AN13027

EdgeLock SE05x Quick start guide with i.MX 8M

Rev. 1.3 — 12 September 2022 Application note

Document information

Information	Content
Keywords	EdgeLock SE05x, Plug & Trust middleware, i.MX 8M
Abstract	This document explains how to get started with the EdgeLock SE05x Plug & Trust middleware using the EdgeLock SE05x development boards and i.MX 8M board. It provides detailed instructions for connecting the boards, installing the software, running the EdgeLock SE05x Plug & Trust project examples and executing the ssscli tool

Revision history

Revision history

Revision number	Date	Description
1.0	2020-10-21	First document release.
1.1	2020-12-07	Updated to latest template and fixed broken links
1.2	2022-03-28	Add EdgeLock SE050E and EdgeLock A5000 product variants. Update Table 1, Figure 1 and Figure 2. Update SD card flash instructions in Section 4.1. Add note in Section 5 (step 6). Add Section Section 6 Product specific CMake build settings Add Section Section 7 Binding EdgeLock SE05x to a host using Platform SCP. Add Section Section 8 Manage access from multiple Linux processes to the EdgeLock SE05x. Added ssscli installation instructions in Section 9 Updated middleware build instructions in Section 10
1.3	2022-09-12	Update to EdgeLock SE Plug & Trust Middleware version 04.02.xx. Update Section 6 Product specific CMake build settings. Update Section 7 Binding EdgeLock SE05x to a host using Platform SCP.

1 How to use this document

The Plug & Trust middleware includes a set of project examples that demonstrate the use of EdgeLock SE05x product family in the latest IoT security use cases. This document provides detailed instructions to run project examples for EdgeLock SE05x in i.MX 8M board. The main body of this document should be used in this sequence:

1. Order board samples. Section 2 contains the ordering details of the demo boards required in this document;
2. Prepare the hardware. Section 3 describes how to setup the OM-SE05xARD and i.MX 8M boards.
3. Flash the microSD card image. Section 4 describes how to create a micro-SD card with the Linux image with the pre-installed Plug & Trust middleware.
4. Run project examples. Section 5 describes how to run EdgeLock SE05x included in Plug & Trust middleware.

Supplementary material is provided in the appendices.

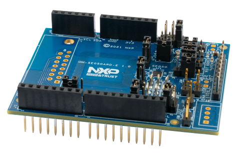
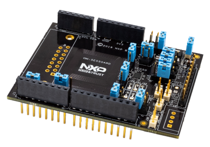
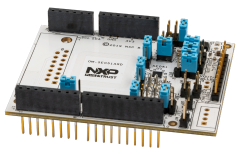
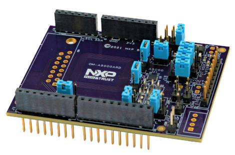
2 Hardware required

The EdgeLock SE05x works as an auxiliary security device attached to a host controller, communicating with through an I²C interface. To follow the instructions provided in this document, you need an EdgeLock SE05x development board and a i.MX 8M MCU board, acting as a host controller.

EdgeLock SE05x development boards ordering details

The EdgeLock SE05x and EdgeLock A5000 product support packages are providing development boards for evaluating EdgeLock SE05x and EdgeLock A5000 features. Select the development board of the product you want to evaluate. Table 1 details the ordering details of the EdgeLock SE05x and EdgeLock A5000 development boards.

Table 1. EdgeLock SE05x development boards.

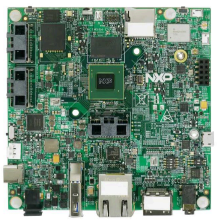
Part number	12NC	Description	Picture
OM-SE050ARD-E	9354 332 66598	SE050E Arduino® compatible development kit	
OM-SE050ARD-F	9354 357 63598	SE050 Arduino® compatible development kit	
OM-SE050ARD	9353 832 82598	SE050F Arduino® compatible development kit	
OM-SE051ARD	9353 991 87598	SE051 Arduino® compatible development kit	
OM-A5000ARD	9354 243 19598	A5000 Arduino® compatible development kit	

Note: The pictures in this guide will show EdgeLock SE05xE, but all boards in Table 1 can be used as well with the same hardware configuration.

i.MX 8M MCU board ordering details

Table 2 details the ordering details for the i.MX 8M board.

Table 2. i.MX 8M development kit details

Part number	12NC	Content	Picture
MCIMX8M-EVKB	935378743598	i.MX 8M evaluation kit	

3 Boards setup

This section explains how to prepare the OM-SE05xARD boards and i.MX 8M board to run the EdgeLock SE05x Plug & Trust middleware project examples. Follow these steps:

1. Connect the OM-SE05xARD to the i.MX 8M board. The i.MX 8M board does not come with an Arduino connector, so you have to connect the Arduino shield of the OM-SE05xARD board to the J801_I2C connector of the i.MX 8M board using wires. Table 3 details the jumper connection

Table 3. Wire connection OM-SE05xARD - i.MX 8M

OM-SE05xARD (# jumper - # pin)	I.MX 8M (# jumper - # pin)	Description
J2 - Pin 10	J801 - Pin 1	SCL
J2 - Pin 7	J801 - Pin 2	Ground
J2 - Pin 9	J801 - Pin 3	SDA
J8 - Pin 3	J801 - Pin 5	3.3 Volt supply

and Figure 1 illustrate the wiring between boards.

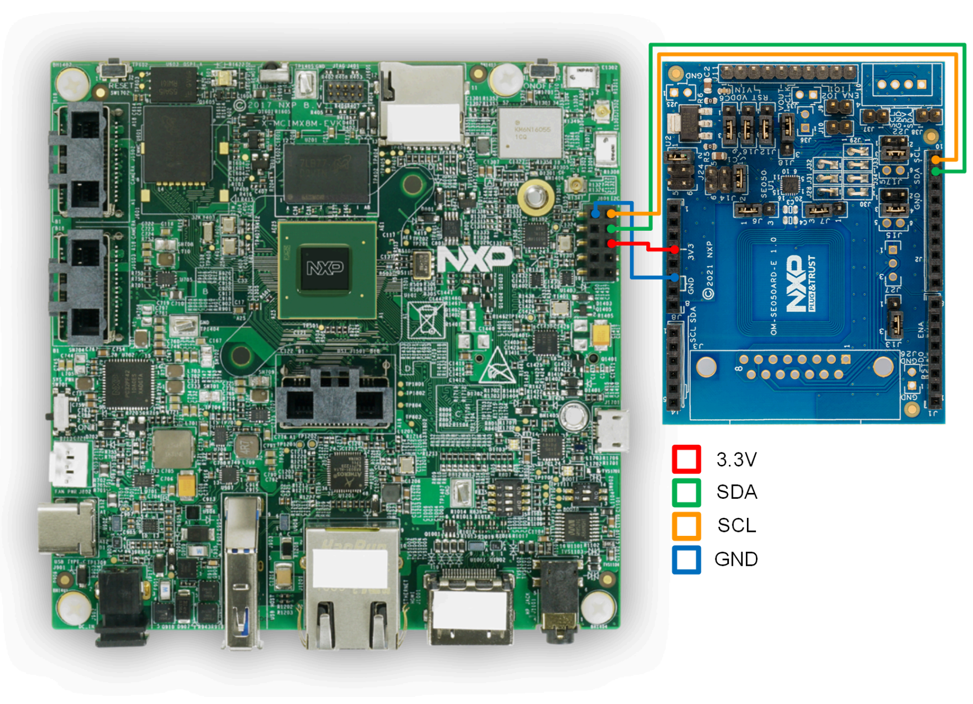


Figure 1. Connection between OM-SE05xARD and i.MX 8M boards

2. Make sure the jumper settings in your OM-SE05xARD board are configured as shown in Figure 2:

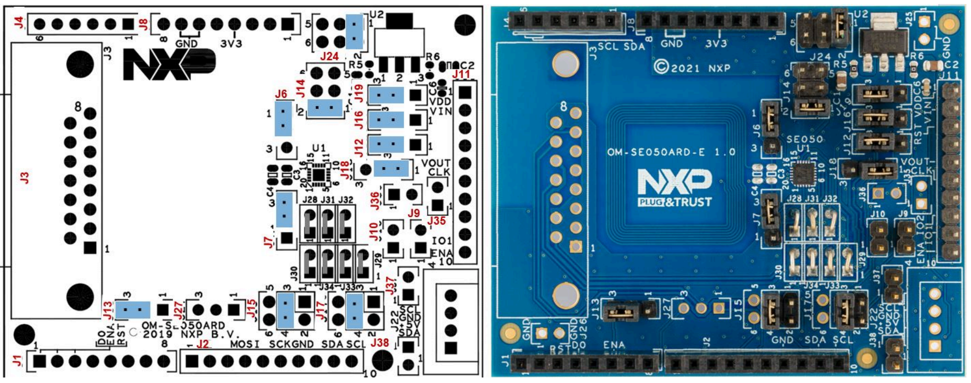


Figure 2. Jumper configuration for i.MX 8M board

Note: *For more information about the jumper settings, refer to AN13539 OM-SE05xARD hardware overview.*

4 Software setup

The software setup consists of:

1. Preparing a micro-SD card with the pre-compiled Linux image with the preinstalled Plug & Trust middleware for i.MX 8M board, as described in Section 4.1.
2. Installing the *USB to UART Bridge VCOM driver* in your laptop, as described in Section 4.2.
3. Installing TeraTerm terminal application, as described in Section 4.3.
4. Booting the i.MX 8M board, as described in Section 4.4.

4.1 Micro-SD card preparation

To prepare the micro-sd card with the pre-compiled Linux image that includes the Plug & Trust middleware, you need to:

1. Download from NXP website the Plug & Trust middleware SD Card Image. This image contains the Plug & Trust middleware pre-installed with already some examples on a bootable SD Card Image.
   Note: *In case the image provided by NXP does not suit your use case, you can find the explanation of how to create an card image using Yocto in the Plug & Trust middleware documentation (simw-top/doc/dev-platforms/platform_imx8_linux.html) .*
2. Download and install Win32 Disk Imager software. Win32 Disk Imager is a Windows open source program to format SD card images. Instead of Win 32 Disk Imager, you could also use any other software for this operation.
3. Plug your micro-SD card in your laptop.

4. Open Win32 Disk Imager, (1) select from your file system the pre-compiled Linux image you downloaded from the website, and (2) click on the ***Write*** button as shown in Figure 3.
   Note: *the SD card image might come in a ZIP or BZ2 package. Make sure to unzip the package first with e.g. 7-ZIP before selecting the uncompressed image file in Win32 Disk Imager.*

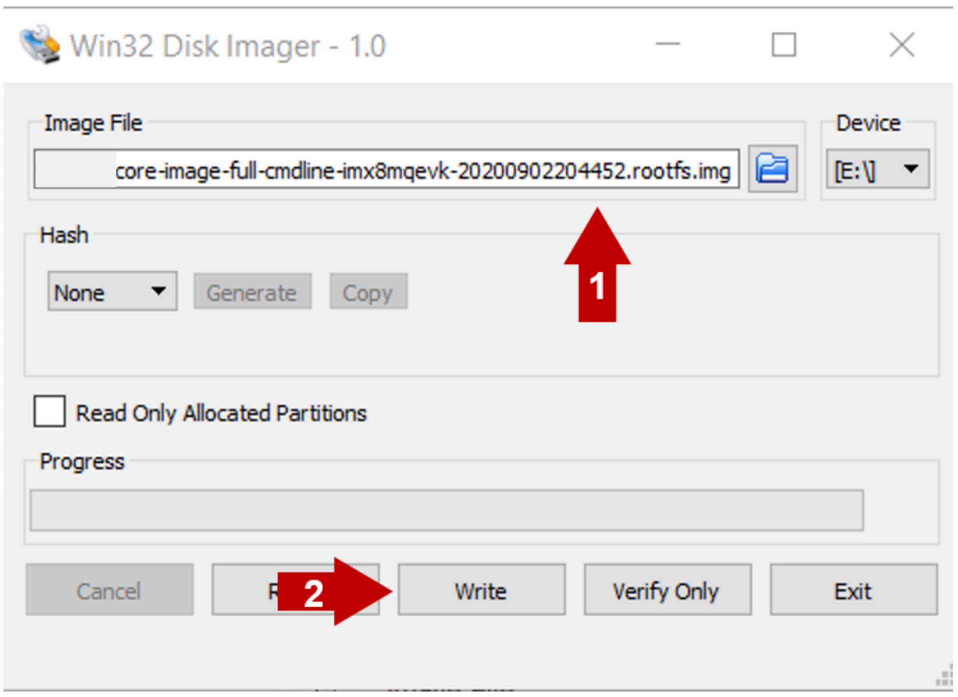


Figure 3. Micro-SD card preparation with Win32 Disk Imager software

4.2 Drivers

To install the i.MX 8M drivers, follow these steps:

