TrustFLEX Step by Step Guide Google Cloud Platform Connect

Table of Contents

1 Introduction

This document gives a detailed walk through of connecting securely to Google Could Platform. If familiar with Jupyter Notebook, can skip this section and move to Section 2.

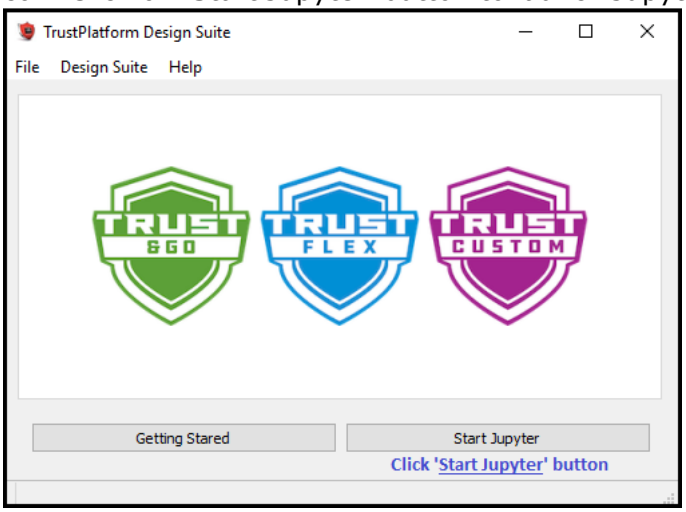
1.1 Getting started with Jupyter Notebook Tutorials

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.

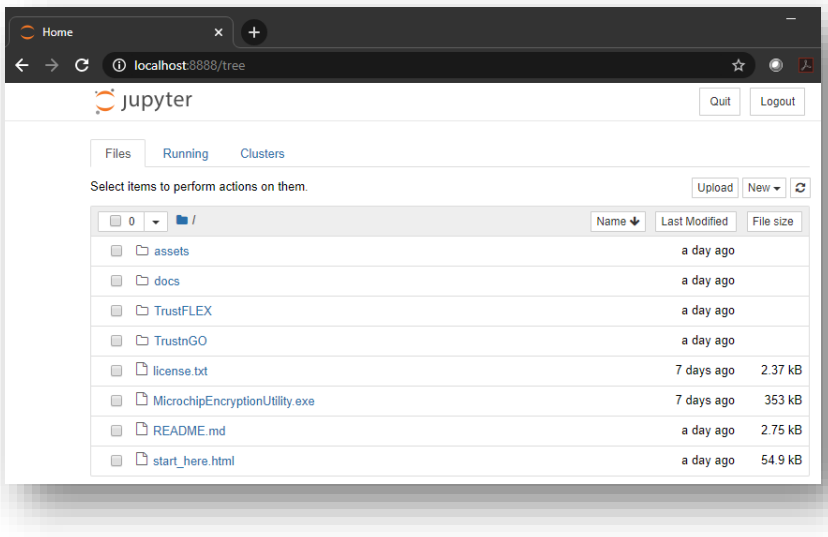
Jupyter notebooks are mainly used to explain/evaluate code in an interactive way.

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

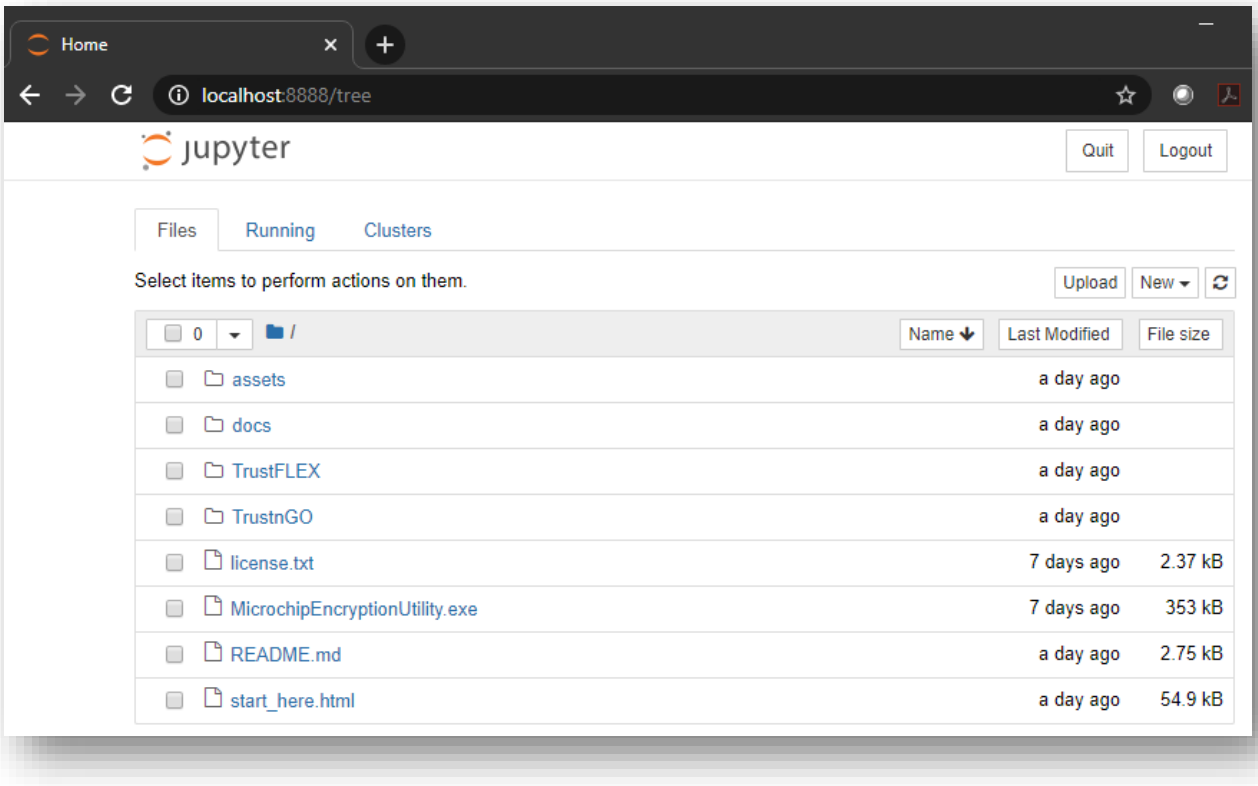
It is recommended to become familiar with Jupyter basic concepts with the online documentation, https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html

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.

