TrustFLEX Step by Step Guide Firmware Validation

Table of Contents

1 Introduction

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

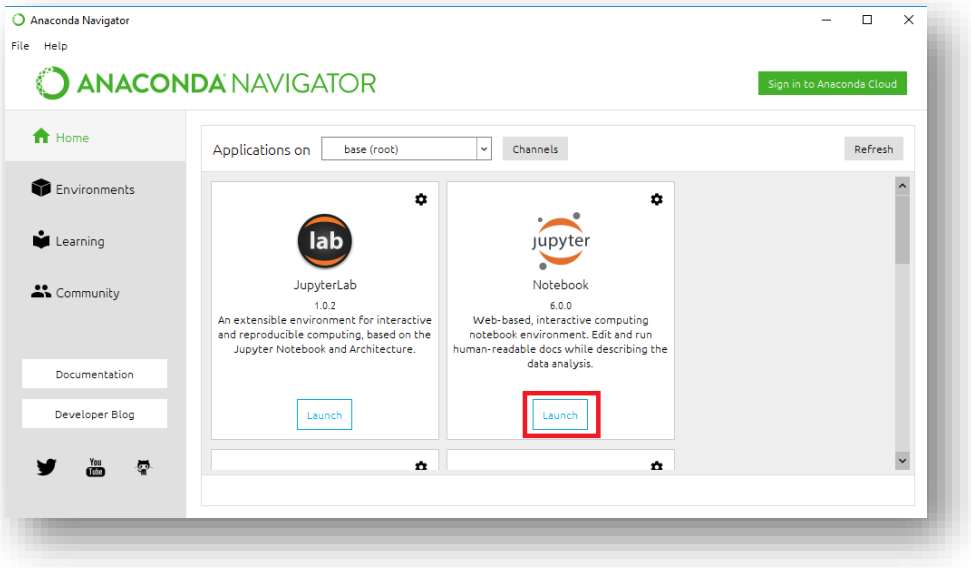
1.1 Getting started with Jupyter Notebook Tutorials

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

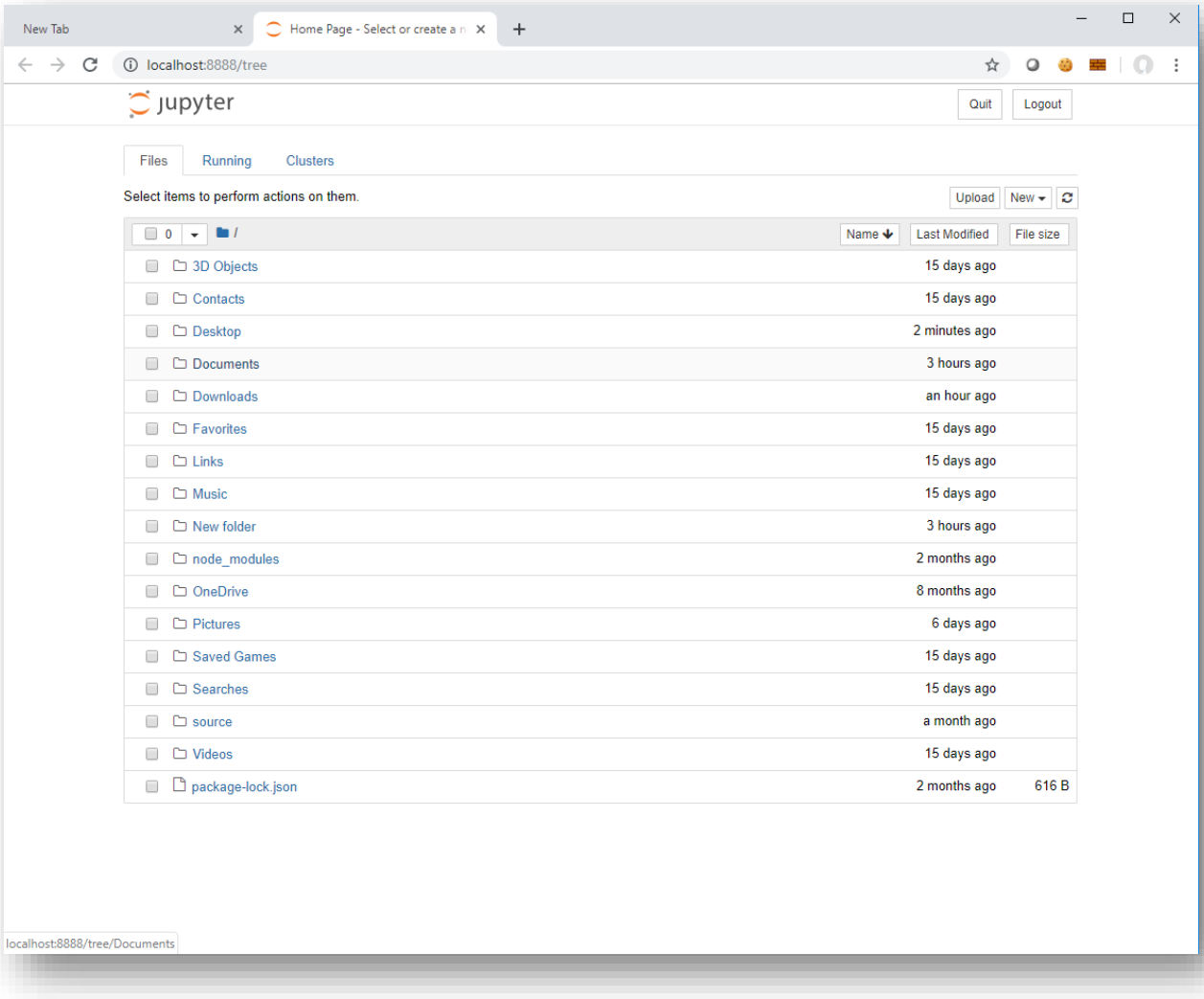
It is recommended to become familiar with Jupyter basic concepts with the online documentation, https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html

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 to the 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 on sub-directories in the notebook list, you can navigate your file system.