1. Plug the power supply and connect the USB cable to your laptop as shown in Figure 4.

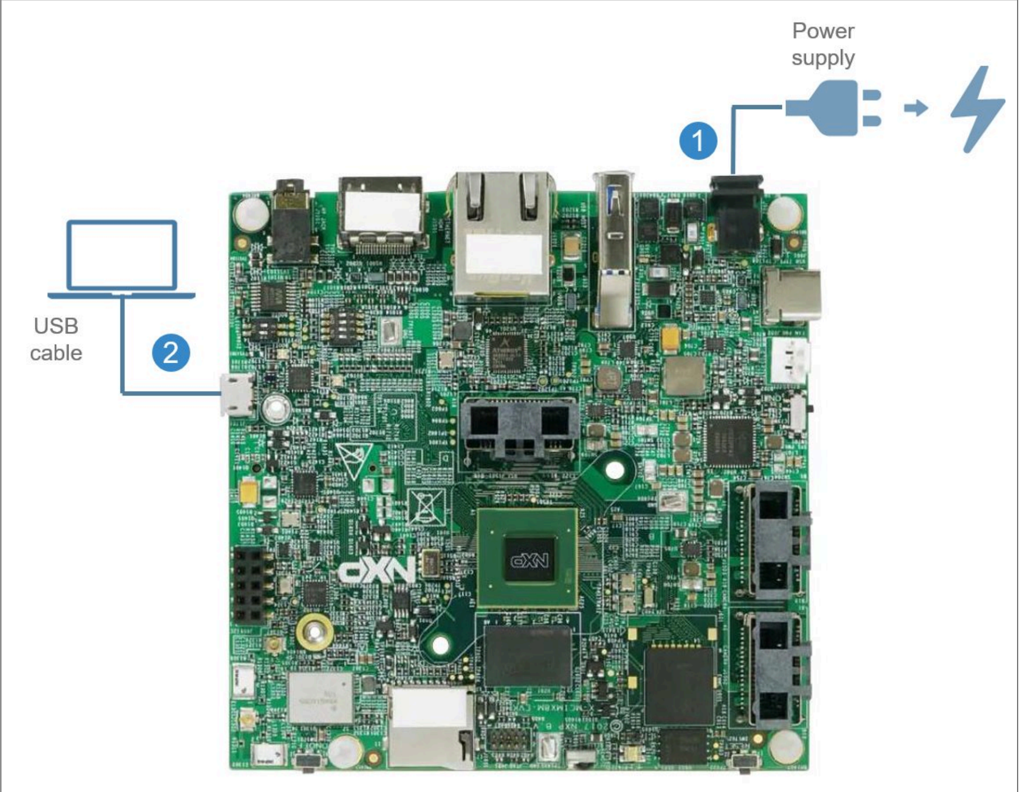


Figure 4. Plug the power supply and connect the USB cable to your laptop

2. Download the USB to UART Bridge VCOM driver for your processor (either 32 or 64 bits). Install the driver by following the setup wizard until it is finished.
3. Unplug and plug your board.

4. Go to your Device Manager, and check that your board is recognized and assigned to a port number (COMxx). Write down the assigned port number (COMxx) as it is needed in the next steps. Your Device Manager should look like Figure 5.
   Note: *if you see more than one COM port, use the one denoted as Enhanced COM port.*

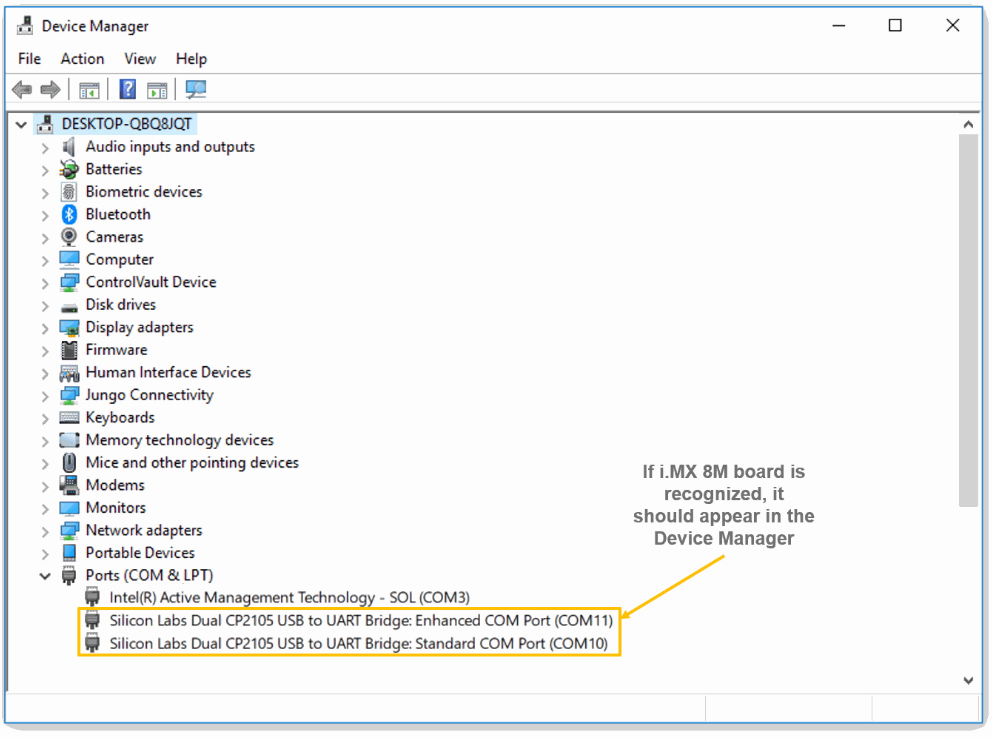

Device Manager
File
Action
View
Help
DESKTOP-QBQ8JQT
Audio inputs and outputs
Batteries
Biometric devices
Bluetooth
Cameras
Computer
ControlVault Device
Disk drives
Display adapters
Firmware
Human Interface Devices
Jungo Connectivity
Keyboards
Memory technology devices
Mice and other pointing devices
Modems
Monitors
Network adapters
Portable Devices
Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
Silicon Labs Dual CP2105 USB to UART Bridge: Enhanced COM Port (COM11)
Silicon Labs Dual CP2105 USB to UART Bridge: Standard COM Port (COM10)
If i.MX 8M board is recognized, it should appear in the Device Manager

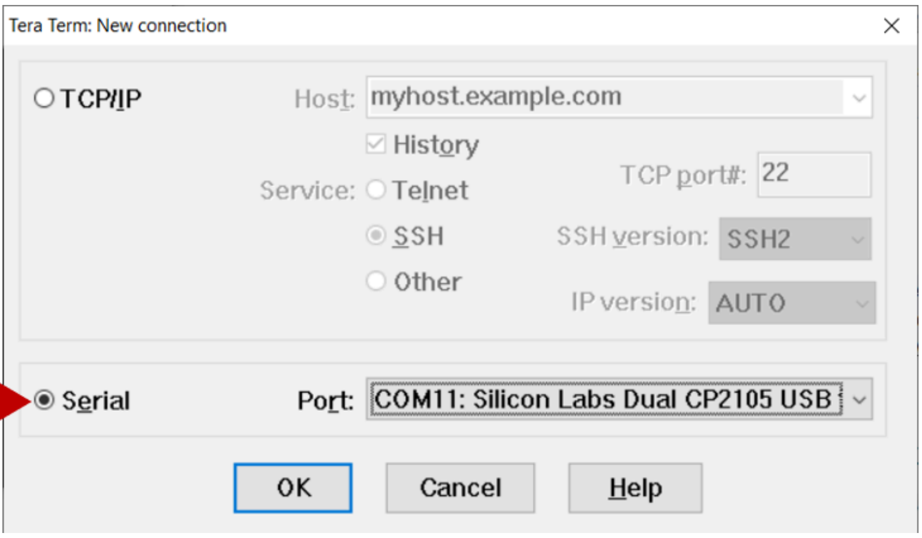
Figure 5. Check that i.MX 8M board is recognized in Device Manager

4.3 Terminal setup

We need to install a terminal application, for instance TeraTerm, to communicate and view the serial output of the i.MX 8M board from our laptop. To setup TeraTerm application:

1. Download TeraTerm and run the installer.
2. Launch TeraTerm, click the ***Serial*** option and select from the dropdown list the Enhanced COM port number assigned to your i.MX 8M board (In this case COM11) as shown in Figure 6. Click the *OK* button to confirm the setup. If you cannot see the

serial port of the board, your i.MX 8M board might not be recognized. In that case, please repeat the driver installation process described in Section 4.2.
Note: *if you can't see the serial port in TeraTerm, make sure that the board is plugged in with the USB cable to the PC and restart TeraTerm.*

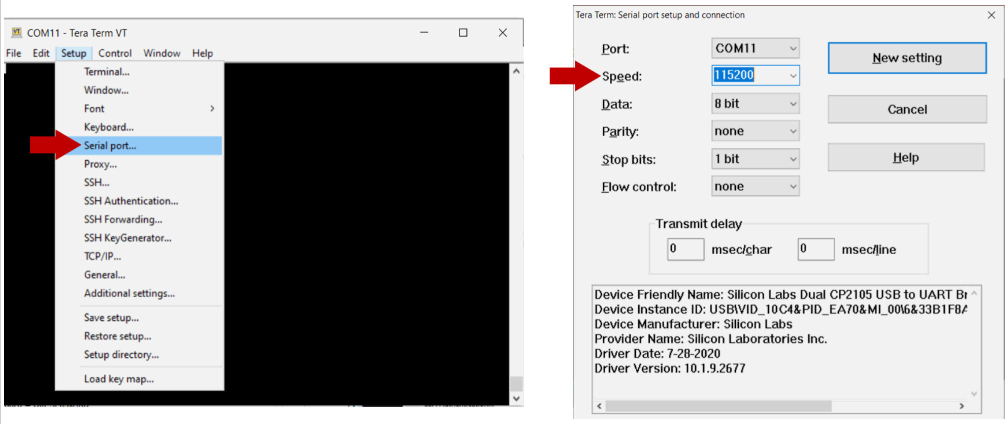

Tera Term: New connection

TCP/IP — Host: myhost.example.com; History; Service: Telnet / SSH / Other; TCP port#: 22; SSH version: SSH2; IP version: AUTO

Serial — Port: COM11: Silicon Labs Dual CP2105 USB

Figure 6. Open a TeraTerm serial connection

3. Go to *Setup → Serial Port* and configure the terminal to *115200 Speed*, *8 data bits*, *no parity* and *1 stop bit* as shown in Figure 7. Click on **New Setting** to confirm the configuration.



Tera Term: Serial port setup and connection

Port: COM11
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none
Transmit delay: 0 msec/char, 0 msec/line

```
Device Friendly Name: Silicon Labs Dual CP2105 USB to UART Br
Device Instance ID: USB\VID_10C4&PID_EA70&MI_00\6&33B1F8A
Device Manufacturer: Silicon Labs
Provider Name: Silicon Laboratories Inc.
Driver Date: 7-28-2020
Driver Version: 10.1.9.2677
```

Figure 7. Configure TeraTerm serial port connection

4.4 Booting the i.MX 8M

To boot the i.MX 8M, follow the steps shown in Figure 8 and Figure 9:

1. Insert the micro-SD card with the pre-compiled Linux image into the i.MX 8M card slot.
2. Configure the board switches as follows:
   - SW801 (Boot Device Select Switch): ON, ON, OFF, OFF (from 1-4 bit)
   - SW802 (Boot Mode Select Switch): ON, OFF (from 1-2 bit)

3. Connect the board to the power supply
4. Connect the board to your laptop using a USB cable and make sure that TeraTerm serial port is configured (see Section 4.3).
5. Turn on the power supply switch to boot up the board.

