TrustFLEX Step by Step Guide AWS IoT with Custom PKI

Table of Contents

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.

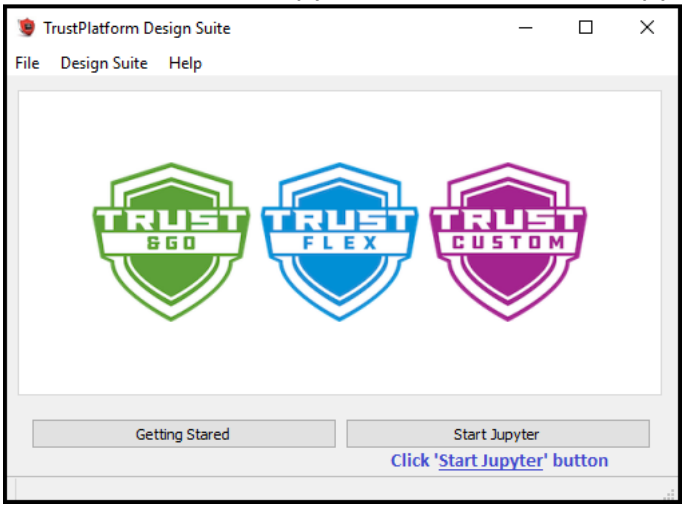
1.1 Getting started with Jupyter Notebook Tutorials

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.

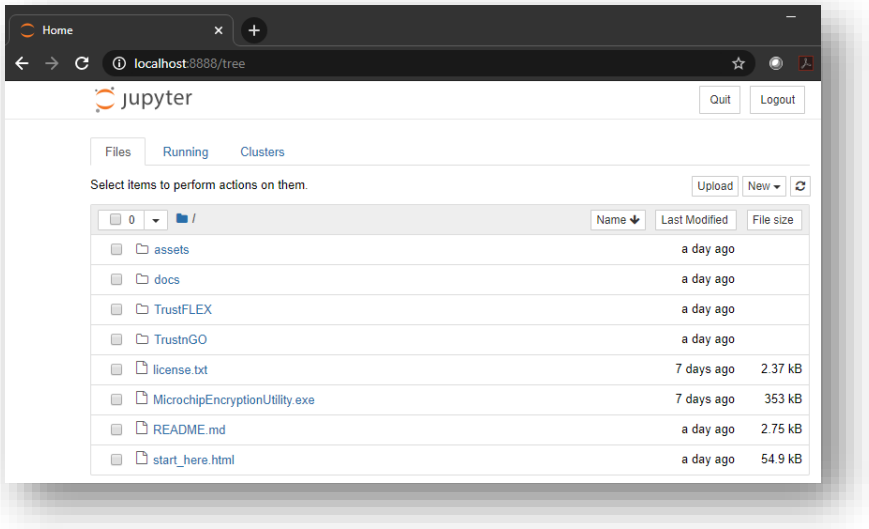
Jupyter notebooks are mainly used to explain/evaluate code in an interactive way.

1.1.1 Starting Jupyter Notebook

Jupyter notebook can be launched from Trust Platform GUI Main window. Run START -> Trust Platform x.x.x icon. Click on 'Start Jupyter' button to launch Jupyter local server.



Clicking on Start Jupyter should be web browser tab like below,



1.2 Jupyter Notebook Basics

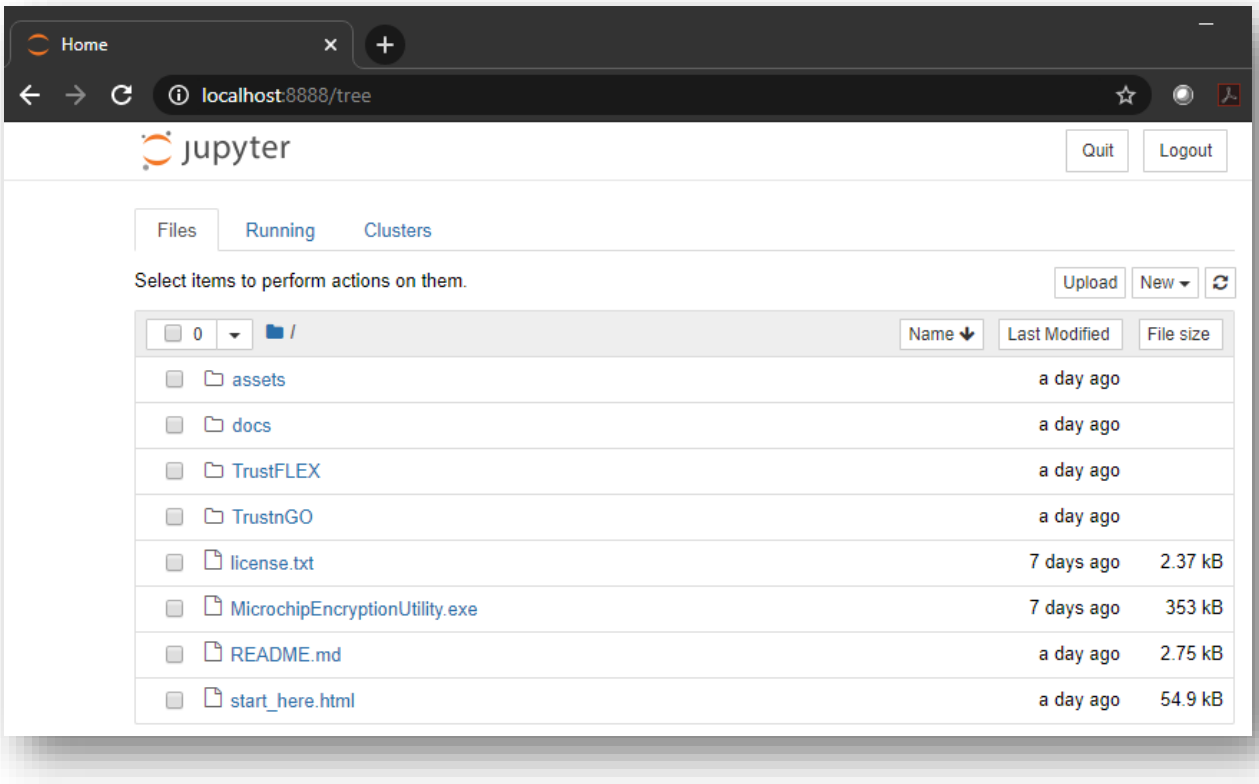
It is recommended to become familiar with Jupyter basic concepts with the online documentation, https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html

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.

