Provisioning and Secure Bootloader User Guide

This guide will explain the necessary steps to perform provisioning and to program your board with a bootable image. Please keep in mind, that **provisioning must be done at least once** (after that the secure keys are stored in the Secure Element). Without provisioning your board, the application will be stuck in a reset loop as it will refuse to run without the proper keys.

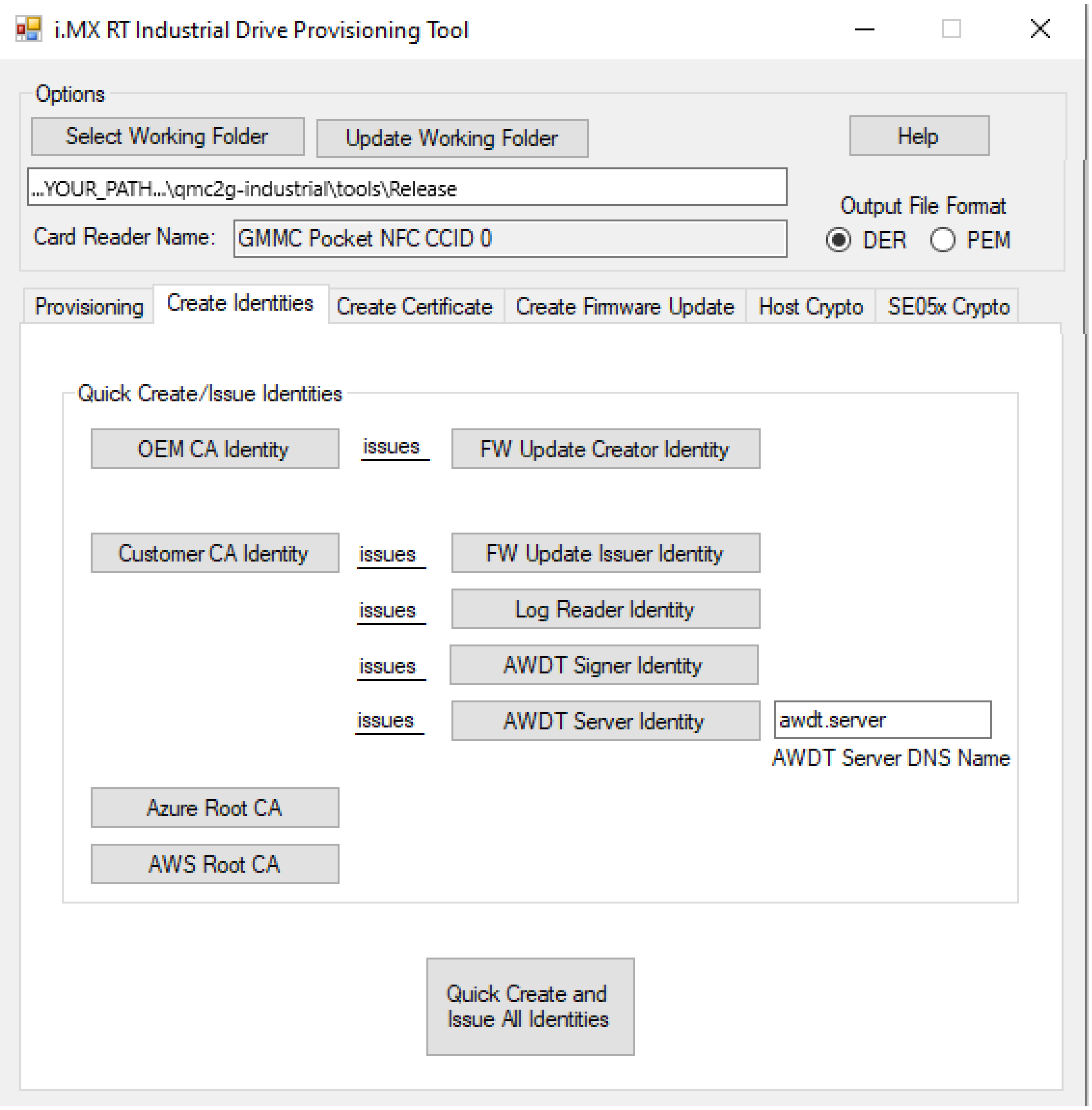
Contents

Provisioning

To provision the on-board Secure Element, you will need to make use of the NFC reader included in your digital board package. Provisioning takes several steps, as described below. If you wish to make use of the full security features provided with the application, please obtain a ACS ACR1252 1S CL Reader PICC 0, open /tools/Release/Qmc2gProvisioningTool.ini, comment out the default CARD READER and uncomment the ACS ACR1252 and skip steps 10 to 14 of this sub-chapter. There will be an alternative path described at the end of the sub-chapter.

1. Power off the HW to enable provisioning via NFC.

2. Insert the NFC reader into your PC and let it install. After successful installation the LED will be blinking red/blue/white color.
3. Go into the **“tools”** folder located at **root** directory and unzip the **“ProvisioningTool.zip”** archive into the **“tools”** folder. **The zip password is “123”.**
4. Run **“ProvisioningTool.exe”.** All files will be placed in and taken from the working folder. You can change the working folder according to your preference (Need to copy image_enc.exe into your customized working folder).
5. Go to the “Create Identities” tab and press “Quick Create and Issue All Identities”. All generated content will be placed into your working folder.



i.MX RT Industrial Drive Provisioning Tool

...YOUR_PATH...\qmc2g-industrial\tools\Release

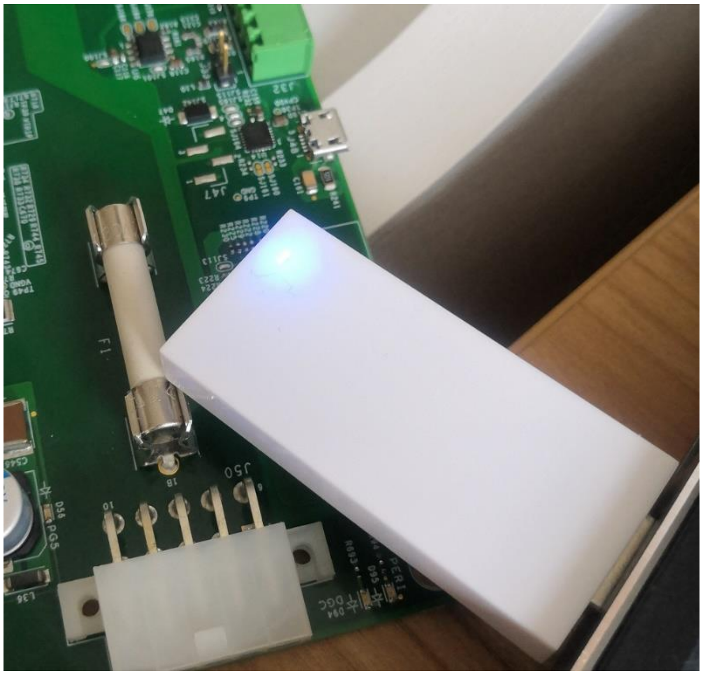
Card Reader Name: GMMC Pocket NFC CCID 0

Output File Format ● DER ○ PEM

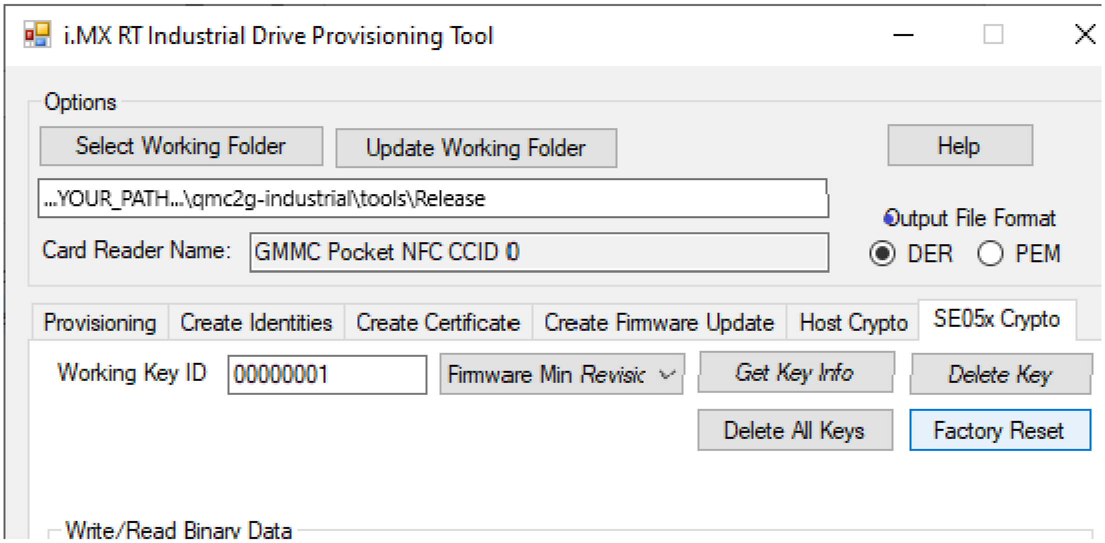
awdt.server
AWDT Server DNS Name

6. Go to the **“SE05x_Crypto”** tab.
7. Take the unpowered Digital board and place the board by the NFC antenna below the NFC reader. The LED color will change to white once a connection is established. Make sure the NFC

reader is positioned properly on top of the digital board's NFC antenna, to ensure a strong and uninterrupted connection.



