
TrustFLEX Step by Step Guide

AWS IoT with Custom PKI

Table of Contents

1	Introduction	3
1.1	Getting started with Jupyter Notebook Tutorials	3
1.1.1	Starting Jupyter Notebook.....	3
1.2	Jupyter Notebook Basics	3
1.2.1	The Notebook dashboard	3
1.3	Introduction to Jupyter Notebook GUI	4
2	Jupyter Notebook Tutorials	6
3	Resource Generation Notebook	7
4	Use Case Prototyping	11
4.1	AWS account credentials	11
4.2	Running Custom PKI example on Jupyter Notebook.....	12
4.3	Running Custom PKI example on Embedded platform	17
4.3.1	Atmel Studio:	17
4.3.2	MPLAB:	21
4.4	CryptoAuth Trust Platform Factory reset.....	24
5	FAQ.....	25

1 Introduction

This document gives a detailed walk through of the custom public key infrastructure use case implementation. If familiar with Jupyter Notebook, can skip this section and move to Section 2.

1.1 Getting started with Jupyter Notebook Tutorials

Jupyter Notebook is open source web application which allows you to create documents that contain code that you can execute in place as well as narrative text. It provides GUI elements, ability to execute code in place, ability to add images and gives it the look and feel that normal code files lack.

Jupyter notebooks are mainly used to explain/evaluate code in an interactive way.

1.1.1 Starting Jupyter Notebook

Jupyter notebook can be launched from the Anaconda Navigator main window.



1.2 Jupyter Notebook Basics

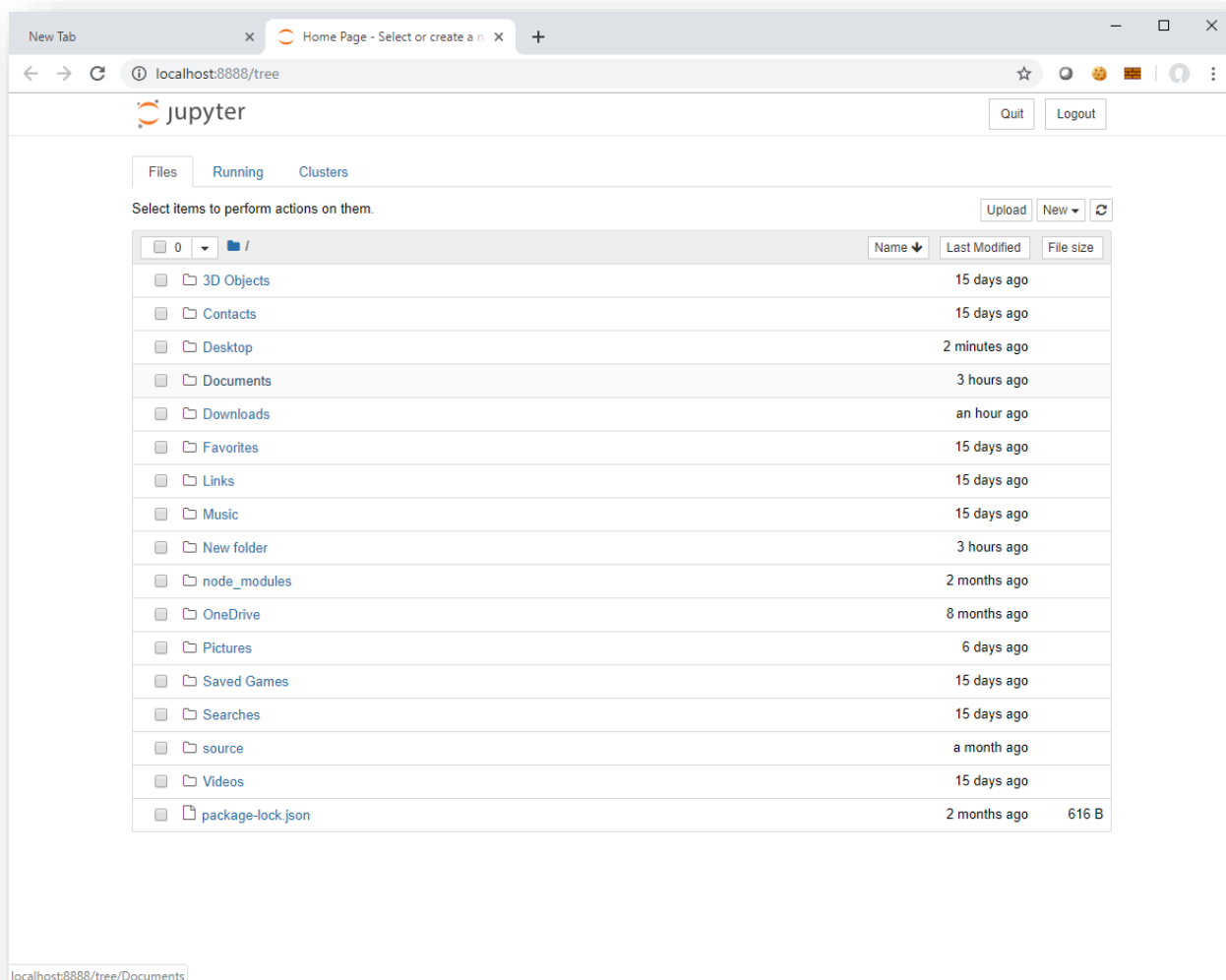
It is recommended to become familiar with Jupyter basic concepts with the online documentation, <https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html>

Some of the content is duplicated here for convenience. The online documentation should always be used as a reference.

1.2.1 The Notebook dashboard

When you first start the notebook server, your browser will open Notebook dashboard. The dashboard serves as a home page for the notebook. Its main purpose is to display the notebooks and files in the current directory.

For example, here is a screenshot of the Jupyter dashboard. The top of the notebook list displays clickable breadcrumbs of the current directory. By clicking on these breadcrumbs or sub-directories in the notebook list, you can navigate your file system.



1.3 Introduction to Jupyter Notebook GUI.

Jupyter Notebooks contain cells where you can either write code or markdown text. Notebooks contain multiple cells, some set as code and others markdown. Code cells contain code that can be executed live, and markdown contains text and images that explains the code.

Below image shows some options in a typical Jupyter Notebook. Individual cells can be executed by pressing on the RUN button as shown in the below image.

All cells in the Notebook can be executed in order by **Kernel->Restart & Run All**.



To run all cells in sequence.



2 Jupyter Notebook Tutorials

The TrustPlatform Design Suite comes with a number of Notebook Tutorials to easily prototype popular use cases for TrustFLEX and Trust&Go devices. Here is the list of Jupyter Notebook Tutorials.