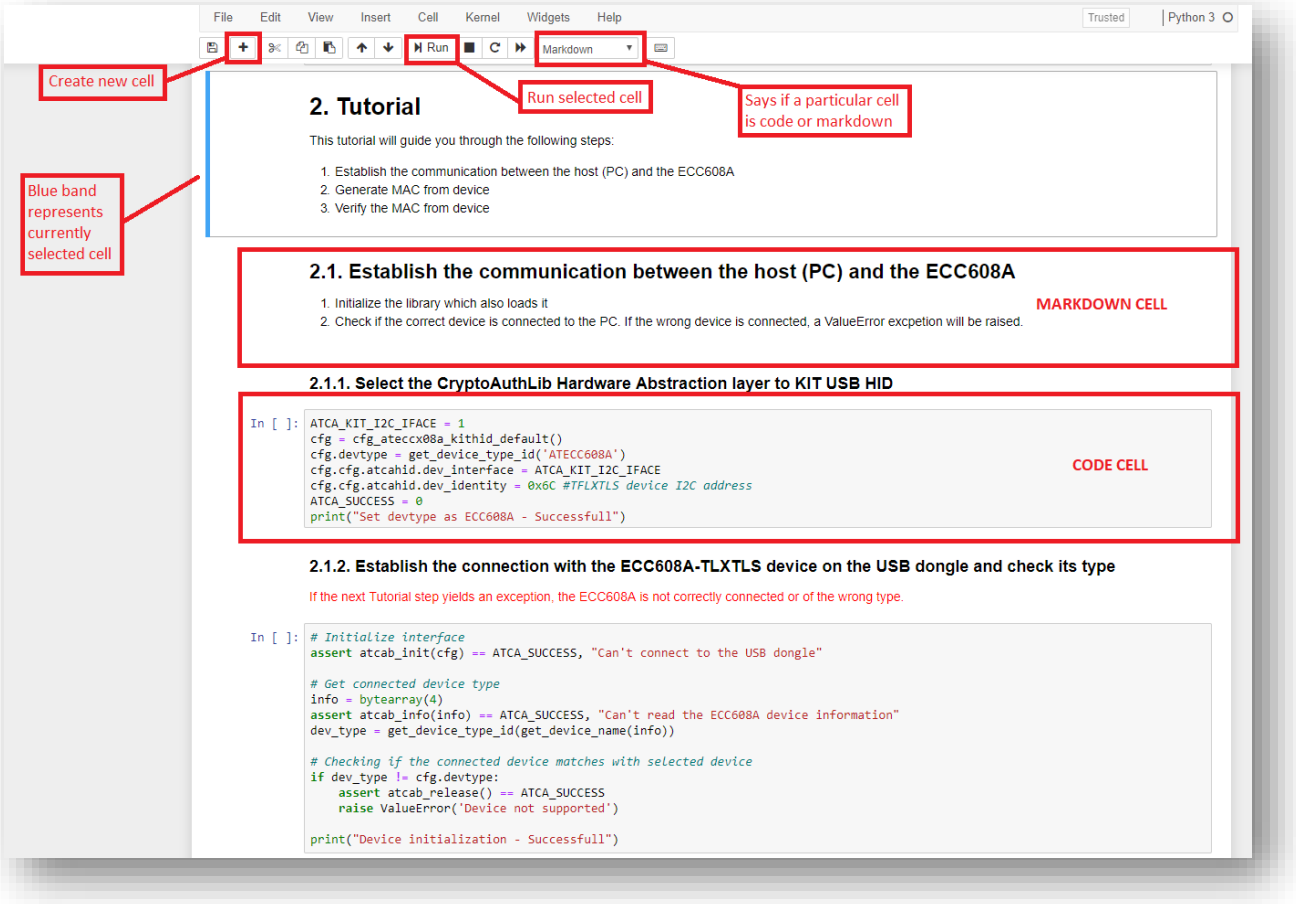


1.3 Introduction to Jupyter Notebook GUI.

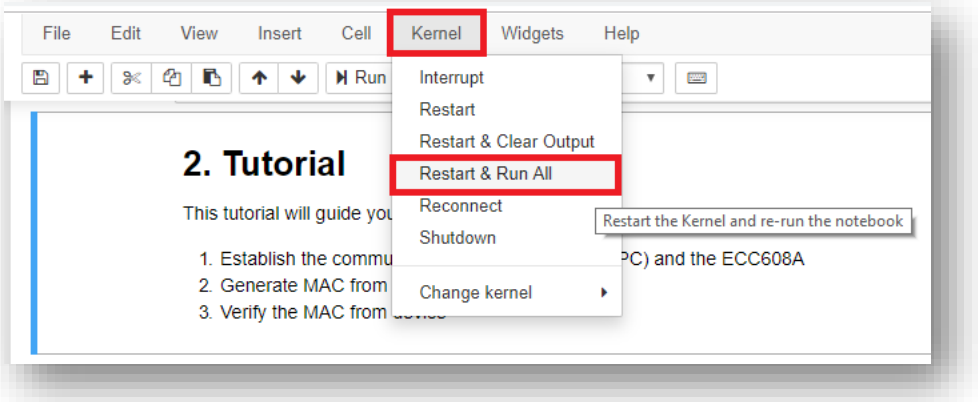
Jupyter Notebooks contain cells where you can either write code or markdown text. Notebooks contain multiple cells, some set as code and others markdown. Code cells contain code that can be executed live, and markdown contains text and images to explain the code.

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 Notebook Tutorials to easily prototype popular use cases for TrustFLEX. 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\firmware-validation\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

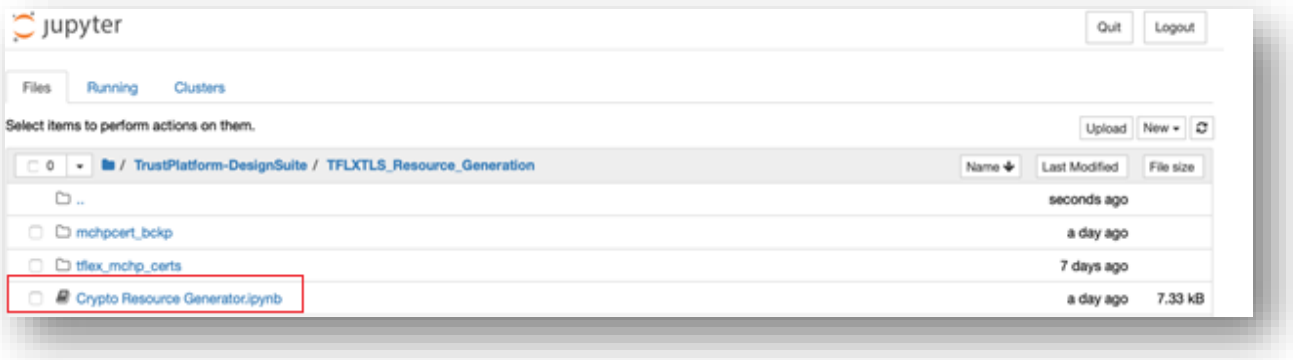
TFLXTLS device is one of the three devices available in the Trust Platform USB Dongle Board.