8. In the provisioning tool press the Factory reset button while having the NFC reader connected to the Secure Element on the Digital Board.



i.MX RT Industrial Drive Provisioning Tool

Options

...YOUR_PATH...\qmc2g-industrial\tools\Release

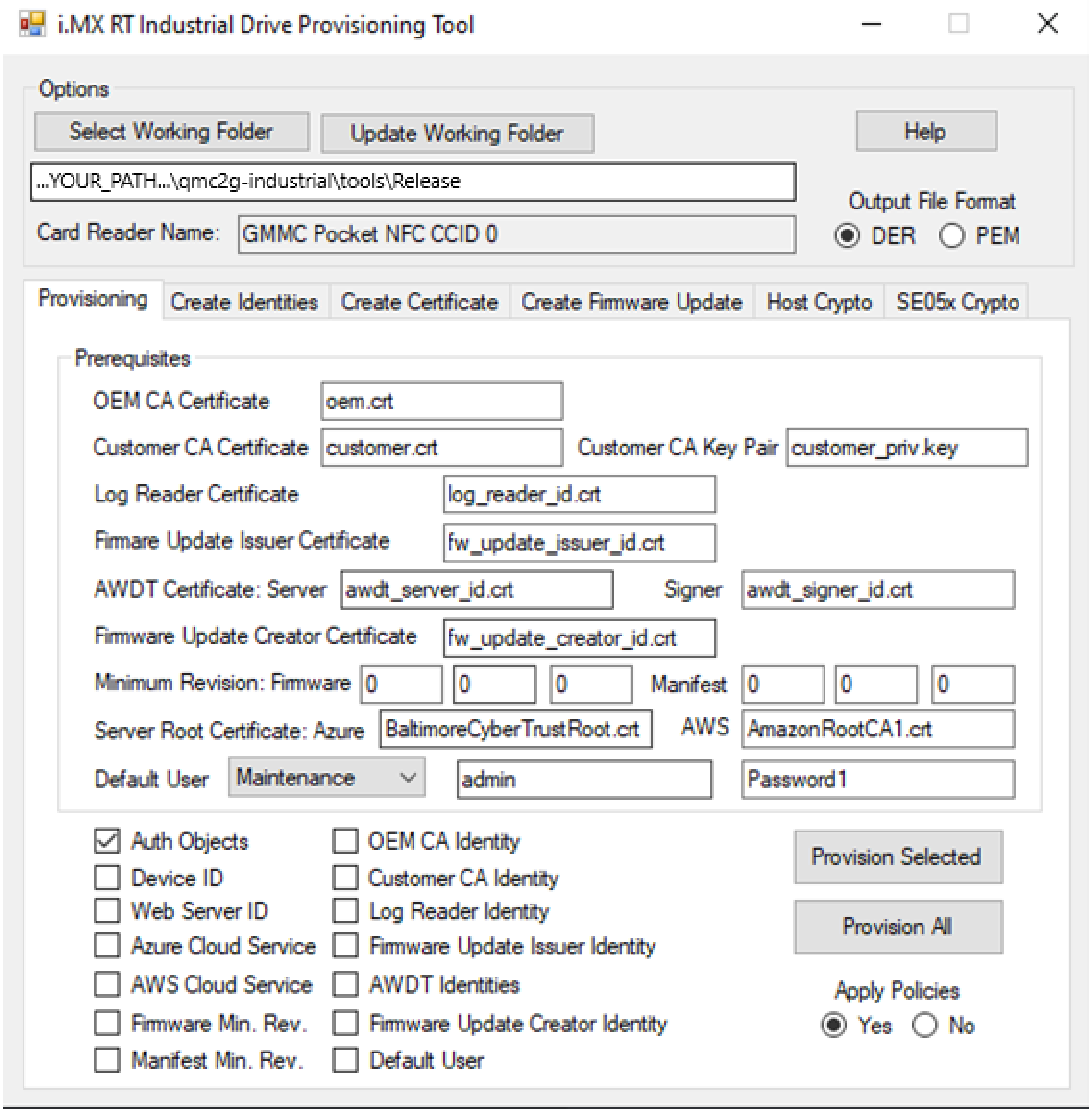
Card Reader Name: GMMC Pocket NFC CCID 0

Output File Format ● DER ○ PEM

Working Key ID 00000001

Firmware Min Revisic

9. Go to the "Provisioning" tab to finally provision the Secure Element with security assets. The provisioning steps must be done in these steps as policies for secure objects have not been fully defined yet.
10. Apply Policies using the "Yes" radio button and select "Auth Objects". Press the "Provision Selected" button while holding the NFC Reader connected the Secure Element on the Digital Board.



...YOUR_PATH...\qmc2g-industrial\tools\Release

Card Reader Name: GMMC Pocket NFC CCID 0

Output File Format ● DER ○ PEM

Prerequisites

OEM CA Certificate: oem.crt

Customer CA Certificate: customer.crt   Customer CA Key Pair: customer_priv.key

Log Reader Certificate: log_reader_id.crt

Firmare Update Issuer Certificate: fw_update_issuer_id.crt

AWDT Certificate: Server: awdt_server_id.crt   Signer: awdt_signer_id.crt

Firmware Update Creator Certificate: fw_update_creator_id.crt

Minimum Revision: Firmware 0 0 0   Manifest 0 0 0

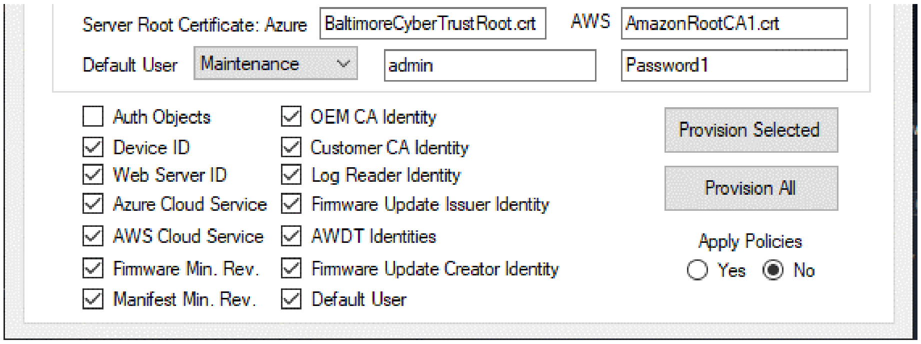
Server Root Certificate: Azure BaltimoreCyberTrustRoot.crt   AWS AmazonRootCA1.crt

Default User: Maintenance   admin   Password1

☑ Auth Objects   □ OEM CA Identity
□ Device ID   □ Customer CA Identity
□ Web Server ID   □ Log Reader Identity
□ Azure Cloud Service   □ Firmware Update Issuer Identity
□ AWS Cloud Service   □ AWDT Identities
□ Firmware Min. Rev.   □ Firmware Update Creator Identity
□ Manifest Min. Rev.   □ Default User

Apply Policies ● Yes ○ No

11. Deselect “Auth Objects” and set Apply Policies to “NO”.
12. Select all remaining checkboxes.
13. Press the “Provision Selected” button again, while still holding the NFC reader connected to the Secure Element on the Digital Board. It takes few seconds to complete the operation.



Server Root Certificate: Azure BaltimoreCyberTrustRoot.crt   AWS AmazonRootCA1.crt

Default User: Maintenance   admin   Password1

□ Auth Objects   ☑ OEM CA Identity
☑ Device ID   ☑ Customer CA Identity
☑ Web Server ID   ☑ Log Reader Identity
☑ Azure Cloud Service   ☑ Firmware Update Issuer Identity
☑ AWS Cloud Service   ☑ AWDT Identities
☑ Firmware Min. Rev.   ☑ Firmware Update Creator Identity
☑ Manifest Min. Rev.   ☑ Default User

Apply Policies ○ Yes ● No

14. The Secure Element on the Digital board is provisioned and you are ready to program your board and try out the application!