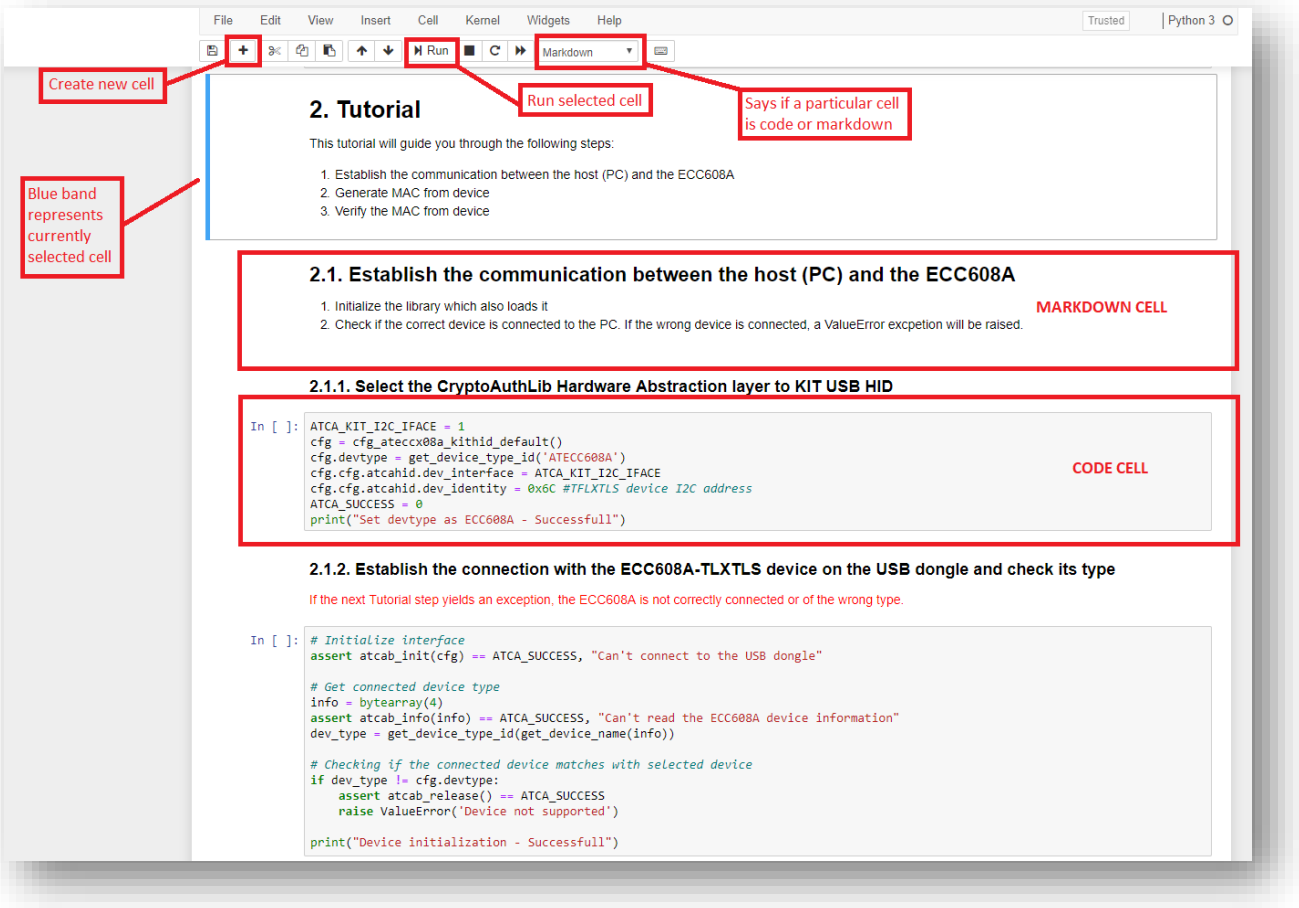


1.3 Introduction to Jupyter Notebook GUI.

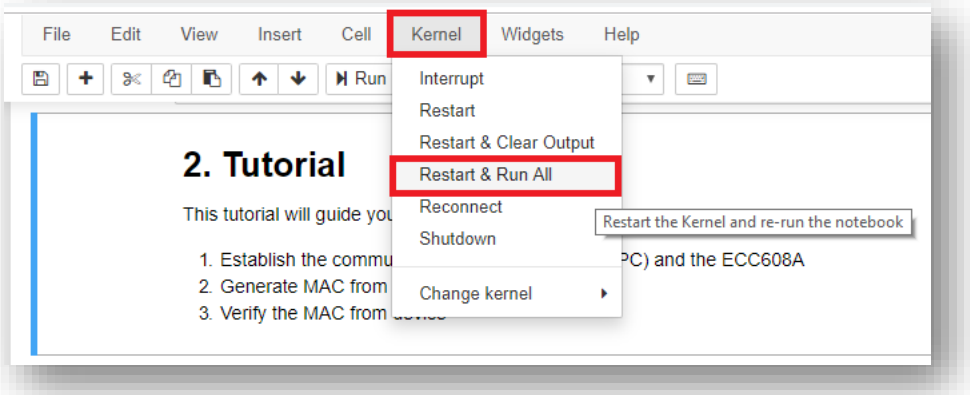
Jupyter Notebooks contain cells where you can either write code or markdown text. Notebooks contain multiple cells, some set as code and others markdown. Code cells contain code that can be executed live, and markdown contains text and images that explains the code.

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 Trust Platform Design Suite comes with several 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	TrustnGO\00_resource_generation\TNGTLS_manifest_file_generation.ipynb	Trust&GO
GCP Connect	TrustnGO\05_cloud_connect\notebook\gcp\TNGTLS_GCP_connect.ipynb	Trust&GO
AWS Connect	TrustnGO\05_cloud_connect\notebook\aws\TNGTLS_aws_connect.ipynb	Trust&GO
Azure Connect	TrustnGO\05_cloud_connect\notebook\azure\ TNGTLS_azure_connect.ipynb	Trust&GO
Resource Generation	TrustFLEX\00_resource_generation\TFLXTLS_resource_generator.ipynb	TrustFLEX
Accessory Authentication	TrustFLEX\01_accessory_authentication\notebook\ TFLXTLS_accessory_authentication.ipynb	TrustFLEX
Firmware Validation	TrustFLEX\02_firmware_validation\notebook\ TFLXTLS_firmware_validation.ipynb	TrustFLEX
IP Protection	TrustFLEX\04_ip_protection\notebook\ TFLXTLS_IP_protection.ipynb	TrustFLEX
Secure Public Key Rotation	TrustFLEX\05_public_key_rotation\notebook\ TFLXTLS_public_key_rotation.ipynb	TrustFLEX
Asymmetric authentication	08_asymmetric_authentication\notebook\ TFLXTLS_asymmetric_authentication.ipynb	TrustFLEX
GCP Connect	TrustFLEX\10_cloud_connect\notebook\gcp\TFLXTLS_GCP_connect.ipynb	TrustFLEX
AWS Custom PKI	TrustFLEX\10_cloud_connect\notebook\aws\ TFLXTLS_aws_connect.ipynb	TrustFLEX
Azure Connect	TrustFLEX\10_cloud_connect\notebook\azure\ TLFXTLS_azure_connect.ipynb	TrustFLEX

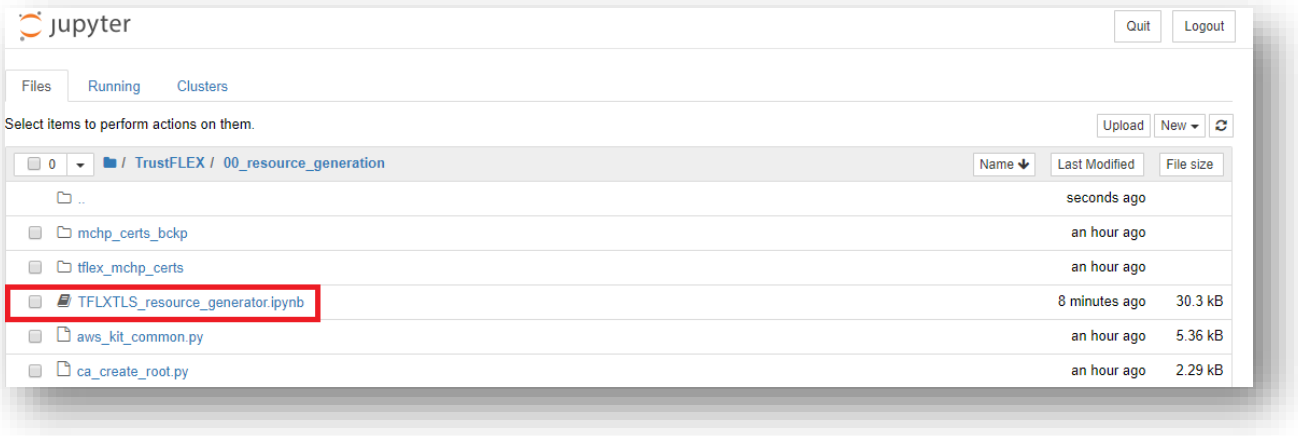
3 Resource Generation Notebook

TFLXTLS device is one of the three devices available in the Crypto Auth Trust Platform 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.

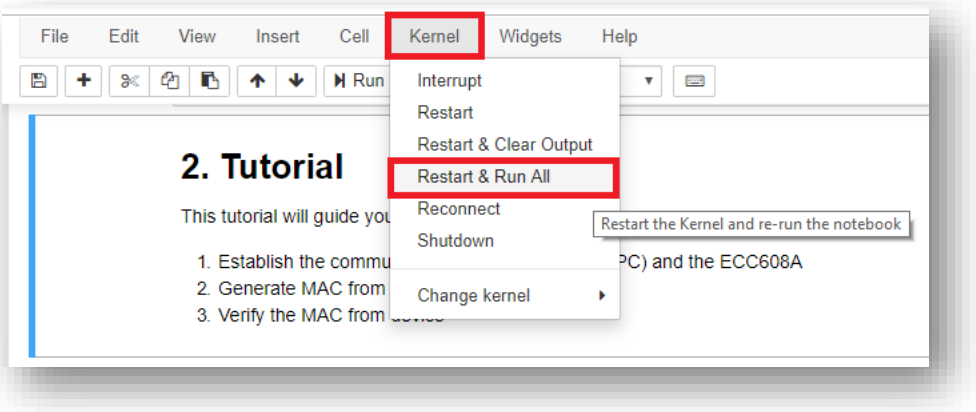
The Resource Generator Notebook will create development keys and certificates for all slots that can be further customized. Keys and Certificate chains are stored in the PC filesystem. These keys should never be used for production purposes as their generation is not handled in a secure environment.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

Within the Jupyter Dashboard, navigate **TrustFLEX\00_resource_generation** folder to open **TFLXTLS_resource_generator.ipynb** notebook