Jupyter Notebook Tutorials	Relative Path	Applicable devices
Manifest Generation	TNGTLS_Manifest_Generation\notebooks\TNGTLS Manifest File Generation.ipynb	Trust&GO
Resource Generation	TFLXTLS_resource_generation\Crypto Resource Generator.ipynb	TrustFLEX
Accessory Authentication	TFLXTLS_Use_Cases\notebooks\ accessory-authentication\ Accessory Authentication.ipynb	TrustFLEX
AWS Custom PKI	TFLXTLS_Use_Cases\notebooks\aws-iot\ aws-iot with ECC608A-TLFXTLS.ipynb	TrustFLEX
Firmware Validation	TFLXTLS_Use_Cases\notebooks\secureboot\ Firmware Validation with ECC608A-TFLXTLS Tutorial.ipynb	TrustFLEX
IP Protection	TFLXTLS_Use_Cases\notebooks\ipprotection\ IP Protection with ECC608A-TFLXTLS Tutorial.ipynb	TrustFLEX
Secure Public Key Rotation	TFLXTLS_Use_Cases\notebooks\public-key-rotation\Public Key Rotation with ECC608A-TFLXTLS Tutorial.ipynb	TrustFLEX

3 Resource Generation Notebook

TFLXTLS device is one of the three devices available in the Trust Platform USB Dongle Board.

TrustFLEX devices come pre-programmed with certificates in slots 10, 11 and 12, also slots 0-4 have pre-generated private keys, other than the mentioned slots all the other slots have no meaningful data in them.

The Resource Generator Notebook will create development keys and certificates for all slots that can be further customized. Keys and Certificate chains are stored in the PC filesystem. These keys should never be used for production purposes as their generation is not handled in a secure environment. These development keys will be later used by the other notebooks to implement the various pre-defined use cases.

By default, Jupyter starts in Users directory (\$HOME for MacOS or Linux systems). For the remainder of this document, it will be assumed that the TrustPlatform-DesignSuite folder is contained in the Documents folder.

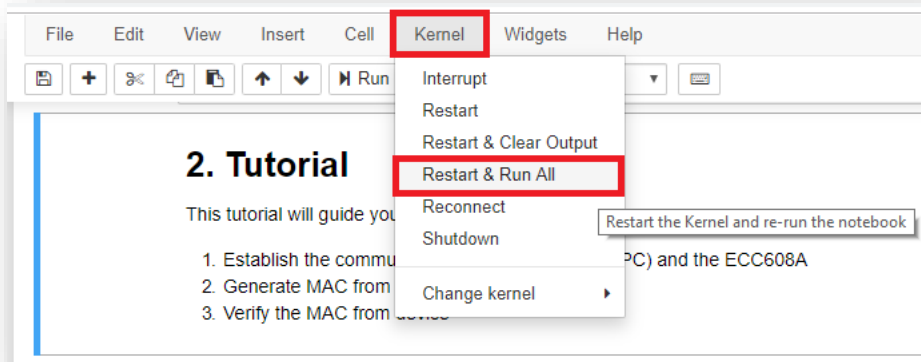
Within the Jupyter Dashboard, navigate to Documents/TrustPlatform-DesignSuite/TFLXTLS_Resource_Generation folder

Select the Crypto Resource Generator.ipynb notebook



Run all cells of the Crypto Resource Generator Notebook: Kernel->Restart & Run All

Note: Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to [4.3 CryptoAuth TrustPlatform Factory reset](#) section for reloading default program.



It will execute and prompt you to choose between MCHP certificate and a custom certificate chain, enter '2' (for custom certificate) and press Enter key.

Now it prompts you to enter the organization name, enter the name that will be used as an Organization Name in the certificate template. The name length is limited to 24 characters.

The Notebook will generate a number of keys and certificates. Make sure you have an error free output before continuing to the next steps of the training.

The output log should resemble this:

Loading root CA key

Generating self-signed root CA certificate
Saving to root-ca.crt

Loading signer CA key

Generating signer CA CSR
Saving to signer-ca.csr

Loading signer CA CSR
Loading from signer-ca.csr

Loading root CA key
Loading from root-ca.key

Loading root CA certificate
Loading from root-ca.crt

Generating signer CA certificate from CSR

Saving to signer-ca.crt

Signer Certificate written successfully to device
Device Certificate written successfully to device
Thing ID c21a93e8d413135f21f452b620a4c83bdf25b589

Generating signer certificate definition header file - cust_def_1_signer.h
Generating device certificate definition header file - cust_def_2_device.h
Generating signer certificate definition source file - cust_def_1_signer.c
Generating device certificate definition source file - cust_def_2_device.c

Custom certificate generation and provisioning - SUCCESS

Root Certificate:

Loading from root-ca.crt
Crypto Authentication Root CA 000
-----BEGIN CERTIFICATE-----
MIIBzzCCAXSgAwIBAgIQQMs/k4fC+GzN1M6BcRRO2DAKBggqhkJOPQQDAjBPMSEw
HwYDVQQKDBhOaWNvbGFzIE9yZyBOYW1lICAgICAgICAgICAgICAgICAgICAgICAg
byBBdXR0ZW50aWNhdGlvbSBSb290IENBIDAuMDAeFw0xOTA5MDQyMzU2MTRaFw00
NDA4MjgyMzU2MTRaME8xITAfBgNVBAoMGE5pY29sYXMgT3JnIE5hbWUgICAgICAg
IDEqMCcGA1UEAwwhQ3J5cHRvIEF1dGhlnbnRyY2F0aW9uIFJvb3QgQ0EgMDAwMFkw
EwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEIqwwOu9E0hjvMFOntRrZtzTWWx0mVwV7
8M07s/VjM/rwu/I486yjOI1vq9k00JkBALrbt0wd5GkZM53nEgfJjKMyMDAwHQYD
VR0OBBYEFFsENZIG+D4+r93QBNORixk/r7OoMA8GA1UdEwEB/wQFMAMBAf8wCgYI
KoZIZj0EAwIDSQAwwRgIhAMyFkf/jAiA5jDncnH7VI9Em5IHlsvUgtKB24OhqxjTi
AiEAlJ0EnZukMELq4xsjG7+v6xxlswV2p8fOj6N2ZcyPbU0=
-----END CERTIFICATE-----

Validate Root Certificate:

OK

TFLXTLS Signer Certificate:

Loading from signer-ca.crt
Crypto Authentication Signer FFFF
-----BEGIN CERTIFICATE-----
MIICAjCCAaigAwIBAgIQV2Ww7Rjyk6b9YwLY0gGENDAKBggqhkJOPQQDAjBPMSEw
HwYDVQQKDBhOaWNvbGFzIE9yZyBOYW1lICAgICAgICAgICAgICAgICAgICAgICAg
byBBdXR0ZW50aWNhdGlvbSBSb290IENBIDAuMDAeFw0xOTA5MDQyMzU2MTRaFw0y
OTA5MDQyMzU2MTRaME8xITAfBgNVBAoMGE5pY29sYXMgT3JnIE5hbWUgICAgICAg
IDEqMCcGA1UEAwwhQ3J5cHRvIEF1dGhlnbnRyY2F0aW9uIFNpZ25lciBGRkZGMFkw
EwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE/P6/EAYOcTP1vokpVfiaDMRsA7oT4G2e
MIK9CdY7UWpdKMKVawmPUN9/VGtxsqTCx56eWy1axyyochG8wHn1N6NmMGQwDgYD
VR0PAQH/BAQDAgGGMbIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVIR0OBBYEFMSkbTjX
FC6EP2hKktjVw6wNRUswMB8GA1UdIwQYMBaAFFsENZIG+D4+r93QBNORixk/r7Oo
MAoGCCqGSM49BAMCA0gAMEUCIDAFcGgep1/zM4EqMuoSVY6yZLUK+nyIzpFuLZLd

4 Use Case Prototyping

This hands-on lab is intended to demonstrate the usage of TrustFLEX to secure an AWS IoT connection based on a custom PKI.