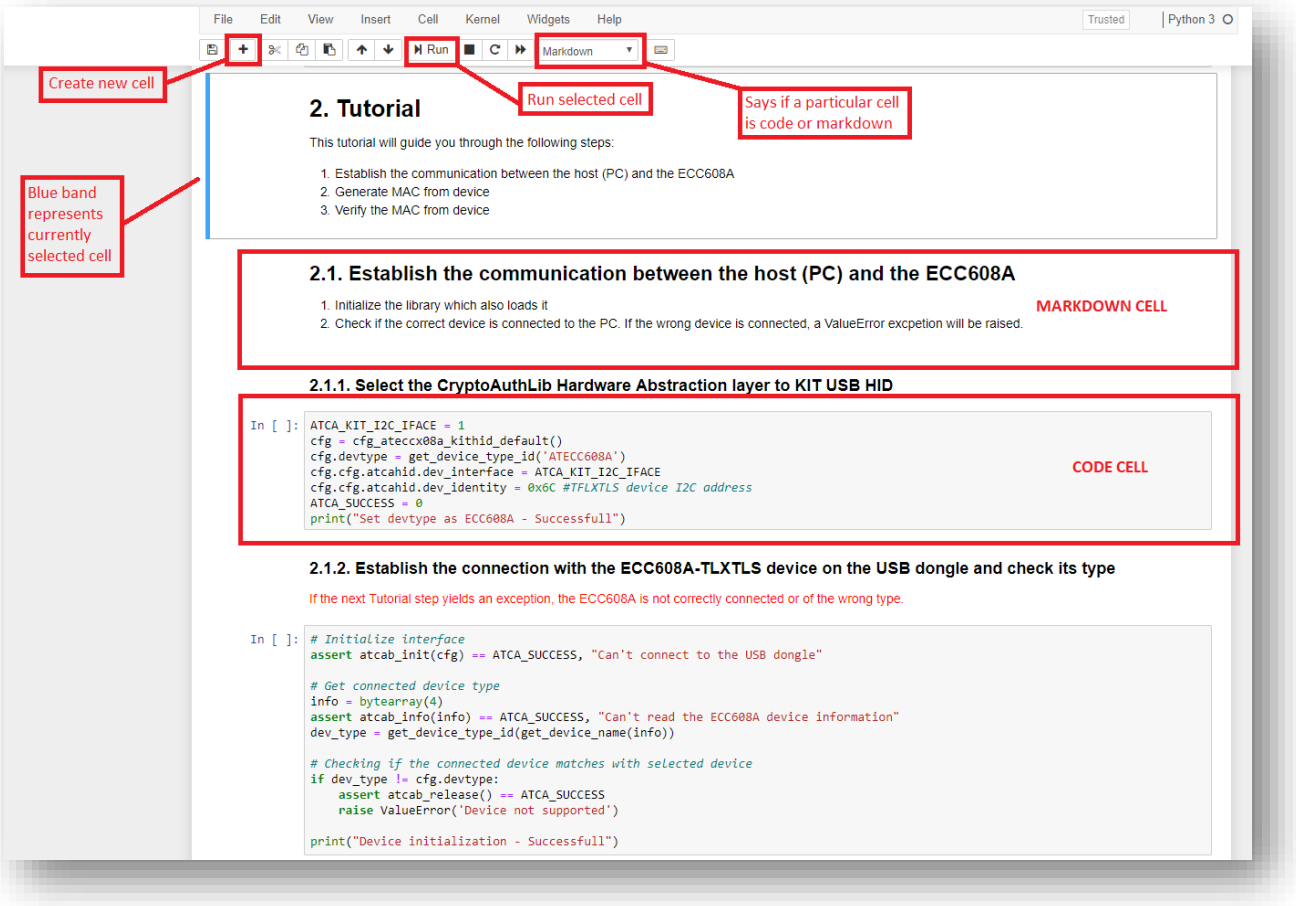


1.3 Introduction to Jupyter Notebook GUI.

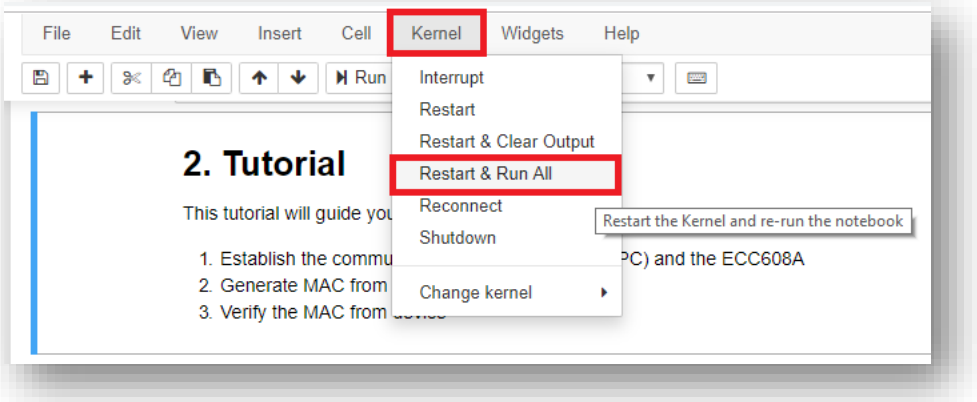
Jupyter Notebooks contain cells where you can either write code or markdown text. Notebooks contain multiple cells, some set as code and others markdown. Code cells contain code that can be executed live, and markdown contains text and images that explains the code.

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 TrustPlatform 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	TrustnGO
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
GCP Connect	TrustFLEX\03_gcp_connect\notebook\TFLXTLS_GCP_connect.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
AWS Custom PKI	TrustFLEX\06_custom_pki_aws\notebook\ TFLXTLS_aws_connect.ipynb	TrustFLEX
Azure Connect	TrustFLEX\07_custom_pki_azure\notebook\ TLFXTLS_azure_connect.ipynb	TrustFLEX

3 Generate Manifest files

In the real scenarios, the Manifest files for Trust&GO and TrustFLEX should be downloaded from microchipDirect. Once devices have shipped, you will be able to download the Manifest file from your Microchip Purchasing & Client Services Account. The file can then be uploaded into your cloud service account.

Kits, demonstration boards do not ship with a Manifest file.

The following sections provide steps to generate manifest files for Trust&GO and TrustFLEX devices during prototyping the Usecases.

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.

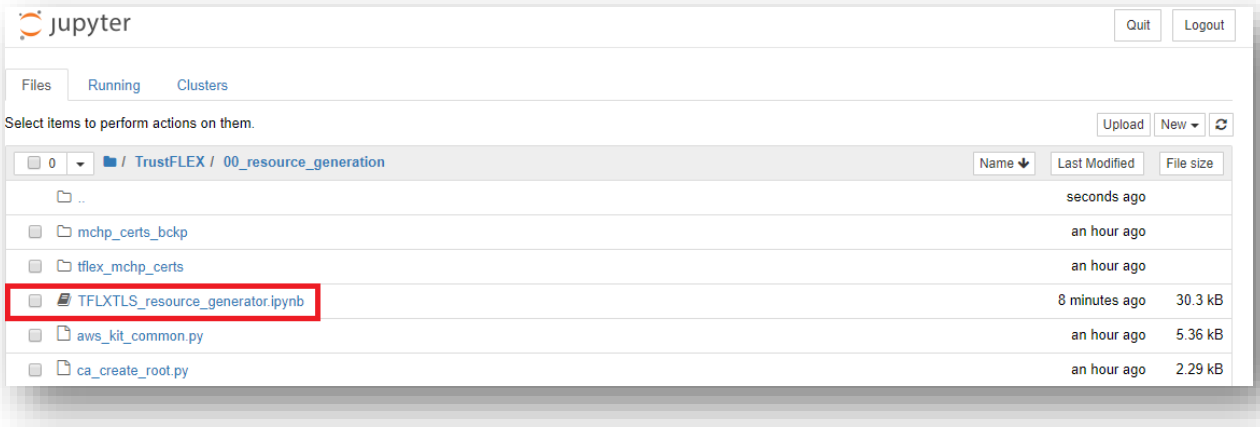
3.1 TrustFLEX – Manifest file generation

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.

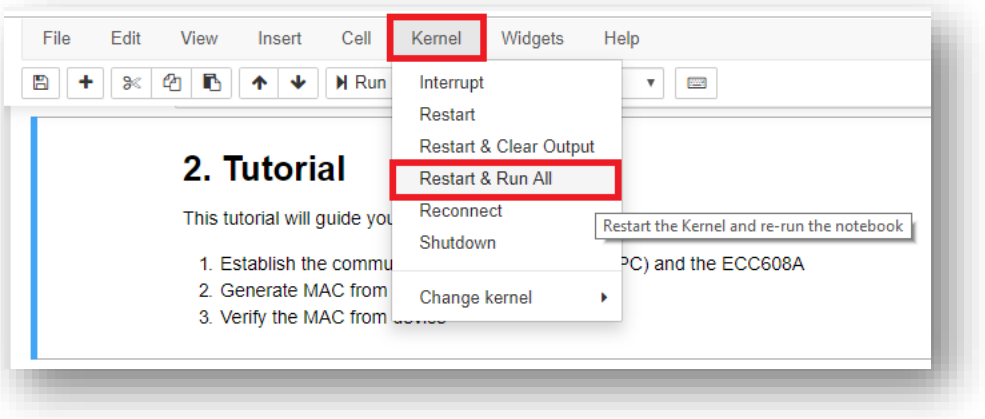
The Resource Generator Notebook will create development keys and certificates for all slots that can be further customized. Keys and Certificate chains are stored in the PC filesystem. These keys should never be used for production purposes as their generation is not handled in a secure environment.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

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

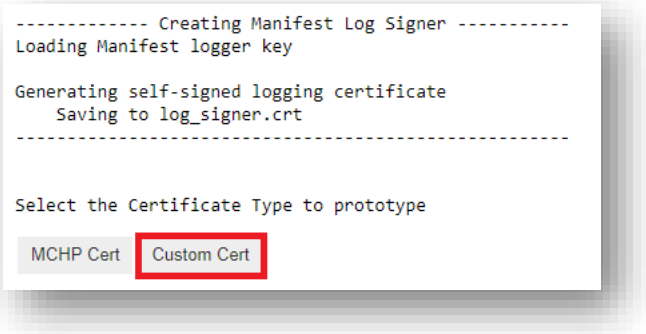


Run all cells of the Crypto Resource Generator Notebook: Kernel->Restart & Run All

Note: Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to 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, press "Custom Cert" option for this use case.