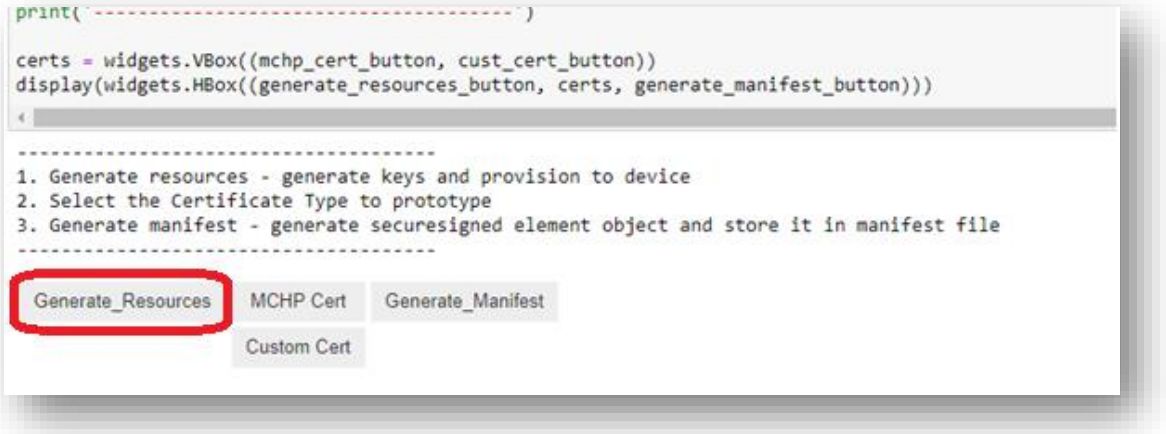
Run all cells of the Crypto Resource Generator Notebook: Kernel->Restart & Run All

Note: Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to Crypto Auth Trust Platform Factory reset section for reloading default program.



Crypto Resource Generator notebook is common for all the use case which comes with option to load the signer certificate and device certificate. The Notebook will generate several keys and certificates. Make sure you have an error free output before continuing to the next steps of the training. Following are 3 different things can be performed,

1. Generating resources to general key slots



The output log should resemble this:

Slot 0 is a private key slot, no action required
Slot 1 is a private key slot, no action required
Slot 2 is a private key slot, no action required
Slot 3 is a private key slot, no action required
Slot 4 is a private key slot, no action required
Slot 6 is a secret key, created slot_6_secret_key.pem and programmed

NOTE: While writing symmetric key into secure element it has to be encrypted with IO prote ction key. So here, Slot 6 (IO protection key) is written before slot 5 (Symmetric key)

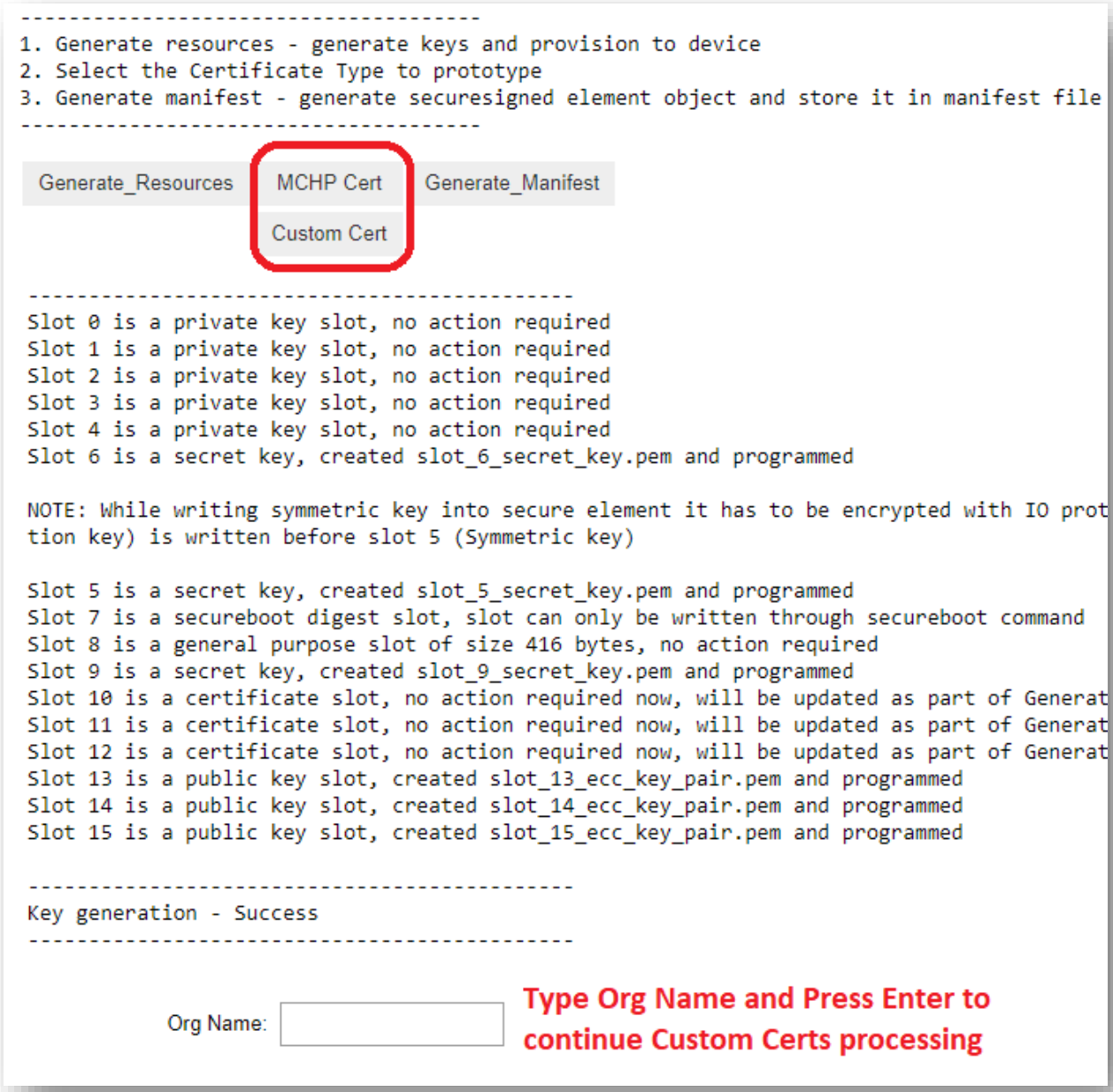
Slot 5 is a secret key, created slot_5_secret_key.pem and programmed

Slot 7 is a secureboot digest slot, slot can only be written through secureboot command
Slot 8 is a general purpose slot of size 416 bytes, no action required
Slot 9 is a secret key, created slot_9_secret_key.pem and programmed
Slot 10 is a certificate slot, no action required now, will be updated as part of Generate Certificates
Slot 11 is a certificate slot, no action required now, will be updated as part of Generate Certificates
Slot 12 is a certificate slot, no action required now, will be updated as part of Generate Certificates
Slot 13 is a public key slot, created slot_13_ecc_key_pair.pem and programmed
Slot 14 is a public key slot, created slot_14_ecc_key_pair.pem and programmed
Slot 15 is a public key slot, created slot_15_ecc_key_pair.pem and programmed

---
Key generation - Success
---

2. Generating MCHP or Custom Certificates

   On selecting Custom certificates, it prompts 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 output log should resemble this:

```
-------------------------------------------
Custom Certs processing...
Device contains custom device and signer certificates
Building new root certificate
Building new signer csr certificate
Building new signer certificate
Read device serial number...OK (SN: 01233E8A1491F2A601)
```