The AWS IoT device reference implementation is provided as an MPLAB X project and the generation of a custom PKI is achieved through the execution of Jupyter Notebook Tutorials

Here are the steps that will be required to complete this Tutorial:

- Retrieve AWS Account credentials
- Retrieve AWS Registration code and register the custom PKI to the AWS account (through a Jupyter Notebook)
- Build the AWS IoT device source code and flash it to the USB Dongle Board

4.1 AWS account credentials

This step assumes the AWS account is already configured for JITR (Just In Time Registration) operation as part of SPG MASTERS AWS account management scheme. If it is not, you can run the CloudFormation template built by Microchip to automate the configuration of the account.

The CloudFormation templates can be found here:

<https://github.com/MicrochipTech/aws-iot-zero-touch-secure-provisioning-kit/tree/master/cloud-formation-templates>

Once executed the resulting subaccount will be configured with a Lambda function that will automatically register a device upon an authenticated incoming TLS connection. The TLS session uses the custom certificate chain generated in section 3 of this tutorial.

As part of the creation of the account, a ZTUser will be generated with the AWS Access Key ID and AWS Secret Access Key.

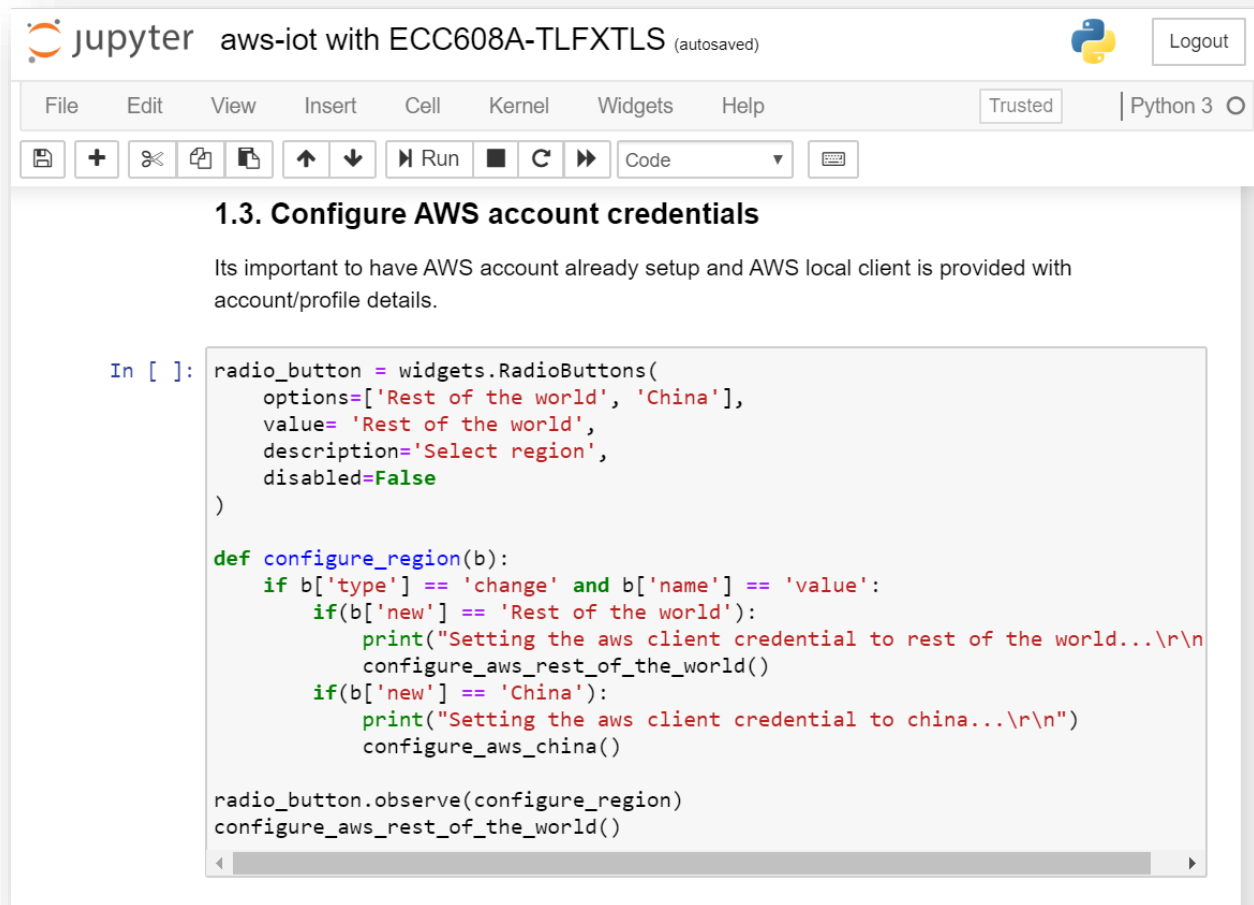
4.2 Running Custom PKI example on Jupyter Notebook

By running this following step, we can configure the AWS command line with AWS credentials, register the signer certificate to AWS IoT and get AWS host endpoint to which device should connect.

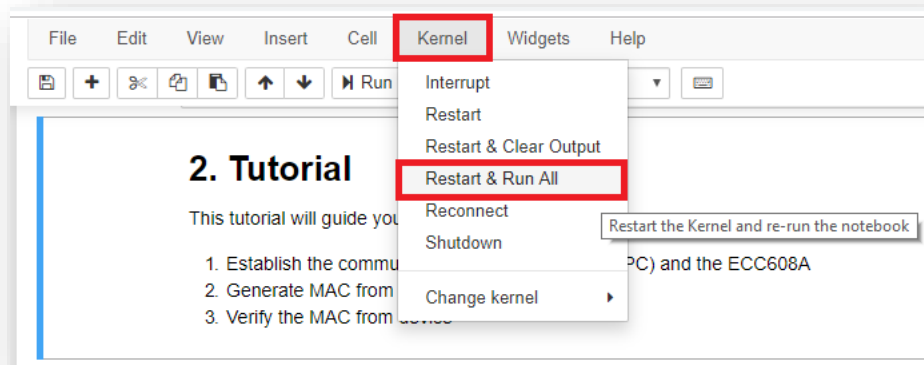
1. From the Jupyter Home page, navigate to **TFLXTLS_Use_Cases\notebooks\aws-iot\aws-iot with ECC608A-TLFXTLS.ipynb** notebook file and open it.