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.pem

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 c162f7cedb44317696f7fcf7c80c43cbee4a6c97

-------------------------------------------

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
Certificate:
   Data:
      Version: 3 (0x2)
      Serial Number:
         46:ae:32:e9:71:02:da:03:24:27:f0:1c:0b:b9:84:9d
      Signature Algorithm: ecdsa-with-SHA256
      Issuer: O = "custom_org            ", CN = Crypto Authentication Root CA 000
```

```
        Validity
            Not Before: Nov 19 06:38:32 2019 GMT
            Not After : Nov 12 06:38:32 2044 GMT
        Subject: O = "custom_org            ", CN = Crypto Authentication Root CA 000
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:f5:9b:35:8c:c5:5b:e8:30:94:ab:c0:d1:5a:0c:
                    40:45:7f:2b:21:a6:05:53:61:a9:d4:7e:fb:b9:7b:
                    a8:e6:7b:d8:ca:82:60:c3:57:f1:f8:a0:cf:f3:df:
                    39:d7:83:ba:d4:78:9f:7d:9b:6b:d3:99:e6:3a:57:
                    2e:9f:cd:c9:78
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                8F:28:E9:74:55:D1:49:52:9D:09:3C:70:00:9F:BB:11:79:9B:FB:3B
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: ecdsa-with-SHA256
        30:45:02:21:00:d9:9c:f1:01:7e:39:09:43:63:e8:8a:62:3b:
        e8:91:4a:b8:28:0e:92:4f:74:e6:f3:fb:42:ff:a2:0f:de:35:
        6d:02:20:01:e4:10:ee:7d:b5:64:ee:ba:18:6a:9d:16:0a:03:
        c8:9a:25:c3:f6:e1:8e:c7:8a:bf:f5:06:d7:90:8b:fc:5a
-----BEGIN CERTIFICATE-----
MIIBzjCCAXSgAwIBAgIQRq4y6XEC2gMkJ/AcC7mEnTAKBggqhkjOPQQDAjBPMSEw
HwYDVQQKDBhjdXN0b21fb3JnICAgICAgICAgICAgICAxKjAoBgNVBAMMIUNyeXB0
byBBdXRoZW50aWNhdGlvbiBSb290IENBIDAwMDAeFw0xOTExMTkwNjM4MzJaFw00
NDExMTIwNjM4MzJaME8xITAfBgNVBAoMGGN1c3RvbV9vcmcgICAgICAgICAgICAg
IDEqMCgGA1UEAwwhQ3J5cHRvIEF1dGhlbnRpY2F0aW9uIFJvb3QgQ0EgMDAwMFkw
EwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE9Zs1jMVb6DCUq8DRWgxARX8rIaYFU2Gp
1H77uXuo5nvYyoJgw1fx+KDP898514O61HiffZtr05nmOlcun83JeKMyMDAwHQYD
VR0OBBYEFI8o6XRV0UlSnQk8cACfuxF5m/s7MA8GA1UdEwEB/wQFMAMBAf8wCgYI
KoZIzj0EAwIDSAAwRQIhANmc8QF+OQlDY+iKYjvokUq4KA6ST3Tm8/tC/6IP3jVt
AiAB5BDufbVk7roYap0WCgPImiXD9uGOx4q/9QbXkIv8Wg==
-----END CERTIFICATE-----

Validate Root Certificate:
OK

Signer Certificate loading from: signer-ca.crt
Certificate:
    Data:
```

```
        Version: 3 (0x2)
        Serial Number:
            67:f4:54:1e:c6:7b:9c:53:1a:eb:6c:71:c6:53:7e:11
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: O = "custom_org            ", CN = Crypto Authentication Root CA 000
        Validity
            Not Before: Nov 19 06:00:00 2019 GMT
            Not After : Nov 19 06:00:00 2029 GMT
        Subject: O = "custom_org            ", CN = Crypto Authentication Signer FFFF
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:79:5c:1d:e1:c4:42:9c:fd:a4:25:26:92:a8:1c:
                    a5:bb:a5:e2:f4:00:24:b9:a9:93:45:44:45:94:d1:
                    61:5f:89:73:92:9d:f5:7d:90:6c:c0:be:93:89:ef:
                    69:cc:ce:80:b7:03:c6:2e:9e:b1:01:de:be:b0:4d:
                    20:26:33:b4:f9
                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:
                1C:27:0D:E4:22:01:30:6A:37:0F:F9:68:E4:D8:91:04:C9:37:D5:82
            X509v3 Authority Key Identifier:
                keyid:8F:28:E9:74:55:D1:49:52:9D:09:3C:70:00:9F:BB:11:79:9B:FB:3B

    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:21:00:c6:8d:c9:fd:a0:07:6f:6d:ab:c3:4e:cd:bf:
         d1:eb:61:a1:80:1b:fd:b0:6f:b3:6e:eb:fe:2c:ef:4b:f7:2d:
         6f:02:20:66:ed:8f:4a:60:1a:34:56:e3:6d:2b:e9:01:ba:3b:
         5b:37:b2:b7:7b:bf:a5:58:d7:37:cd:55:9b:a8:e4:15:eb
-----BEGIN CERTIFICATE-----
MIICAjCCAaigAwIBAgIQZ/RUHsZ7nFMa62xxxlN+ETAKBggqhkjOPQQDAjBPMSEw
HwYDVQQKDBhjdXN0b21fb3JnICAgICAgICAgICAgICAxKjAoBgNVBAMMIUNyeXB0
byBBdXRoZW50aWNhdGlvbiBSb290IENBIDAwMDAeFw0xOTExMTkwNjAwMDBaFw0y
OTExMTkwNjAwMDBaME8xITAfBgNVBAoMGGN1c3RvbV9vcmcgICAgICAgICAgICAg
IDEqMCgGA1UEAwwhQ3J5cHRvIEF1dGhlbnRpY2F0aW9uIFNpZ25lciBGRkZGMFkw
EwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEeVwd4cRCnP2kJSaSqBylu6Xi9AAkuamT
RURFlNFhX4lzkp31fZBswL6Tie9pzM6AtwPGLp6xAd6+sE0gJjO0+aNmMGQwDgYD
VR0PAQH/BAQDAgGGMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFBwnDeQi
```

```
ATBqNw/5aOTYkQTJN9WCMB8GA1UdIwQYMBaAFI8o6XRV0UlSnQk8cACfuxF5m/s7
MAoGCCqGSM49BAMCA0gAMEUCIQDGjcn9oAdvbavDTs2/0ethoYAb/bBvs27r/izv
S/ctbwIgZu2PSmAaNFbjbSvpAbo7Wzeyt3u/pVjXN81Vm6jkFes=
-----END CERTIFICATE-----

Validate Signer Certificate:
OK

Device Certificate loading from: device_cert.crt
Certificate:
   Data:
      Version: 3 (0x2)
      Serial Number:
         4b:58:a6:d8:9c:be:92:41:bc:f8:57:54:df:02:2b:3e
      Signature Algorithm: ecdsa-with-SHA256
      Issuer: O = "custom_org             ", CN = Crypto Authentication Signer FFFF
      Validity
         Not Before: Nov 19 06:00:00 2019 GMT
         Not After : Nov 19 06:00:00 2029 GMT
      Subject: O = "custom_org             ", CN = 01239E0A9490433401_ATECC
      Subject Public Key Info:
         Public Key Algorithm: id-ecPublicKey
            Public-Key: (256 bit)
            pub:
               04:0f:eb:6a:61:45:3e:f7:97:58:f9:dc:98:13:98:
               e0:20:10:99:7a:49:3e:c8:ab:29:01:79:bc:4e:13:
               7a:1e:cf:c3:e3:2e:89:66:7b:6e:cf:9c:0f:c3:10:
               15:c0:35:64:ac:28:b8:9f:8e:f4:68:d5:f3:d9:a2:
               4f:17:28:a9:0d
            ASN1 OID: prime256v1
            NIST CURVE: P-256
      X509v3 extensions:
         X509v3 Basic Constraints: critical
            CA:FALSE
         X509v3 Key Usage: critical
            Digital Signature, Certificate Sign, CRL Sign
         X509v3 Subject Key Identifier:
            C1:62:F7:CE:DB:44:31:76:96:F7:FC:F7:C8:0C:43:CB:EE:4A:6C:97
         X509v3 Authority Key Identifier:
            keyid:1C:27:0D:E4:22:01:30:6A:37:0F:F9:68:E4:D8:91:04:C9:37:D5:82