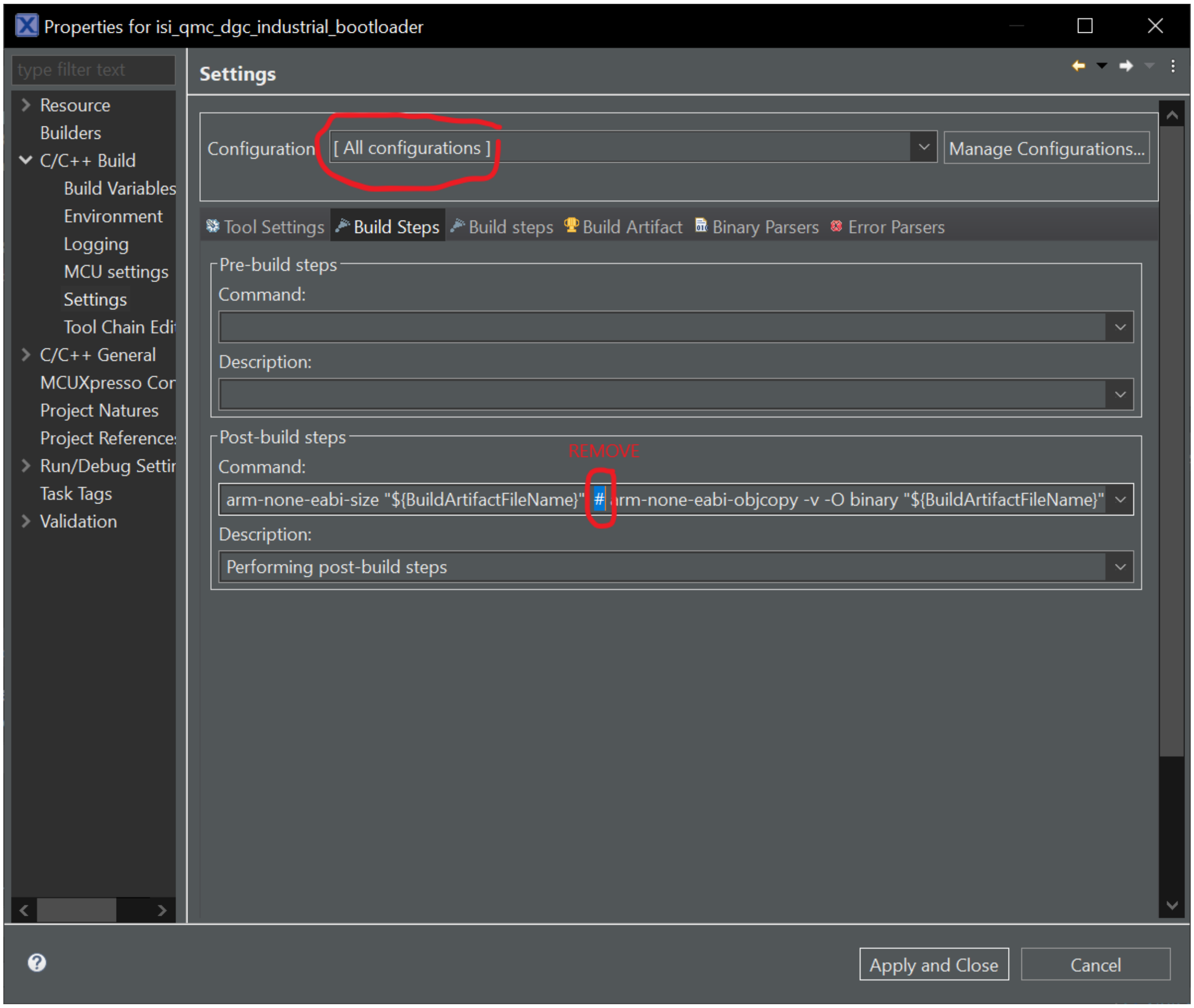
OPTIONAL STEP FOR ACS ACR1252 1S CL Reader PICC 0:

10. Apply Policies using the “Yes” radio button and press the “Provision All” button while holding the NFC Reader connected to the Secure Element on the Digital Board.

11. The Secure Element on the Digital board is provisioned and you are ready to program your board and try out the application!

Preparing the binaries

1. Open the QMC2G project in MCUXpresso.
2. Open the project properties window for the CM7 and Bootloader projects, go to *Properties > C/C++ Build > Settings > Build Steps* and remove the **#** symbol in front of *arm-none-eabi-objcopy*, so that *.bin file creation, needed later, is enabled (make sure to do it for all configurations):



Properties for isi_qmc_dgc_industrial_bootloader

Settings

Configuration [All configurations]

Pre-build steps

Command:

Description:

Post-build steps

REMOVE

Command:

```
arm-none-eabi-size "${BuildArtifactFileName}" # arm-none-eabi-objcopy -v -O binary "${BuildArtifactFileName}"
```

Description: Performing post-build steps

3. Compile the **isi_qmc_dgc_industrial_bootloader** project using **Debug_Non_Secure** or **Debug (Secure)** target for debugging, development and evaluation or **Release (Secure)** for production.
4. Next, compile the CM4 and then the CM7 project using **the Release_SBL** targets.

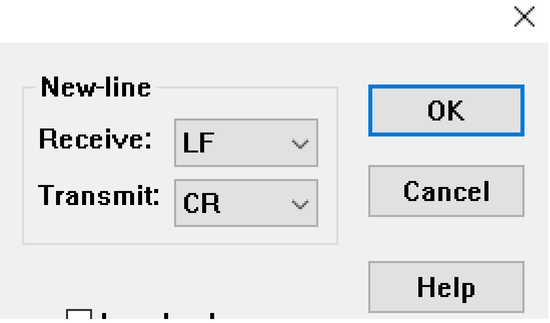
5. Go into the **"tools"** folder located in the application SW pack directory, which contains two folders with batch files:
   a. **development-unsecured** contains batch files intended for the development phase:
      i. **blhost_usb_cmd_xip_app.bat –** programs the board with both the main FW and SBL with disabled security. **(Edit the file paths on lines 78 and 79 to point to your binaries.)**
      ii. **blhost_usb_cmd_xip_app_secure.bat –** programs the SBL with enabled security and all essential fuses + PUF KeyStore. The main FW will be programmed by the SBL from the SD card. **(Edit the file path on line 111 to point to your binary.)**
      iii. **blhost_usb_erase_octal_flash.bat** – erases the octal flash.

   b. **production-secured/5th_step** contains batch files for production programming and enabling of all device and application security. It requires steps 1-4 described later in this document to be performed first.
      i. **blhost_hab_enc_xip_release.bat** - programs the SBL with a signed and encrypted image with enabled security and all essential fuses for enabling HAB, Encrypted XiP, PUF KeyStore and disabling the debug port. The main FW can be programmed by SBL from the SD card or directly placed into the FWU location in the octal flash (default). **(Edit the file path on line 45 to point to your binary folder.)**

Non-secure Bootloader: blhost_usb_cmd_xip_app.bat

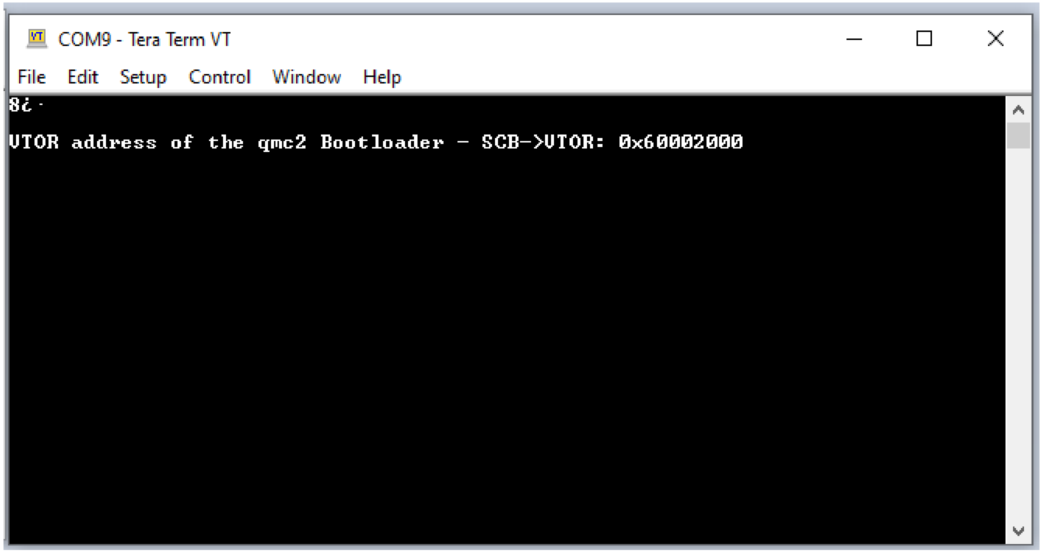
Note: if you wish to use the Debug_SBL targets for CM4 and CM7, edit the path in the blhost_usb_cmd_xip_app.bat file on line 79.

1. Configure the switch SW4 on the daughter card to SDP mode (1-ON/2-OFF/3-OFF/4-OFF).
2. Connect a micro-USB cable to J3 on the daughter card and press reset (SW2).
3. Execute the **blhost_usb_cmd_xip_app.bat.**
4. Configure the switch SW4 on the daughter card to Boot from fuses mode (1-OFF/2-OFF/3-OFF/4-OFF).
5. Connect a micro-USB cable to J48 on the digital board.
6. Power up the board if needed.
7. Use an application to establish serial communication for both COM ports (115200) (Like PuTTY or Tera Term). See the picture below for the recommended line-ending settings:

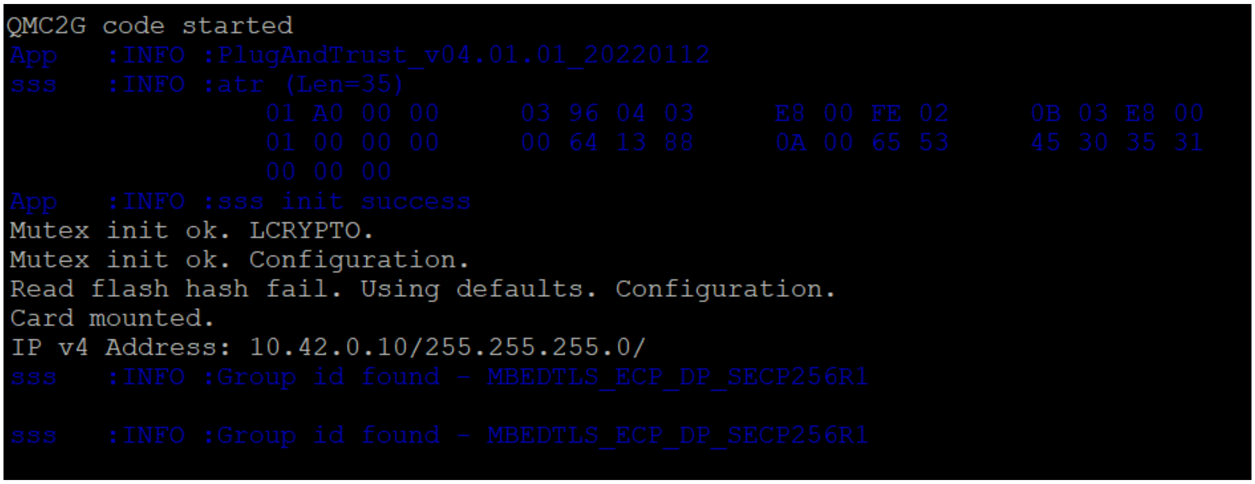


New-line
Receive: LF
Transmit: CR

8. Press the reset button on the daughter card (SW2) and you should see similar prints in the consoles after ~5 seconds:



```
8¿ ·
VTOR address of the qmc2 Bootloader - SCB->VTOR: 0x60002000
```