TrustFlex devices come with pre-programmed 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 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.

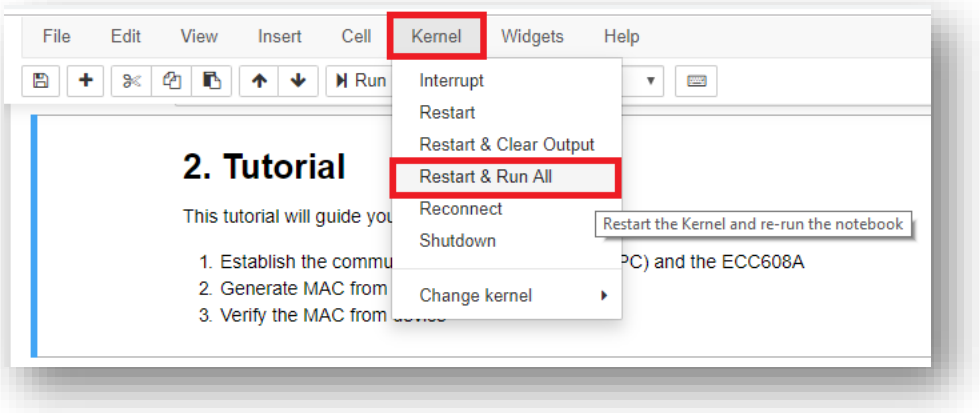
By default, Jupyter starts in Users directory ($HOME for MacOS or Linux systems). For the remainder of this document, it will be assumed that the trust_platform folder is contained in Users directory. If this is not the case, please move trust_platform folder to your Users directory

Within the Jupyter Dashboard, navigate trust_platform\DesignTools\TFLXTLS_Resource_Generation folder to open Crypto Resource Generator.ipynb notebook



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

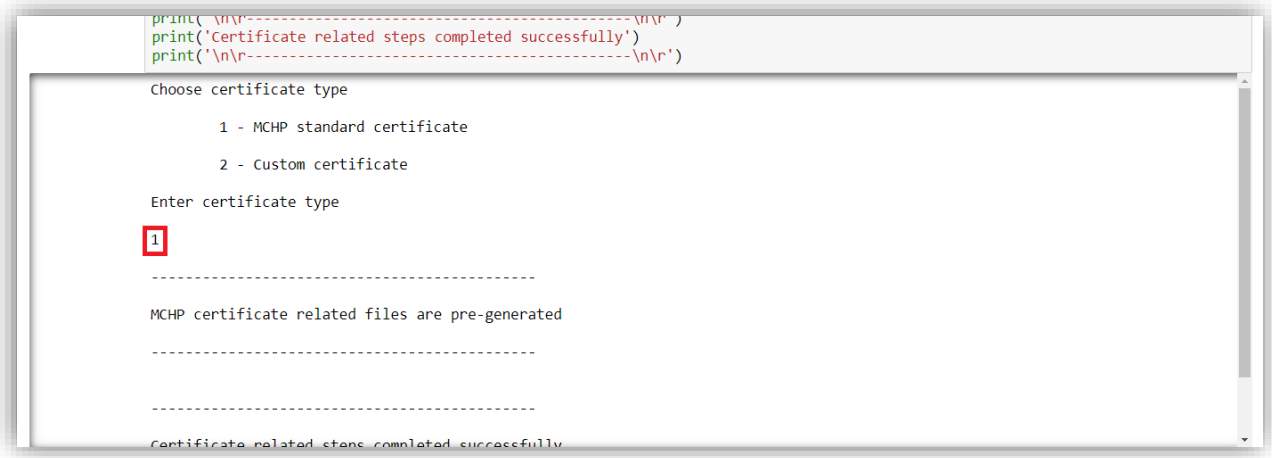
Note: Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to 4.3 CryptoAuth TrustPlatform Factory reset section for reloading default program.



Crypto Resource Generator notebook is common for all the use case which comes with option to load the signer certificate and device certificate.

It will execute and prompt you to choose between MCHP certificate and a custom certificate chain, enter '1' (for MCHP certificate) and press Enter key for this use case.

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.