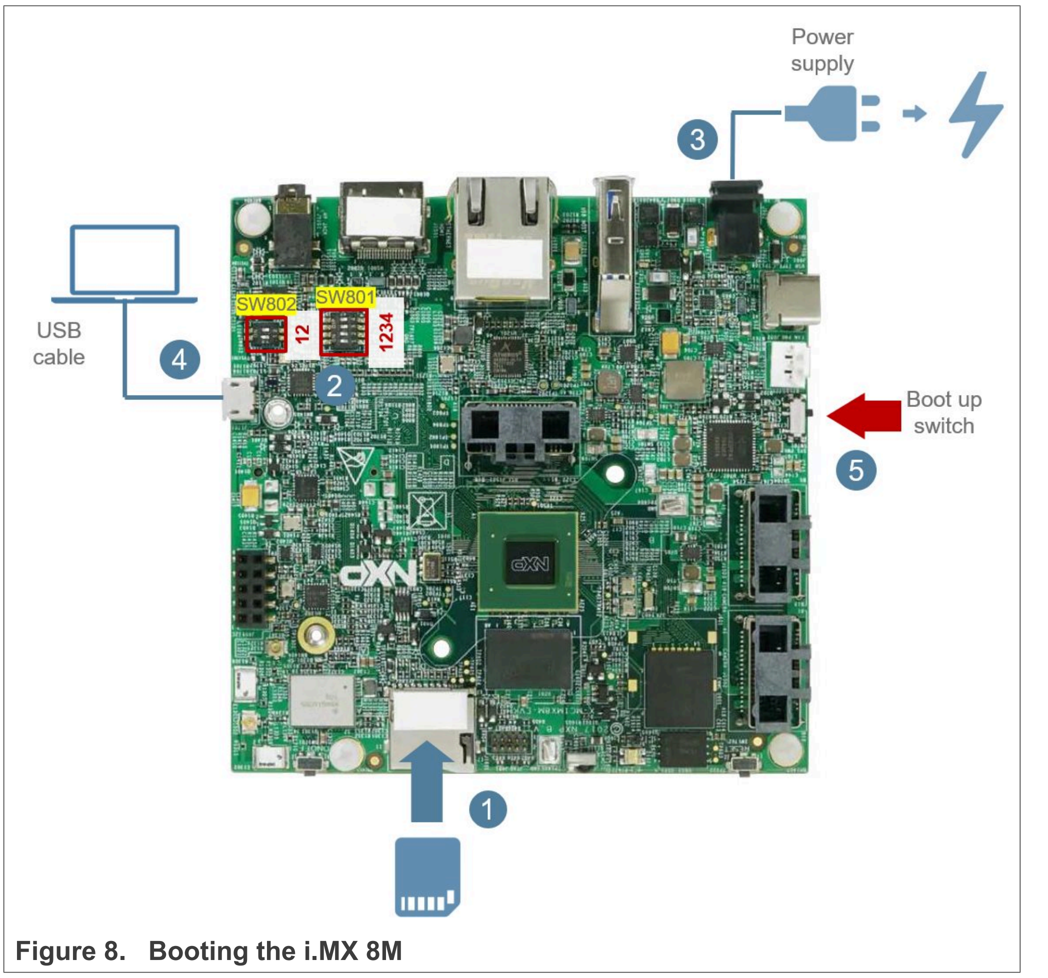
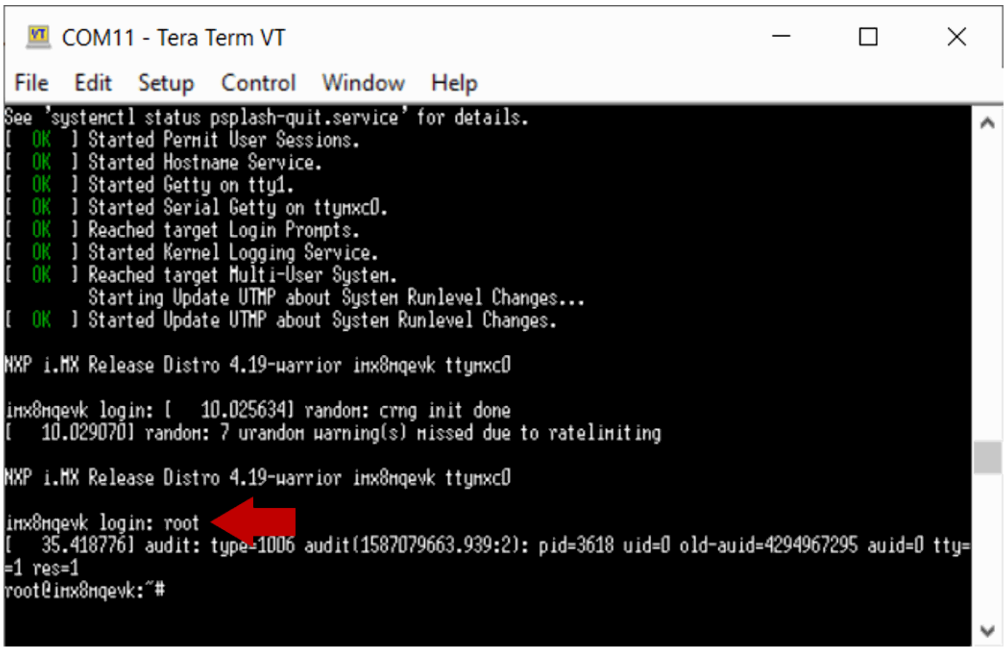


Figure 8. Booting the i.MX 8M

6. During the boot process, the operating system status information will be displayed in TeraTerm as shown in Figure 9. When the process is complete, the user can login with the following credentials:
   - Account name: root
   - Password: not required



```
See 'systemctl status psplash-quit.service' for details.
[  OK  ] Started Permit User Sessions.
[  OK  ] Started Hostname Service.
[  OK  ] Started Getty on tty1.
[  OK  ] Started Serial Getty on ttymxc0.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started Kernel Logging Service.
[  OK  ] Reached target Multi-User System.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

NXP i.MX Release Distro 4.19-warrior imx8mqevk ttymxc0

imx8mqevk login: [   10.025634] random: crng init done
[   10.029070] random: 7 urandom warning(s) missed due to ratelimiting

NXP i.MX Release Distro 4.19-warrior imx8mqevk ttymxc0

imx8mqevk login: root
[   35.418776] audit: type=1006 audit(1587079663.939:2): pid=3618 uid=0 old-auid=4294967295 auid=0 tty=
=1 res=1
root@imx8mqevk:~#
```

Figure 9. Sign in in the OS

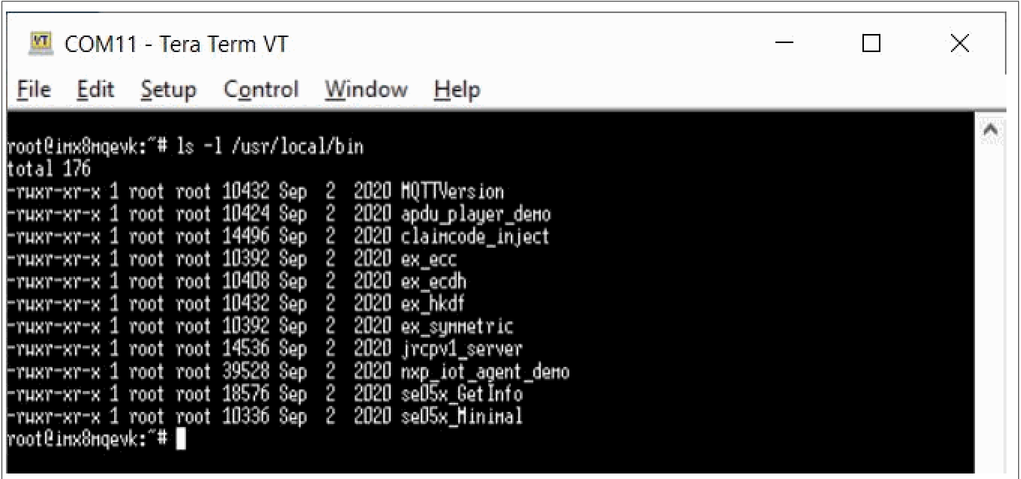
Note: In the case that the precompiled Plug & Trust middleware version does not suit your use case and a different version is needed, it must be downloaded and build into the device. For more information refer to Appendix B.

5 Run preinstalled Plug & Trust middleware test examples

The Plug & Trust middleware comes with several test examples used to verify atomic EdgeLock SE05x security IC features. This section explains how to run the Plug & Trust middleware test example called `se05x_Minimal`.

Note: The default build configuration of the Plug & Trust middleware ≤ `V04.01.0x` generates code for the OM-SE050ARD development board. You need to adapt the CMake settings in case you are using a different EdgeLock secure element development board or a different secure element product IC. The settings are described in Section 6.

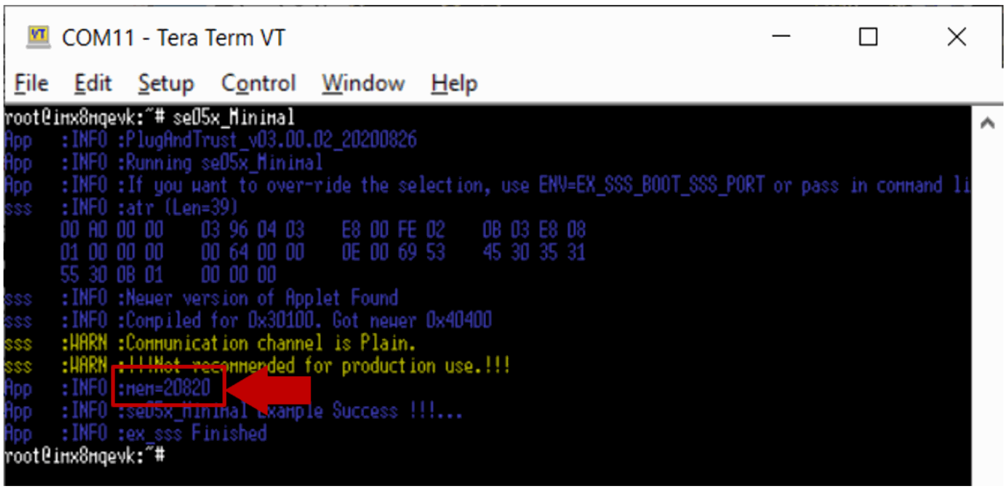
1. After booting the board, you will be in the */usr/local/bin* directory of the file system. To see the list of examples that are available in this directory use the following command: Send > `ls -l /usr/local/bin`.
   The TeraTerm window should be similar to Figure 10:



```
root@imx8mqevk:~# ls -l /usr/local/bin
total 176
-rwxr-xr-x 1 root root 10432 Sep  2  2020 MQTTVersion
-rwxr-xr-x 1 root root 10424 Sep  2  2020 apdu_player_demo
-rwxr-xr-x 1 root root 14496 Sep  2  2020 claimcode_inject
-rwxr-xr-x 1 root root 10392 Sep  2  2020 ex_ecc
-rwxr-xr-x 1 root root 10408 Sep  2  2020 ex_ecdh
-rwxr-xr-x 1 root root 10432 Sep  2  2020 ex_hkdf
-rwxr-xr-x 1 root root 10392 Sep  2  2020 ex_symmetric
-rwxr-xr-x 1 root root 14536 Sep  2  2020 jrcpv1_server
-rwxr-xr-x 1 root root 39528 Sep  2  2020 nxp_iot_agent_demo
-rwxr-xr-x 1 root root 18576 Sep  2  2020 se05x_GetInfo
-rwxr-xr-x 1 root root 10336 Sep  2  2020 se05x_Minimal
root@imx8mqevk:~#
```

Figure 10. Go to the Plug & Trust middleware test example directory

2. Execute the `se05x_Minimal` test example. This test example outputs the memory left in EdgeLock SE05x security IC.
   Send > `se05x_Minimal`.
   The TeraTerm logs should indicate the available memory in EdgeLock SE05x security IC as can be seen in Figure 11 (in this case, 20820).



```
root@imx8mqevk:~# se05x_Minimal
App   :INFO :PlugAndTrust_v03.00.02_20200826
App   :INFO :Running se05x_Minimal
App   :INFO :If you want to over-ride the selection, use ENV=EX_SSS_BOOT_SSS_PORT or pass in command li
sss   :INFO :atr (Len=39)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0E 00 69 53    45 30 35 31
      55 30 0B 01    00 00 00
sss   :INFO :Newer version of Applet Found
sss   :INFO :Compiled for 0x30100. Got newer 0x40400
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
App   :INFO :mem=20820
App   :INFO :se05x_Minimal Example Success !!!...
App   :INFO :ex_sss Finished
root@imx8mqevk:~#
```

Figure 11.   Run `se05x_minimal` test example

6 Product specific CMake build settings

The NXP Plug & Trust middleware supports the SE05x Secure Elements, the A5000 Secure Authenticator, and the legacy A71CH products.

The EdgeLock Plug & Trust middleware is delivered with CMake files that include the set of directives and instructions describing the project's source files and the build targets. The CMake files are used to select a dedicated EdgeLock product IC and the corresponding IoT applet or Authenticator application.

The SE050 product identification can be obtained as described in AN12436 chapter 1 *Product Information*. AN12973 describes the same procedure for the SE051 product family.

The following tables show the required `PTMW` CMake options to build a dedicated product variant. The `SSSFTR_ SE05X_RSA` CMake option is used to optimize the memory footprint for product variants that do not support RSA.

Table 4. CMake Settings for SE050E product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050E Dev. Board OM-SE050ARD-E	A921	`SE050_E`	`None`	`07_02`	any option	`None` or `SCP03_ SSS`	disabled
SE050E2	A921						

Table 5. CMake Settings for SE050F product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050F Dev.Board OM-SE050ARD-F	A92A	`SE05X_C`	`SE050`	`03_XX`	`PlatfSCP03` or `UserID_PlatfSCP03` or `AESKey_PlatfSCP03` or `ECKey_PlatfSCP03`	`SCP03_ SSS`	enabled
SE050F2	A92A						

Table 6. CMake Settings for SE050 Previous Generation product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050A1	A204	`SE05X_A`	`None`	`03_XX`	any option	`None` or `SCP03_ SSS`	disabled
SE050A2	A205						
SE050B1	A202	`SE05X_B`	`None`	`03_XX`	any option	`None` or `SCP03_ SSS`	enabled
SE050B2	A203						

Table 6. CMake Settings for SE050 Previous Generation product variants*...continued*

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050C1	A200	`SE05X_C`	`None`	`03_XX`	any option	`None` or `SCP03_SSS`	enabled
SE050C2	A201						
SE050 Dev Board OM-SE050ARD	A1F4						
SE050F2	A77E[1]	`SE05X_C`	`SE050`	`03_XX`	`PlatfSCP03` or `UserID_PlatfSCP03` or `AESKey_PlatfSCP03` or `ECKey_PlatfSCP03`	`SCP03_SSS`	enabled

[1] All SE050F2 with variant A77E have date code in year 2021. All the SE050F2 with date code in the year 2022 have the variant identifier A92A.