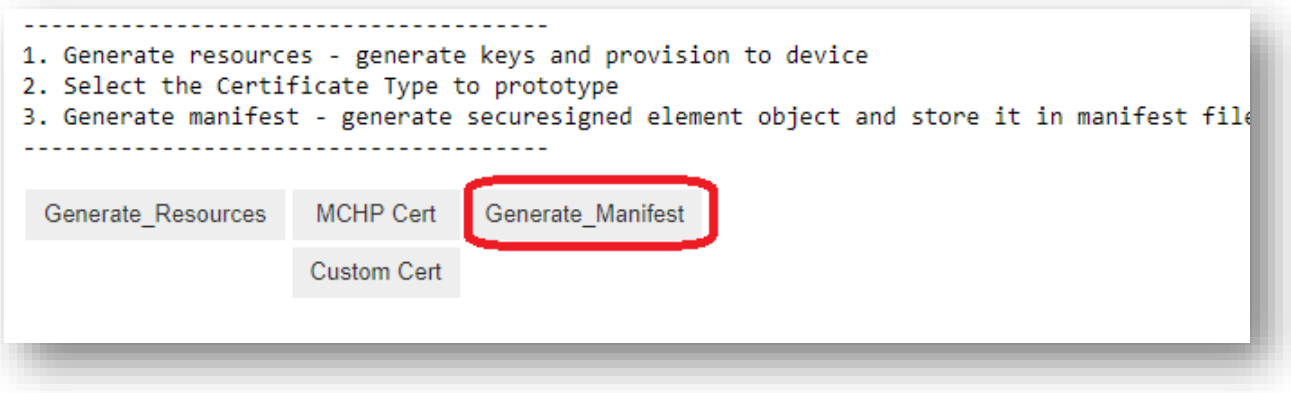
```
Read device public key from slot 0...OK (Public Key: CF1988BC3A6C252026FE70FB34397AD
85A39AE811C722BFA6E5EC1E9CDA9133B3F0E91FD3877F25B8C893B311BAF0203CB5100C
4CDABEBAFDAF3EBD550B00125)
Generating device certificate...OK (saved to device_01233E8A1491F2A601.crt)
Saving signer certificate to device...OK
Saving device certificate to device...OK
Thing ID eabc56113c70227a18c0a62f7c285fc68d75f9cd

-------------------------------------------
Custom certificate generation and provisioning - SUCCESS
-------------------------------------------

Validate root certificate...OK
-----BEGIN CERTIFICATE-----
MIIByjCCAW+gAwIBAgIQeoueybRh8XWwzOkoixtW1jAKBggqhkjOPQQDAjA7MQ0w
CwYDVQQKDAR0ZXN0MSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gUm9v
dCBDQSAwMDIwIBcNMjAwNzAxMDgwNTE5WhgPMjA2MDA2MjEwODA1MTlaMDsxDTAL
BgNVBAoMBHRlc3QxKjAoBgNVBAMMIUNyeXB0byBBdXRoZW50aWNhdGlvbiBSb290
IENBIDAwMjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABFf6qcSyPv8iY0uccoTX
SlSstaz0ECCUxXUoqky8Xo40vsOCbPPt5QtlvNHnyy8tAbwza6DsAiz2sGLzDI5h
QhqjUzBRMB0GA1UdDgQWBBRHVPQoljiq65JOG4vu5l32JzmkSTAfBgNVHSMEGDAW
gBRHVPQoljiq65JOG4vu5l32JzmkSTAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49
BAMCA0kAMEYCIQCB7FKx5K33xK9E0PsWGKZRaaQxxSRypC66y4hVqWVmMAIhAMlG
22zNUKPHccHQxfQssYH5LfR5SVE+WC3Hyxem/EVj
-----END CERTIFICATE-----

Certificate:
   Data:
      Version: 3 (0x2)
      Serial Number:
         7a:8b:9e:c9:b4:61:f1:75:b0:cc:e9:28:8b:1b:56:d6
      Signature Algorithm: ecdsa-with-SHA256
      Issuer: O=test, CN=Crypto Authentication Root CA 002
      Validity
         Not Before: Jul  1 08:05:19 2020 GMT
         Not After : Jun 21 08:05:19 2060 GMT
      Subject: O=test, CN=Crypto Authentication Root CA 002
      Subject Public Key Info:
         Public Key Algorithm: id-ecPublicKey
            Public-Key: (256 bit)
            pub:
               04:57:fa:a9:c4:b2:3e:ff:22:63:4b:9c:72:84:d7:
               4a:54:ac:b5:ac:f4:10:20:94:c5:75:28:aa:4c:bc:
               5e:8e:34:be:c3:82:6c:f3:ed:e5:0b:65:bc:d1:e7:
```

```
            cb:2f:2d:01:bc:33:6b:a0:ec:02:2c:f6:b0:62:f3:
            0c:8e:61:42:1a
          ASN1 OID: prime256v1
          NIST CURVE: P-256
      X509v3 extensions:
        X509v3 Subject Key Identifier:
          47:54:F4:28:96:38:AA:EB:92:4E:1B:8B:EE:E6:5D:F6:27:39:A4:49
        X509v3 Authority Key Identifier:
          keyid:47:54:F4:28:96:38:AA:EB:92:4E:1B:8B:EE:E6:5D:F6:27:39:A4:49

        X509v3 Basic Constraints: critical
          CA:TRUE
  Signature Algorithm: ecdsa-with-SHA256
      30:46:02:21:00:81:ec:52:b1:e4:ad:f7:c4:af:44:d0:fb:16:
      18:a6:51:69:a4:31:c5:24:72:a4:2e:ba:cb:88:55:a9:65:66:
      30:02:21:00:c9:46:db:6c:cd:50:a3:c7:71:c1:d0:c5:f4:2c:
      b1:81:f9:2d:f4:79:49:51:3e:58:2d:c7:cb:17:a6:fc:45:63


Validate signer certificate...OK
-----BEGIN CERTIFICATE-----
MIIB3TCCAYKgAwIBAgIQV/RpeXxWfquIIYFCFTDc/TAKBggqhkjOPQQDAjA7MQ0w
CwYDVQQKDAR0ZXN0MSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gUm9v
dCBDQSAwMDIwIBcNMjAwNzAxMDgwMDAwWhgPMjA0MDA3MDEwODAwMDBaMDsxDTAL
BgNVBAoMBHRlc3QxKjAoBgNVBAMMIUNyeXB0byBBdXRoZW50aWNhdGlvbiBTaWdu
ZXIgRkZGRjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABCEubbOfXDakettxvfKu
kfG5UhQNDHrPrZiURytSZmQ8p38VacZ682akSAC6XQYDzhiy5/5O4eAHBCuN5rOt
vnOjZjBkMA4GA1UdDwEB/wQEAwIBhjASBgNVHRMBAf8ECDAGAQH/AgEAMB0GA1Ud
DgQWBBRycA/sc+NWXwp0wLudepyPtQtzFzAfBgNVHSMEGDAWgBRHVPQoljiq65JO
G4vu5l32JzmkSTAKBggqhkjOPQQDAgNJADBGAiEA1ThacjiYboKYh69+NllQKiX2
wb7Jztq8zMsY61H/NkYCIQDQc2TQfOl9HBDUoDzUtTZNgIksElkU7ysiSgBhumAA
zQ==
-----END CERTIFICATE-----

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      57:f4:69:79:7c:56:7e:ab:88:21:81:42:15:30:dc:fd
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: O=test, CN=Crypto Authentication Root CA 002
    Validity
      Not Before: Jul  1 08:00:00 2020 GMT
      Not After : Jul  1 08:00:00 2040 GMT
```

```
        Subject: O=test, CN=Crypto Authentication Signer FFFF
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:21:2e:6d:b3:9f:5c:36:a4:7a:db:71:bd:f2:ae:
                    91:f1:b9:52:14:0d:0c:7a:cf:ad:98:94:47:2b:52:
                    66:64:3c:a7:7f:15:69:c6:7a:f3:66:a4:48:00:ba:
                    5d:06:03:ce:18:b2:e7:fe:4e:e1:e0:07:04:2b:8d:
                    e6:b3:ad:be:73
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Subject Key Identifier:
                72:70:0F:EC:73:E3:56:5F:0A:74:C0:BB:9D:7A:9C:8F:B5:0B:73:17
            X509v3 Authority Key Identifier:
                keyid:47:54:F4:28:96:38:AA:EB:92:4E:1B:8B:EE:E6:5D:F6:27:39:A4:49

    Signature Algorithm: ecdsa-with-SHA256
        30:46:02:21:00:d5:38:5a:72:38:98:6e:82:98:87:af:7e:36:
        59:50:2a:25:f6:c1:be:c9:ce:da:bc:cc:cb:18:eb:51:ff:36:
        46:02:21:00:d0:73:64:d0:7c:e9:7d:1c:10:d4:a0:3c:d4:b5:
        36:4d:80:89:2c:12:59:14:ef:2b:22:4a:00:61:ba:60:00:cd


Validate device certificate...OK
-----BEGIN CERTIFICATE-----
MIIByDCCAW+gAwIBAgIQdxkpBswUT+e4dShL6tp00jAKBggqhkjOPQQDAjA7MQ0w
CwYDVQQKDAR0ZXN0MSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gU2ln
bmVyIEZGRkYwIBcNMjAwNzAxMDYwMDAwWhgPMjA0ODA3MDEwNjAwMDBaMC4xDTAL
BgNVBAoMBHRlc3QxHTAbBgNVBAMMFHNuMDEyMzNFOEExNDkxRjJBNjAxMFkwEwYH
KoZIzj0CAQYIKoZIzj0DAQcDQgAEzxmIvDpsJSAm/nD7NDl62Fo5roEcciv6bl7B
6c2pEzs/DpH9OHfyW4yJOzEbrwIDy1EAxM2r66/a8+vVULABJaNgMF4wDAYDVR0T
AQH/BAIwADAOBgNVHQ8BAf8EBAMCA4gwHQYDVR0OBBYEFOq8VhE8cCJ6GMCmL3wo
X8aNdfnNMB8GA1UdIwQYMBaAFHJwD+xz41ZfCnTAu516nI+1C3MXMAoGCCqGSM49
BAMCA0cAMEQCIANn/QrqxwmwrRsrcYyQpWJ0o4AxLzGoeCZjfJ5o0FAbAiBFne67
iEzuh6dqwrdQYvgB6+qTxfLyei1kwoFcfVnHvA==
-----END CERTIFICATE-----

Certificate:
```

```
Data:
    Version: 3 (0x2)
    Serial Number:
        77:19:29:06:cc:14:4f:e7:b8:75:28:4b:ea:da:74:d2
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: O=test, CN=Crypto Authentication Signer FFFF
    Validity
        Not Before: Jul  1 06:00:00 2020 GMT
        Not After : Jul  1 06:00:00 2048 GMT
    Subject: O=test, CN=sn01233E8A1491F2A601
    Subject Public Key Info:
        Public Key Algorithm: id-ecPublicKey
            Public-Key: (256 bit)
            pub:
                04:cf:19:88:bc:3a:6c:25:20:26:fe:70:fb:34:39:
                7a:d8:5a:39:ae:81:1c:72:2b:fa:6e:5e:c1:e9:cd:
                a9:13:3b:3f:0e:91:fd:38:77:f2:5b:8c:89:3b:31:
                1b:af:02:03:cb:51:00:c4:cd:ab:eb:af:da:f3:eb:
                d5:50:b0:01:25
            ASN1 OID: prime256v1
            NIST CURVE: P-256
    X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:FALSE
        X509v3 Key Usage: critical
            Digital Signature, Key Agreement
        X509v3 Subject Key Identifier:
            EA:BC:56:11:3C:70:22:7A:18:C0:A6:2F:7C:28:5F:C6:8D:75:F9:CD
        X509v3 Authority Key Identifier:
            keyid:72:70:0F:EC:73:E3:56:5F:0A:74:C0:BB:9D:7A:9C:8F:B5:0B:73:17

Signature Algorithm: ecdsa-with-SHA256
     30:44:02:20:03:67:fd:0a:ea:c7:09:b0:ad:1b:2b:71:8c:90:
     a5:62:74:a3:80:31:2f:31:a8:78:26:63:7c:9e:68:d0:50:1b:
     02:20:45:9d:ee:bb:88:4c:ee:87:a7:6a:c2:b7:50:62:f8:01:
     eb:ea:93:c5:f2:f2:7a:2d:64:c2:81:5c:7d:59:c7:bc
```