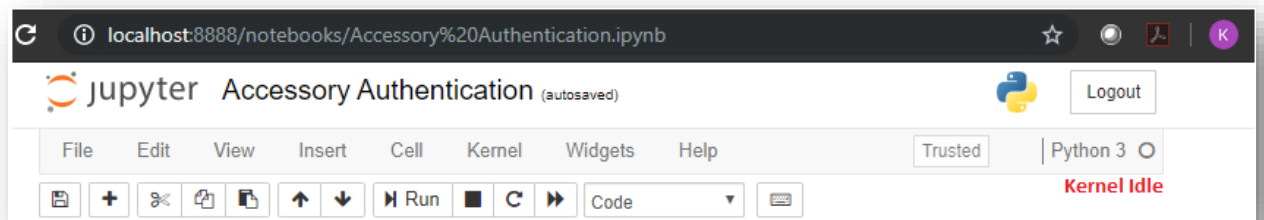
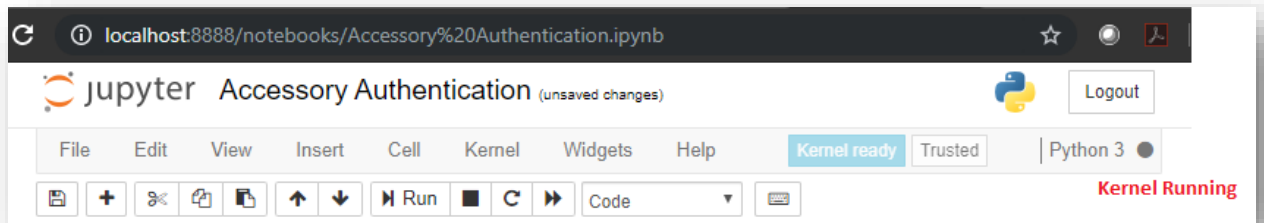
Opening the Jupyter notebook example should load the following on the browser.



2. Run All Cells by using Kernel -> Restart & Run All



It may take a while to complete, wait for the kernel to complete all processing i.e. from Kernel Running to Kernel Idle state (Check circle above **RED** text)



3. Navigate through different cells output for the description of the step and result from the execution.

4. There are 4 major steps:

Configure AWS command line interface:

Before we can interact with AWS, we need to configure the tools with the appropriate AWS credentials. These credentials are composed of the **Access Key ID** and the **Secret Access Key**. Outside of the classroom, they will be generated when creating the IAM user.

Below screenshot display the configured AWS credentials

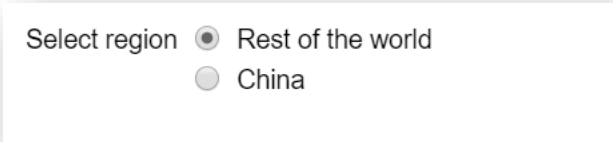
```
Setting aws access key...
Setting aws secret access key...
Setting aws region...

      Name              Value              Type      Location
      ----              -
      profile            <not set>          None      None
      access_key          *****HGMA        shared-credentials-file
      secret_key          *****Qtqr        shared-credentials-file
      region              us-east-2          config-file  ~/.aws/config
```

Select region:

Select your region from the option available either rest of the world or china.

Below screenshot display the option to select region

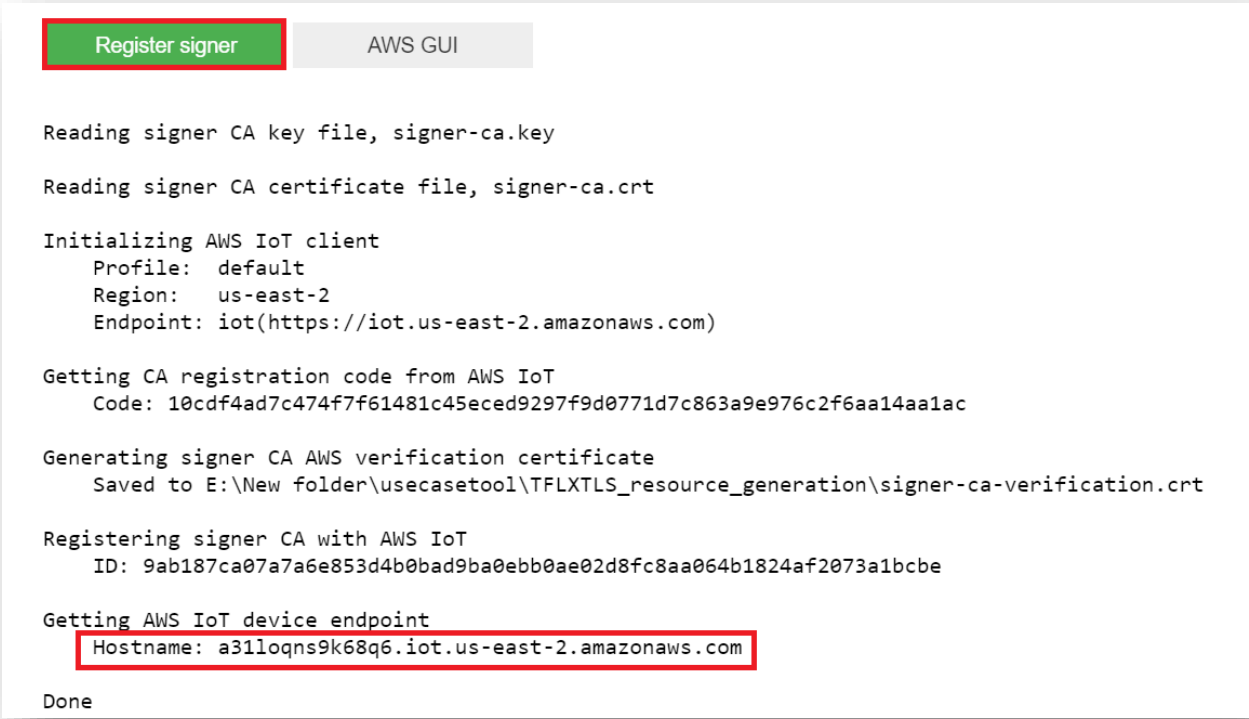


Select region ☒ Rest of the world
☐ China

Register Signer Module:

Code block of this step generates “**Register signer**” button. Clicking the button, it registers the custom PKI signer module to the AWS account, and gives an AWS host endpoint. To establish a secure communication with AWS IoT, we need to register the Custom PKI signer certificate to AWS IoT.

Upon successful execution, the log should look like this. The marked URL is the endpoint to which the device must connect. It will be added to the device firmware in the next section



```
Register signer    AWS GUI

Reading signer CA key file, signer-ca.key

Reading signer CA certificate file, signer-ca.crt

Initializing AWS IoT client
  Profile: default
  Region:  us-east-2
  Endpoint: iot(https://iot.us-east-2.amazonaws.com)

Getting CA registration code from AWS IoT
  Code: 10cdf4ad7c474f7f61481c45eced9297f9d0771d7c863a9e976c2f6aa14aa1ac

Generating signer CA AWS verification certificate
  Saved to E:\New folder\usecasetool\TFLXTLS_resource_generation\signer-ca-verification.crt

Registering signer CA with AWS IoT
  ID: 9ab187ca07a7a6e853d4b0bad9ba0ebb0ae02d8fc8aa064b1824af2073a1bcbe

Getting AWS IoT device endpoint
  Hostname: a31loqns9k68q6.iot.us-east-2.amazonaws.com

Done
```

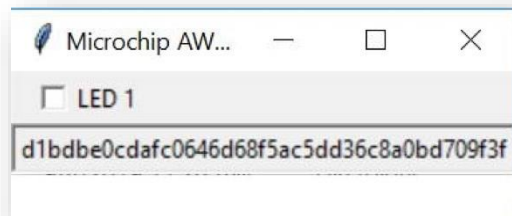
Once this step is completed, signer module is successfully registered to AWS IoT. Before running the last cell, we need to program the Crypto Trust Platform. So next step is to program the Crypto Trust Platform.