The output log should look like this.

```
Choose certificate type

       1 - MCHP standard certificate

       2 - Custom certificate

Enter certificate type

1

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

MCHP certificate related files are pre-generated

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


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

Certificate related steps completed successfully

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

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

After running this Notebook, it generates the required resources and program data zone with required secrets, keys and certificates. For this use case, IO protection key and firmware validation public key are loaded into TrustFLEX device in the slot 6 and 15 respectively.

4 Use Case Prototyping

This hands-on lab is intended to demonstrate the usage of TrustFLEX device to validate firmware that going to run on HostMCU. It uses asymmetric authentication.

To validate the firmware, following steps to be followed

1. Generating a firmware Signing Key pair
2. Signing the firmware
3. Updating the firmware to product
4. Verifying the firmware image

OEM to take care of first 2 things in a controlled environment. To have firmware validation functionality, once the firmware implementation is completed it will be signed by the OEM signer to make the image authentic. Typically, firmware signer's public key will be loaded to secure element and locked permanently.

On the product side, the digest and signature generated in the previous step will be provided to secure element using Secure boot command. Secure boot command will be executed on secure element with option set to store (Full Copy) on successful validation of the digest and signature.

On TrustFLEX device secure boot configuration is set as "FullDig", which stores the firmware digest on the device (slot 7 on TrustFLEX). On subsequent boots, the digest is compared without ECC verify operations. While sending the digest to TrustFLEX device, the digest is encrypted with IO protection key to avoid man in the middle attack.

This lab is setup such a way firmware sign operation taken care by notebook, update and verify operations can be done both in notebook and embedded project. Firmware sign operations are NOT done in embedded project as it's the role of OEM but not the product.

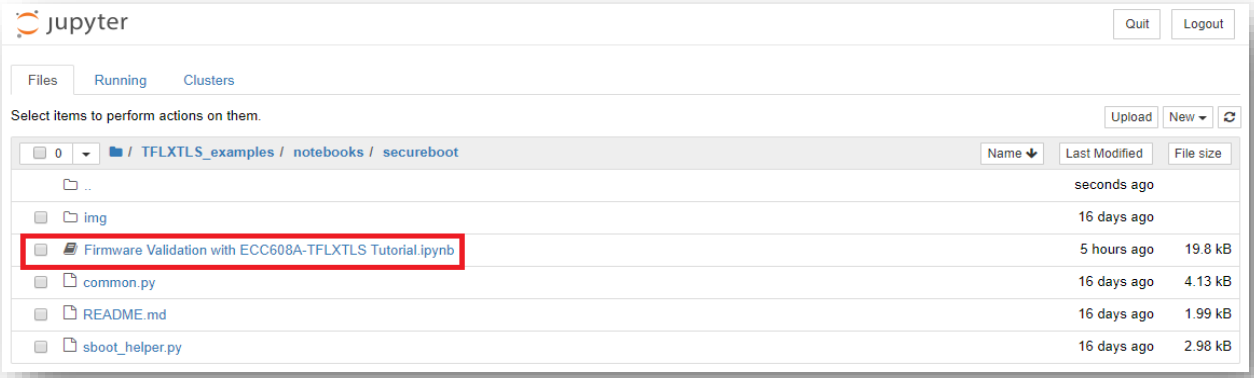
The resource generation for TrustFLEX device will load

1. A prototyping firmware signing key
2. A prototyping IO protection key to Slot6
3. Signers public key to Slot15 respectively

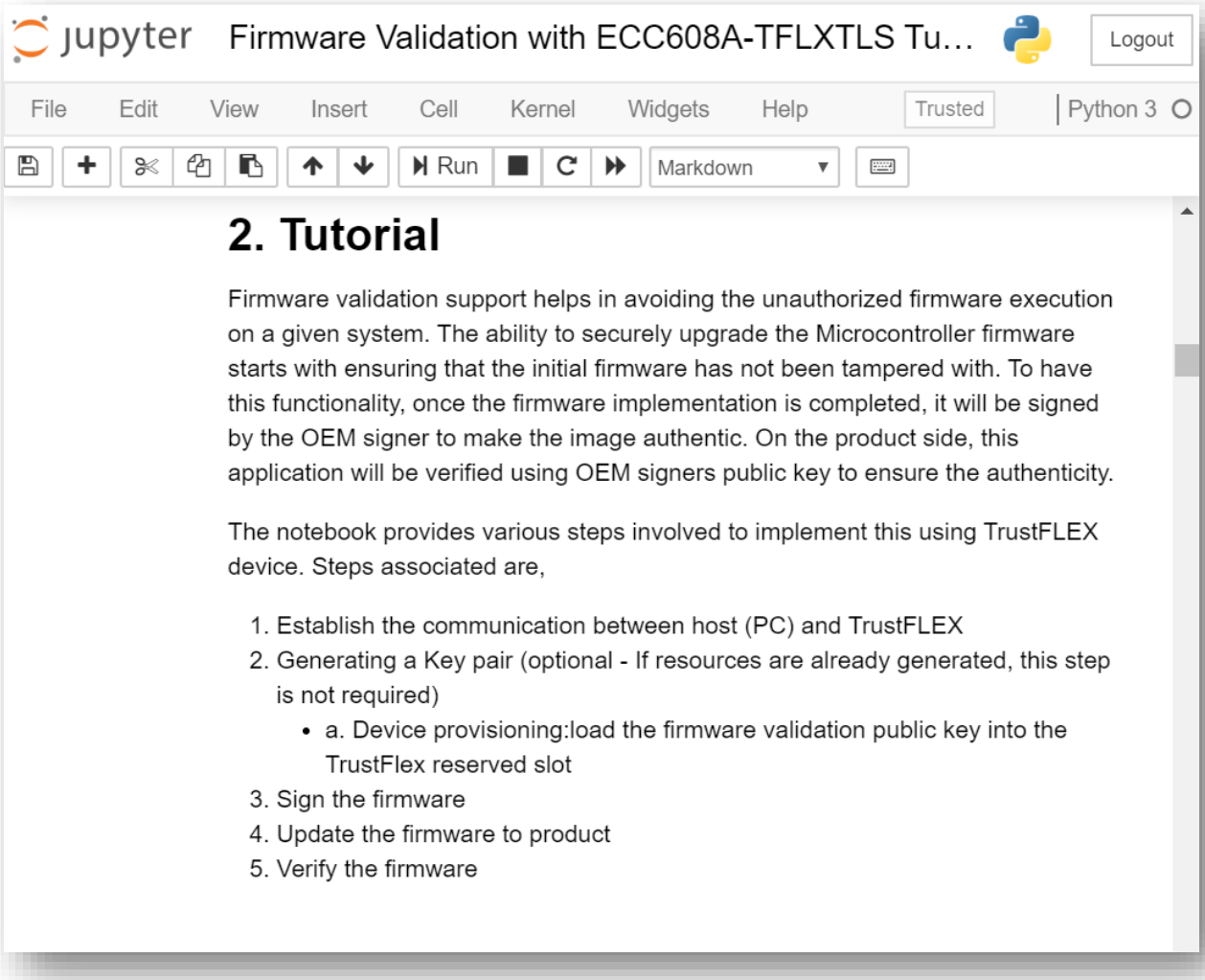
Following sections provide detail steps to execute the Usecase both on Jupyter Notebook and on Embedded project

4.1 Running Firmware Validation example on Jupyter Notebook:

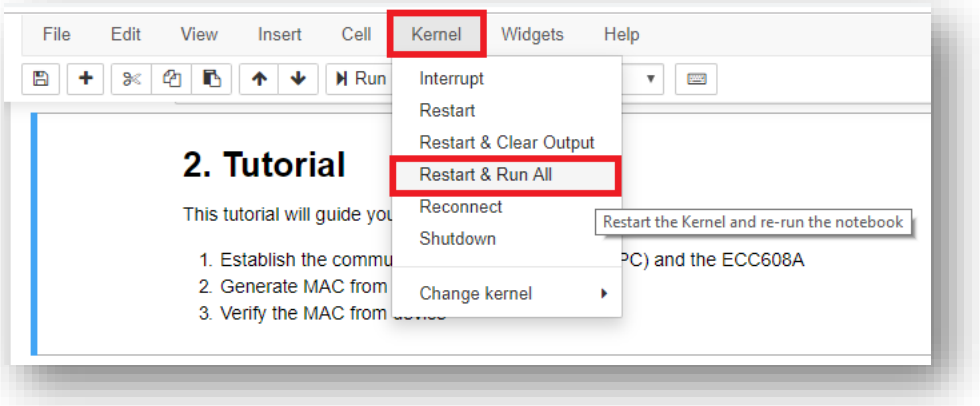
1. From the Jupyter Home page, navigate to **TFLXTLS_Use_Cases\notebooks\ firmware-validation\Firmware Validation with ECC608A-TFLXTLS tutorial.ipynb** notebook file and open it.