Table 7. CMake Settings for SE051 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE051A2	A920	`SE05X_A`	`None`	`07_02`	any option	`None` or `SCP03_SSS`	disabled
SE051C2	A8FA	`SE05X_C`	`None`	`07_02`	any option	`None` or `SCP03_SSS`	enabled
SE051W2	A739	`SE05X_C`	`None`	`07_02`	any option	`None` or `SCP03_SSS` or `SCP03_SSS`	enabled
SE051A2	A565	`SE05X_A`	`None`	`06_00`	any option	`None` or `SCP03_SSS`	disabled
SE051C2	A564	`SE05X_C`	`None`	`06_00`	any option	`None` or `SCP03_SSS`	enabled

Table 8. CMake Settings for A5000 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
OM-A5000ARD	A736	`AUTH`	`None`	`07_02`	any option	`None` or `SCP03_SSS`	disabled
A5000	A736						

6.1 Example: SE050E CMake build settings

To build the Plug & Trust Middleware to support the SE050E Secure Element applet the following CMake setting needs to be modified before building the middleware

according to Table 4:

- Select `SE050_E` *for the CMake option* `PTWM_Applet`.
- Select `None` for the CMake option `PTWM_FIPS`.
- Select `07_02`*for the CMake option* `PTWM_SE05X_Ver`
- Disable the CMake option `SSSFTR_SE05X_RSA`

In this example we use plain communication. Plain communication for the example execution is enabled by selecting the following options:

- Select `None` for the CMake option `PTMW_SE05X_Auth`.
- Select `None` for the CMake option `PTMW_SCP`.

How to enable Platform SCP is described in How to enable Platform SCP in the CMake-based build system.

Run the following commands to update the CMake settings and rebuild the Plug & Trust middleware:

```
cd ~/se_mw/simw-top_build/imx_native_se050_t1oi2c
```

```
cmake -DPTMW_Applet=SE050_E -DPTMW_FIPS=None -
DPTMW_SE05X_Ver=07_02 -DSSSFTR_SE05X_RSA=0 -DPTMW_SCP=None -
DPTMW_SE05X_Auth=None .
```

```
cmake --build .
```

```
sudo make install
```

```
sudo ldconfig /usr/local/lib/
```

7 Binding EdgeLock SE05x to a host MCU/MPU using Platform SCP

Binding is a process to establish a pairing between the IoT device host MPU/MCU and EdgeLock SE05x, so that only the paired MPU/MCU is able to use the services offered by the corresponding EdgeLock SE05x and vice versa.

A mutually authenticated, encrypted channel will ensure that both parties are indeed communicating with the intended recipients and that local communication is protected against local attacks, including man-in-the-middle attacks aimed at intercepting the communication between the MPU/MCU and the EdgeLock SE05x and physical tampering attacks aimed at replacing the host MPU/MCU or EdgeLock SE05x .

EdgeLock SE05x natively supports Global Platform Secure Channel Protocol 03 (SCP03) for this purpose. PlatformSCP uses SCP03 and can be enabled to be mandatory.

This chapter describes the required steps to enable Platform SCP in the middlware for EdgeLock SE05x.

The following topics are discussed:

- Section 7.1 Introduction to the Global Platform Secure Channel Protocol 03 (SCP03)
- How to configure the product specific default Platform SCP keys How to configure the EdgeLock SE05x product specific SCP keys in the Plug & Trust middleware
- How to enable Platform SCP in the CMake-based build system How to enable Platform SCP in the Plug & Trust middleware

7.1 Introduction to the Global Platform Secure Channel Protocol 03 (SCP03)

The Secure Channel Protocol SCP03 authenticates and protects locally the bidirectional communication between host and EdgeLock SE05x against eavesdropping on the physical I2C interface.

EdgeLock SE05x can be bound to the host by injecting in both the host and EdgeLock SE05x the same unique SCP03 AES key-set and by enabling the Platform SCP feature in the Plug & Trust middleware. The AN12662 *Binding a host device to EdgeLock SE05x* describes in detail the concept of secure binding.

SCP03 is defined in Global Platform Secure Channel Protocol '03' - Amendment D v1.2 specification.

SCP03 can provide the following three security goals:

- **Mutual authentication (MA)**
  - Mutual authentication is achieved through the process of initiating a Secure Channel and provides assurance to both the host and the EdgeLock SE05x entity that they are communicating with an authenticated entity.
- **Message Integrity**
  - The Command- and Response-MAC are generated by applying the CMAC according to NIST SP 800-38B.
- **Confidentiality**
  - The message data field is encrypted across the entire data field of the command message to be transmitted to the EdgeLock SE05x, and across the response transmitted from the EdgeLock SE05x.

The SCP03 secure channel is set up via the EdgeLock SE05x Java Card OS Manager using the standard ISO7816-4 secure channel APDUs.

The establishment of an SCP03 channel requires three static 128-bit AES keys shared between the two communicating parties: `Key-ENC`, `Key-MAC` and `Key-DEK`. These keys are stored in the Java Card Supplementary Security Domain (SSD) and not in the secure authenticator applet.

`Key-ENC` and `Key-MAC` keys are used during the SCP03 channel establishment to generate the session keys. Session Keys are generated to ensure that a different set of keys are used for each Secure Channel Session to prevent replay attacks.

`Key-ENC` is used to derive the session key `S-ENC`. The `S-ENC` key is used for encryption/decryption of the exchanged data. The session keys `S-MAC` and `R-MAC` are derived from `Key-MAC` and used to generate/verify the integrity of the exchanged data (C-APDU and R-APDU).

`Key-DEK` key is used to encrypt new SCP03 keys in case they get updated.

Table 9. Static SCP03 keys

Key	Description	Usage	Key Type
`Key-ENC`	Static Secure Channel Encryption Key	Generate session key for Decryption/ Encryption (AES)	AES 128
`Key-MAC`	Static Secure Channel Message Authentication Code Key	Generate session key for Secure Channel authentication and Secure Channel MAC Verification/Generation (AES)	AES 128
`Key-DEK`	Data Encryption Key	Sensitive Data Decryption (AES)	AES 128

The session key generation is performed by the Plug & Trust middleware host crypto.

Table 10. SCP03 session keys

Key	Description	Usage	Key Type
`S-ENC`	Session Secure Channel Encryption Key	Used for data confidentiality	AES 128
`S-MAC`	Secure Channel Message Authentication Code Key for Command	Used for data and protocol integrity	AES 128
`S-RMAC`	Secure Channel Message Authentication Code Key for Response	User for data and protocol integrity	AES 128

Note: *For further details please refer to Global Platform Secure Channel Protocol '03' - Amendment D v1.2.*

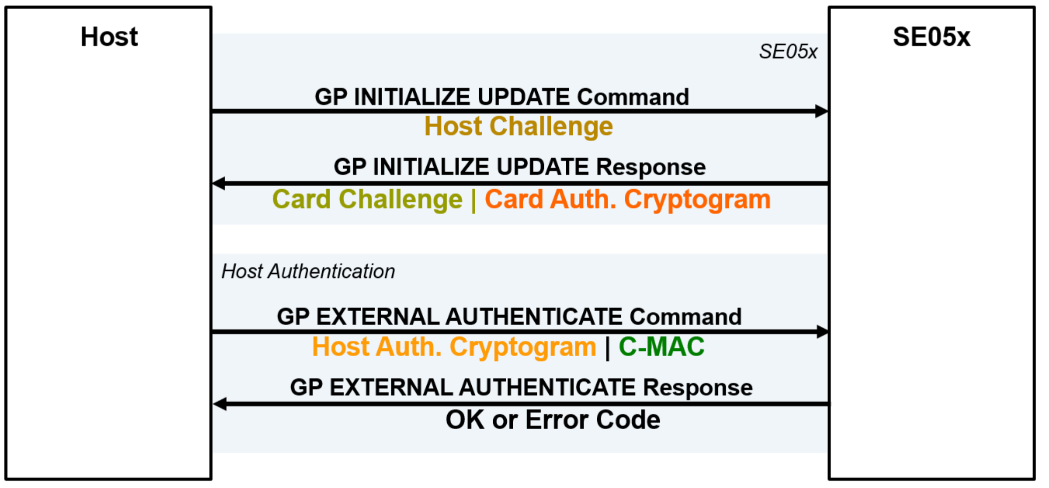

Host
SE05x
SE05x
GP INITIALIZE UPDATE Command
Host Challenge
GP INITIALIZE UPDATE Response
Card Challenge | Card Auth. Cryptogram
Host Authentication
GP EXTERNAL AUTHENTICATE Command
Host Auth. Cryptogram | C-MAC
GP EXTERNAL AUTHENTICATE Response
OK or Error Code

Figure 12. SPC03 mutual authentication – principle

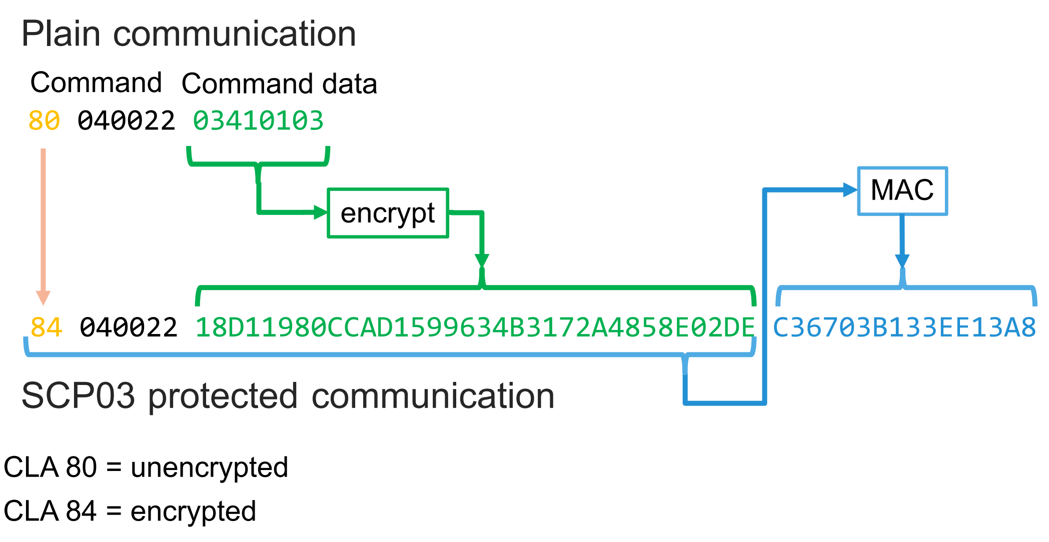


Figure 13. SPC03 Encryption and MACing principle

7.2 How to configure the product specific default Platform SCP keys

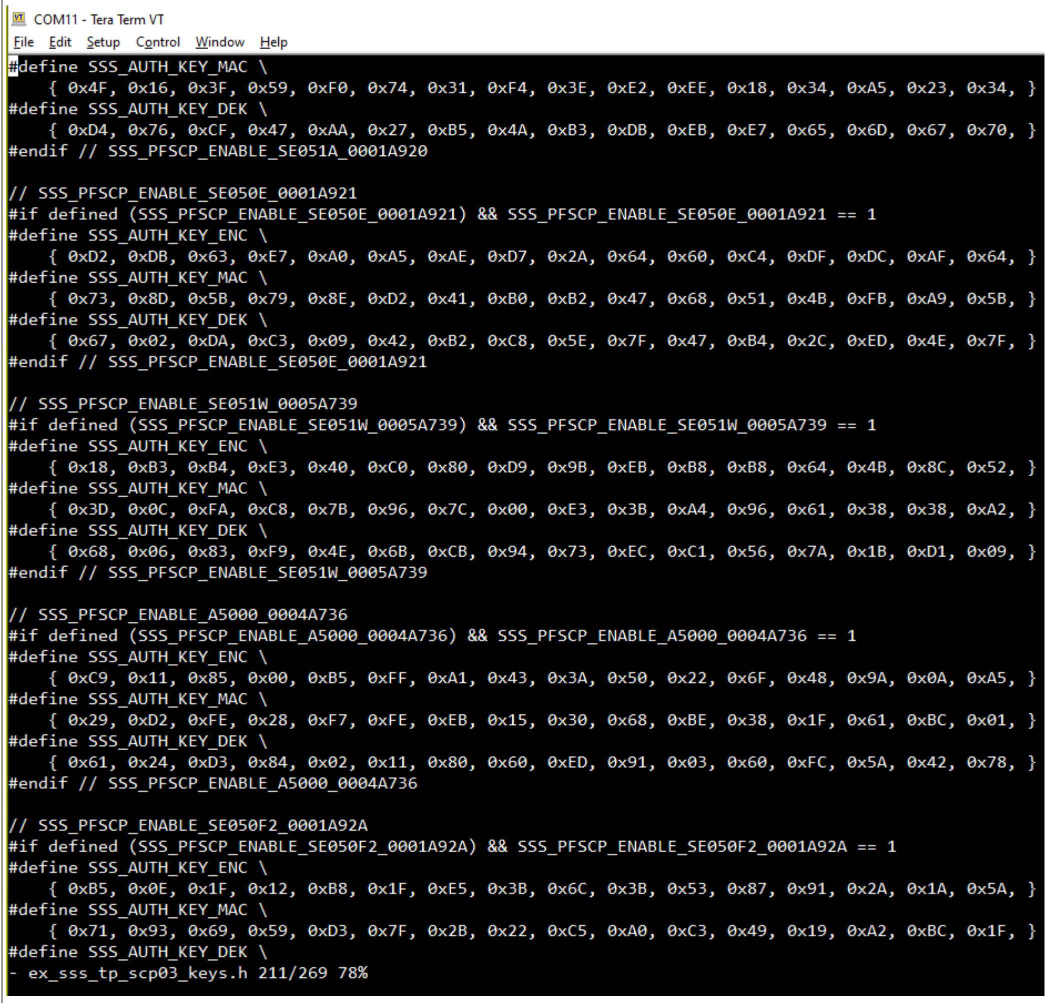
The default Platform SCP key values are described for the EdgeLock SE05x product variants in AN12436 and for the EdgeLock A5000 variants in AN12973.