NOTE: Make sure that you executed C project successfully before executing the next step in the Jupyter notebook. To execute C project, refer "[Running AWS IoT example on Embedded platform](#)" next section.

AWS GUI:

Code block of this step generates "**AWS GUI**" button. Clicking the button, it will create a very basic graphical interface that will display the device ID and will allow to switch the board LED status.

Below screenshot display the graphical interface



Using this interface, Custom PKI CryptoAuth Trust Platform can able to communicate with AWS IoT. Upon successful communication, you have now a device connected to AWS IoT through a secure TLS session with a custom PKI using a Crypto Trust Platform.

4.3 Running Custom PKI example on Embedded platform

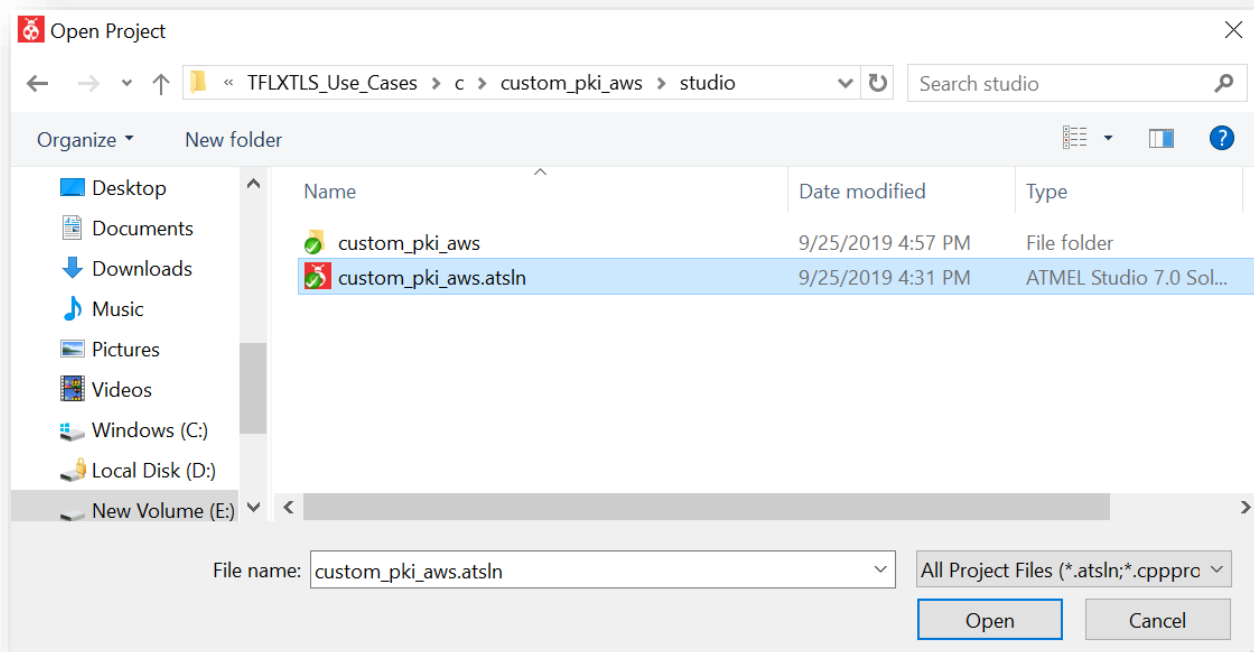
Once the resources are generated both Atmel Studio and MPLAB projects provided can be used to run the use case on CryptoAuth Trust Platform.

This project can configure the Wi-Fi credentials, establish a TLS connection, subscribe to MQTT and register device certificate but not register the signer module to AWS IoT. It is required to use the AWS IoT Jupyter notebook to register the signer module and get the AWS endpoint to which device must connect.

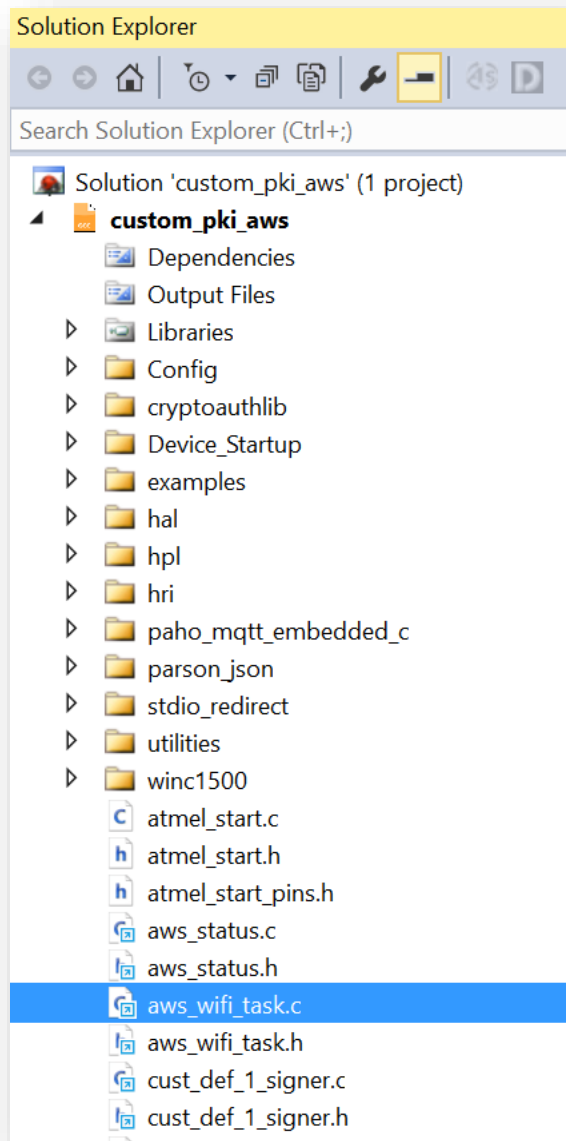
Once the signer module is registered and AWS endpoint is available then these embedded projects can be executed.