   Signature Algorithm: ecdsa-with-SHA256
       30:45:02:20:3f:e6:c7:a5:17:80:57:d6:69:04:36:01:34:5e:
       0e:c2:f3:8a:17:2f:a5:6b:41:ca:9e:79:23:42:1d:60:1e:6e:
```

```
        02:21:00:bb:d6:59:98:a0:a1:34:4e:81:b8:c9:1d:ae:54:18:
        7f:f2:57:46:34:5e:e9:03:ff:7f:39:4e:eb:d2:a3:72:8d
-----BEGIN CERTIFICATE-----
MIIB8zCCAZmgAwIBAgIQS1im2Jy+kkG8+FdU3wIrPjAKBggqhkjOPQQDAjBPMSEw
HwYDVQQKDBhjdXN0b21fb3JnICAgICAgICAgICAgICAxKjAoBgNVBAMMIUNyeXB0
byBBdXRoZW50aWNhdGlvbiBTaWduZXIgRkZGRjAeFw0xOTExMTkwNjAwMDBaFw0y
OTExMTkwNjAwMDBaMEYxITAfBgNVBAoMGGN1c3RvbV9vcmcgICAgICAgICAgICAg
IDEhMB8GA1UEAwwYMDEyMzlFMEE5NDkwNDMzNDAxX0FURUNDMFkwEwYHKoZIzj0C
AQYIKoZIzj0DAQcDQgAED+tqYUU+95dY+dyYE5jgIBCZekk+yKspAXm8ThN6Hs/D
4y6JZntuz5wPwxAVwDVkrCi4n470aNXz2aJPFyipDaNgMF4wDAYDVR0TAQH/BAIw
ADAOBgNVHQ8BAf8EBAMCAYYwHQYDVR0OBBYEFMFi987bRDF2lvf898gMQ8vuSmyX
MB8GA1UdIwQYMBaAFBwnDeQiATBqNw/5aOTYkQTJN9WCMAoGCCqGSM49BAMCA0gA
MEUCID/mx6UXgFfWaQQ2ATReDsLzihcvpWtByp55I0IdYB5uAiEAu9ZZmKChNE6B
uMkdrlQYf/JXRjRe6QP/fzlO69Kjco0=
-----END CERTIFICATE-----
```

```
Validate Device Certificate:
OK

Generated the manifest file 01239e0a9490433401_manifest.json
Custom Certificate processing completed successfully