3. Generating Manifest file



The output log should resemble this:

```
------------------------------------------------------
Generating manifest data...OK (saved to TFLXTLS_devices_manifest.json)
------------------------------------------------------
```

At the end of the execution, a Custom PKI chain will be generated on your PC and TrustFLEX device specific slots (10 through 12) will be overwritten with the custom certificates.

The Notebook has also generated a manifest file to be uploaded into the public cloud of your choice (Google GCP, AWS IoT and Microsoft Azure).

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:

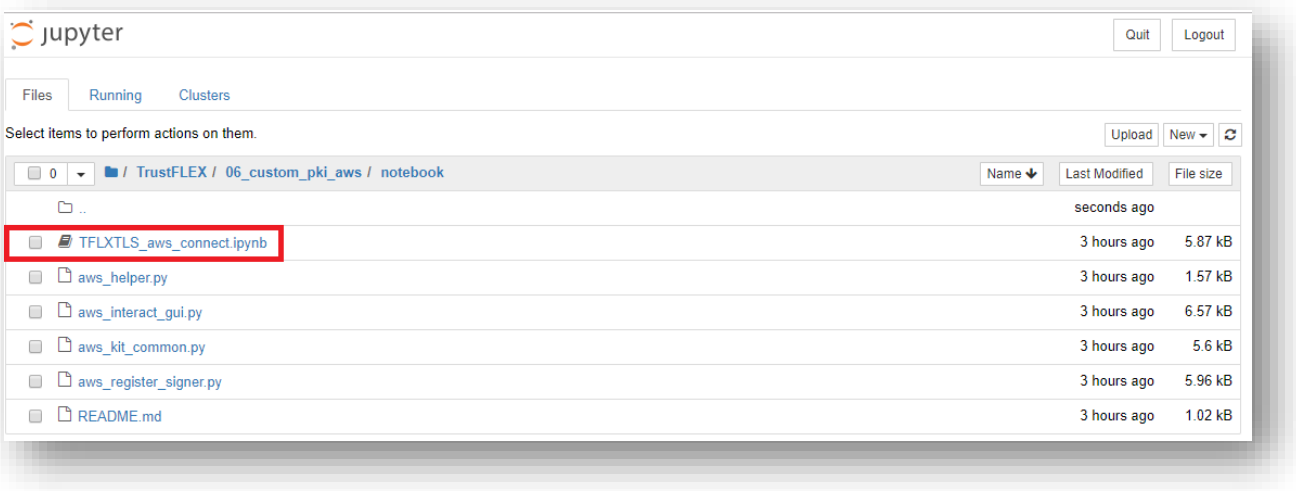
- Configure AWS CLI
- Register Custom PKI signer
- Build the AWS IoT device source code and flash it to the Crypto Auth Trust Platform board

Note: It is required to have an AWS IoT test account setup. Instruction to setup the account quickly is provided in **docs\TrustFLEX AWS demo account setup instructions.pdf**.

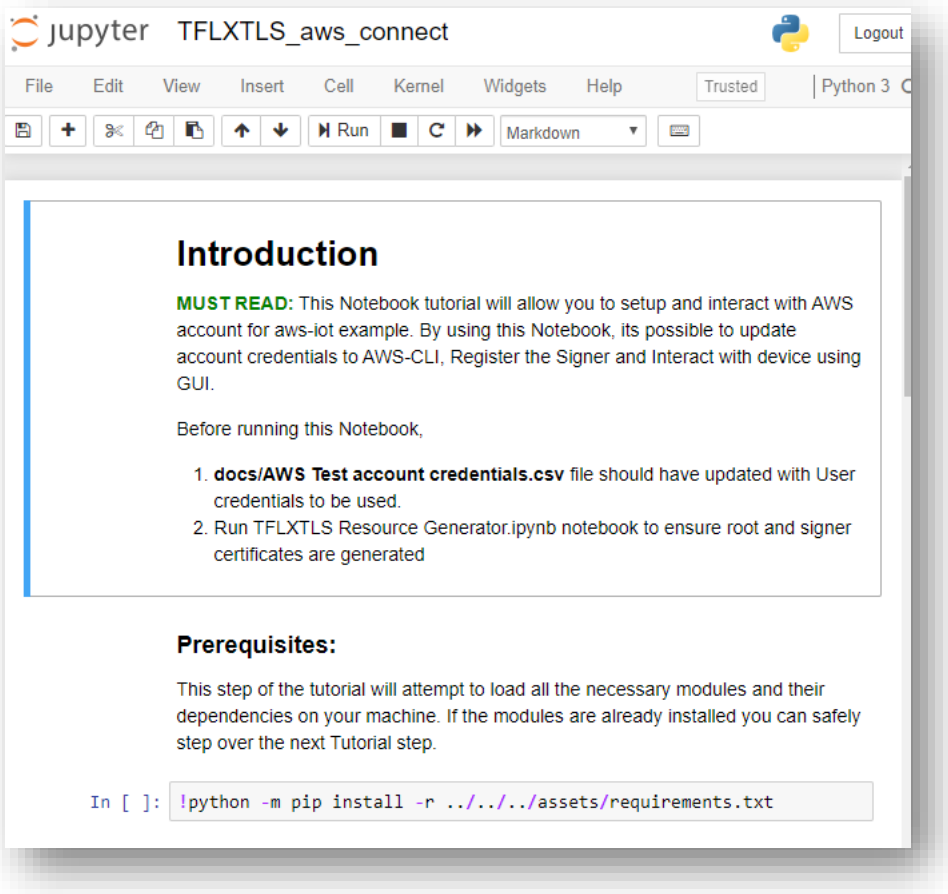
4.1 Running Custom PKI example on Jupyter Notebook

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.