4.3.1 Atmel Studio:

1. Open **custom_pki_aws.atsln** project by navigating
TFLXTLS_Use_Cases\c\custom_pki_aws\studio\custom_pki_aws.atsln



2. In the project navigate to **custom_pki_aws -> aws_wifi_task.c** file



update the following constants before building the project:

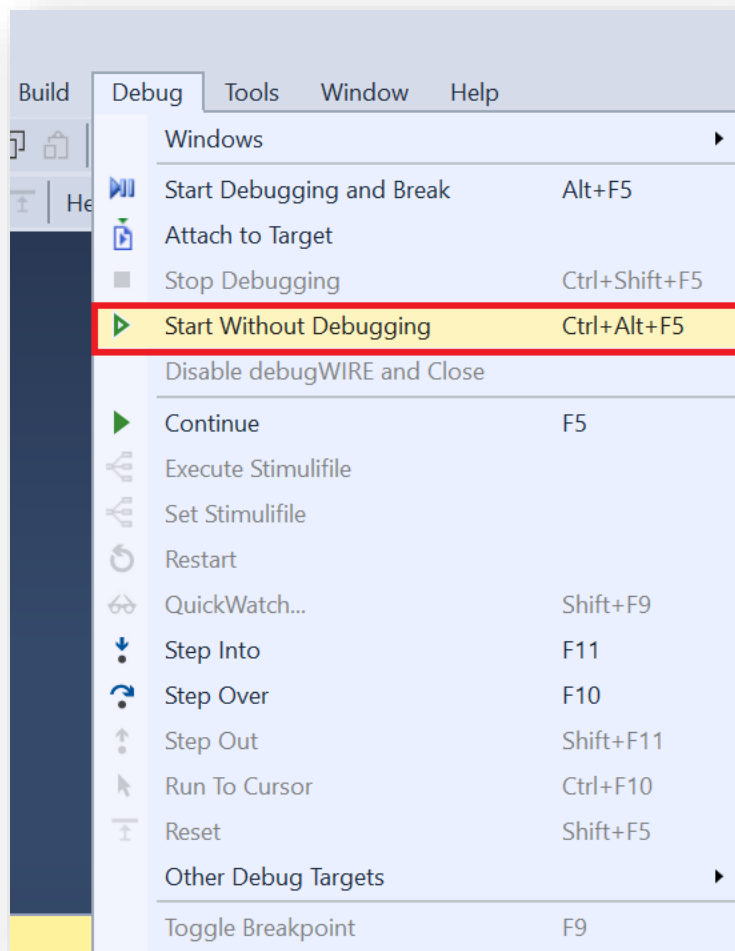
- MAIN_WLAN_SSID
- MAIN_WLAN_PSK
- AWS_HOST_ENDPOINT

The AWS_HOST_ENDPOINT string should be set to the value reported by the Jupyter Notebook of the section 4.3 (field Hostname)

```
#define MAIN_WLAN_SSID      "xxxxxx"
#define MAIN_WLAN_AUTH      M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK       "xxxxxxxxxx"

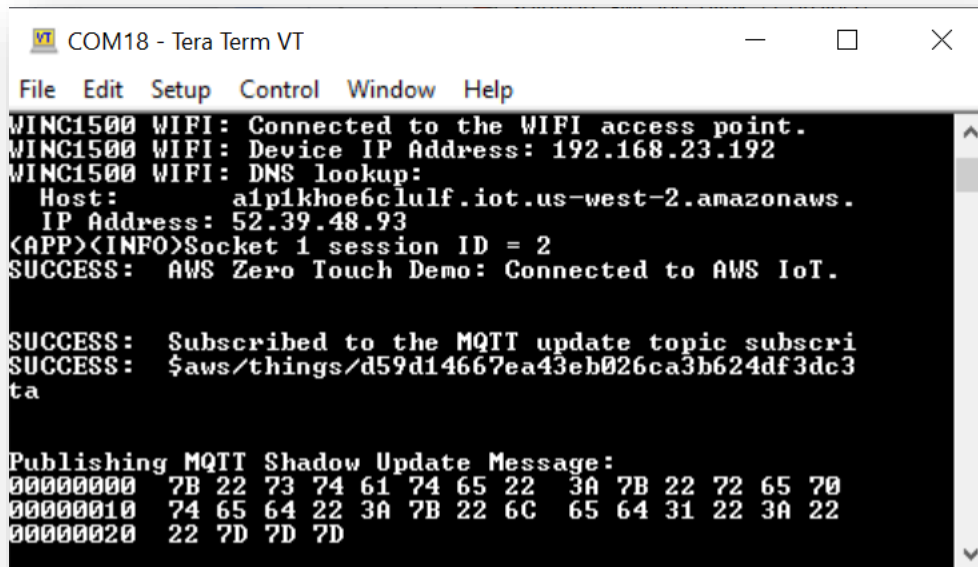
#define AWS_HOST_ENDPOINT   "xxxxxxx.iot.us-west-2.amazonaws.com"
```

3. Program the CryptoAuth Trust Platform by navigating to **Debug -> Start Without Debugging**



This step may take some time, wait for Atmel Studio to compile and program the device.

Once the programming is done, reset the hardware (press the reset button) and view the Console messages by using applications like 'Tera Term'. Open the application with the COM related to CryptoAuth Trust Platform with 115200-8-N-1 settings.

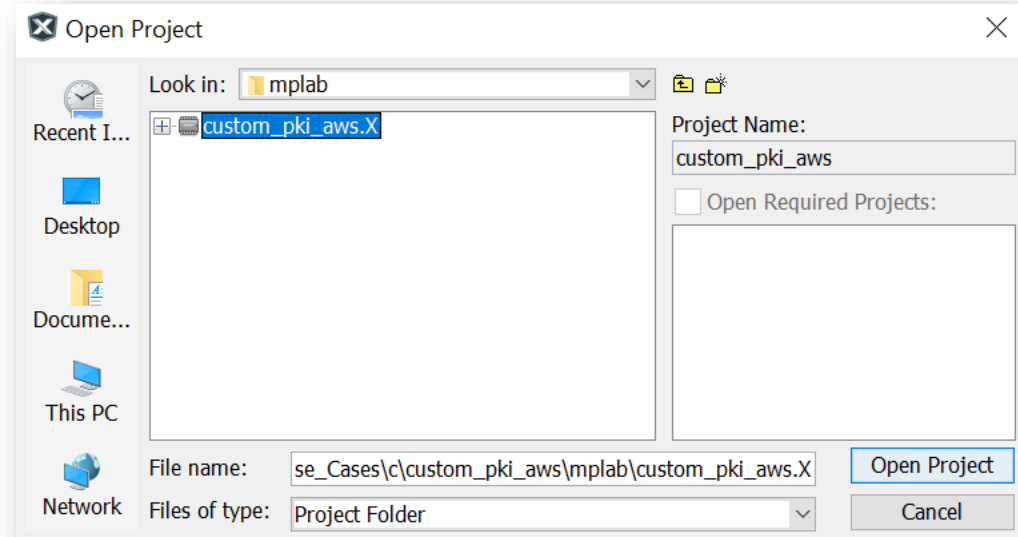
A screenshot of a Tera Term VT window titled 'COM18 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main area displays the following text:

```
WINC1500 WIFI: Connected to the WIFI access point.  
WINC1500 WIFI: Device IP Address: 192.168.23.192  
WINC1500 WIFI: DNS lookup:  
  Host:      alpikhoe6clulf.iot.us-west-2.amazonaws.  
  IP Address: 52.39.48.93  
<APP><INFO>Socket 1 session ID = 2  
SUCCESS:  AWS Zero Touch Demo: Connected to AWS IoT.  
  
SUCCESS:  Subscribed to the MQTT update topic subscri  
SUCCESS:  $aws/things/d59d14667ea43eb026ca3b624df3dc3  
ta  
  
Publishing MQTT Shadow Update Message:  
00000000  7B 22 73 74 61 74 65 22  3A 7B 22 72 65 70  
00000010  74 65 64 22 3A 7B 22 6C  65 64 31 22 3A 22  
00000020  22 7D 7D 7D
```