For evaluation purpose, the Platform SCP keys can be defined either in the Plug & Trust middleware source code (see Section 7.2.1) or provided as text file (see Section 7.2.2).

7.2.1 Defining the deault Platfrom SCP keys in the Plug & Trust middleware source code

The Plug & Trust middleware header file `ex_sss_tp_scp03_keys.h` contains the default values of all EdgeLock SE05x, EdgeLock A5000, A5000 and A71CH product variants.

The `ex_sss_tp_scp03_keys.h` header file location in the following location: `~/se_mw/simw-top/sss/ex/inc/`

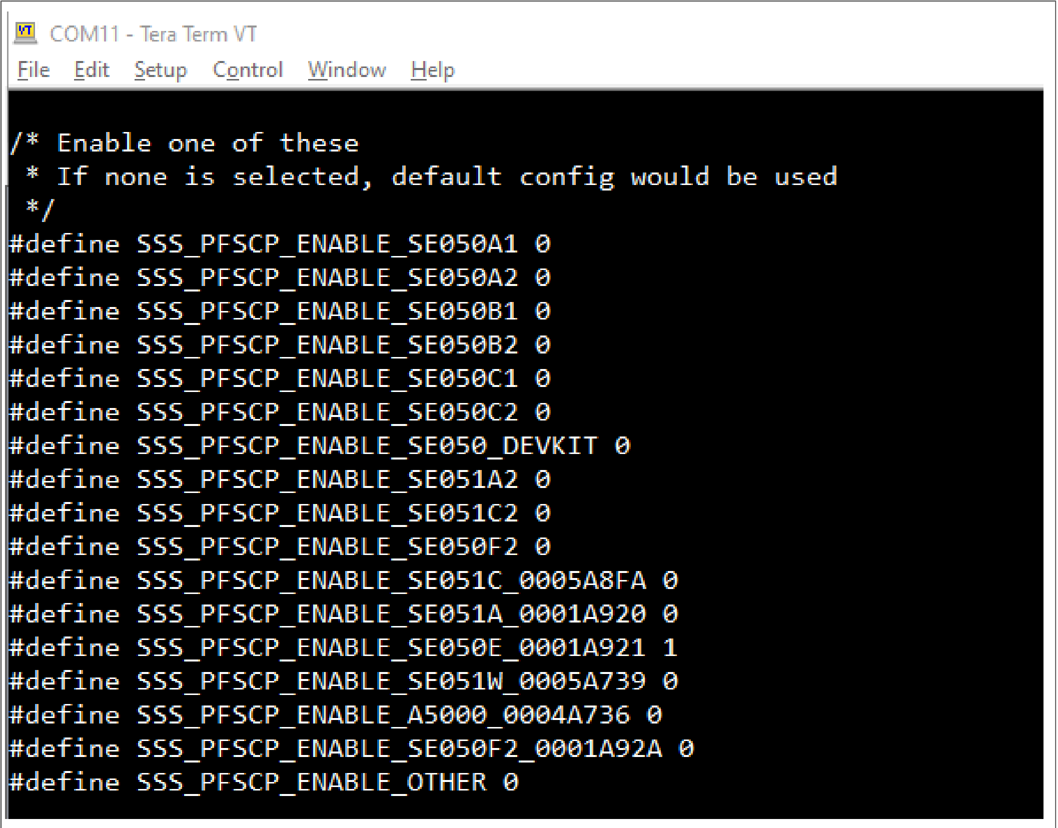


```
#define SSS_AUTH_KEY_MAC \
    { 0x4F, 0x16, 0x3F, 0x59, 0xF0, 0x74, 0x31, 0xF4, 0x3E, 0xE2, 0xEE, 0x18, 0x34, 0xA5, 0x23, 0x34, }
#define SSS_AUTH_KEY_DEK \
    { 0xD4, 0x76, 0xCF, 0x47, 0xAA, 0x27, 0xB5, 0x4A, 0xB3, 0xDB, 0xEB, 0xE7, 0x65, 0x6D, 0x67, 0x70, }
#endif // SSS_PFSCP_ENABLE_SE051A_0001A920

// SSS_PFSCP_ENABLE_SE050E_0001A921
#if defined (SSS_PFSCP_ENABLE_SE050E_0001A921) && SSS_PFSCP_ENABLE_SE050E_0001A921 == 1
#define SSS_AUTH_KEY_ENC \
    { 0xD2, 0xDB, 0x63, 0xE7, 0xA0, 0xA5, 0xAE, 0xD7, 0x2A, 0x64, 0x60, 0xC4, 0xDF, 0xDC, 0xAF, 0x64, }
#define SSS_AUTH_KEY_MAC \
    { 0x73, 0x8D, 0x5B, 0x79, 0x8E, 0xD2, 0x41, 0xB0, 0xB2, 0x47, 0x68, 0x51, 0x4B, 0xFB, 0xA9, 0x5B, }
#define SSS_AUTH_KEY_DEK \
    { 0x67, 0x02, 0xDA, 0xC3, 0x09, 0x42, 0xB2, 0xC8, 0x5E, 0x7F, 0x47, 0xB4, 0x2C, 0xED, 0x4E, 0x7F, }
#endif // SSS_PFSCP_ENABLE_SE050E_0001A921

// SSS_PFSCP_ENABLE_SE051W_0005A739
#if defined (SSS_PFSCP_ENABLE_SE051W_0005A739) && SSS_PFSCP_ENABLE_SE051W_0005A739 == 1
#define SSS_AUTH_KEY_ENC \
    { 0x18, 0xB3, 0xB4, 0xE3, 0x40, 0xC0, 0x80, 0xD9, 0x9B, 0xEB, 0xB8, 0xB8, 0x64, 0x4B, 0x8C, 0x52, }
#define SSS_AUTH_KEY_MAC \
    { 0x3D, 0x0C, 0xFA, 0xC8, 0x7B, 0x96, 0x7C, 0x00, 0xE3, 0x3B, 0xA4, 0x96, 0x61, 0x38, 0x38, 0xA2, }
#define SSS_AUTH_KEY_DEK \
    { 0x68, 0x06, 0x83, 0xF9, 0x4E, 0x6B, 0xCB, 0x94, 0x73, 0xEC, 0xC1, 0x56, 0x7A, 0x1B, 0xD1, 0x09, }
#endif // SSS_PFSCP_ENABLE_SE051W_0005A739

// SSS_PFSCP_ENABLE_A5000_0004A736
#if defined (SSS_PFSCP_ENABLE_A5000_0004A736) && SSS_PFSCP_ENABLE_A5000_0004A736 == 1
#define SSS_AUTH_KEY_ENC \
    { 0xC9, 0x11, 0x85, 0x00, 0xB5, 0xFF, 0xA1, 0x43, 0x3A, 0x50, 0x22, 0x6F, 0x48, 0x9A, 0x0A, 0xA5, }
#define SSS_AUTH_KEY_MAC \
    { 0x29, 0xD2, 0xFE, 0x28, 0xF7, 0xFE, 0xEB, 0x15, 0x30, 0x68, 0xBE, 0x38, 0x1F, 0x61, 0xBC, 0x01, }
#define SSS_AUTH_KEY_DEK \
    { 0x61, 0x24, 0xD3, 0x84, 0x02, 0x11, 0x80, 0x60, 0xED, 0x91, 0x03, 0x60, 0xFC, 0x5A, 0x42, 0x78, }
#endif // SSS_PFSCP_ENABLE_A5000_0004A736

// SSS_PFSCP_ENABLE_SE050F2_0001A92A
#if defined (SSS_PFSCP_ENABLE_SE050F2_0001A92A) && SSS_PFSCP_ENABLE_SE050F2_0001A92A == 1
#define SSS_AUTH_KEY_ENC \
    { 0xB5, 0x0E, 0x1F, 0x12, 0xB8, 0x1F, 0xE5, 0x3B, 0x6C, 0x3B, 0x53, 0x87, 0x91, 0x2A, 0x1A, 0x5A, }
#define SSS_AUTH_KEY_MAC \
    { 0x71, 0x93, 0x69, 0x59, 0xD3, 0x7F, 0x2B, 0x22, 0xC5, 0xA0, 0xC3, 0x49, 0x19, 0xA2, 0xBC, 0x1F, }
#define SSS_AUTH_KEY_DEK \
- ex_sss_tp_scp03_keys.h 211/269 78%
```

Figure 14. Default Platform SCP keys are defined in ex_sss_tp_scp03_keys.h

The `fsl_sss_ftr.h.in` file includes options to select one of the predefined default Platform SCP keys. This file is located in: `~/se_mw/simw-top/sss/inc`. Select the desired value of the compilation option by setting exclusively the corresponding C-preprocessor define `SSS_PFSCP_ENABLE_xx` to `1` (enable).

All other values for the same option (represented by C-preprocessor defines `SSS_PFSCP_ENABLE_xx`) must be set to `0`.



```
/* Enable one of these
 * If none is selected, default config would be used
 */
#define SSS_PFSCP_ENABLE_SE050A1 0
#define SSS_PFSCP_ENABLE_SE050A2 0
#define SSS_PFSCP_ENABLE_SE050B1 0
#define SSS_PFSCP_ENABLE_SE050B2 0
#define SSS_PFSCP_ENABLE_SE050C1 0
#define SSS_PFSCP_ENABLE_SE050C2 0
#define SSS_PFSCP_ENABLE_SE050_DEVKIT 0
#define SSS_PFSCP_ENABLE_SE051A2 0
#define SSS_PFSCP_ENABLE_SE051C2 0
#define SSS_PFSCP_ENABLE_SE050F2 0
#define SSS_PFSCP_ENABLE_SE051C_0005A8FA 0
#define SSS_PFSCP_ENABLE_SE051A_0001A920 0
#define SSS_PFSCP_ENABLE_SE050E_0001A921 1
#define SSS_PFSCP_ENABLE_SE051W_0005A739 0
#define SSS_PFSCP_ENABLE_A5000_0004A736 0
#define SSS_PFSCP_ENABLE_SE050F2_0001A92A 0
#define SSS_PFSCP_ENABLE_OTHER 0
```

Figure 15. Select the acutal Platform SCP keys in infsl_sss_ftr.h.in

The Plug & Trust Middleware uses a feature file to select/detect used/enabled features within the middleware stack. The file `fsl_sss_ftr.h` is automatically generated into the used build directory.. CMake is overwritting the `fsl_sss_ftr.h` file every time CMake is invoked. CMake is using the SCP key settings of the `fsl_sss_ftr.h.in` file as input to generate the the `fsl_sss_ftr.h` file. You do not have to manually edit the `fsl_sss_ftr.h` feature file. Selections from CMake edit cache automatically updates the generated feature file.

Note: *The Platform SCP key selection in the `fsl_sss_ftr.h.in` CMake input file is persistent.*

The location of the generated fsl_sss_ftr.h feature header file is: `~/se_mw/simw-top_build/imx_native_se050_t1oi2c`

The following tables contains the the Platform SCP key header file define to be set to `1` (enable) for the different secure element and secure authenticator product variants.

Table 11. Platform SCP key define prefix for SE050E product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050E Dev. Board OM-SE050ARD-E	A921	`SSS_PFSCP_ENABLE_SE050E_0001A921`
SE050E2	A921	`SSS_PFSCP_ENABLE_SE050E_0001A921`

Table 12. Platform SCP key define prefix for SE050F product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050F Dev.Board OM-SE050ARD-F	A92A	`SSS_PFSCP_ENABLE_SE050F2_0001A92A`
SE050F2	A92A	`SSS_PFSCP_ENABLE_SE050F2_0001A92A`

Table 13. Platform SCP key define prefix for SE050 Previous Generation product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050A1	A204	`SSS_PFSCP_ENABLE_SE050A1`
SE050A2	A205	`SSS_PFSCP_ENABLE_SE050A2`
SE050B1	A202	`SSS_PFSCP_ENABLE_SE050B1`
SE050B2	A203	`SSS_PFSCP_ENABLE_SE050B2`
SE050C1	A200	`SSS_PFSCP_ENABLE_SE050C1`
SE050C2	A201	`SSS_PFSCP_ENABLE_SE050C2`
SE050 Dev Board OM-SE050ARD	A1F4	`SSS_PFSCP_ENABLE_SE050_DEVKIT`
SE050F2	A77E[1]	`SSS_PFSCP_ENABLE_SE050F2`

[1] All SE050F2 with variant A77E have date code in year 2021. All the SE050F2 with date code in the year 2022 have the variant identifier A92A.