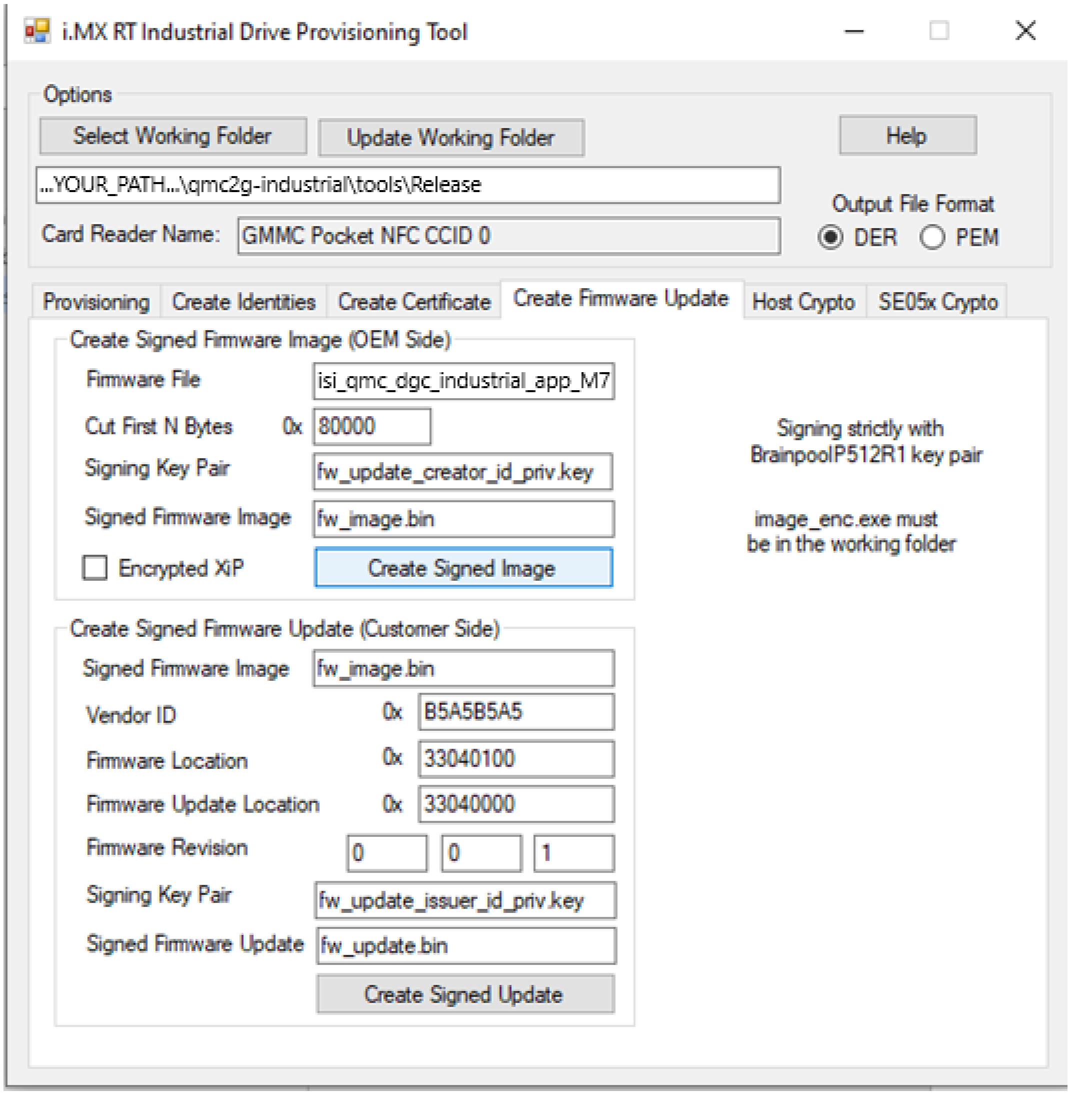


```
QMC2G code started
App   :INFO :PlugAndTrust_v04.01.01_20220112
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :sss init success
Mutex init ok. LCRYPTO.
Mutex init ok. Configuration.
Read flash hash fail. Using defaults. Configuration.
Card mounted.
IP v4 Address: 10.42.0.10/255.255.255.0/
sss   :INFO :Group id found - MBEDTLS_ECP_DP_SECP256R1

sss   :INFO :Group id found - MBEDTLS_ECP_DP_SECP256R1
```

Secure Bootloader: blhost_usb_cmd_xip_app_secure.bat

1. First, the Secure FW update image can be generated. Copy the "**Release_SBL\isi_qmc_dgc_industrial_app_M7MASTER.bin**" into your Provisioning Tool working folder where all the security assets were generated.
2. Go into the "Create Firmware Update" tab.
3. Deselect "Encrypted XiP" as we do not use this feature for now.
4. Increase Manifest firmware revision. It must be greater than minimum Manifest revision configured during the provisioning step (default 0.0.0).



i.MX RT Industrial Drive Provisioning Tool

Options

...YOUR_PATH...\qmc2g-industrial\tools\Release

Card Reader Name: GMMC Pocket NFC CCID 0

Output File Format ● DER ○ PEM

Create Signed Firmware Image (OEM Side)

Firmware File: isi_qmc_dgc_industrial_app_M7

Cut First N Bytes 0x 80000

Signing Key Pair: fw_update_creator_id_priv.key

Signed Firmware Image: fw_image.bin

□ Encrypted XiP

Signing strictly with BrainpoolP512R1 key pair

image_enc.exe must be in the working folder

Create Signed Firmware Update (Customer Side)