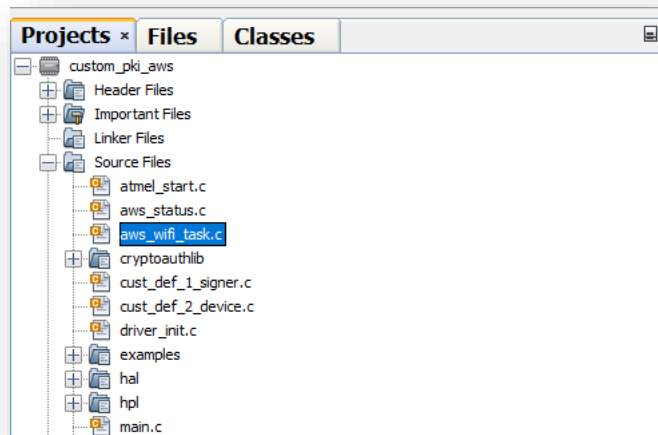
Once successfully programmed the CryptoAuth Trust Platform, now we can run the last step in the Jupyter Notebook. Just navigate to previous section 4.3 to run the last step (AWS GUI) in the Jupyter Notebook.

4.3.2 MPLAB:

1. Open **custom_pki_aws.X** project by navigating to MPLAB -> File -> Open Project -> **TFLXTLS_Use_Cases\c\custom_pki_aws\mplab\custom_pki_aws.X**



2. Open **aws_wifi_task.c** file by navigating to **custom_pki_aws -> Source Files -> aws_wifi_task.c**



update the following constants before building the project:

- MAIN_WLAN_SSID
- MAIN_WLAN_PSK
- AWS_HOST_ENDPOINT

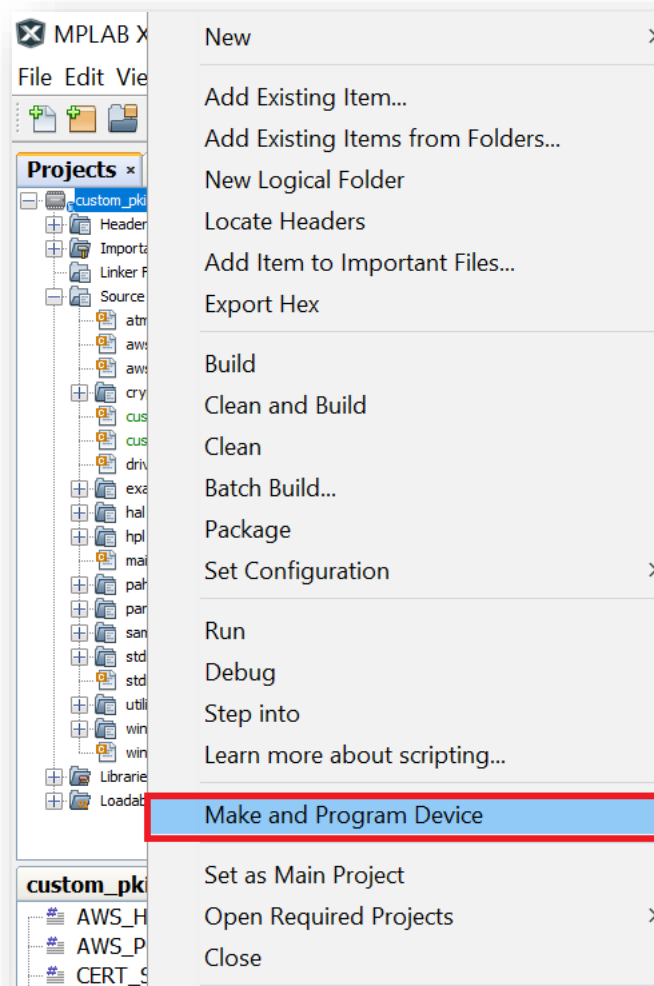
The AWS_HOST_ENDPOINT string should be set to the value reported by the Notebook of the section 4.3 (field Hostname)

```

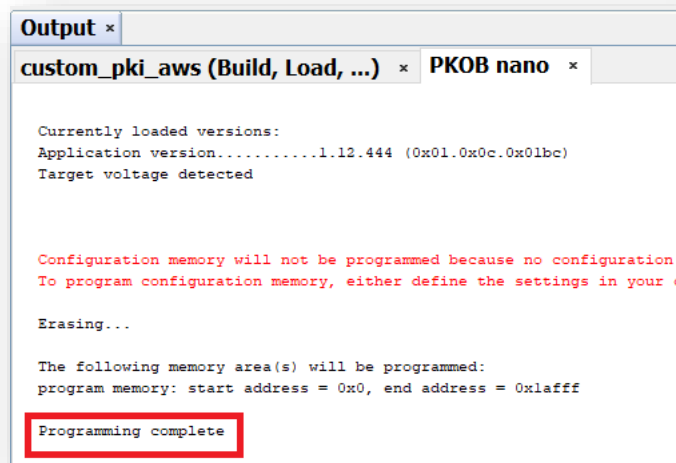
49
50 #define MAIN_WLAN_SSID      "XXXXXXX"
51 #define MAIN_WLAN_AUTH     M2M_WIFI_SEC_WPA_PSK
52 #define MAIN_WLAN_PSK      "XXXXXXX"
53
54 #define AWS_HOST_ENDPOINT   "XXXXXXXXXX.iot.us-west-2.amazonaws.com"
55

```

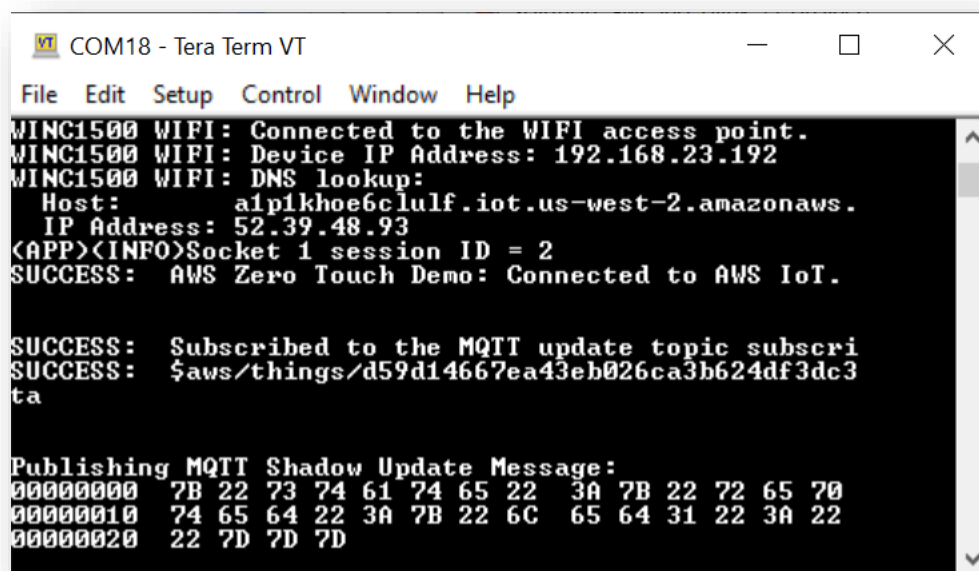
3. Program the CryptoAuth Trust platform by navigating to **custom_pki_aws -> Make and Program Device**



This step may take some time, wait for MPLAB to program the device. Once it is done programming you will see "**Programming complete**" message in Output Window.