-------------------------------------------
```

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/Trust&GO to secure a Google Cloud Platform connection.

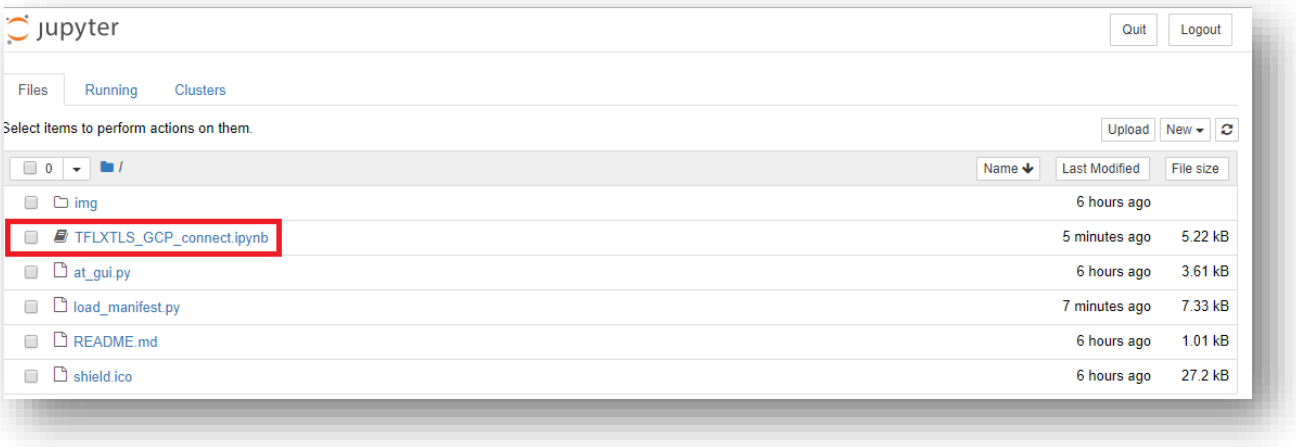
The reference implementation is provided with Embedded projects and Notebooks. The generation of manifest can be achieved through the execution of Jupyter Notebook Tutorials.

Note: It is required to have Google account test account setup prior to running this. Instruction to setup the account is provided in **docs\TrustFLEX GCP Account setup instructions.pdf**.

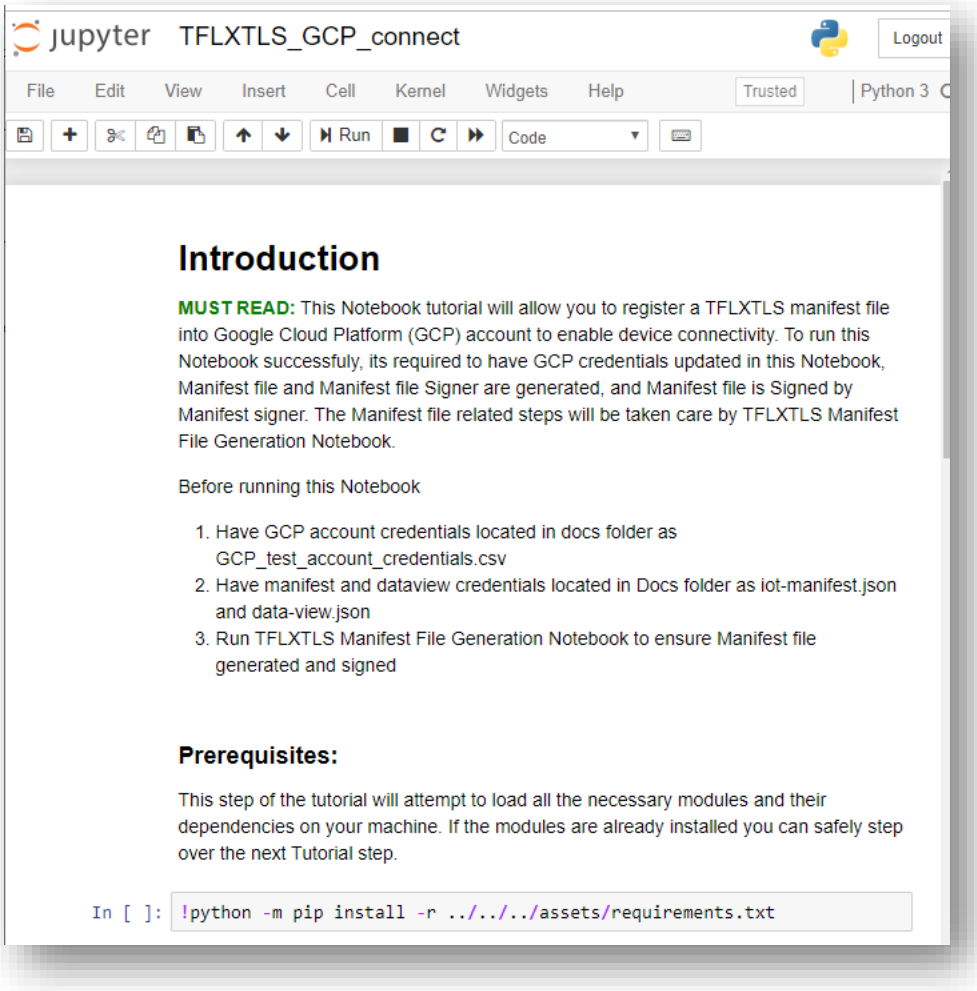
4.1 Running GCP example on Jupyter Notebook

By running this step, one should be able to register the secure element to Google account by uploading device manifest file generated in the previous section. To run this Notebook, its required to have device manifest file (generated in previous section), google account credentials for manifest and data view (saved as part of GCP account setup).

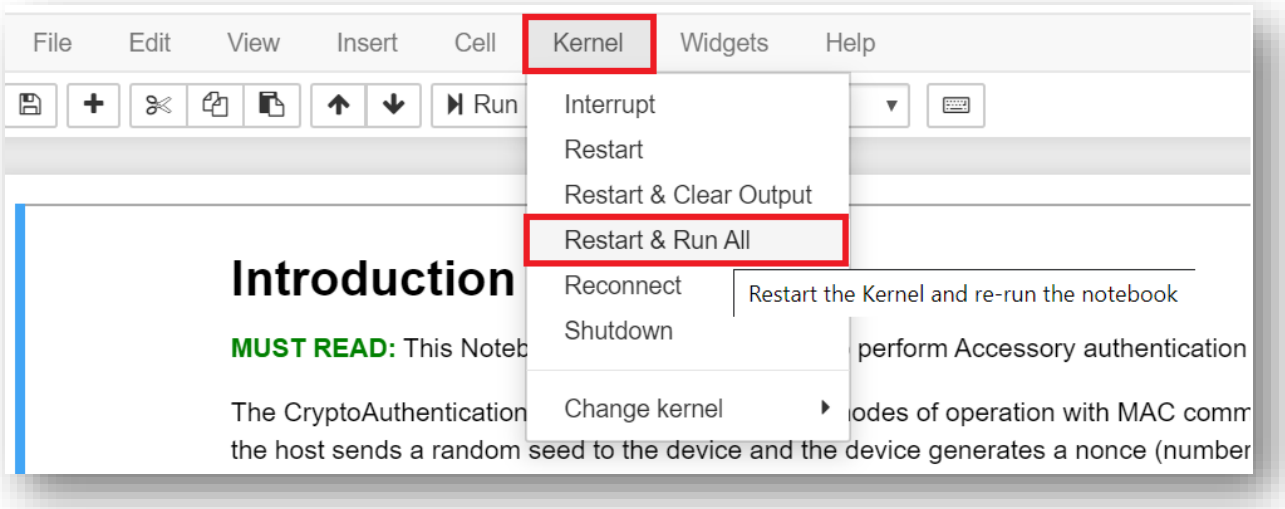
1. From the Jupyter Home page, navigate to **TrustFLEX\03_gcp_connect\notebook\TFLXTLS_GCP_connect.ipynb** notebook file and open it.



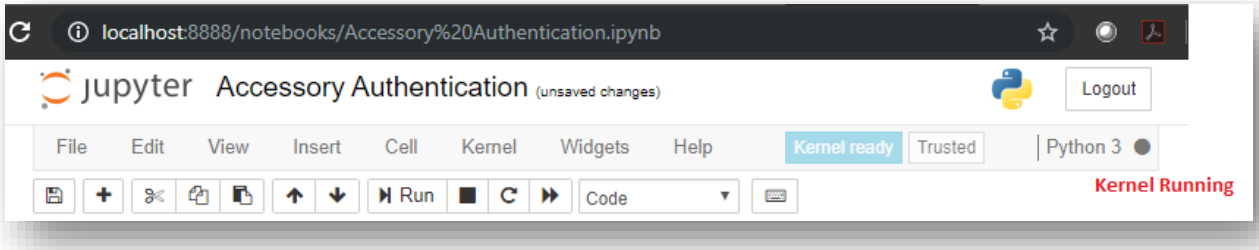
Opening the Jupyter notebook example should load the following on the browser.

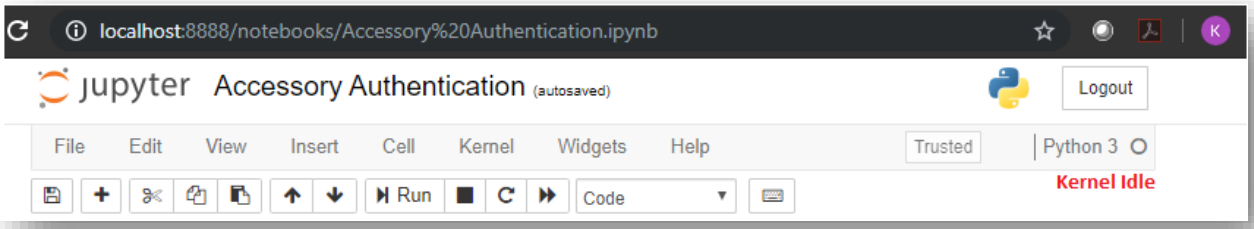


2. Run All Cells by using Kernel -> Restart & Run All



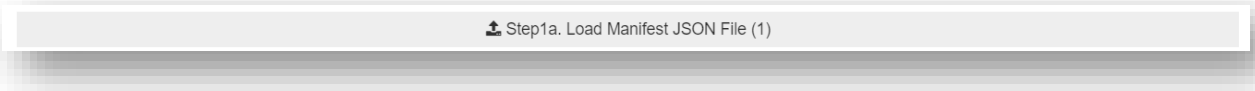
It may take a while to complete, wait for the kernel to complete all processing i.e. from Kernel Running to Kernel Idle state (Check circle above **RED** text)



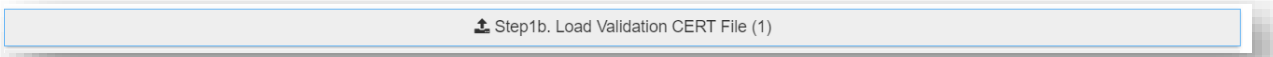


3. Navigate through different cells output for the description of the step and result from the execution.

4. There are 3 major steps:
   Load Manifest File:
   Under the section **Upload Manifest File**, click the button '**Load Manifest JSON File**' and select the
   manifest file generated from the TrustFlex Resource generation notebook.



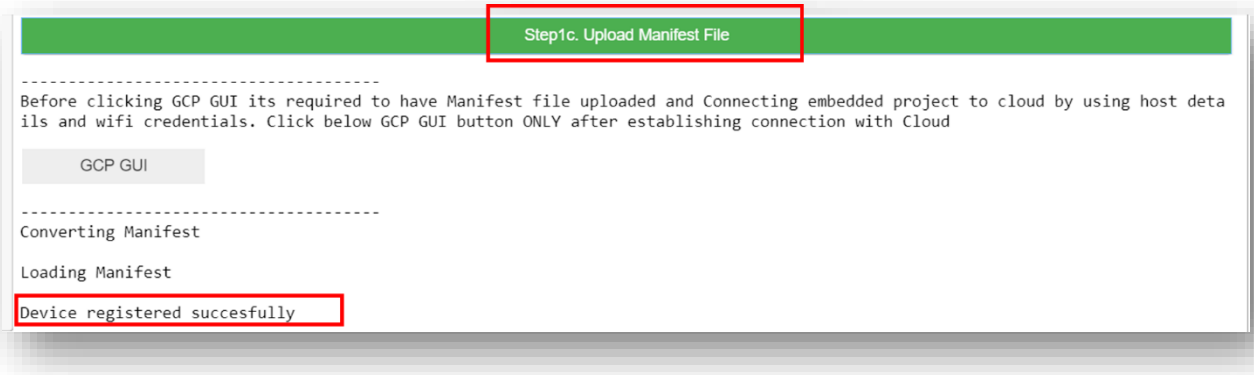
   Load validation certificate:
   click the button '**Load Validation CERT File**' and
   select the validation certificate which signed the manifest file and it should be present in the following folder with name log_signer.crt