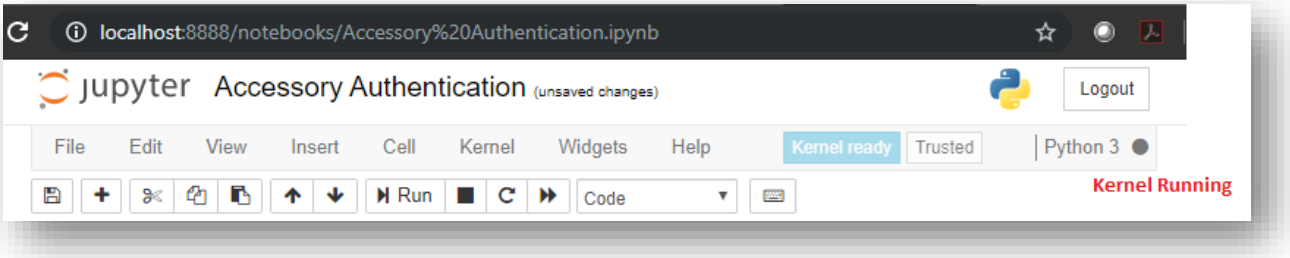
Opening the notebook from Jupyter home page should load the following on the browser,

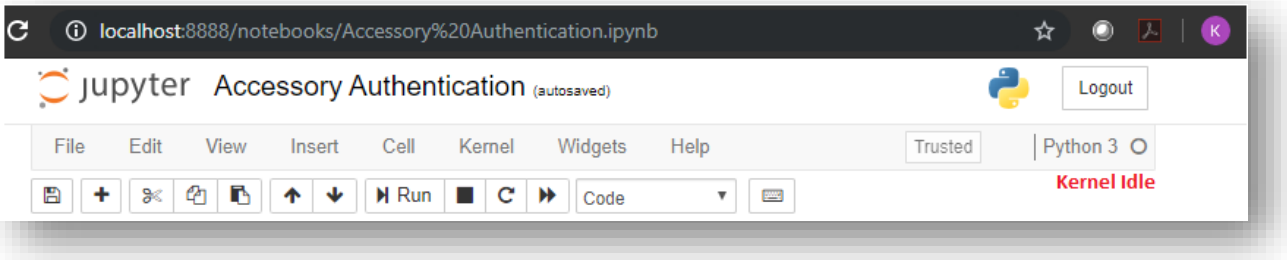


2. Run All Cells by using Kernel -> Restart & Run All



3. 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)





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

5. There are 4 major steps in this lab
   Generating a Firmware Validation key pair
   This step setups a temporary firmware signer to perform firmware validation process. This key generation is already taken care part of resource generation.

   Sign the Firmware
   This step generates firmware digest by hash the example firmware image with SHA 256 algorithm and get it signed with firmware signer's private key. Then digest will be encrypted with IO protection key to avoid man in the middle attack before host send digest to the device.

Here is how the memory of the Microcontroller is portioned. Microcontroller has a 256KB flash starting from 0x0000 0000, supporting address range from 0x0000 0000 to 0x0003 FFFF.

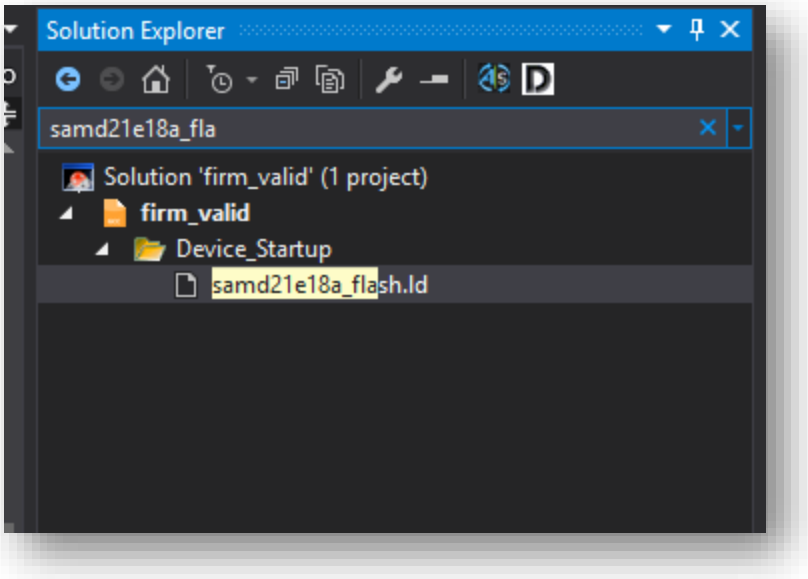
Firmware validation image	0x0000 0000 to 0x0000 BFFFF
Application image	0x0000 C000 to 0x0003 FBFF
Signature data	0x0003 FC00 to 0x0003 FFFF

The firmware validation image and the application image can be obtained by building (compile + link) the respective projects in the correct address spaces, the signature will be calculated and stitched with the other images through Jupyter Notebooks.

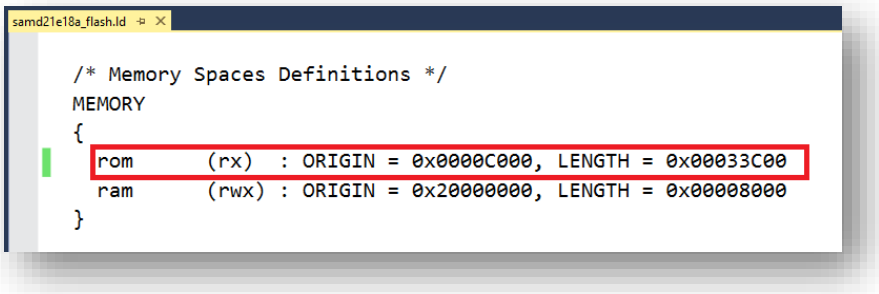
To get the firmware validation hex,
open the following Studio project and compile it.
DesignTool\TFLXTLS_Use_Cases\c\firm_valid\studio. After successful compilation it will create .hex file under Debug folder i.e.
DesignTool\TFLXTLS_Use_Cases\c\firm_valid\studio\firm_valid\Debug\firm_valid.hex . We will be using this file in future steps.

To get the application HEX,
any of the Usecase example applications can be used as application hex. Here, let's use ip_protect as application. open Atmel Studio project under the following directory i.e. DesignTool\TFLXTLS_Use_Cases\c\ip_protect\studio

This project was setup such that the start address is 0x0000 0000 which is the default address for SAM D21 ICs. But we need to modify start address of this project to 0x0000 C000 as mentioned in the above table. This change needs to be made in the linker settings. In solution explorer find the "samd21e18a_flash.ld" file.



Open samd21e18a_flash.ld to change the start address to 0x0000 C000.



After changing the linker file, you can compile the project. After successful compilation it will create .hex file under Debug folder i.e. DesignTool\TFLXTLS_Use_Cases\c\ip_protect\studio\ip_protect\Debug\ ip_protect.hex.

Now that we have all the binaries available, go back to the Firmware Validation Juyter Notebook. Go to step 2.3, this step accepts two hex files, combines then and appends signature to it. Follow the below snapshots for reference. Make sure the correct images are selected on Upload buttons.

Load firmware validation binary (HEX)