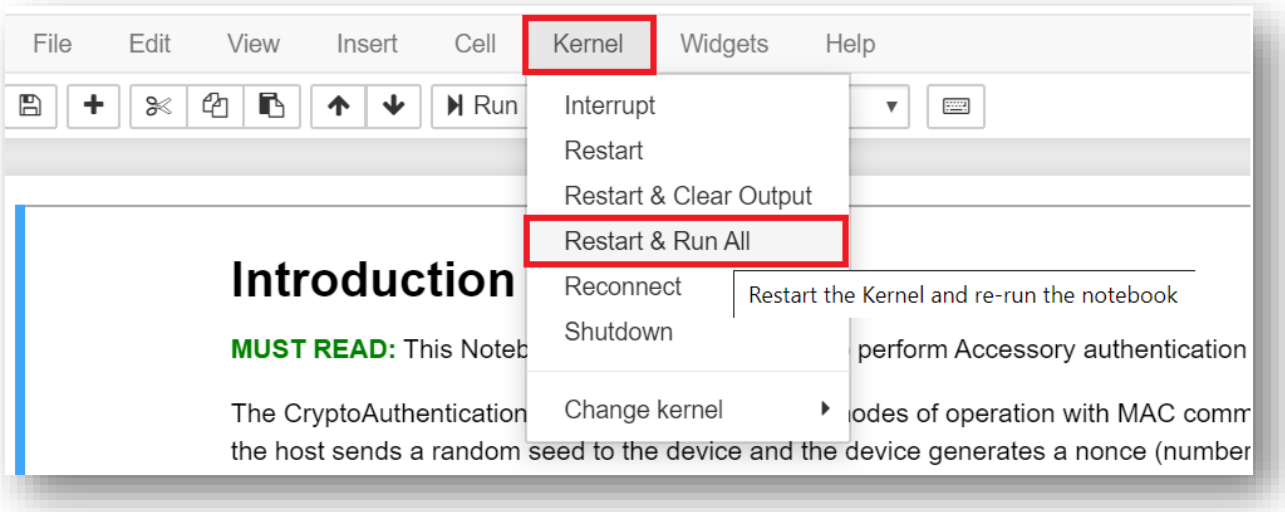
1. From the Jupyter Home page, navigate to **TrustFLEX\10_cloud_connect\notebook\aws\TFLXTLS_aws_connect.ipynb** notebook file and open it.



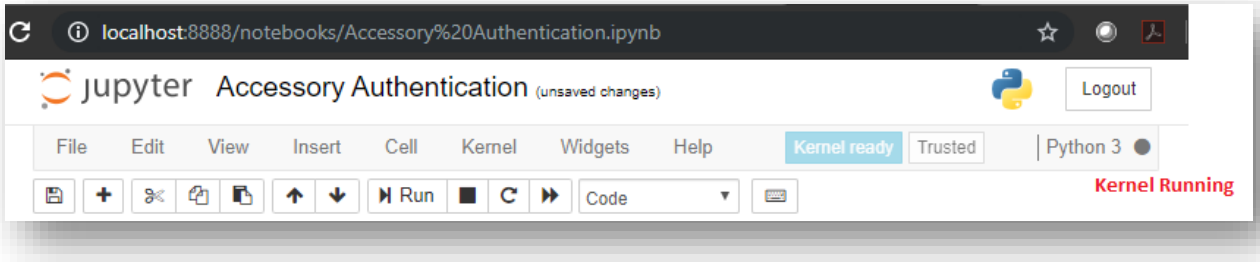
Opening the Jupyter notebook example should load the following on the browser.

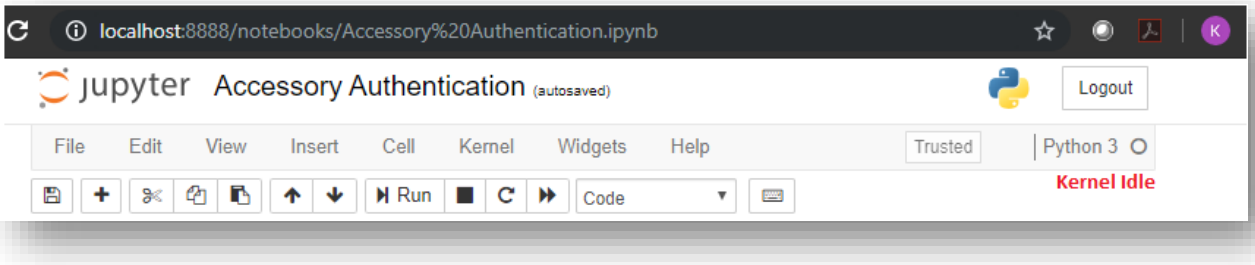


2. Run All Cells by using Kernel -> Restart & Run All



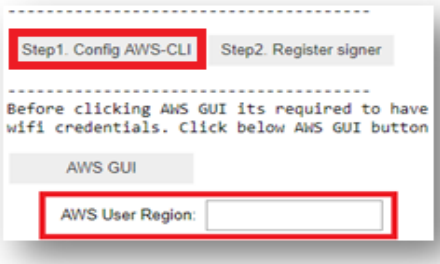
It may take a while to complete, wait for the kernel to complete all processing i.e. from Kernel Running to Kernel Idle state (Check circle above **RED** text)



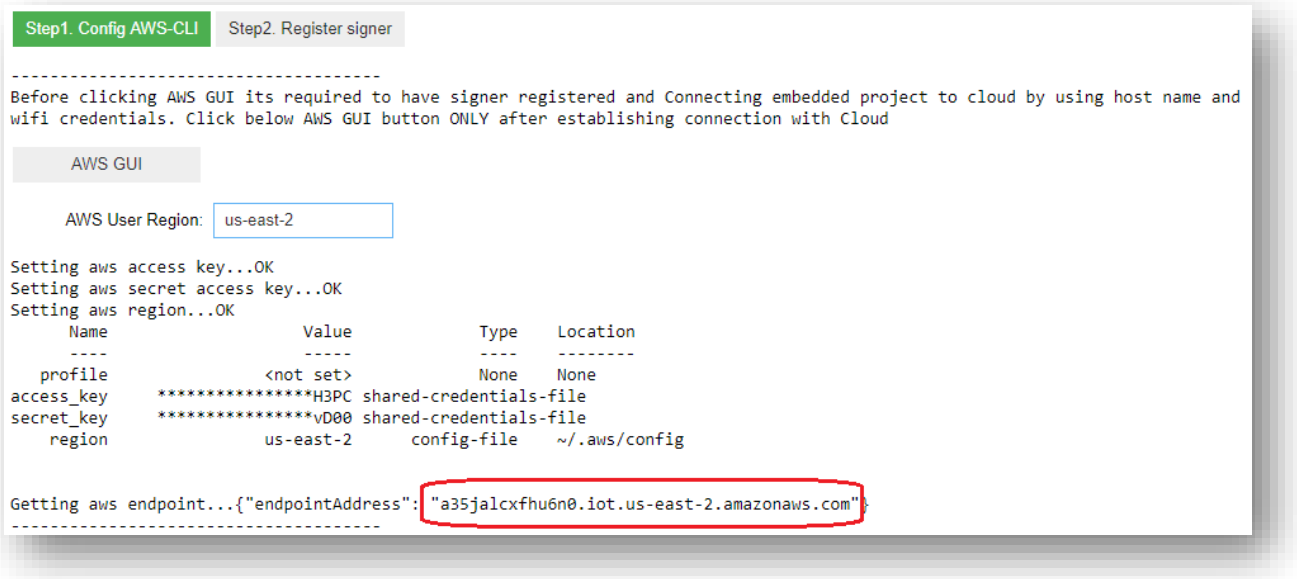


3. Navigate through different cells output for the description of the step and result from the execution.

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**. On clicking the config AWS-CLI button, it will prompt to enter your region.
   Enter your region:
   Below screenshot display the option to enter your region



Below screenshot display the result of config AWS-CLI,



On successful aws configure, it will fetch the AWS host endpoint and display along with the aws config details. Marked URL is the aws host endpoint to which device must connect. This endpoint will be added to device's embedded c project in next section.