   For TrustFLEX – TrustFLEX\00_resource_generation\
   For Trust&GO - TrustnGO\00_resource_generation\



   Register device manifest file:
   Code block of this step generates "**Upload manifest File**" button. Clicking the button, it registers the device manifest file to the GCP account. Once the manifest

file is registered, the gcp cloud authorizes the Trust Platform device and it will be able to
communicate to them.

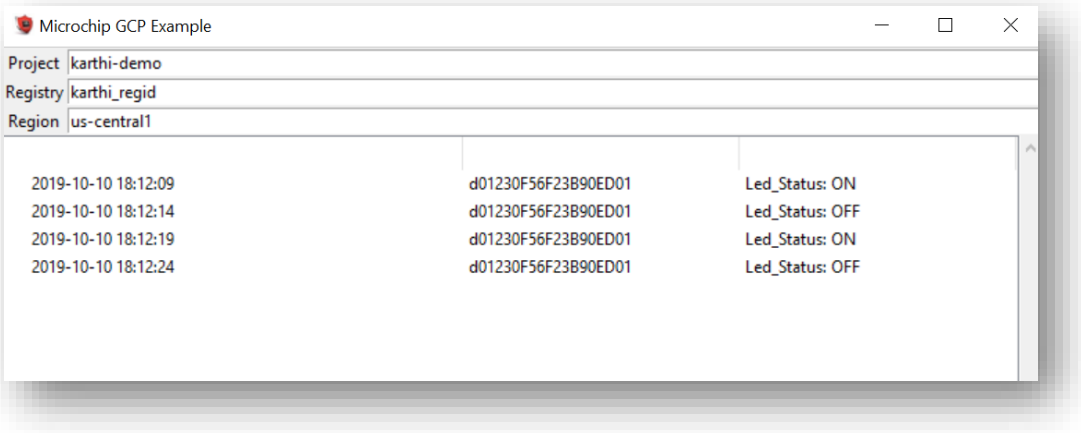
Upon successful execution, the log should look like this.



WARNING: It is required to execute C project successfully before executing the next step in the Jupyter notebook. To execute C project, refer "Running GCP IoT example on Embedded platform" next section.

GCP GUI:
Code block of this step generates "**GCP GUI**" button. Clicking the button, it will create a very basic graphical interface that will display the trust platform board LED status.

Below screenshot display the graphical interface



This GUI displays the packets exchanged between CryptoAuth Trust Platform and GCP.

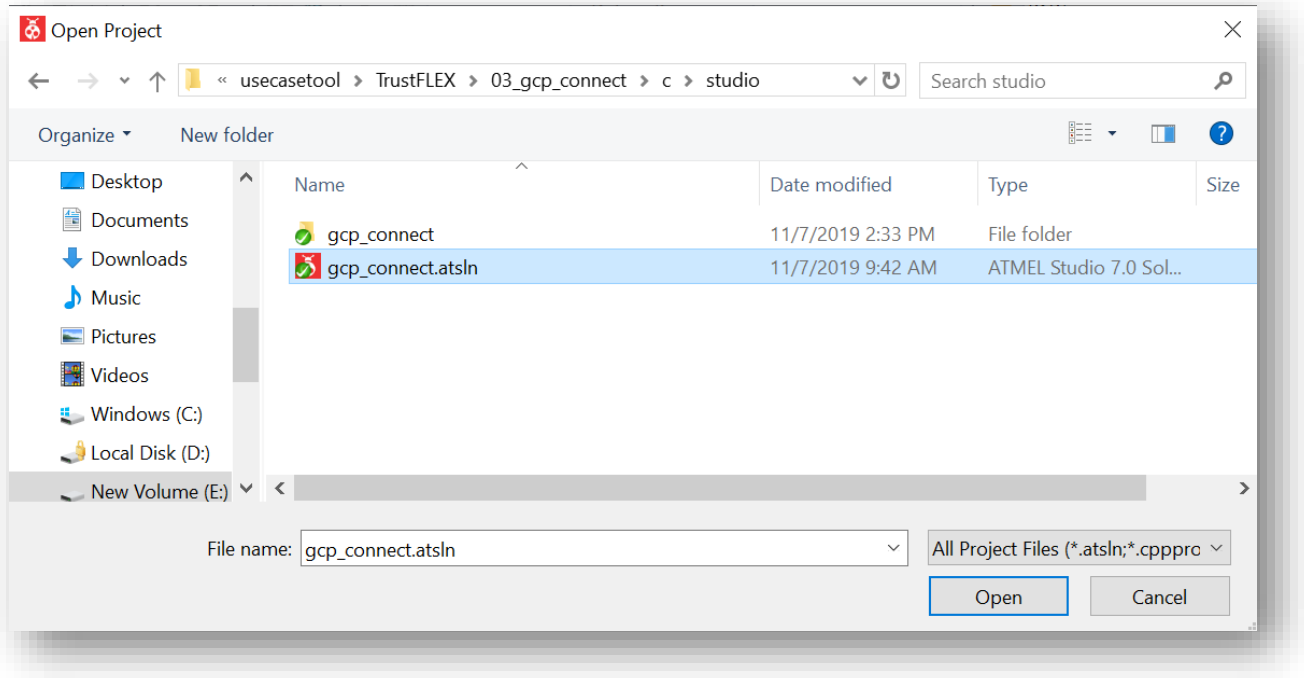
4.2 Running GCP example on Embedded platform

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

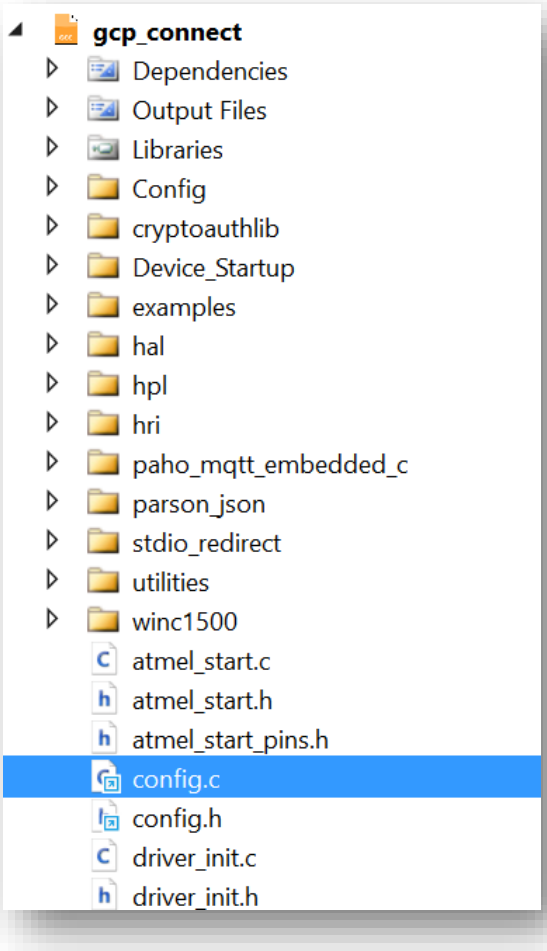
This project establishes a TLS connection and subscribe to MQTT. It is required to use the GCP IoT Jupyter notebook to register the device through manifest file. Prior to executing the application, it is required to update Wifi credentials, GCP account details. Following steps provides the instructions for the same,

4.2.1 Atmel Studio:

1. Open **gcp_connect.atsln** project by navigating **TrustFLEX\03_gcp_connect\c\studio\gcp_connect.atsln**



2. In the project navigate to **gcp_connect -> config.c** file

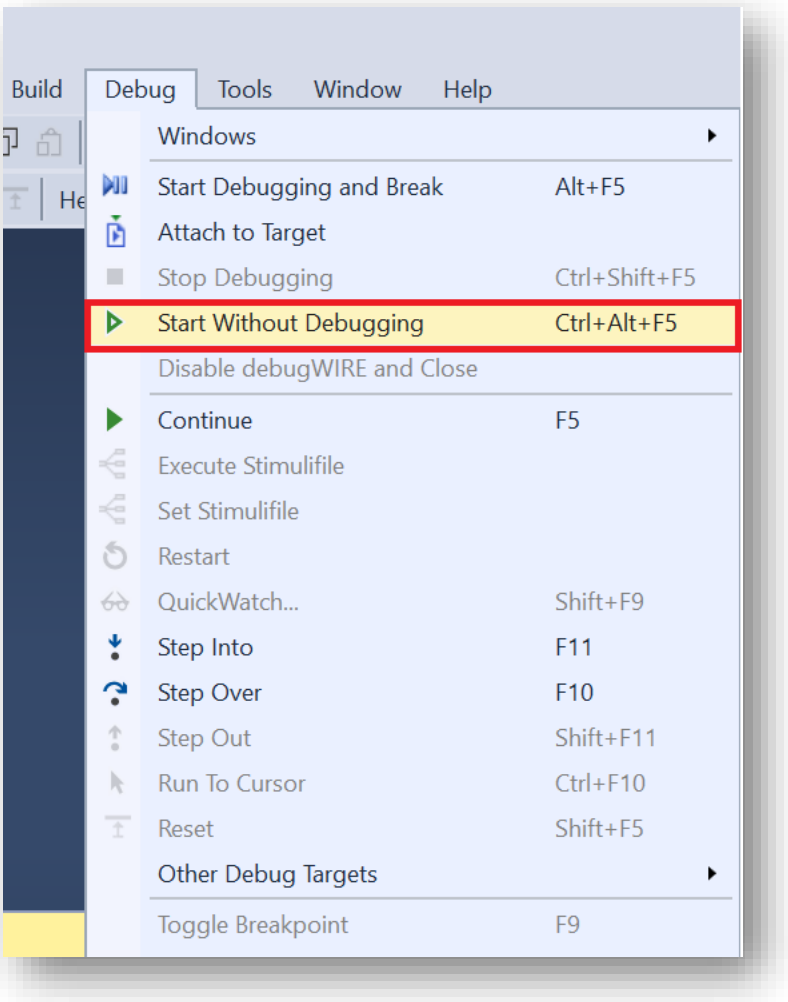


update the following constants before building the project:
The project id, region id and registry id should be same as in the gcp account setup.

- `config_demo_ssid`
- `config_demo_pass`
- `config_gcp_project_id`
- `config_gcp_region_id`
- `config_gcp_registry_id`

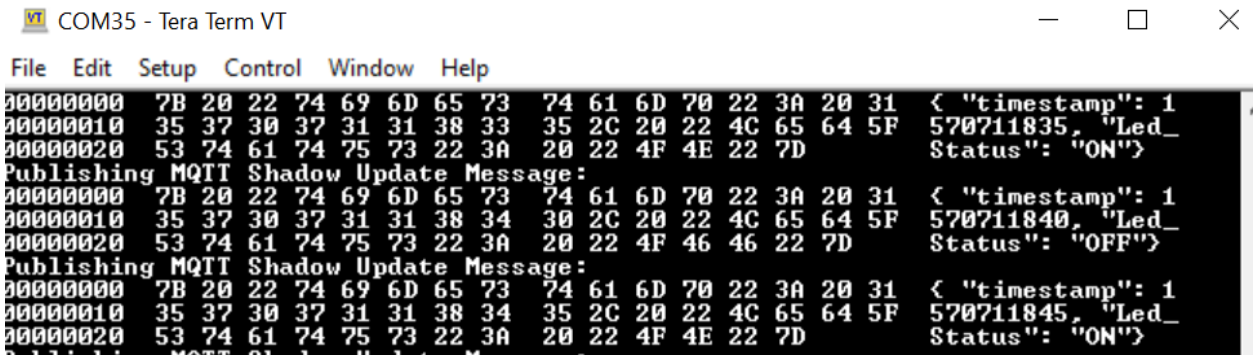
```
/* Example Configuration Data  Global Variables */
const char config_demo_ssid[] = "xxxxxxxxx";
const char config_demo_pass[] = "xxxxxxxxx";