Table 14. Platform SCP key define prefix for SE051 product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE051A2	A920	`SSS_PFSCP_ENABLE_SE051A_0001A920`
SE051C2	A8FA	`SSS_PFSCP_ENABLE_SE051C_0005A8FA`
SE051W2	A739	`SSS_PFSCP_ENABLE_SE051W_0005A739`
SE051A2	A565	`SSS_PFSCP_ENABLE_SE051A2`
SE051C2	A564	`SSS_PFSCP_ENABLE_SE051C2`

Table 15. Platform SCP key define prefix for A5000 product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
A5000 Dev. Board OM-A5000ARD	A736	`SSS_PFSCP_ENABLE_A5000_0004A736`
A5000	A736	`SSS_PFSCP_ENABLE_A5000_0004A736`

7.2.2 Defining the deault Platfrom SCP keys in a text file

For evaluation purpose the Plug & Trust middleware supports to store the Platform SCP key in a plain text file. For further details see Plug & Trust middleware documentation chapter *11.10 Using own Platform SCP03 keys*.

The following Linux commands can be used to create the Platform SCP key text file (`se050_Dev_Kit_scp_keys.txt`):

The Platform SCP key text file can be stored in any location. In this example the file is stored in: `~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin`

```
cd ~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin
```

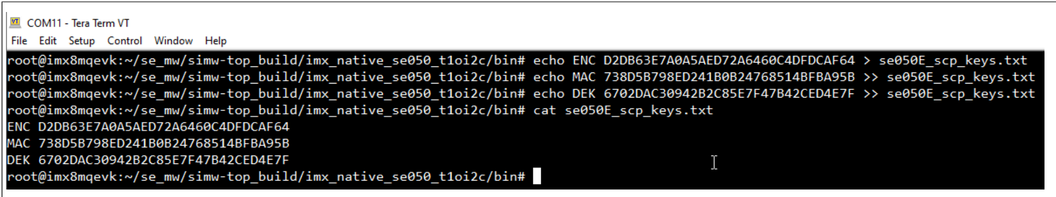
```
echo ENC D2DB63E7A0A5AED72A6460C4DFDCAF64 > se050E_scp_keys.txt
echo MAC 738D5B798ED241B0B24768514BFBA95B >> se050E_scp_keys.txt
echo DEK 6702DAC30942B2C85E7F47B42CED4E7F >> se050E_scp_keys.txt
```

Check the `se050E_scp_keys.txt`file content:

```
cat se050E_scp_keys.txt
```

The Linux environment variable `EX_SSS_BOOT_SCP03_PATH` is used to define the Platform SCP key textfile (filename and location).

```
export EX_SSS_BOOT_SCP03_PATH=~/se_mw/simw-top_build/
imx_native_se050_t1oi2c/bin/se050E_scp_keys.txt
```



```
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# echo ENC D2DB63E7A0A5AED72A6460C4DFDCAF64 > se050E_scp_keys.txt
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# echo MAC 738D5B798ED241B0B24768514BFBA95B >> se050E_scp_keys.txt
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# echo DEK 6702DAC30942B2C85E7F47B42CED4E7F >> se050E_scp_keys.txt
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# cat se050E_scp_keys.txt
ENC D2DB63E7A0A5AED72A6460C4DFDCAF64
MAC 738D5B798ED241B0B24768514BFBA95B
DEK 6702DAC30942B2C85E7F47B42CED4E7F
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin#
```

Figure 16. EdgeLock SE05xPlatform SCP plain text key file

Note: *In this example the MCIMX8M-EVKB is used for evaluation purpose only. Because different host MCU/MPU platforms are providing different hardware security mechanisms to protect keys it is not in the scope of this document to demonstrate how to store the Platform SCP shared binding keys securely. For commercial deployment the secure storage of Platform SCP keys must be adapted accordingly.*

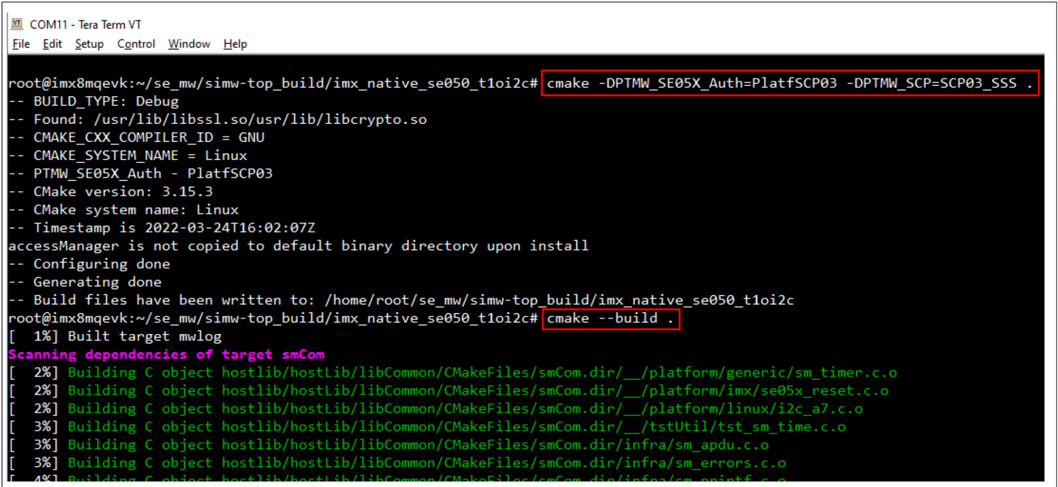
7.3 How to enable Platform SCP in the CMake-based build system

To enable Platform SCP is required to rebuild the SDK with the following CMake options:

- Select `SCP03_SSS` for the CMake option `PTMW_SCP`.
- Select `PlatfSCP03` for the CMake option `PTMW_SE05X_Auth`.

Run the following commands to update the CMake settings and rebuild the Plug & Trust middleware:

```
cd ~/se_mw/simw-top_build/imx_native_se050_t1oi2c
cmake -DPTMW_SE05X_Auth=PlatfSCP03 -DPTMW_SCP=SCP03_SSS .
cmake --build .
sudo make install
sudo ldconfig /usr/local/lib/
```



COM11 - Tera Term VT

```
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c# cmake -DPTMW_SE05X_Auth=PlatfSCP03 -DPTMW_SCP=SCP03_SSS .
-- BUILD_TYPE: Debug
-- Found: /usr/lib/libssl.so/usr/lib/libcrypto.so
-- CMAKE_CXX_COMPILER_ID = GNU
-- CMAKE_SYSTEM_NAME = Linux
-- PTMW_SE05X_Auth - PlatfSCP03
-- CMake version: 3.15.3
-- CMake system name: Linux
-- Timestamp is 2022-03-24T16:02:07Z
accessManager is not copied to default binary directory upon install
-- Configuring done
-- Generating done
-- Build files have been written to: /home/root/se_mw/simw-top_build/imx_native_se050_t1oi2c
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c# cmake --build .
[  1%] Built target mwlog
Scanning dependencies of target smCom
[  2%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/platform/generic/sm_timer.c.o
[  2%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/platform/imx/se05x_reset.c.o
[  2%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/platform/linux/i2c_a7.c.o
[  3%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/tstUtil/tst_sm_time.c.o
[  3%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/infra/sm_apdu.c.o
[  3%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/infra/sm_errors.c.o
```

Figure 17. SE050E CMake Settings - PlatformSCP enabled

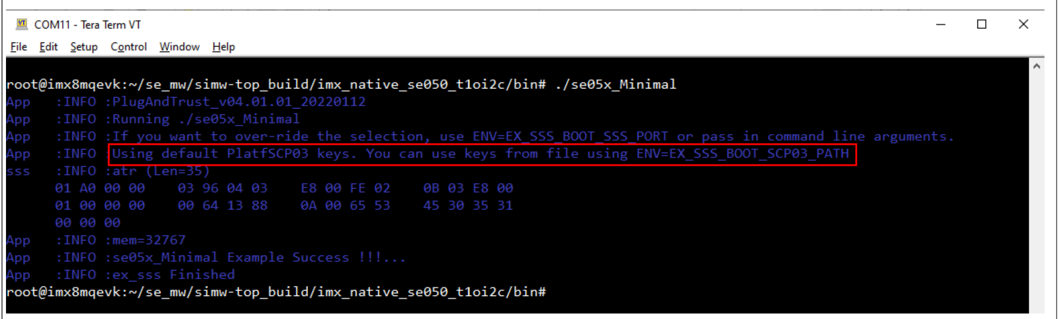
Note: *In this document the MCIMX8M-EVKB is used for evaluation purpose only. Because different host MCU/MPU platforms are providing different hardware security mechanisms to protect keys it is not in the scope of this document to demonstrate how to store the Platform SCP shared binding keys securely. For commercial deployment the secure storage of Platform SCP keys must be adapted accordingly.*

In the next step we can verify if we successfully enabled Platform SCP. For this purpose we run again the se05x_minimal example:

```
cd bin
```

```
./se05x_Minimal
```

Figure 18 shows the log output in case the Platform SCP keys are defined in the Plug & Trust middleware source code (see Defining the deault Platfrom SCP keys in the Plug & Trust middleware source code).

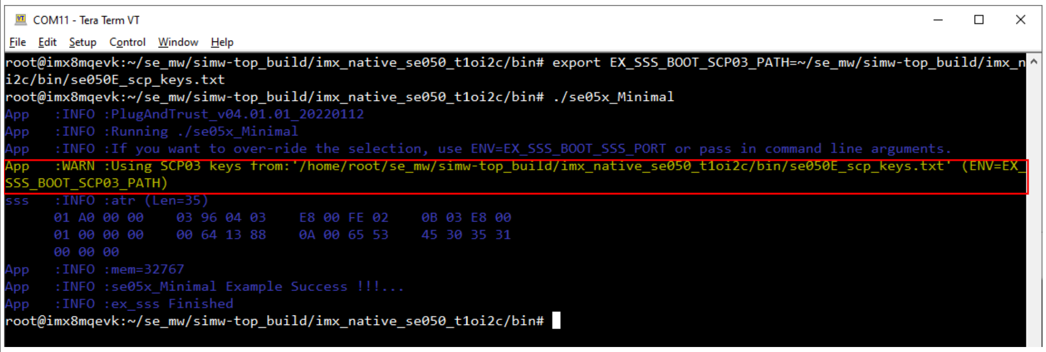


COM11 - Tera Term VT

```
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# ./se05x_Minimal
App   :INFO :PlugAndTrust_v04.01.01_20220112
App   :INFO :Running ./se05x_Minimal
App   :INFO :If you want to over-ride the selection, use ENV=EX_SSS_BOOT_SSS_PORT or pass in command line arguments.
App   :INFO :Using default PlatfSCP03 keys. You can use keys from file using ENV=EX_SSS_BOOT_SCP03_PATH
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 :mem=32767
App   :INFO :se05x_Minimal Example Success !!!...
App   :INFO :ex_sss Finished
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin#
```

Figure 18. Run se05x_minimal example with platformSCP enabled - SCP keys defined in the Plug & Trust middleware source

The log output for defining the Platform SCP keys via a text file (see Defining the default Platfrom SCP keys in a text file) is shown in Figure 19.



```
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# export EX_SSS_BOOT_SCP03_PATH=~/se_mw/simw-top_build/imx_n
i2c/bin/se050E_scp_keys.txt
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin# ./se05x_Minimal
App   :INFO :PlugAndTrust_v04.01.01_20220112
App   :INFO :Running ./se05x_Minimal
App   :INFO :If you want to over-ride the selection, use ENV=EX_SSS_BOOT_SSS_PORT or pass in command line arguments.
App   :WARN :Using SCP03 keys from:'/home/root/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin/se050E_scp_keys.txt' (ENV=EX_
SSS_BOOT_SCP03_PATH)
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 :mem=32767
App   :INFO :se05x_Minimal Example Success !!!...
App   :INFO :ex_sss Finished
root@imx8mqevk:~/se_mw/simw-top_build/imx_native_se050_t1oi2c/bin#
```

Figure 19. Run se05x_minimal example with platformSCP enabled - SCP keys defined in a text file

The Plug & Trust Middleware provides the following additional examples to rotate the PlatformSCP Keys and to mandate Platform SCP.

- **SE05X Rotate PlatformSCP Keys example**: Showcases authentication with default Platform SCP keys and the rotation (update) of those keys with user defined keys. The example documentation is available in the EdgeLock SE05x Plug & Trust Middleware documentation (`simw-top/doc/demos/se05x/se05x_RotatePlatformSCP03Keys/Readme.html`). The example source code is available at `/simw-top/demos/se05x/se05x_RotatePlatformSCP03Keys`.
- **SE05X Mandate SCP example**: Showcases how to make Platform SCP authentication mandatory in EdgeLock SE05x. The example documentation is available in the EdgeLock SE05x Plug & Trust Middleware documentation (`/simw-top/doc/demos/se05x/se05x_MandatePlatformSCP/Readme.html`). The example source code is available at `/simw-top/demos/se05x/se05x_MandatePlatformSCP`.
- **SE05x AllowWithout PlatformSCP example**: This project demonstrates how to configure SE05X to allow without platform SCP. The example documentation is available in´the EdgeLock SE05x Plug & Trust Middleware documentation (`~/se_mw/simwtop/doc/demos/se05x/se05x_AllowWithoutPlatformSCP/Readme.html`). The example source code is available at `~/se_mw/simw-top/demos/se05x/se05x_AllowWithoutPlatformSCP`.

