformation deploy --template-file
create_dynamodb.package.yaml --stack-name createDynamoDB --capabilities CAPABILITY_NAMED_IAM --profile ZTUser

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - createDynamoDB
```

3. Go to https://console.aws.amazon.com/cloudformation and find the stack “createDynamoDB”. A DynamoDB Table “SensorBoardAcctTable” is successfully created. Developer can check this in the AWS DynamoDB console.

Figure 10-24. Status of createDynamoDB



11. Document Revision History

Revision	Date	Section	Description
A	05/2019	Document	Initial Revision

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

© 2019 Microchip Technology Inc.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2019, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-4498-5

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820