```
In [6]: firmvalid_img_object = FileUpload(accept='*.hex', multiple=False)
        display(firmvalid_img_object)
```

Upload (1) **Select firm_valid.hex**

Load application binary (HEX)

```
In [7]: app_img_object = FileUpload(accept='*.hex', multiple=False)
        display(app_img_object)
```

Upload (1) **Select ip_protect.hex**

Combining HEX files and apending signature

```
In [8]: combine_and_sign = widgets.Button(description = "Combine HEX")

        def combine_hex(b):
            combine_sign_hex(firmvalid_img_object, app_img_object)

        combine_and_sign.on_click(combine_hex)
        display(combine_and_sign)
```

Combine HEX **Click on Combine HEX after selecting firmware validation hex and application hex**

```
Firmware validation binary size: 26412
Application binary size: 23768

Application digest:
    ED B5 26 9F 6C 11 5C 8B 78 3C 74 87 ED 50 8A 85
    7E AE 12 40 12 B4 B4 2B 97 1B 6C A2 BF 0B D3 0A

Successfully Signed the firmware digest
Calculated signature:
0x30, 0x31, 0xBF, 0xAC, 0xF6, 0xDD, 0x88, 0xC1, 0x11, 0x39, 0x69, 0xF7, 0x3E, 0x21, 0x2F, 0xCA,
0x1D, 0x49, 0xF1, 0x20, 0x2B, 0xC9, 0x2F, 0x5E, 0x14, 0x5F, 0xCC, 0xC6, 0x99, 0xD3, 0xB7, 0x46,
0xBE, 0x98, 0x1E, 0xB6, 0x22, 0xE4, 0x6B, 0x81, 0x44, 0x91, 0x1E, 0x4A, 0x63, 0x87, 0xDB, 0xA8,
0x68, 0x94, 0x3E, 0x3E, 0x46, 0x6B, 0x0F, 0xE9, 0x1A, 0x89, 0x6E, 0x1E, 0xDE, 0x6B, 0x60, 0x17,
```

"Combine HEX" will combine the firmware validation hex, ip protection hex and will append the signature to it. The combined hex file will be store in the PC at DesignTool\TFLXTLS_Use_Cases\notebooks\firmware-validation\mergerd.hex

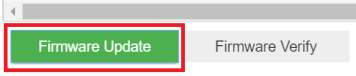
At this step, we have the combined image available for firmware validation update and verify operations. Both update and verify can be performed on the Notebook itself or on embedded projects. Refer to section 4.2 for instructions on embedded projects.

Update the firmware to product

Before verifying the firmware's validity, the firmware digest should be verified and stored to secure element. In this step host sends the encrypted firmware digest and signature to device to validate the firmware. Here the firmware is validated by verifying the signature using firmware signer's public key. Upon successful validation, the device stores the digest to Secureboot digest slot i.e. slot7.

```
def secureboot_verify(b):
    # Generating a random number to use
    host_random = os.urandom(32)
    is_verified = AtcaReference(False)
    # Perform Secureboot operation on the application file
    print('Perform Application validation request... ')
    assert atcab_secureboot_mac(SECUREBOOT_MODE_FULL_STORE, digest_verify, app_sign, host_random, io_prot_key, is_verified)
    if 1 == bool(is_verified.value):
        print('Secureboot Verify Success...')
        firmware_verify.button_style = 'success'
    else:
        firmware_verify.button_style = 'danger'
        print('Secureboot Verify failed...')

firmware_update.on_click(secureboot_update)
firmware_verify.on_click(secureboot_verify)
display(widgets.HBox((firmware_update, firmware_verify)))
```




```
Perform Application upgrade request...
Secureboot Update Success...
```

Clicking on "**Firmware Update"** will perform the above steps between host (PC) and the TrustFLEX device. Once firmware update is completed successfully, current firmware digest will be stored in the Secureboot digest slot.

Verifying the firmware image

This step recalculates the digest from the example bin (secureboot_test_app.bin). The encrypted digest will be sent to TrustFLEX. Upon successful validation, the device returns MAC value corresponding to this verify request.