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.

```
Reading signer CA key file, signer_FFFF.key

Reading signer CA certificate file, signer_FFFF.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: 28295a5f232009d9aa3bc8f2a871d888745fa23119b4c5fa6e564a2b560f5e8b

Generating signer CA AWS verification certificate
    Saved to E:\Desktop\TrustPlatform\tp_designsuite\TrustFLEX\00_resource_generation\signer_FFFF-verification.crt

Registering signer CA with AWS IoT
    ID: a2c410258fa77b1ee5f292a396093e006654ac6e271d37f47df294c4d88eeba5

Getting AWS IoT device endpoint
    Hostname: a35jalcxfhu6n0.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.

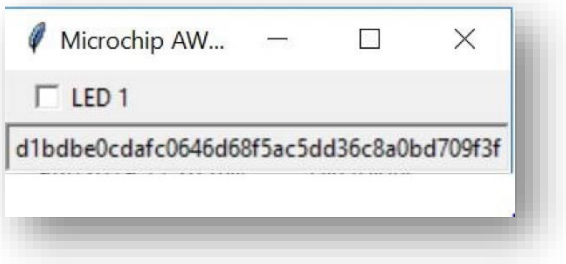
NOTE: Make sure that you executed C project successfully before executing the next step in the Jupyter notebook. To execute C project, refer "Running AWS IoT example on Embedded platform" next section.

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 Crypto Auth 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.

4.2 Running AWS IoT example on Embedded platform

Once the resources are generated MPLAB projects provided can be used to run the use case on Crypto Auth Trust Platform.

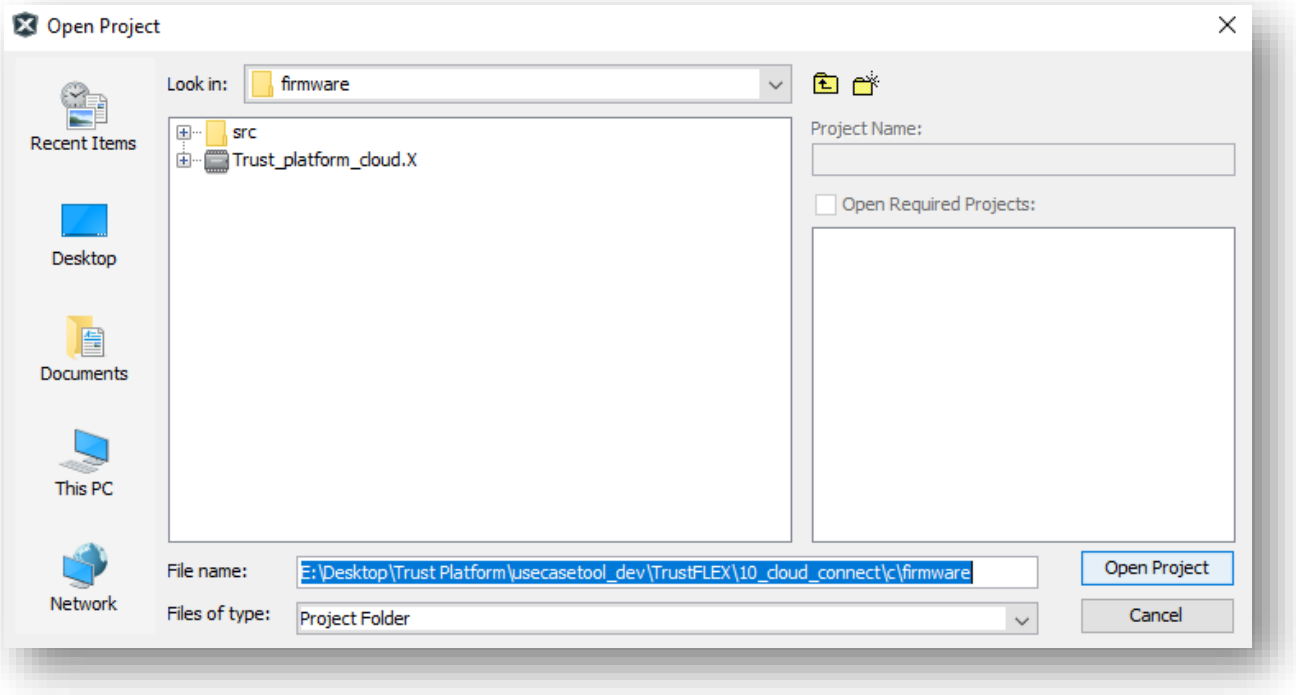
This project can configure the Wi-Fi credentials, establish a TLS connection, subscribe to MQTT and register device certificate but not register the signer module to AWS IoT. It is required to use the AWS IoT Jupyter notebook to register the signer module and get the AWS endpoint to which device must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

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

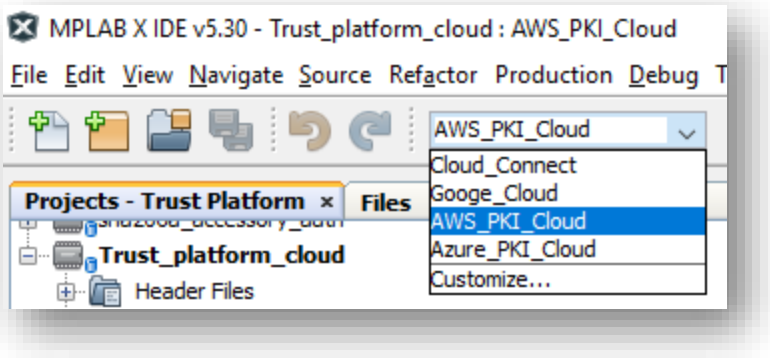
Prerequisite: It is required that WINC firmware is updated to latest version / version that is available in this package. Update the WINC firmware using package available in cloned repository at **assets\winc_firmware_upgrade**

4.2.1 MPLAB:

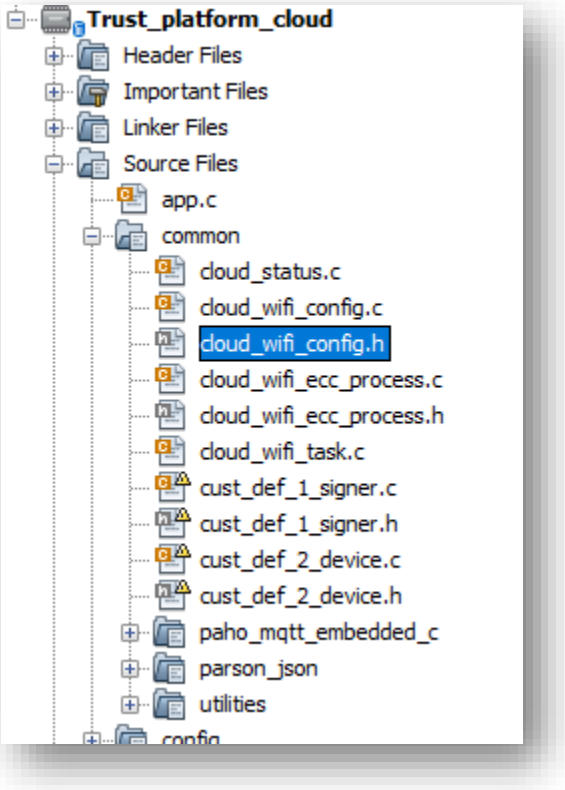
1. Open **Trust_platform_cloud.X** project by navigating to MPLAB -> File -> Open Project -> **TrustFLEX\10_cloud_connect\firmware**



1. Select Build configuration as AWS_PKI_Cloud



2. Open **cloud_wifi_config.h** file by navigating to **Trust_platform_cloud -> Source Files**



update the following constants before building the project:

- MAIN_WLAN_SSID
- MAIN_WLAN_PSK
- CLOUD_ENDPOINT

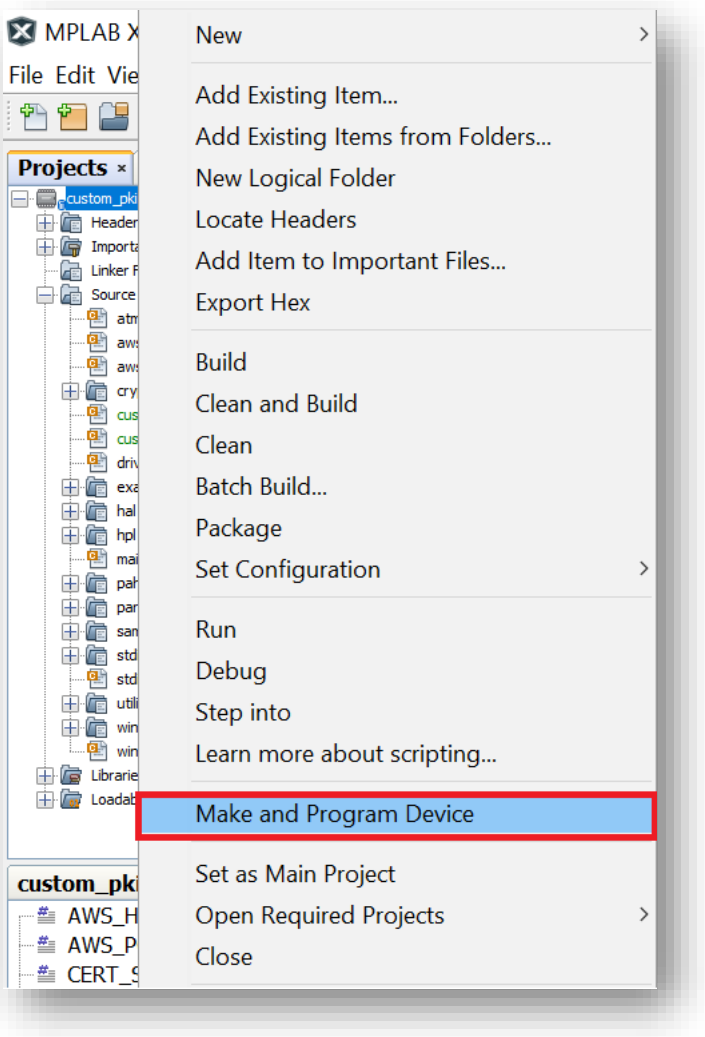
```
#define WLAN_SSID           "xxxxxxxxxxxx"
#define WLAN_AUTH_WPA_PSK
#define WLAN_PSK            "xxxxxxxxxxxx"
```