const char config_gcp_project_id[] = "xxxxxxxxx";
const char config_gcp_region_id[] = "xxxxxxxxx";
const char config_gcp_registry_id[] = "xxxxxxxxx";
```

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.

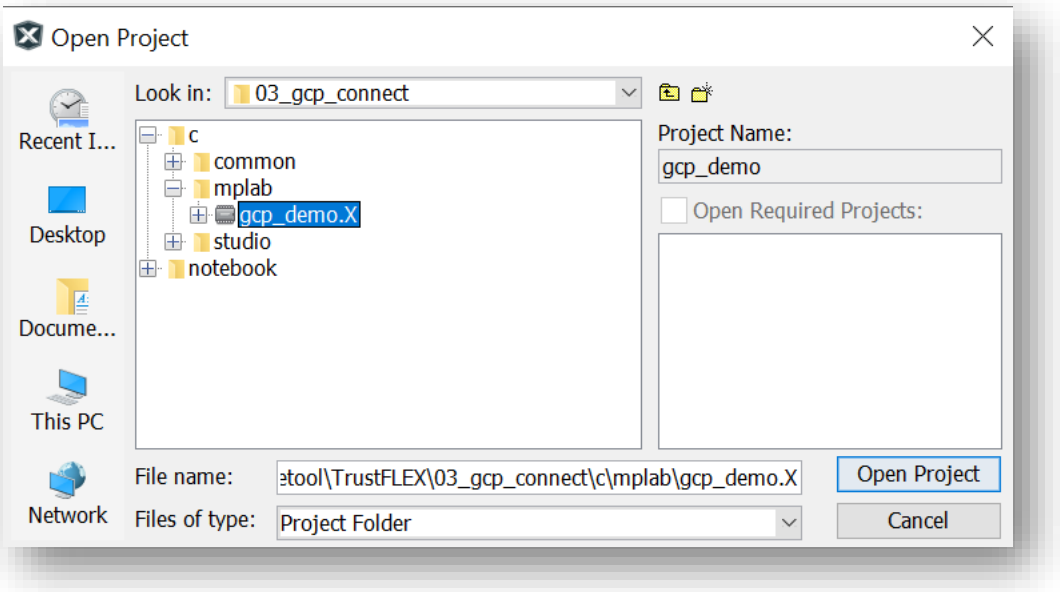
Once the programming is done, reset the hardware (press the reset button) and view the Console messages by using applications like 'Tera Term'. Open the application with the COM related to CryptoAuth Trust Platform with 115200-8-N-1 settings.



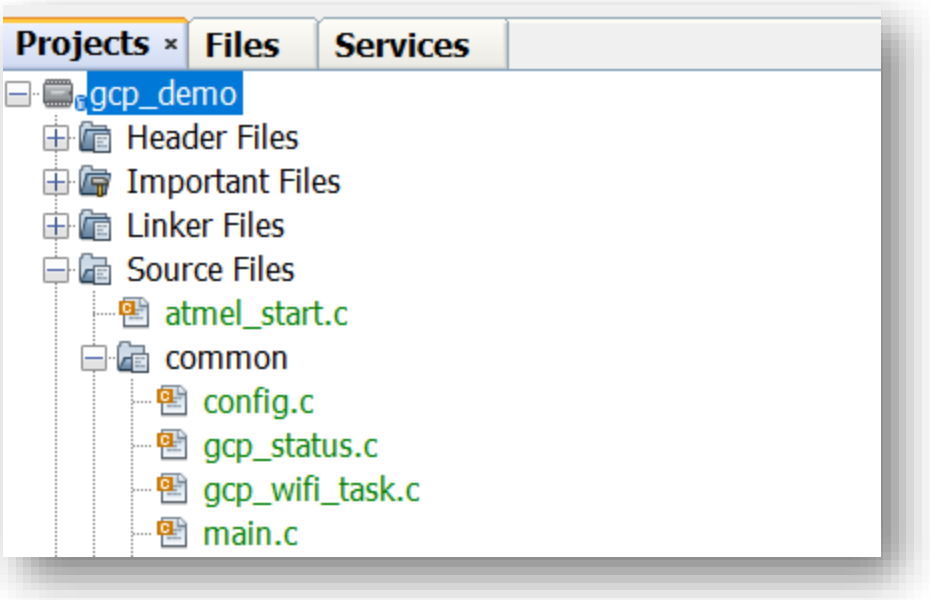
Once successfully programmed the CryptoAuth Trust Platform, navigate to previous section 1.7 to run the last step (GCP GUI) in the Jupyter Notebook.

4.2.2 MPLAB:

1. Open **gcp_demo.X** project by navigating to MPLAB -> File -> Open Project -> **TrustFLEX\03_gcp_connect\c\mplab\gcp_demo.X**



2. Open **config.c file** by navigating to **gcp_demo-> Source Files ->common-> config.c**

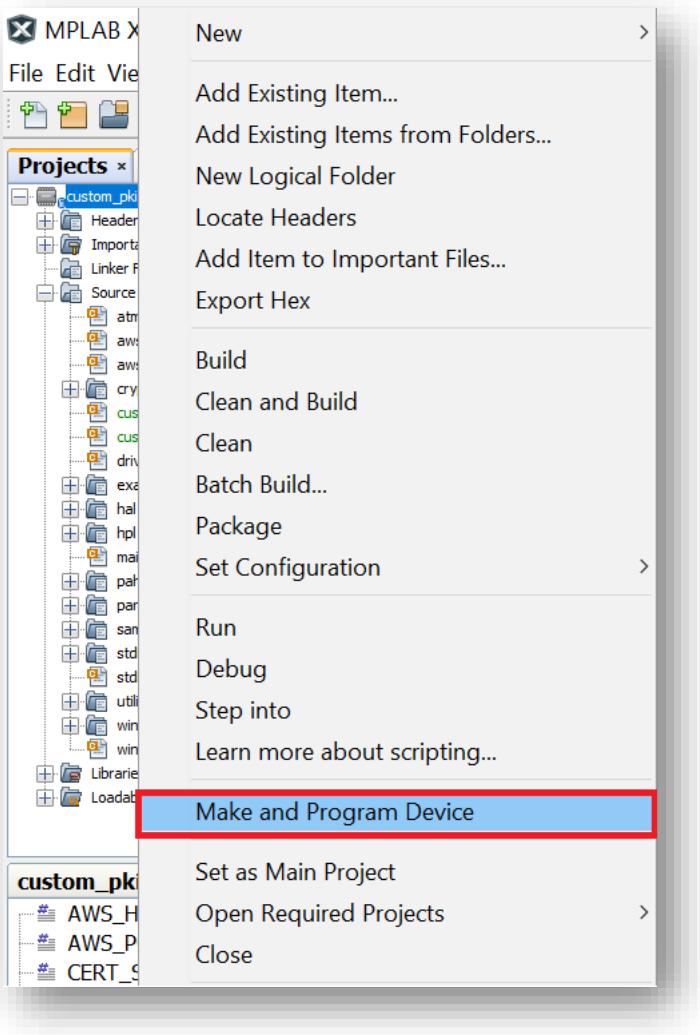


update the following constants before building the project:
The project id, region id and registry id should be same as in the gcp account setup.

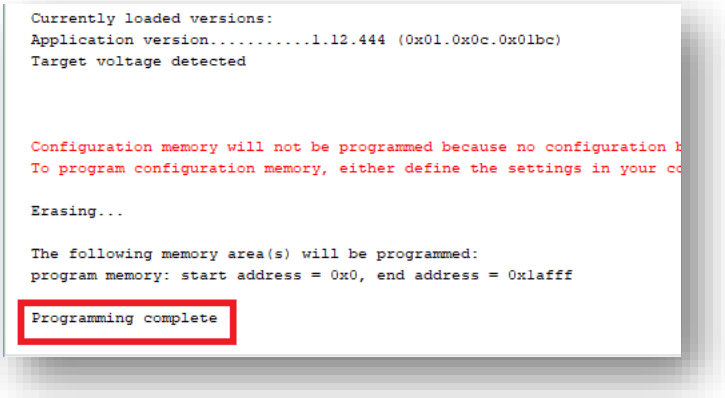
- config_demo_ssid
- config_demo_pass
- config_gcp_project_id
- config_gcp_region_id
- config_gcp_registry_id