Signed Firmware Image: fw_image.bin

Vendor ID 0x B5A5B5A5

Firmware Location 0x 33040100

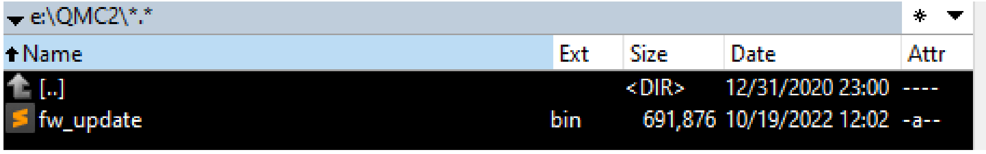
Firmware Update Location 0x 33040000

Firmware Revision 0 0 1

Signing Key Pair: fw_update_issuer_id_priv.key

Signed Firmware Update: fw_update.bin

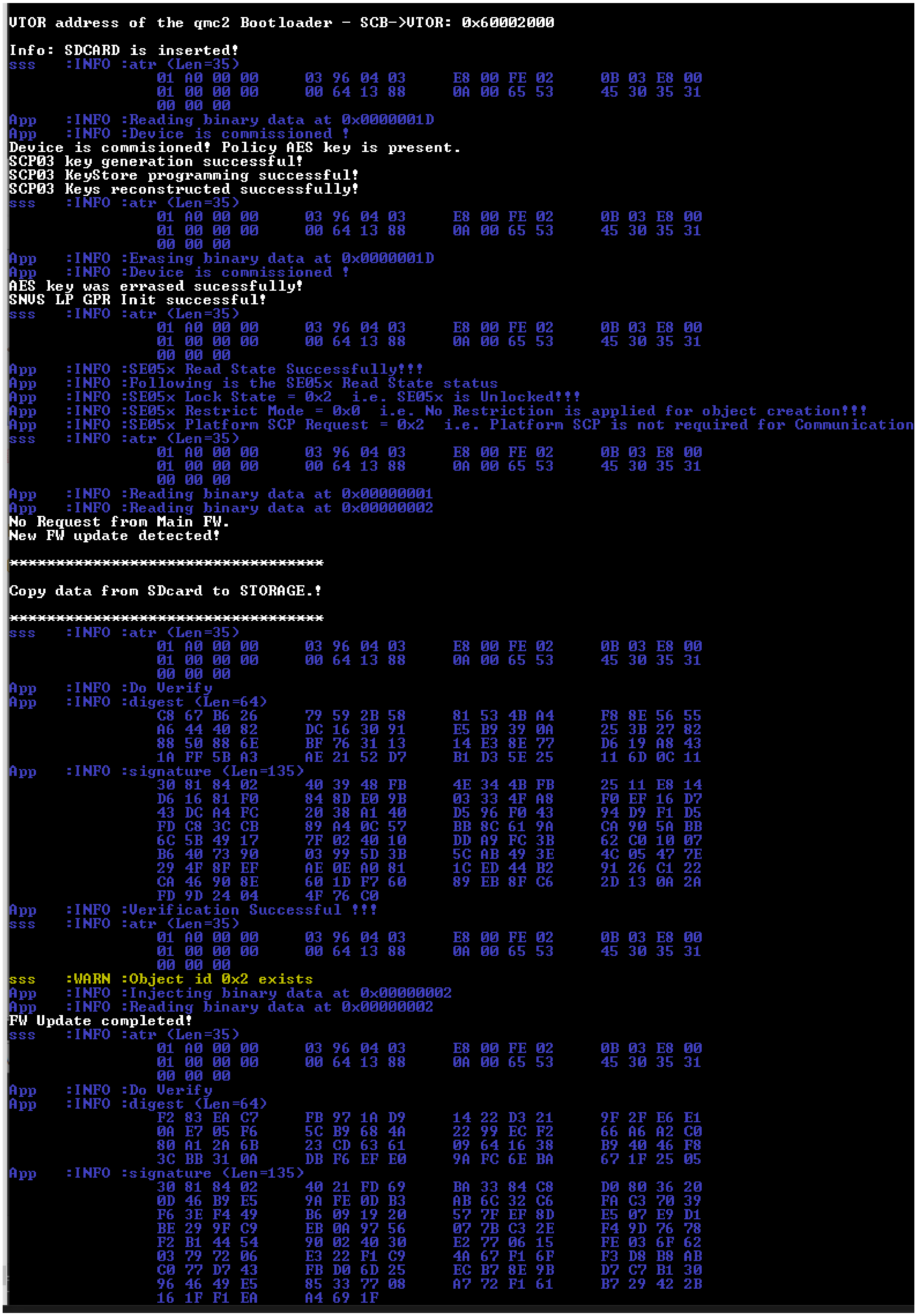
5. Press the **"Create Signed Image"** button.
6. Press the **"Create Signed Update"** button.
7. Copy the resulting **"fw_update.bin"** from your working folder to the SD card into the **"QMC2" folder**



e:\QMC2\*.*

Name	Ext	Size	Date	Attr
[..]		<DIR>	12/31/2020 23:00	----
fw_update	bin	691,876	10/19/2022 12:02	-a--

8. Insert the SD card into the Daughter card’s SD card slot (located below the battery slot).
9. You can now close the Provisioning Tool.
10. Configure the SW4 switch on the daughter card to SDP mode (1-ON/2-OFF/3-OFF/4-OFF).
11. Connect a micro-USB cable to J3 on the daughter card and press reset (SW2).
12. Execute the **blhost_usb_cmd_xip_app_secure.bat. The script will program all the essential fuses, PUF keystore and the secure version of SBL.**
13. Configure the switch SW4 on the daughter card to Boot from fuses mode (1-OFF/2-OFF/3-OFF/4-OFF).
14. Connect a micro-USB cable to the J48 on the digital board.
15. Power up the board if needed.
16. Use an application to establish serial communication for both COM ports (115200) (Like PuTTY or Tera Term).
17. Press the reset button on the daughter card (SW2) and you should see similar prints in the consoles after ~5 seconds:



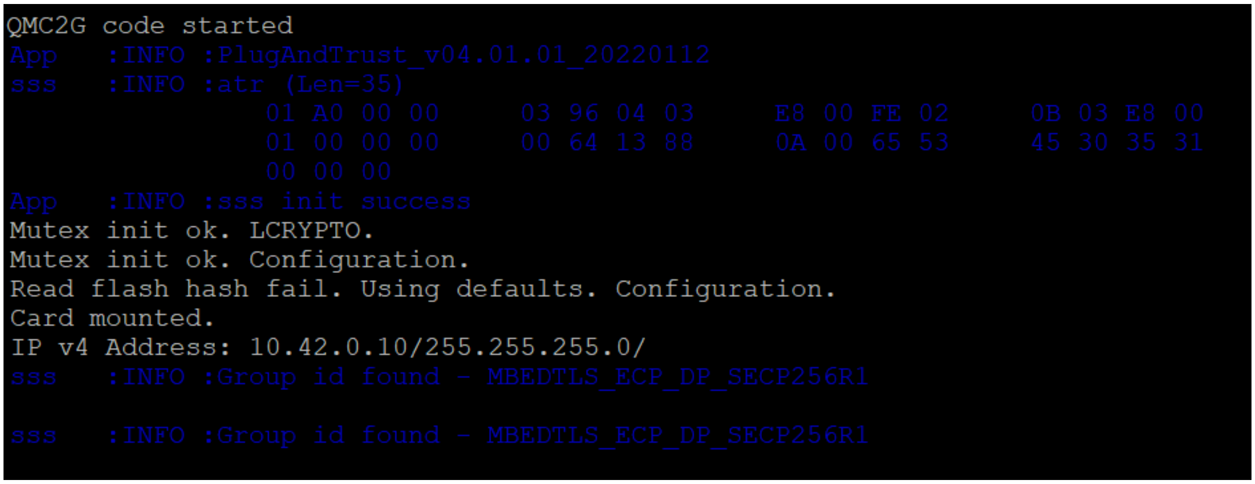
```
VTOR address of the qmc2 Bootloader - SCB->VTOR: 0x60002000

Info: SDCARD is inserted!
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :Reading binary data at 0x0000001D
App   :INFO :Device is commissioned !
Device is commisioned! Policy AES key is present.
SCP03 key generation successful!
SCP03 KeyStore programming successful!
SCP03 Keys reconstructed successfully!
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :Erasing binary data at 0x0000001D
App   :INFO :Device is commissioned !
AES key was errased sucessfully!
SNVS LP GPR Init successful!
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :SE05x Read State Successfully!!!
App   :INFO :Following is the SE05x Read State status
App   :INFO :SE05x Lock State = 0x2  i.e. SE05x is Unlocked!!!
App   :INFO :SE05x Restrict Mode = 0x0  i.e. No Restriction is applied for object creation!!!
App   :INFO :SE05x Platform SCP Request = 0x2  i.e. Platform SCP is not required for Communication
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :Reading binary data at 0x00000001
App   :INFO :Reading binary data at 0x00000002
No Request from Main FW.
New FW update detected!

**********************************

Copy data from SDcard to STORAGE.!

**********************************
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :Do Verify
App   :INFO :digest (Len=64)
                C8 67 B6 26     79 59 2B 58     81 53 4B A4     F8 8E 56 55
                A6 44 40 82     DC 16 30 91     E5 B9 39 0A     25 3B 27 82
                88 50 88 6E     BF 76 31 13     14 E3 8E 77     D6 19 A8 43
                1A FF 5B A3     AE 21 52 D7     B1 D3 5E 25     11 6D 0C 11
App   :INFO :signature (Len=135)
                30 81 84 02     40 39 48 FB     4E 34 4B FB     25 11 E8 14
                D6 16 81 F0     84 8D E0 9B     03 33 4F A8     F0 EF 16 D7
                43 DC A4 FC     20 38 A1 40     D5 96 F0 43     94 D9 F1 D5
                FD C8 3C CB     89 A4 0C 57     BB 8C 61 9A     CA 90 5A BB
                6C 5B 49 17     7F 02 40 10     DD A9 FC 3B     62 C0 10 07
                B6 40 73 90     03 99 5D 3B     5C AB 49 3E     4C 05 47 7E
                29 4F 8F EF     AE 0E A0 81     1C ED 44 B2     91 26 C1 22
                CA 46 90 8E     60 1D F7 60     89 EB 8F C6     2D 13 0A 2A
                FD 9D 24 04     4F 76 C0
App   :INFO :Verification Successful !!!
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
sss   :WARN :Object id 0x2 exists
App   :INFO :Injecting binary data at 0x00000002
App   :INFO :Reading binary data at 0x00000002
FW Update completed!
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App   :INFO :Do Verify
App   :INFO :digest (Len=64)
                F2 83 EA C7     FB 97 1A D9     14 22 D3 21     9F 2F E6 E1
                0A E7 05 F6     5C B9 68 4A     22 99 EC F2     66 A6 A2 C0
                80 A1 2A 6B     23 CD 63 61     09 64 16 38     B9 40 46 F8
                3C BB 31 0A     DB F6 EF E0     9A FC 6E BA     67 1F 25 05
App   :INFO :signature (Len=135)
                30 81 84 02     40 21 FD 69     BA 33 84 C8     D0 80 36 20
                0D 46 B9 E5     9A FE 0D B3     AB 6C 32 C6     FA C3 70 39
                F6 3E F4 49     B6 09 19 20     57 7F EF 8D     E5 07 E9 D1
                BE 29 9F C9     EB 0A 97 56     07 7B C3 2E     F4 9D 76 78
                F2 B1 44 54     90 02 40 30     E2 77 06 15     FE 03 6F 62
                03 79 72 06     E3 22 F1 C9     4A 67 F1 6F     F3 D8 B8 AB
                C0 77 D7 43     FB D0 6D 25     EC B7 8E 9B     D7 C7 B1 30
                96 46 49 E5     85 33 77 08     A7 72 F1 61     B7 29 42 2B
                16 1F F1 EA     A4 69 1F
```



```
QMC2G code started
App    :INFO :PlugAndTrust_v04.01.01_20220112
sss    :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
App    :INFO :sss init success
Mutex init ok. LCRYPTO.
Mutex init ok. Configuration.
Read flash hash fail. Using defaults. Configuration.
Card mounted.
IP v4 Address: 10.42.0.10/255.255.255.0/
sss    :INFO :Group id found - MBEDTLS_ECP_DP_SECP256R1

sss    :INFO :Group id found - MBEDTLS_ECP_DP_SECP256R1
```

Signed and Encrypted Secure Bootloader: blhost_hab_enc_xip_release.bat

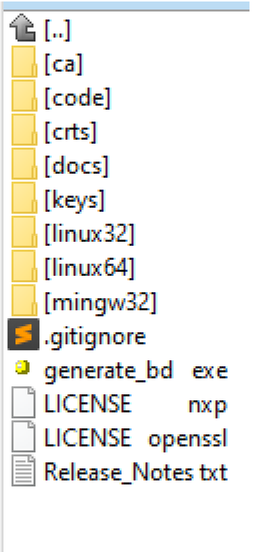
This batch file located in the **“tools\production-secured\5th_step\evkmimxrt1170\qspiflash\”** shall be used for production programming. It sets the device to OEM_CLOSED by enabling HAB which ensures authenticity and integrity. SBL confidentiality is achieved by Encrypted XiP mode also enabled within this script. Moreover, the script disables the debug port entirely.