```
def secureboot_verify(b):
    # Generating a random number to use
    host_random = os.urandom(32)
    is_verified = AtcaReference(False)
    # Perform Secureboot operation on the application file
    print('Perform Application validation request... ')
    assert atcab_secureboot_mac(SECUREBOOT_MODE_FULL_STORE, digest_verify, app_sign, host_random, io_prot_key, is_verified)
    if 1 == bool(is_verified.value):
        print('Secureboot Verify Success...')
        firmware_verify.button_style = 'success'
    else:
        firmware_verify.button_style = 'danger'
        print('Secureboot Verify failed...')

firmware_update.on_click(secureboot_update)
firmware_verify.on_click(secureboot_verify)
display(widgets.HBox((firmware_update, firmware_verify)))
```


```
Perform Application upgrade request...
Secureboot Update Success...
Perform Application validation request...
Secureboot Verify Success...
```

Clicking on "**Firmware Verify"** will perform the above steps between host (PC) and the TrustFLEX device.

Pressing “Firmware Update” and “Firmware Verify” should turn to green to indicate successful firmware update and verify operations.

4.2 Running Firmware Validation on Embedded platform

This usecase can also be executed on Embedded platform. Once the resources are generated as described in previous section, Atmel Studio project provided can be used to run the usecase on CryptoAuth Trust Platform.

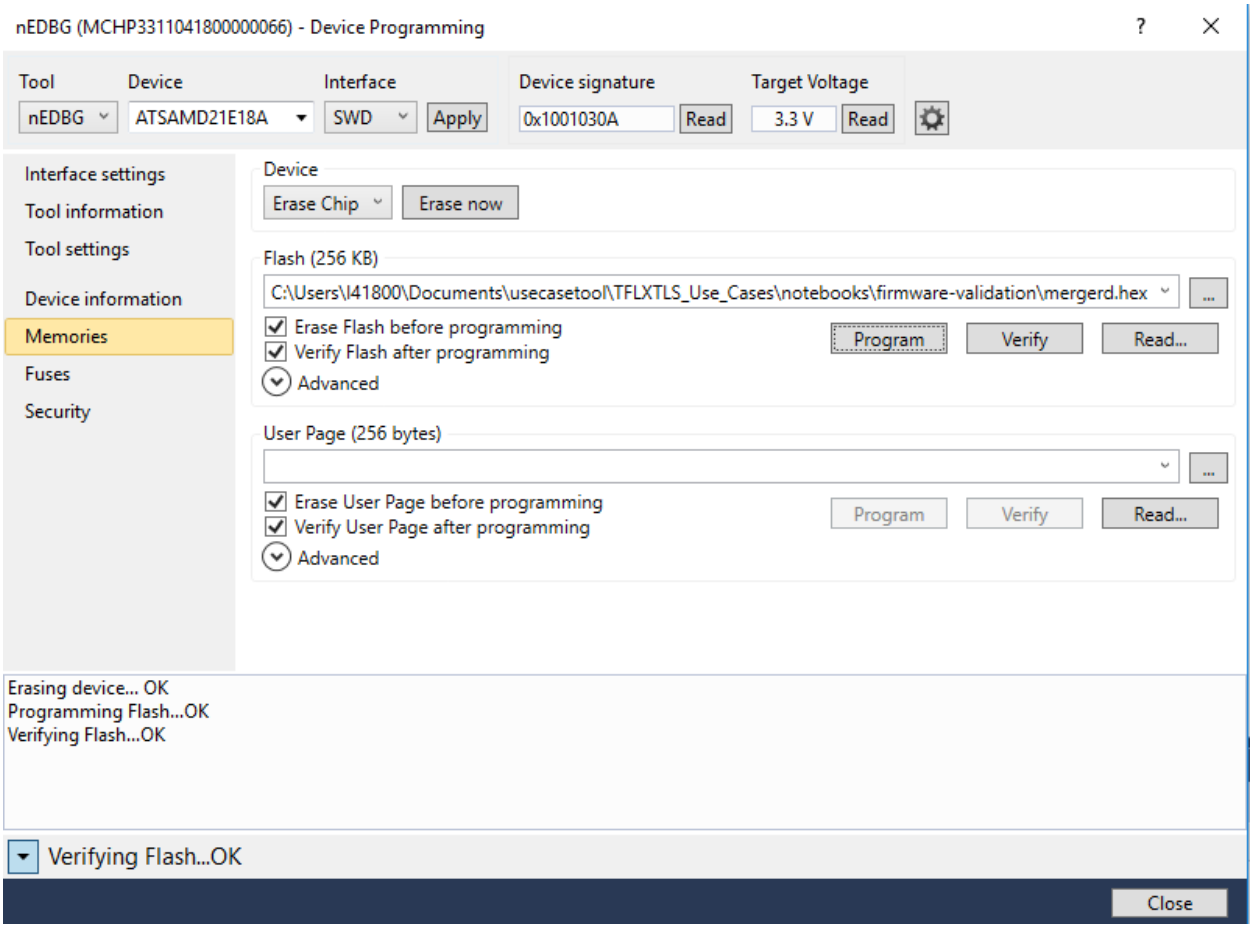
4.2.1 Atmel Studio:

All the necessary build steps are done as part of the previous steps. All that needs be done is to program the generated .hex file available at DesignTool\TFLXTLS_Use_Cases\notebooks\firmware-validation\mergerd.hex