8 Manage access from multiple Linux processes to the EdgeLock SE05x

The Plug & Trust middleware provides the Access Manager to support concurrent access from multiple linux processes to the EdgeLock SE05x IoT applet. The Access Manager can establish a connection to the EdgeLock SE05x IoT applet either as a plain connection or using Platform SCP.

Client processes are connecting over the JRCPv1 protocol to the Access Manager.

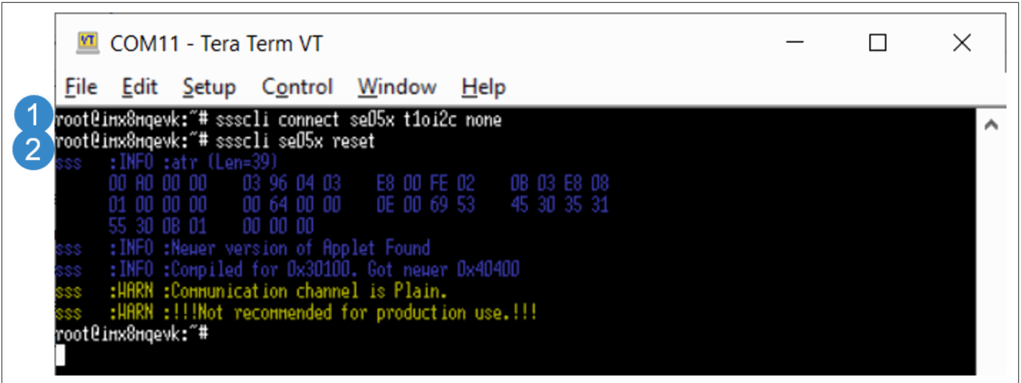
Please refer to the Plug & Trust middleware documentation chapter Access Manager: Manage access from multiple (Linux) processes to an SE05x IoT Applet for more details.

9 Appendix A: Using the ssscli tool

In Section 3 and Section 4 we have prepared the hardware setup and the software setups respectively. To validate that the whole process was done correctly and that your setup is fully operational, we are going to run the `ssscli` tool. This tool can be used to interact with the EdgeLock SE05x security IC without having to write any code.

To start the `ssscli` tool, send the commands shown in Figure 20:

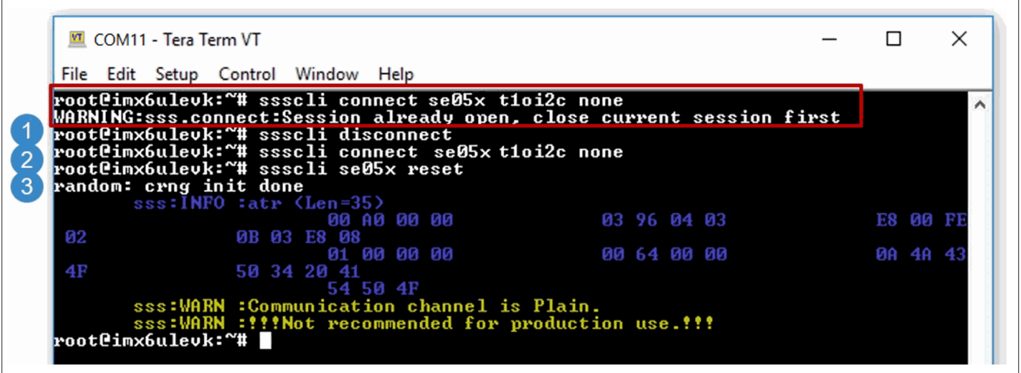
1. Open the connection:
   Send: >`ssscli connect se05x t1oi2c none`
   Note: In case ssscli did not get installed automatically in the image, execute these steps to finish the installation first:
   >`cd simw-top/pycli/src`
   >`python3 setup.py develop`
2. Send the reset command:
   Send: >`ssscli se05x reset`



```
root@imx8mqevk:~# ssscli connect se05x t1oi2c none
root@imx8mqevk:~# ssscli se05x reset
sss   :INFO :atr (Len=39)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0E 00 69 53    45 30 35 31
      55 30 0B 01    00 00 00
sss   :INFO :Newer version of Applet Found
sss   :INFO :Compiled for 0x30100. Got newer 0x40400
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
root@imx8mqevk:~#
```

Figure 20. Start the ssscli tool

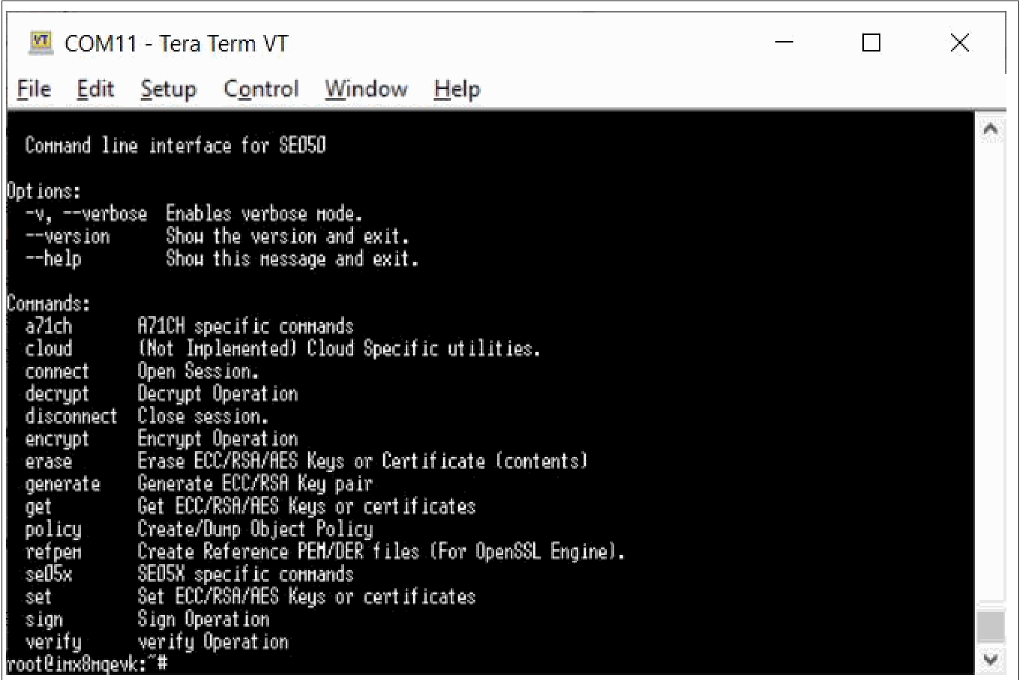
Note: If you see the following message: *WARNING:sss.connect:Session already open, close current session first* as shown in Figure 21, it means that you have a session open. To close it, send: (1) > `ssscli disconnect` and then send once again (2) >`ssscli connect se05x t1oi2c none` and later (3) >`ssscli se05x reset`:



```
root@imx6ulevk:~# ssscli connect se05x t1oi2c none
WARNING:sss.connect:Session already open, close current session first
root@imx6ulevk:~# ssscli disconnect
root@imx6ulevk:~# ssscli connect se05x t1oi2c none
root@imx6ulevk:~# ssscli se05x reset
random: crng init done
       sss:INFO :atr (Len=35)
                  00 A0 00 00            03 96 04 03            E8 00 FE
 02             0B 03 E8 08
                  01 00 00 00            00 64 00 00            0A 4A 43
 4F             50 34 20 41
                  54 50 4F
       sss:WARN :Communication channel is Plain.
       sss:WARN :!!!Not recommended for production use.!!!
root@imx6ulevk:~#
```

Figure 21. Close an already openned session

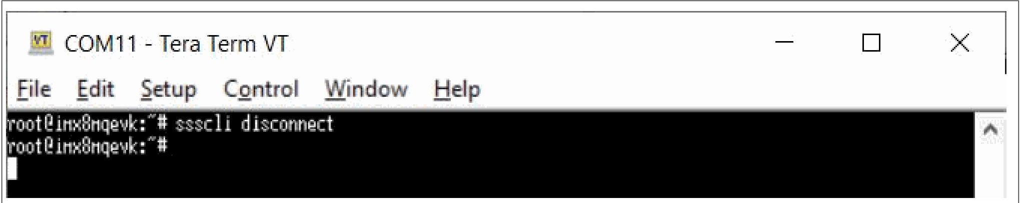
3. The SE05x ssscli tool supports several operations. To check which commands are supported by the ssscli tool:
   (Figure 22) Send: > `ssscli --help`



```
 Command line interface for SE050

Options:
  -v, --verbose  Enables verbose mode.
  --version      Show the version and exit.
  --help         Show this message and exit.

Commands:
  a71ch       A71CH specific commands
  cloud       (Not Implemented) Cloud Specific utilities.
  connect     Open Session.
  decrypt     Decrypt Operation
  disconnect  Close session.
  encrypt     Encrypt Operation
  erase       Erase ECC/RSA/AES Keys or Certificate (contents)
  generate    Generate ECC/RSA Key pair
  get         Get ECC/RSA/AES Keys or certificates
  policy      Create/Dump Object Policy
  refpem      Create Reference PEM/DER files (For OpenSSL Engine).
  se05x       SE05X specific commands
  set         Set ECC/RSA/AES Keys or certificates
  sign        Sign Operation
  verify      verify Operation
root@imx8mqevk:~#
```

Figure 22. ssscli info

4. Once you are done using the ssscli tool, close the session with the EdgeLock SE05x security IC:
   (Figure 23) Send: > `ssscli disconnect`



```
root@imx8mqevk:~# ssscli disconnect
root@imx8mqevk:~#
```

Figure 23. ssscli disconnect

The ssscli tool uses the installed sss library (/usr/local/lib/libsssapisw.so) for communication with the secure element. In order to be able to connect with ssscli to a specific interface type (e.g. T1oI2C or JRCP) or secure element type (A71CH or SE05x) the ssslibrary needs to be compiled and installed with this specific interface/ secure element type selected in the compilation options. This can be done by recompiling the Plug & Trust middleware as shown in the third step of the Appendix B (Section 7.2) .

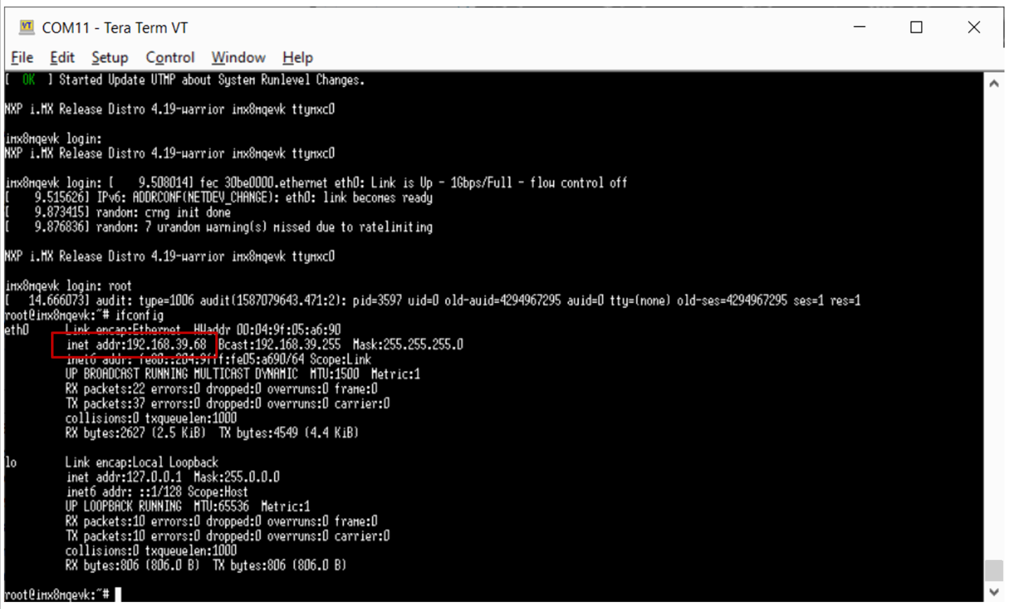
10 Appendix B: Update Plug & Trust middleware

In this section it will be described the process needed for downloading, compiling and building a different Plug & Trust middleware version from the one preinstalled in the SD card image seen at Section 5.

10.1 Obtain the latest Plug & Trust middleware version

We need to compile the Plug & Trust middleware into the Linux software image we flashed before. Follow these steps:

1. Download latest version of the Plug & Trust middleware using this link.
2. Connect to the board using a terminal application such as TeraTerm and run the `ifconfig` command to determine the IP address of your board as shown in Figure 24. We will use the board IP address to transfer the Plug & Trust middleware using the SCP protocol.