Be aware that after this script is executed, only applications signed by the corresponding private key can be executed. Access via the debugger will not be possible anymore.

The following sub-chapters describe all the steps necessary to enable HAB. The files referenced in the following sub-chapters are located in the “**tools\production-secured\**” folder.

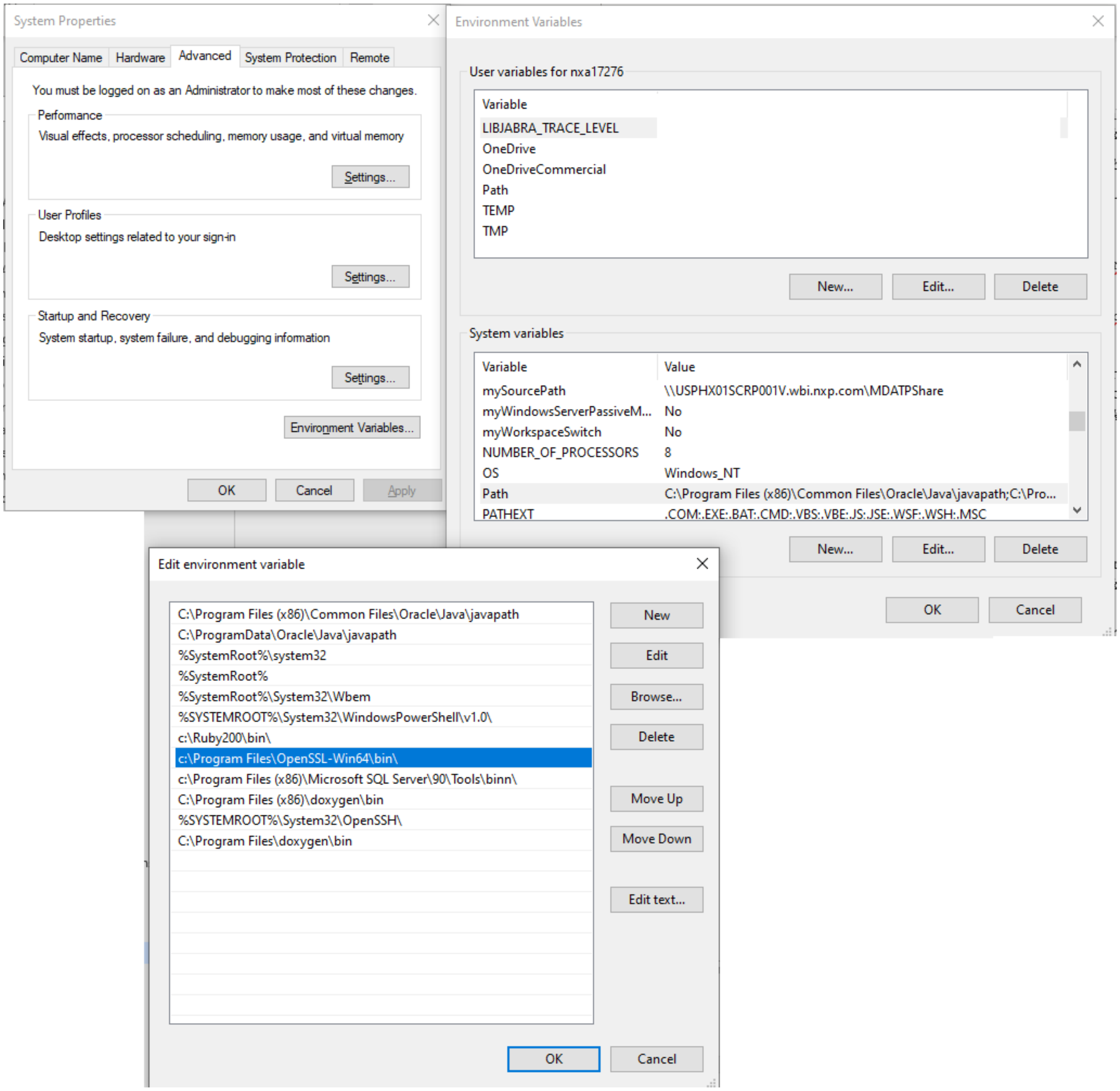
HAB Enablement - 1st step

1. Download the **cst-3.1.0.tgz** package from https://www.nxp.com/webapp/sps/download/license.jsp?colCode=IMX_CST_TOOL
2. Copy the content of the **cst-3.1.0.tgz** package into the **“tools\utils\evkmimxrt1170\cst\”** folder:



```
[..]
[ca]
[code]
[crts]
[docs]
[keys]
[linux32]
[linux64]
[mingw32]
.gitignore
generate_bd   exe
LICENSE       nxp
LICENSE  openssl
Release_Notes txt
```

3. Download and install OpenSSL from this link.
4. Add the following bin folder path into your system variables. For example, the default installation path is **c:\ProgramFiles\OpenSSL-Win64\bin\**



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

User variables for nxa17276

Variable
LIBJABRA_TRACE_LEVEL
OneDrive
OneDriveCommercial
Path
TEMP
TMP

System variables

Variable	Value
mySourcePath	\\USPHX01SCRP001V.wbi.nxp.com\MDATPShare
myWindowsServerPassiveM...	No
myWorkspaceSwitch	No
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Pro...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

```
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
c:\Ruby200\bin\
c:\Program Files\OpenSSL-Win64\bin\
c:\Program Files (x86)\Microsoft SQL Server\90\Tools\binn\
C:\Program Files (x86)\doxygen\bin
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\doxygen\bin
```

HAB Enablement - 2nd step

This step uses batch files that generate keys and certificates and covers ECDSA algorithms. For a given device you should only go through this process one time. In this step you'll use the "**production-secured\2nd_step\evkmimxrt1170\ecc-p521\key_gen _ecc.bat"** file that does the following actions:

- Runs **hab4_pki_tree.bat** to generate new key pairs.
- Runs **srktool** to create a key table and hash value to blow into fuses.
- Modifies the **enable_hab.bd** file to use the generated hash values.
- Creates the **enable_hab.sb** file for the SRKH and SEC_CONFIG fuses programming.

During the script execution following, several questions will be asked:

1. Do you want to use an existing CA key (y/n) ?: **n**
2. Do you want to use Elliptic Curve Cryptography (y/n) ?: **y**
    i. n – if RSA is desired.
3. Enter key length in bits for PKI tree**: p521**
    i. 4096 – if RSA is desired.
4. Enter PKI tree duration (years): **10**
    i. Expiration is not verified.
5. How many Super Root Keys should be generated? **4**
    i. It is recommended to generate 4 keys at once to support SRK revoke feature.
6. Do you want the SRK certificates to have the CA flag set? (y/n) ?: **y**
    i. n – is for fast boot. IMG and CSF keys won't be generated in this case.

NOTE

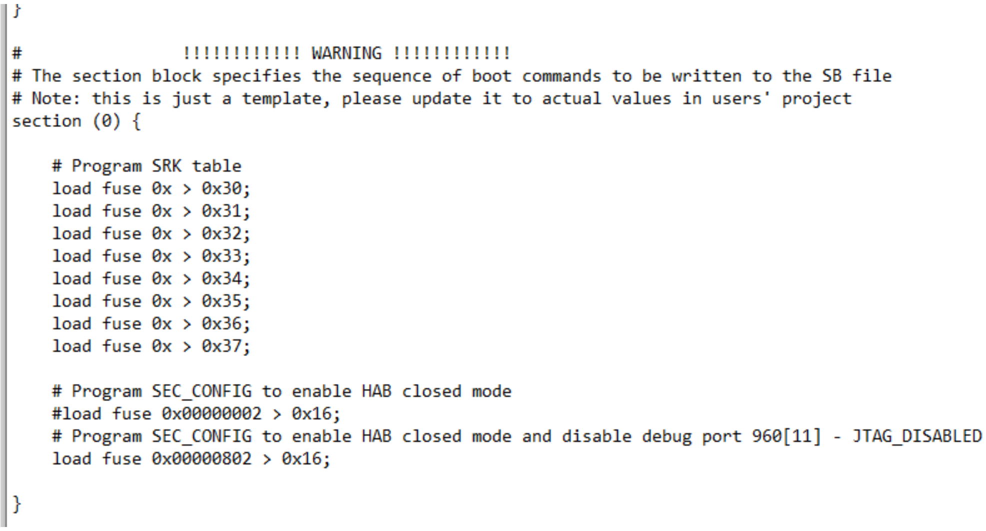
If different key length than p521 is desired, bd files and batch files must be updated accordingly.