The mergerd.hex contains firmware_validation, application images and the signature.

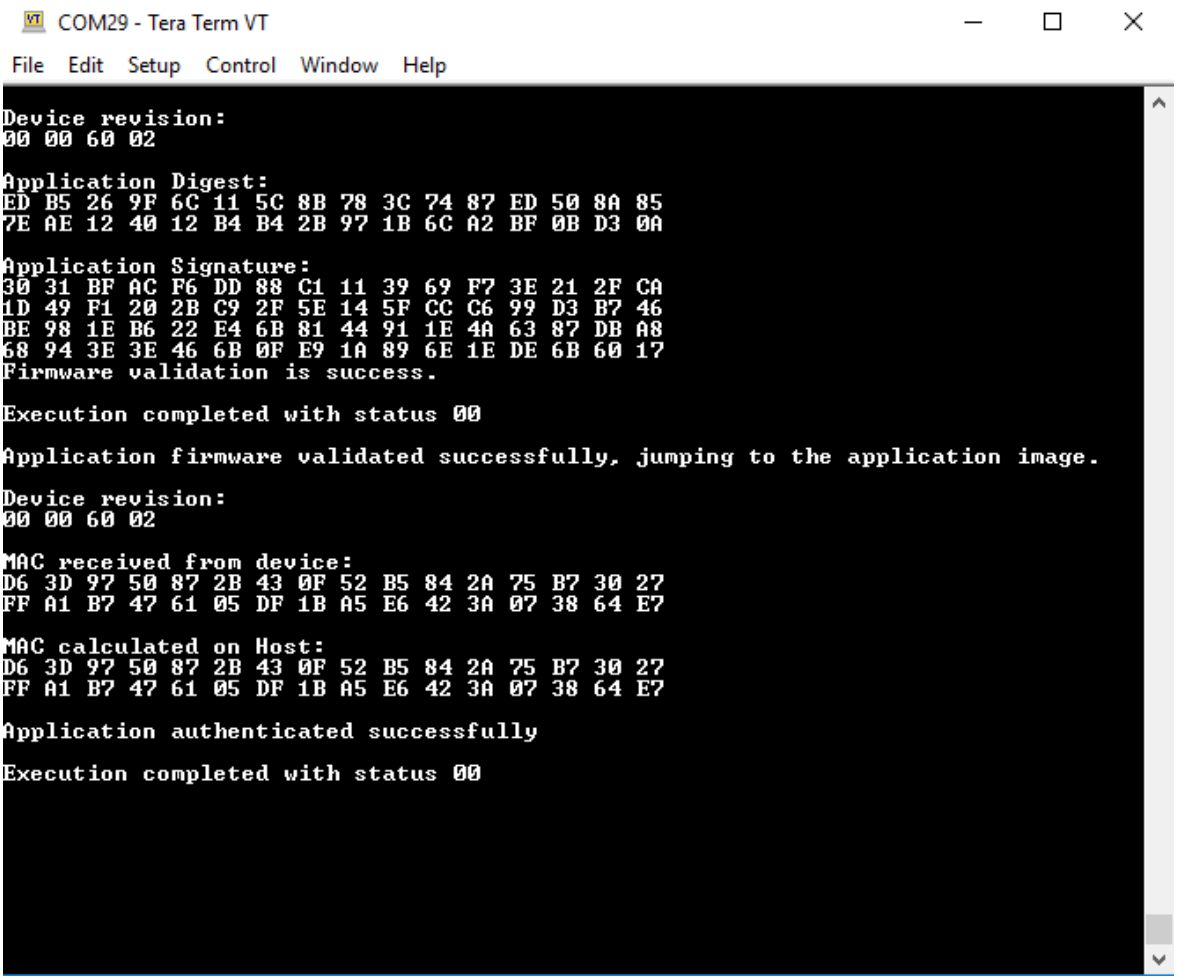
To program using Atmel Studio:

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. Select mergerd.hex file
5. Check both Erase Flash and Verify Flash
6. Click on Program



The program output can be viewed using a serial terminal. Terminal needs to be opened with 115200-8-N-1 settings on TrustPlatform's serial port.

Output on the serial terminal would look like the image below,



On any error, LED blinks five times every second.

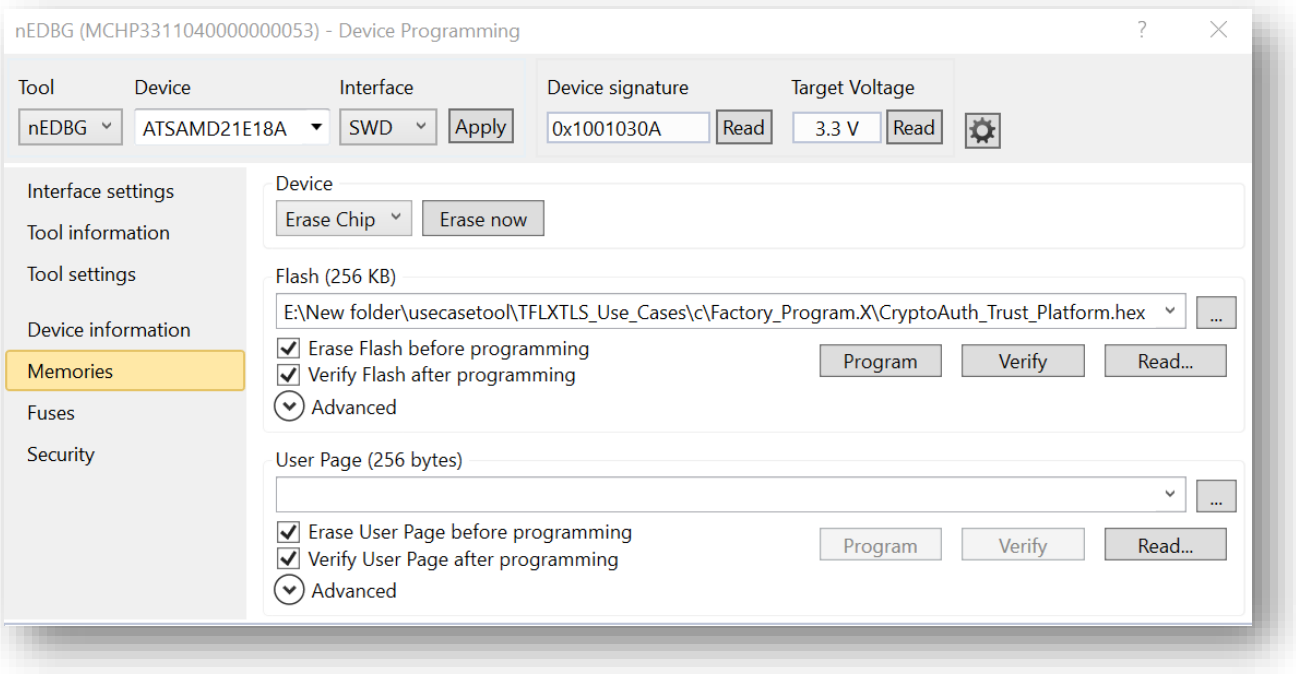
4.3 CryptoAuth TrustPlatform kit Factory reset

Once any of the embedded project is loaded to CrytoAuth TrustPlatform, the default program that enables interaction with TrustPlatform 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
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, Crypto Trust Platform contains factory programmed 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.

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 is firmware validation project fails with error "Firmware validation is failed! with status 01"?**
   There are many possibilities like,
   a. The resources on TrustFLEX device and on the host (PC) could be different. Rerun "Resource Generation Notebook" section for reloading it.
   b. Firmware digest is not matched. Make sure that firmware Update step is exec uted using Notebook prior to running 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