The CLOUD_ENDPOINT string should be set to the value reported during Config AWS-CLI step in the Usecase notebook.

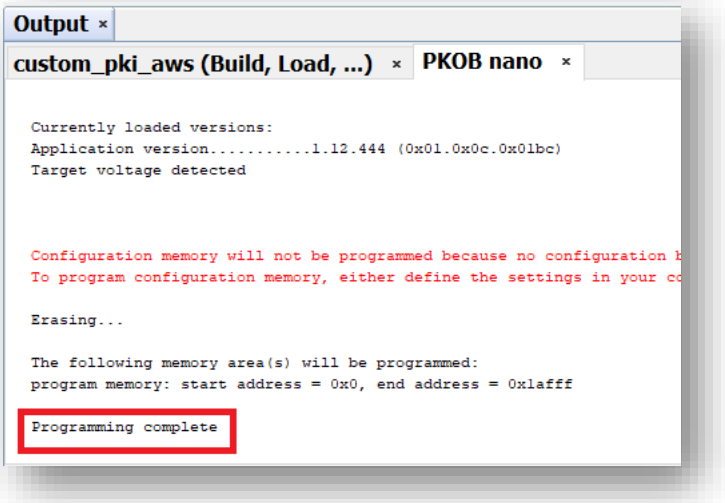
```
#elif defined(CLOUD_CONFIG_AWS)
    #define SSL_CIPHER_SUITE_SELECTION          SSL_ECC_ONLY_CIPHERS
    #define CLOUD_ENDPOINT                      "xxxxxxxxxxxx.iot.xxxxxxx.amazonaws.com"
```

For ex:
#define CLOUD_ENDPOINT "a35jalcxfhu6n0.iot.us-west-2.amazonaws.com"

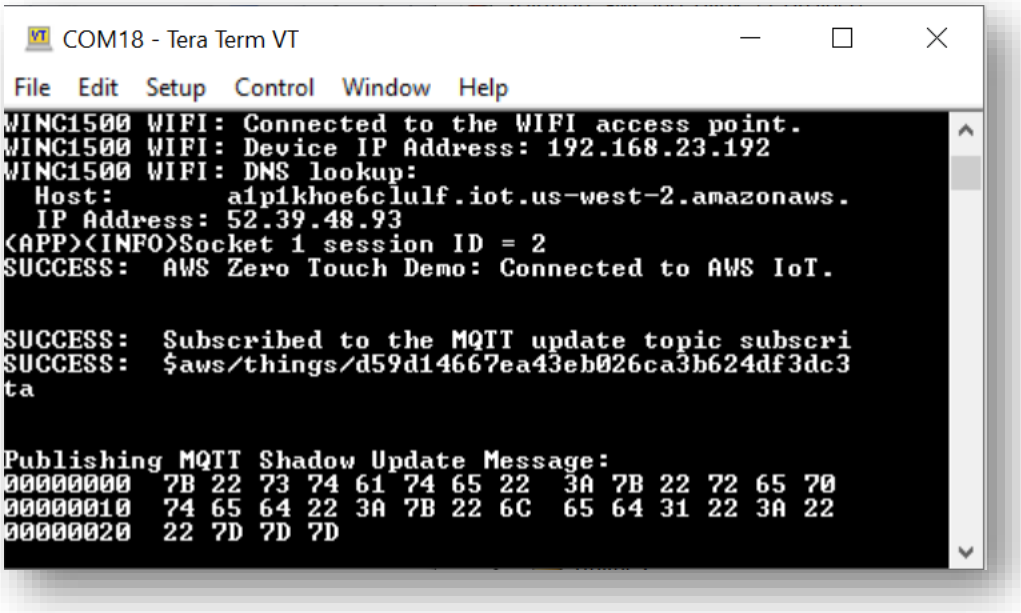
2. Program the Crypto Auth Trust platform by navigating to **Trust_platform_cloud -> Make and Program Device**



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 Crypto Auth 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 Crypto Auth Trust Platform Factory reset

Once any of the embedded project is loaded to Crypto Auth Trust Platform, the default program that enables interaction with Trust Platform tools will be erased.

Before using the Platform with any other notebook or tools on PC, its required to reprogram the default .hex file. Default hex file is available at
assets\Factory_Program.X\CryptoAuth_Trust_Platform.hex

If Trust Platform GUI is provided with MPLAB X IDE installation location, notebooks can program the Factory reset hex file if its not available by default.

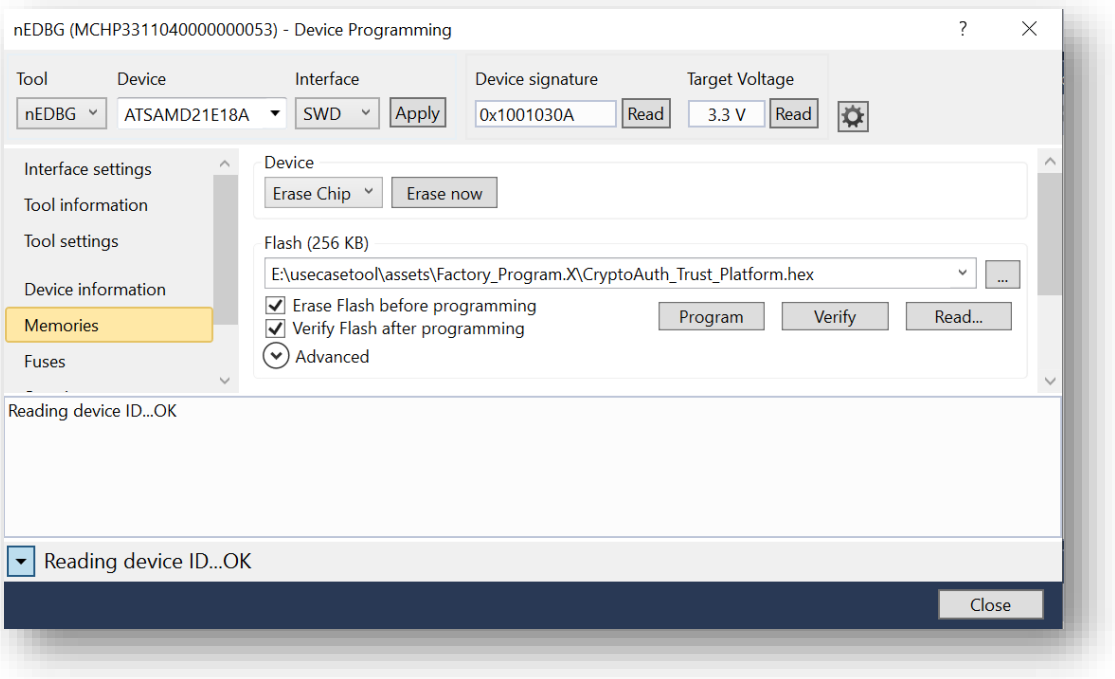
This can also be done manually by MPLAB and Atmel Studio

To reprogram using MPLAB:

1. Open **assets\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**

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



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

5 FAQ

1. **What are the reasons for "AssertionError: Cannot connect to CryptoAuth Trust Platform, check USB connection" error?**
   There are many possibilities like,
   1. Crypto Trust Platform is having different application than factory reset firmware. Refer to "Crypto Auth Trust Platform 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 "Crypto Auth Trust Platform Factory reset" section any usecase TrustFLEX Guide for reloading it.

3. **Why does my C projects generates No such file or directory with ../../ 00_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.
   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

Trademarks

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

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

ASIA/PACIFIC

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

ASIA/PACIFIC

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

EUROPE

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