- Certificates with public keys will be generated in the *„Tools\utils\evkmimxrt1170\cst\crts"* folder.
- Key files with private keys will be generated in the *„Tools\utils\evkmimxrt1170\cst\keys"* folder.
- The SRK_1_2_3_4_fuse.bin and SRK_1_2_3_4_table.bin files will be created in the *„Tools\utils\evkmimxrt1170\cst\keys"* folder.
- In the utils/bd_file/imx10xx folder enable_hab.bd is generated/modified to use the fuse values from SRK_1_2_3_4_fuse.bin
- The enable_hab.sb and file will be generated in the *„Tools\utils\evkmimxrt1170\*" folder.

HAB Enablement - 3rd step

The batch file in this step programs SRKH, JTAG_DISABLE and SEC_CONFIG into fuses. By the end of this step, HAB will be configured into the closed mode. It means that only images signed with the valid private key will be executed. It is also mandatory for applications programmed via SDP, such as NXP Flashloader, which is why the signed NXP Flashloader must be used in the next steps. To program SRKH, JTAG_DISABLE and SEC_CONFIG fuses the **„program_srkh_sec_conf.bat“** must be executed.

The following template has been updated to also disable the debug port:



```
}

#               !!!!!!!!!!!! WARNING !!!!!!!!!!!!
# The section block specifies the sequence of boot commands to be written to the SB file
# Note: this is just a template, please update it to actual values in users' project
section (0) {

    # Program SRK table
    load fuse 0x > 0x30;
    load fuse 0x > 0x31;
    load fuse 0x > 0x32;
    load fuse 0x > 0x33;
    load fuse 0x > 0x34;
    load fuse 0x > 0x35;
    load fuse 0x > 0x36;
    load fuse 0x > 0x37;

    # Program SEC_CONFIG to enable HAB closed mode
    #load fuse 0x00000002 > 0x16;
    # Program SEC_CONFIG to enable HAB closed mode and disable debug port 960[11] - JTAG_DISABLED
    load fuse 0x00000802 > 0x16;

}
```

NOTE

The board must be in SDP (SW4 to ON-OFF-OFF-OFF and power cycle) mode prior executing the batch file.

HAB Enablement - 4th step

This step is used for signature verification using the NXP Flashloader. The Flashloader can be signed using the following script. This step only needs to be performed once for a given key set.

- **sign_flashloader_ecc.bat**
  - Using keys and certificates for ECDSA

After the NXP Flashloader is signed, Blhost commands can be used to verify the signature. The following script can be used for verification.

- **setup_flash_usb_sec.bat**
  - If the signature is valid. The NXP Flashloader will be executed and Blhost commands issued. Make sure that the batch file is executed from the Command Prompt (CMD) in order to issue Blhost commands.

NOTE

The board must be in SDP (SW4 to ON-OFF-OFF-OFF and power cycle) mode prior executing the batch file.

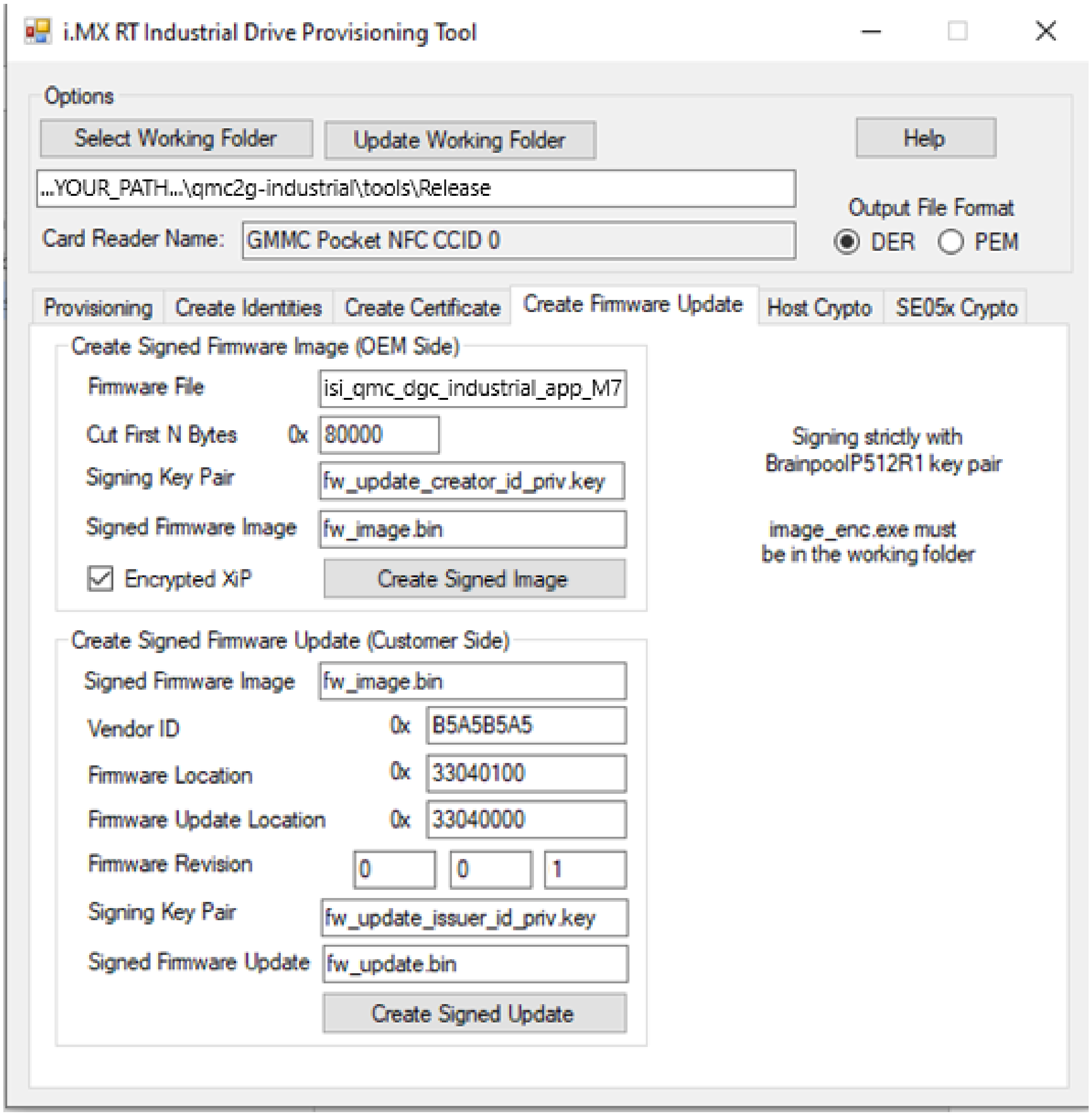
HAB Enablement- 5th step

The main batch files responsible for the signing/programming of an image and configuring fuses. Steps 1-4 can be executed only once per product. The 5th step can be issued as many times as needed.

- **blhost_hab_enc_xip_release.bat**
  - this batch file enables HAB + Encrypted XiP. It disables the debug port.
  - To enable fuse programming, set **"PGM_OTP_FUSE=1"** in the batch file.

Before executing the script make sure that steps 1-4 were successfully performed. If so, proceed with the following steps to prepare SBL and Main FW images:

1. Build the "**Release_Secure**" target of the bootloader project
2. Build and the copy "**Release_SBL\isi_qmc_dgc_industrial_app_M7MASTER.bin**" into your Provisioning Tool working folder where all the security assets were generated.
3. Go into the "Create Firmware Update" tab.
4. Increase the Manifest firmware revision. It must be greater than minimum Manifest revision configured during the provisioning step (default 0.0.0).


i.MX RT Industrial Drive Provisioning Tool

Options

...YOUR_PATH...\qmc2g-industrial\tools\Release

Card Reader Name: GMMC Pocket NFC CCID 0

Output File Format ● DER ○ PEM

Create Signed Firmware Image (OEM Side)

Firmware File: isi_qmc_dgc_industrial_app_M7

Cut First N Bytes 0x 80000

Signing Key Pair: fw_update_creator_id_priv.key

Signed Firmware Image: fw_image.bin

☑ Encrypted XiP

Signing strictly with
BrainpoolP512R1 key pair

image_enc.exe must
be in the working folder

Create Signed Firmware Update (Customer Side)