```
/* Example Configuration Data  Global Variables */
const char config_demo_ssid[] = "xxxxxxxxx";
const char config_demo_pass[] = "xxxxxxxxx";

const char config_gcp_project_id[] = "xxxxxxxxx";
const char config_gcp_region_id[] = "xxxxxxxxx";
const char config_gcp_registry_id[] = "xxxxxxxxx";
```

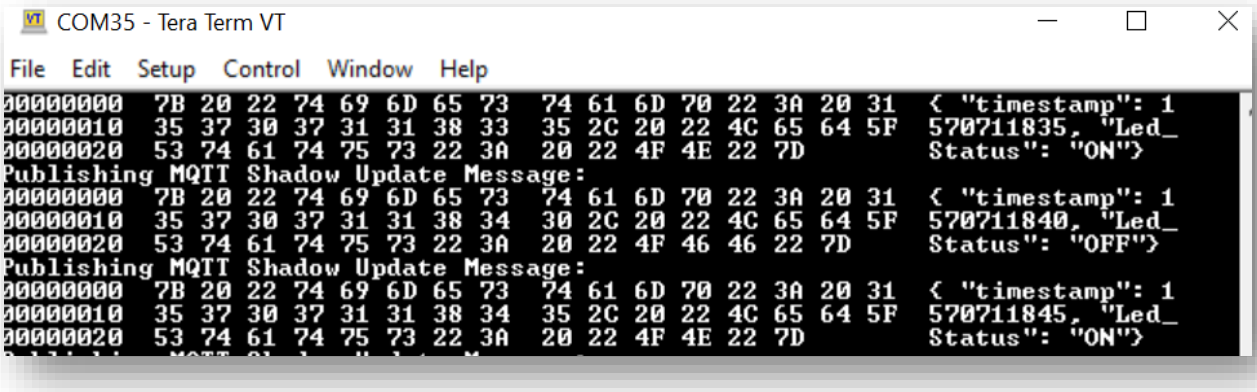
3. Program the CryptoAuth Trust platform by navigating to **gcp_connect -> Make and Program Device**



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, navigate to previous section 1.7 to run the last step (GCP GUI) in the Jupyter Notebook.

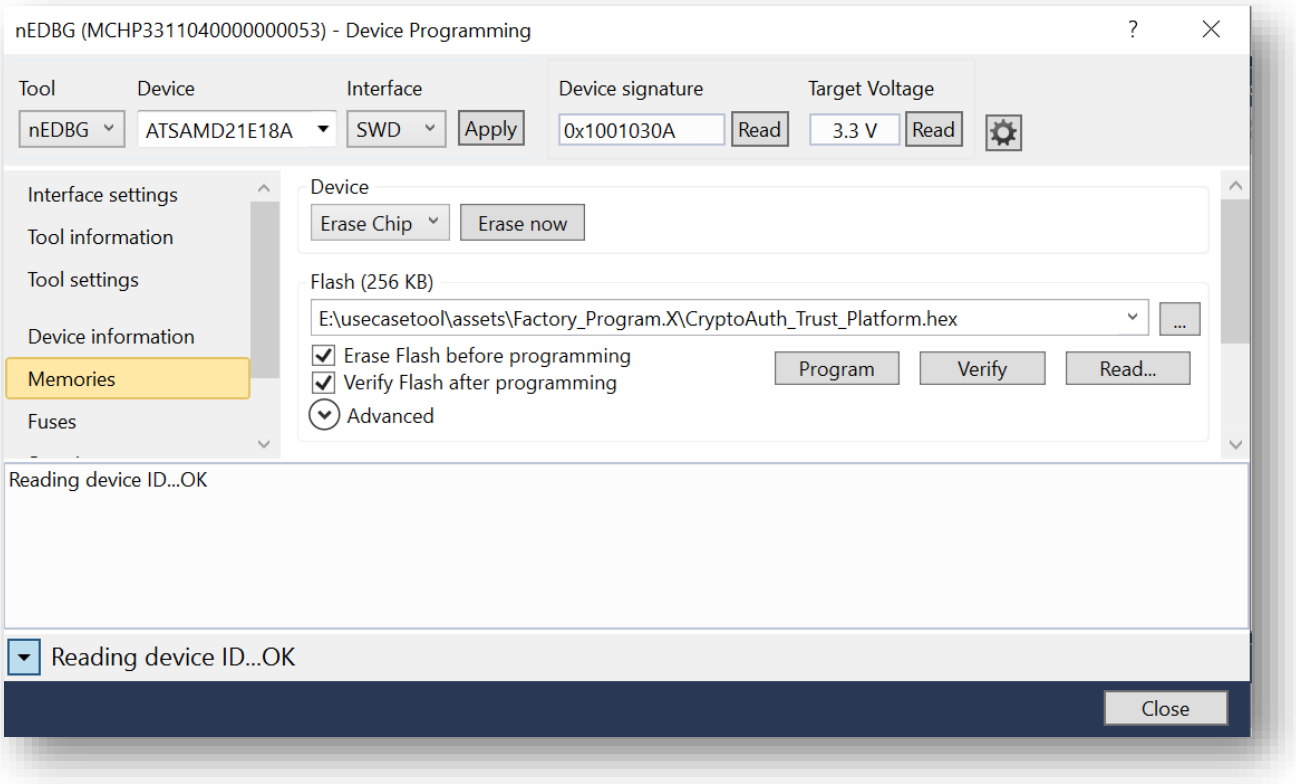
4.3 CryptoAuth Trust Platform Factory reset

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 **assets\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 **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**

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.

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