Once the programming is done, reset the hardware (press the reset button) and view the Console messages by using applications like 'Tera Term'. Open the application with the COM related to CryptoAuth Trust Platform with 115200-8-N-1 settings.



Once successfully programmed the CryptoAuth Trust Platform, now we can run the last step in the Jupyter Notebook. Just navigate to previous section 4.3 to run the last step (AWS GUI) in the Jupyter Notebook

4.4 CryptoAuth Trust Platform Factory reset

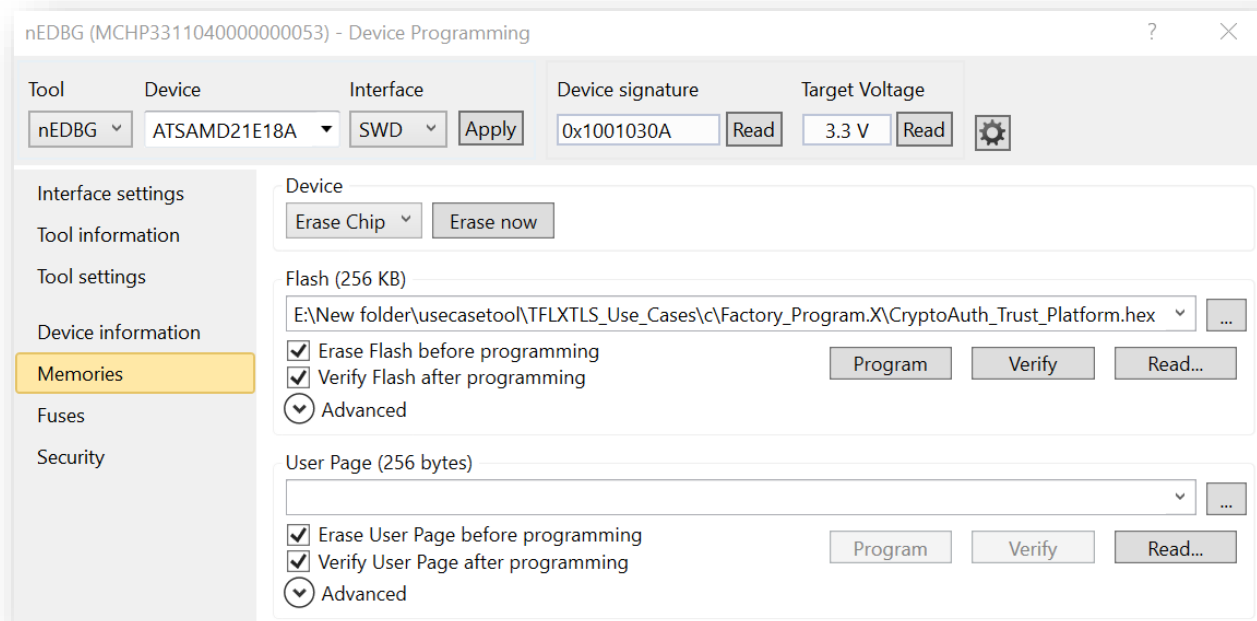
If any embedded project is loaded to CryptoAuth Trust Platform, the default program that enables interaction with CryptoAuth Trust Platform tools will be erased.

Before using the CryptoAuth Trust Platform with any other notebook or tools on PC, its required to reprogram the default firmware. Default hex file is available at

TFLXTLS_Use_Cases\c\Factory_Program.X\CryptoAuth_Trust_Platform.hex

To reprogram using Atmel Studio:

1. Navigate to AtmelStudio -> Tools -> Device Programming
2. Select Tool as nEDBG and Apply
3. Go to Memories and navigate to above path under Flash dropdown
4. Check both Erase Flash and Verify Flash
5. Click on Program



To reprogram using MPLAB:

1. Open **TFLXTLS_Use_Cases\c\Factory_Program.X** project in MPLAB IDE
2. Program the Crypto Trust platform by navigating to
CryptoAuth_Trust_Platform_Factory_Program -> Make and Program Device

Now, CryptoAuth Trust Platform contains factory application that enables interactions with Notebooks and/or PC tools.

5 FAQ

1. What are the reasons for “**AssertionError: Can't connect to the USB dongle**” error?

There are many possibilities like,

1. Crypto Trust Platform is having different application than factory reset firmware. Refer to “CryptoAuth TrustPlatform Factory reset” section any usecase TrustFLEX Guide for reloading it
2. Check the switch positions on Crypto Trust Platform and/or ATECC608A Trust board
 - a. Correct Trust device should be connected and only one device of that type is allowed on the I2C bus. Multiple devices with same address results in error
3. Check USB connections to Crypto Trust Platform

2. How to reload factory default application to Crypto Trust Platform?

Refer to “CryptoAuth TrustPlatform Factory reset” section any usecase TrustFLEX Guide for reloading it.

3. Why does my C projects generates No such file or directory with ../../../../TFLXTLS_resource_generation/?

C project generates this error when the resources are not generated prior to using embedded projects. Running the resource generation notebook ensures these files and secrets are generated.

4. Before running any use case notebook and/or C project, why is it mandate to execute resource generation?

When resource generation notebook is executed, it generates and programs the required resources like secrets, keys and certificates. These are only prototyping keys and cannot be used for production. These keys will be used part of Usecase notebooks and C projects

5. How to know the resources being used in a use case?

Refer to individual Usecase description html for details on transaction diagrams, resources being used and other details. The resources required for given use case is mentioned in INFER CRYPTOGRAPHIC ASSETS section.

6. When should I select Custom certificates while doing resource generation?

Custom certificates are required when user wants to have their own root, signer instead of MCHP provided. The difference would be organization name, common name and validity are configurable

7. How to know whether C project is executing on Trust Platform or not after programming?

Once the programming is done, the firmware will do use case operation. Depending on the use case operation's output, the Crypto Trust Platform board's status LED will blink at different rates.

If use case operation succeeds, LED blinks once every second. If it fails, LED blinks five times every second.

It is also possible to view the Console messages by using applications like TeraTerm. Open the application with the COM related to Crypto Trust Platform with 115200-8-N-1 settings

8. Why AWS demo application is not getting connected to cloud?

There are many possibilities like,

- a. Signer registration is not done to the right account
- b. aws client region is select incorrectly
- c. WiFi credentials are not populated or in correct in C project
- d. aws-iot endpoint is not populated or in-correct in C project

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.
- 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, Helder, JukeBlox, KeeLoq, Klear, 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, KlearNet, KlearNet 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.

© 2018, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN:

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 France - Saint Cloud Tel: 33-1-30-60-70-00 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: 47-7289-7561 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