Signed Firmware Image: fw_image.bin

Vendor ID 0x B5A5B5A5

Firmware Location 0x 33040100

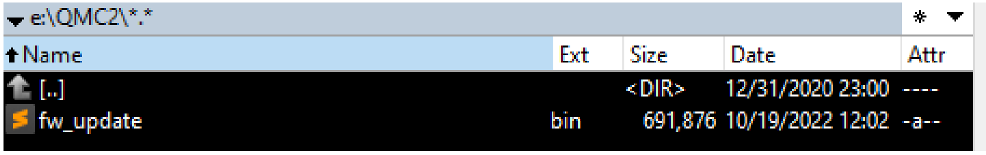
Firmware Update Location 0x 33040000

Firmware Revision 0 0 1

Signing Key Pair: fw_update_issuer_id_priv.key

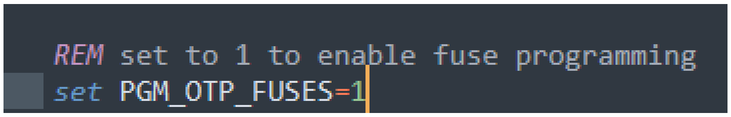
Signed Firmware Update: fw_update.bin

5. Check the Encrypted XiP check box.
6. Press the **"Create Signed Image"** button.
7. Press the **"Create Signed Update"** button which generates the **fw_update.bin** file. The file will be copied by the script into the FWU storage in the Octal flash. SBL will perform the FW update during the first boot.
8. To perform the FW Update from the SD Card, copy the resulting **"fw_update.bin"** from your working folder to the SD card into the **"QMC2"** folder


```
e:\QMC2\*.*
Name                    Ext   Size      Date              Attr
[..]                          <DIR>     12/31/2020 23:00  ----
fw_update               bin   691,876   10/19/2022 12:02  -a--
```

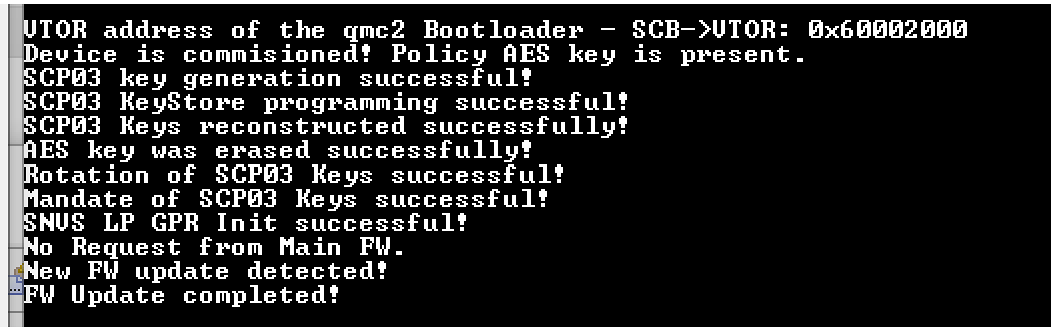
9. Insert the SD card into the Daughter card's SD memory slot (located below the battery slot).
10. You can now close the Provisioning Tool.

11. Configure the SW4 switch on the daughter card to SDP mode (1-ON/2-OFF/3-OFF/4-OFF).
12. Connect a micro-USB cable to J3 on the daughter card and press reset (SW2).
13. Edit the **blhost_hab_enc_xip_release.bat** script to enable fuse programming.



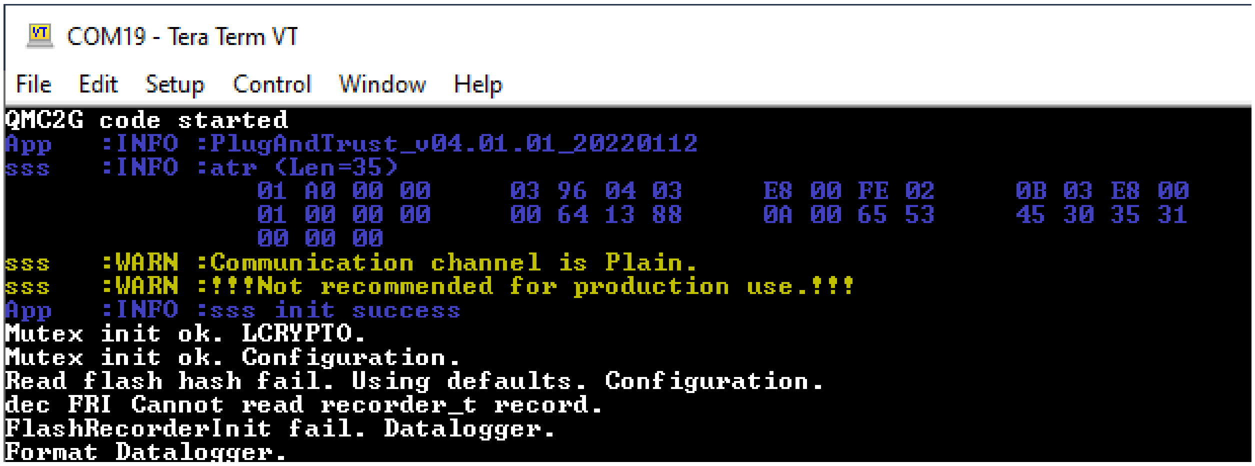
```
REM set to 1 to enable fuse programming
set PGM_OTP_FUSES=1
```

14. Execute the script**. The script programs all essential fuses, PUF keystore, Enables Encrypted XiP, signs and encrypts the secure version of SBL, and copies the fw_update.bin file into FWU storage in the octal flash.**
15. Configure the SW4 switch on the daughter card to Boot from fuses mode (1-OFF/2-OFF/3-OFF/4-OFF). Wait until the super cap is fully discharged if populated (No LEDs are lit on the daughter card).
16. Connect a micro-USB cable to the J48 on the digital board.
17. Power up the board if needed.
18. Use an application to establish serial communication for both COM ports (115200).
19. Press the reset button on daughter card (SW2) and after ~5 seconds you should see similar prints in the consoles:



```
VTOR address of the qmc2 Bootloader - SCB->VTOR: 0x60002000
Device is commisioned! Policy AES key is present.
SCP03 key generation successful!
SCP03 KeyStore programming successful!
SCP03 Keys reconstructed successfully!
AES key was erased successfully!
Rotation of SCP03 Keys successful!
Mandate of SCP03 Keys successful!
SNVS LP GPR Init successful!
No Request from Main FW.
New FW update detected!
FW Update completed!
```

4

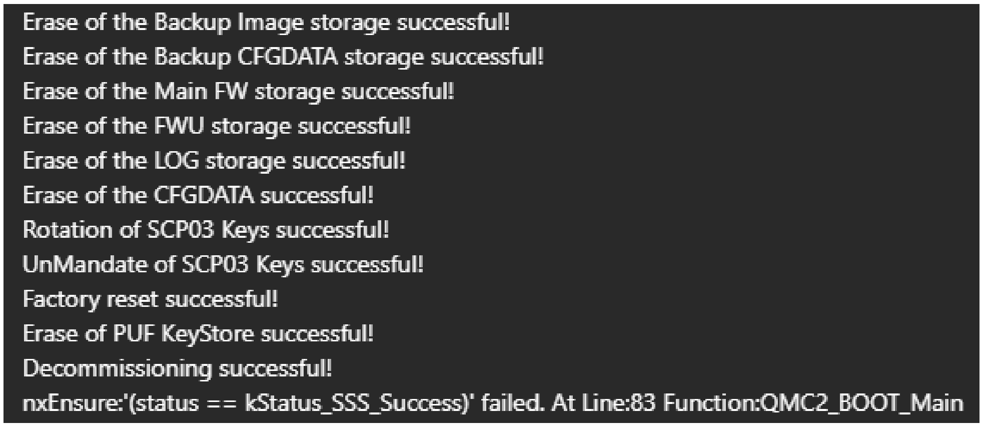


```
QMC2G code started
App   :INFO :PlugAndTrust_v04.01.01_20220112
sss   :INFO :atr (Len=35)
                01 A0 00 00     03 96 04 03     E8 00 FE 02     0B 03 E8 00
                01 00 00 00     00 64 13 88     0A 00 65 53     45 30 35 31
                00 00 00
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
App   :INFO :sss init success
Mutex init ok. LCRYPTO.
Mutex init ok. Configuration.
Read flash hash fail. Using defaults. Configuration.
dec FRI Cannot read recorder_t record.
FlashRecorderInit fail. Datalogger.
Format Datalogger.
```

Decommissioning

This operation will erase all code from external memory except the bootloader. The SC03 keys will be rotated back into their default state. SCP03 will be un-mandated. Application's PUF key store will be erased. After boot, SBL will report an error message that the device is not commissioned.

1. Create an empty **"decommission.bin"** file and put it on the SD card at the following path **"/QMC2/decommission.bin"**.
2. Press and hold all 4 buttons on the Digital board.
3. Reset the board.
4. Buttons can be released once this message "Decommissioning has started!" is printed on the console.
5. The SBL enters the decommissioning state. It will take some time to finish the process. You should see the following messages in the console.



```
Erase of the Backup Image storage successful!
Erase of the Backup CFGDATA storage successful!
Erase of the Main FW storage successful!
Erase of the FWU storage successful!
Erase of the LOG storage successful!
Erase of the CFGDATA successful!
Rotation of SCP03 Keys successful!
UnMandate of SCP03 Keys successful!
Factory reset successful!
Erase of PUF KeyStore successful!
Decommissioning successful!
nxEnsure:'(status == kStatus_SSS_Success)' failed. At Line:83 Function:QMC2_BOOT_Main
```

Troubleshooting

- If you start the application and see a provisioning error, it means the bootloader was reflashed but the device wasn’t recommissioned. You need to do the whole provisioning process again.
- Each FW update must have a higher version number than the last committed FW update.
- Each time you generate a FW update (Create Signed Update button), you must first generate the FW Image (Create Signed Image button).
- If you get errors while trying to provision the Secure Element, you probably have issues with the NFC connection. Take the NFC reader away from the board, connect it back and retry the provisioning.