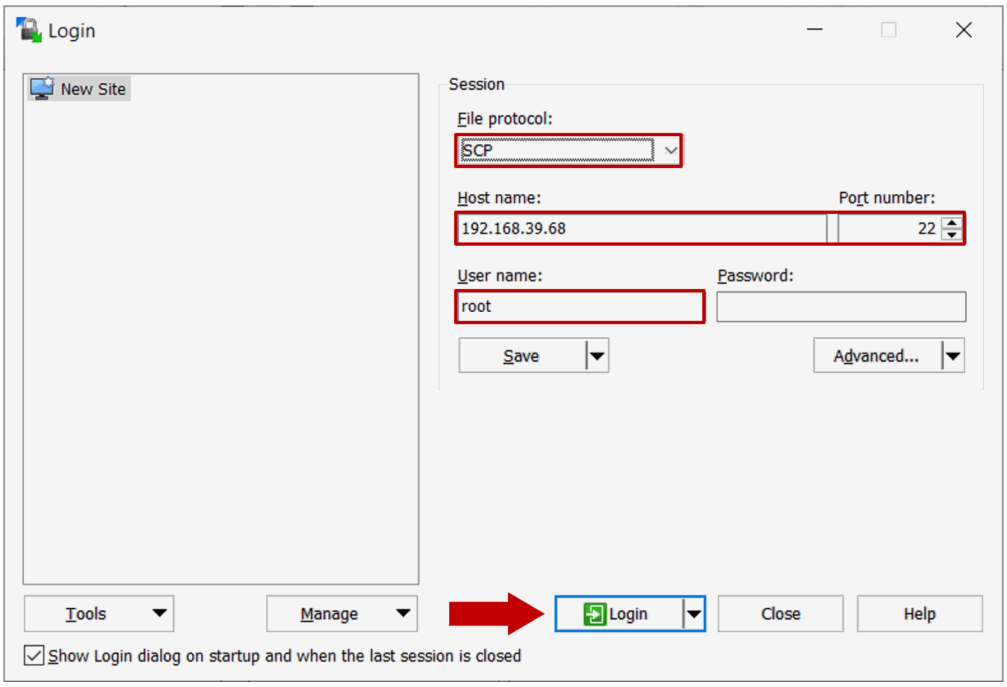


```
[  OK  ] Started Update UTMP about System Runlevel Changes.

NXP i.MX Release Distro 4.19-warrior imx8mqevk ttymxc0

imx8mqevk login:
NXP i.MX Release Distro 4.19-warrior imx8mqevk ttymxc0

imx8mqevk login: [    9.508014] fec 30be0000.ethernet eth0: Link is Up - 1Gbps/Full - flow control off
[    9.515626] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[    9.873415] random: crng init done
[    9.876836] random: 7 urandom warning(s) missed due to ratelimiting

NXP i.MX Release Distro 4.19-warrior imx8mqevk ttymxc0

imx8mqevk login: root
[   14.666073] audit: type=1006 audit(1587079643.471:2): pid=3597 uid=0 old-auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=1 res=1
root@imx8mqevk:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:9f:05:a6:90
          inet addr:192.168.39.68  Bcast:192.168.39.255  Mask:255.255.255.0
          inet6 addr: fe80::204:9fff:fe05:a690/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2627 (2.5 KiB)  TX bytes:4549 (4.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:806 (806.0 B)  TX bytes:806 (806.0 B)

root@imx8mqevk:~#
```

Figure 24. Obtain board IP address

3. Download and install a file transfer software supporting SCP such as WinSCP in case you are using a Windows host machine.

4. **Windows host:** open WinSCP and configure the SCP connection as shown in Figure 25. The Host name corresponds to the IP address of the board obtained previously. Click on the *Login* button to establish the connection.



File protocol: SCP

Host name: 192.168.39.68

Port number: 22

User name: root

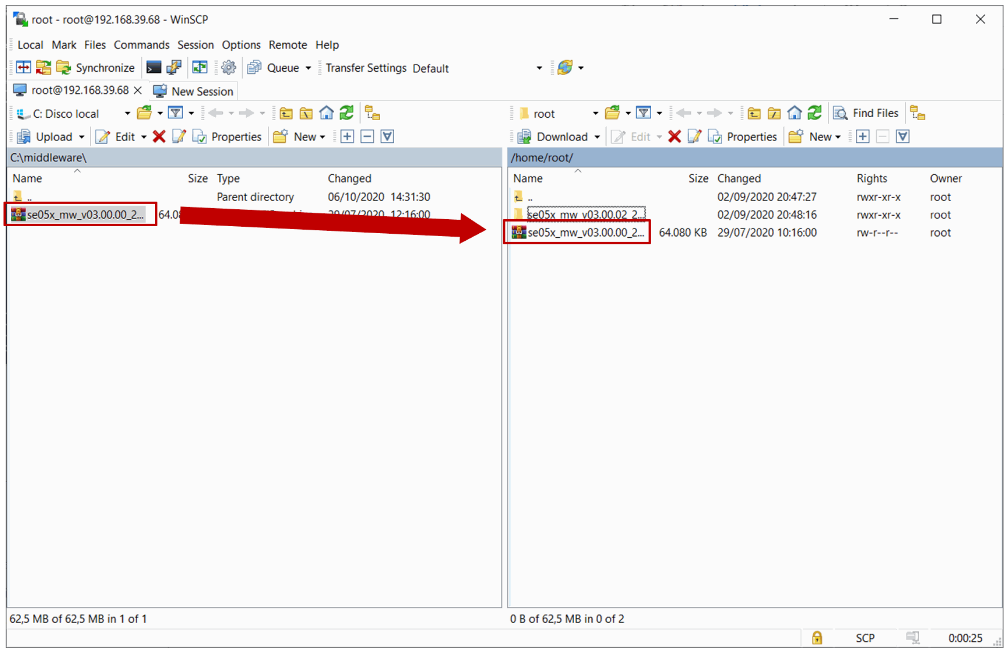
Password:

☑ Show Login dialog on startup and when the last session is closed

Figure 25. Connect to the board using SCP with WinSCP

5. Once the connection is established, you should see on the right pane the board file system and, on the left pane the file system of the host machine. It is recommended to delete the folder named *se05x_mw_vxxx* in order to avoid confusion. Navigate to the folder where you downloaded the Plug & Trust middleware package and then drag

and drop the package to the right pane as shown in Figure 26. A copy of the Plug & Trust middleware file should now be in the `/home/root folder` of the board.


root - root@192.168.39.68 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize Queue Transfer Settings Default
root@192.168.39.68 New Session
C: Disco local
Upload Edit Properties New
root
Find Files
Download Edit Properties New
SCP
0:00:25

C:\middleware\

Name	Size	Type	Changed
..		Parent directory	06/10/2020 14:31:30
se05x_mw_v03.00.00_2...	64.0[illegible]	[illegible]	[illegible] 12:16:00

62,5 MB of 62,5 MB in 1 of 1

/home/root/

Name	Size	Changed	Rights	Owner
..		02/09/2020 20:47:27	rwxr-xr-x	root
se05x_mw_v03.00.02_2...		02/09/2020 20:48:16	rwxr-xr-x	root
se05x_mw_v03.00.00_2...	64.080 KB	29/07/2020 10:16:00	rw-r--r--	root

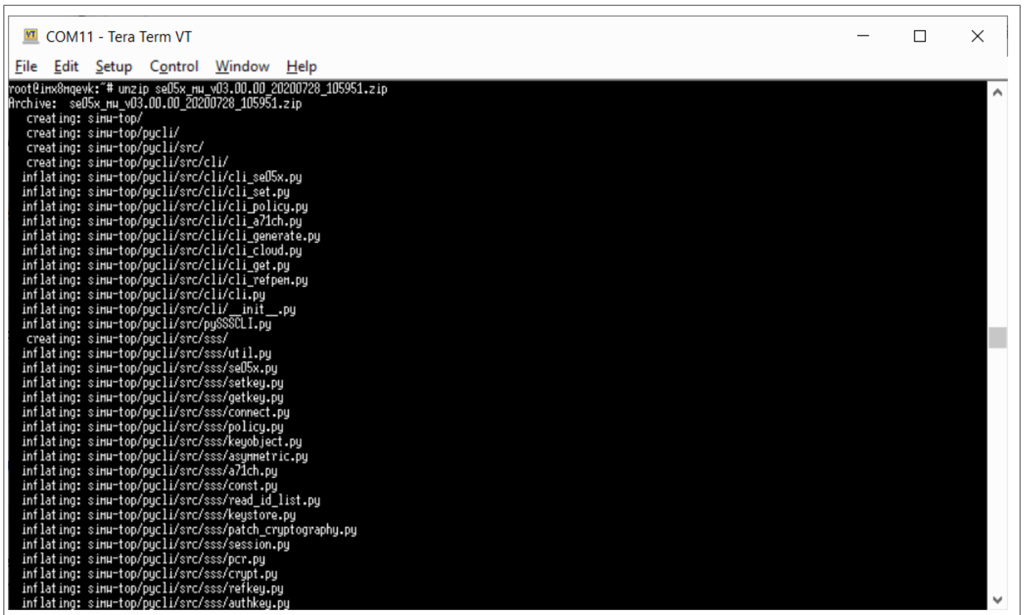
0 B of 62,5 MB in 0 of 2

Figure 26. Copy the middleware to the board

10.2 Build the Plug & Trust middleware

To build the Plug & Trust middleware, open the TeraTerm and follow the steps listed below:

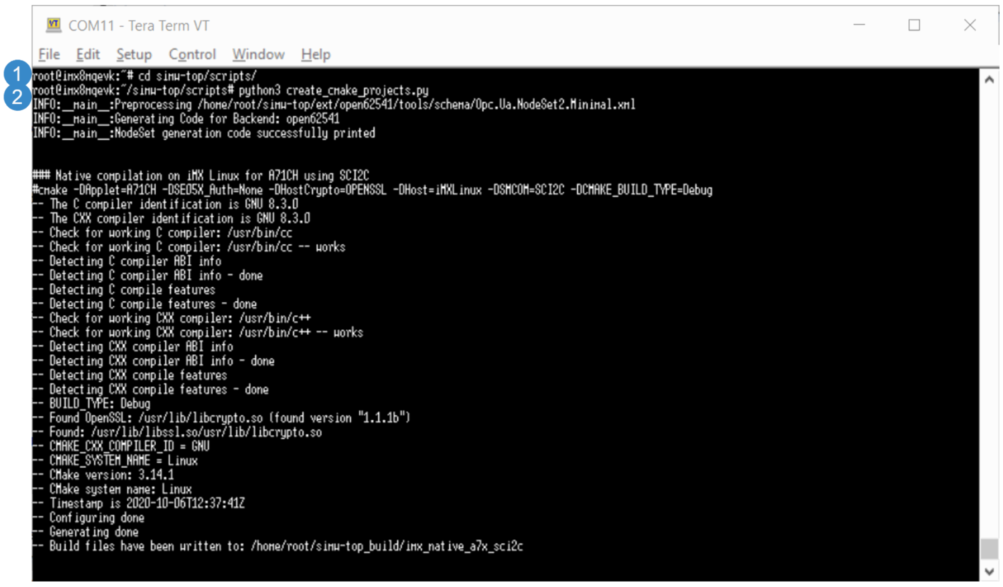
1. Unzip the Plug & Trust middleware file that you have transferred to the board in Section 10.1.
   Send `unzip <middleware_file_name>.zip` as shown in Figure 27.



```
root@imx8mqevk:~# unzip se05x_mw_v03.00.00_20200728_105951.zip
Archive:  se05x_mw_v03.00.00_20200728_105951.zip
   creating: simw-top/
   creating: simw-top/pycli/
   creating: simw-top/pycli/src/
   creating: simw-top/pycli/src/cli/
  inflating: simw-top/pycli/src/cli/cli_se05x.py
  inflating: simw-top/pycli/src/cli/cli_set.py
  inflating: simw-top/pycli/src/cli/cli_policy.py
  inflating: simw-top/pycli/src/cli/cli_a71ch.py
  inflating: simw-top/pycli/src/cli/cli_generate.py
  inflating: simw-top/pycli/src/cli/cli_cloud.py
  inflating: simw-top/pycli/src/cli/cli_get.py
  inflating: simw-top/pycli/src/cli/cli_refpem.py
  inflating: simw-top/pycli/src/cli/cli.py
  inflating: simw-top/pycli/src/cli/__init__.py
  inflating: simw-top/pycli/src/pySSSCLI.py
   creating: simw-top/pycli/src/sss/
  inflating: simw-top/pycli/src/sss/util.py
  inflating: simw-top/pycli/src/sss/se05x.py
  inflating: simw-top/pycli/src/sss/setkey.py
  inflating: simw-top/pycli/src/sss/getkey.py
  inflating: simw-top/pycli/src/sss/connect.py
  inflating: simw-top/pycli/src/sss/policy.py
  inflating: simw-top/pycli/src/sss/keyobject.py
  inflating: simw-top/pycli/src/sss/asymmetric.py
  inflating: simw-top/pycli/src/sss/a71ch.py
  inflating: simw-top/pycli/src/sss/const.py
  inflating: simw-top/pycli/src/sss/read_id_list.py
  inflating: simw-top/pycli/src/sss/keystore.py
  inflating: simw-top/pycli/src/sss/patch_cryptography.py
  inflating: simw-top/pycli/src/sss/session.py
  inflating: simw-top/pycli/src/sss/pcr.py
  inflating: simw-top/pycli/src/sss/crypt.py
  inflating: simw-top/pycli/src/sss/refkey.py
  inflating: simw-top/pycli/src/sss/authkey.py
```

Figure 27. Unzip the middleware

2. A new folder called *simw-top* should have been created in the root directory. You can now create the middleware CMake projects for the board as shown in Figure 28 :
   (1) Navigate to the `simw-top/scripts` folder.
   Send `cd simw-top/scripts/.`
   (2) Send the command `python3 create_cmake_projects.py` and wait until the command is finished executing.
   Note: *This command may take a few seconds to complete.*

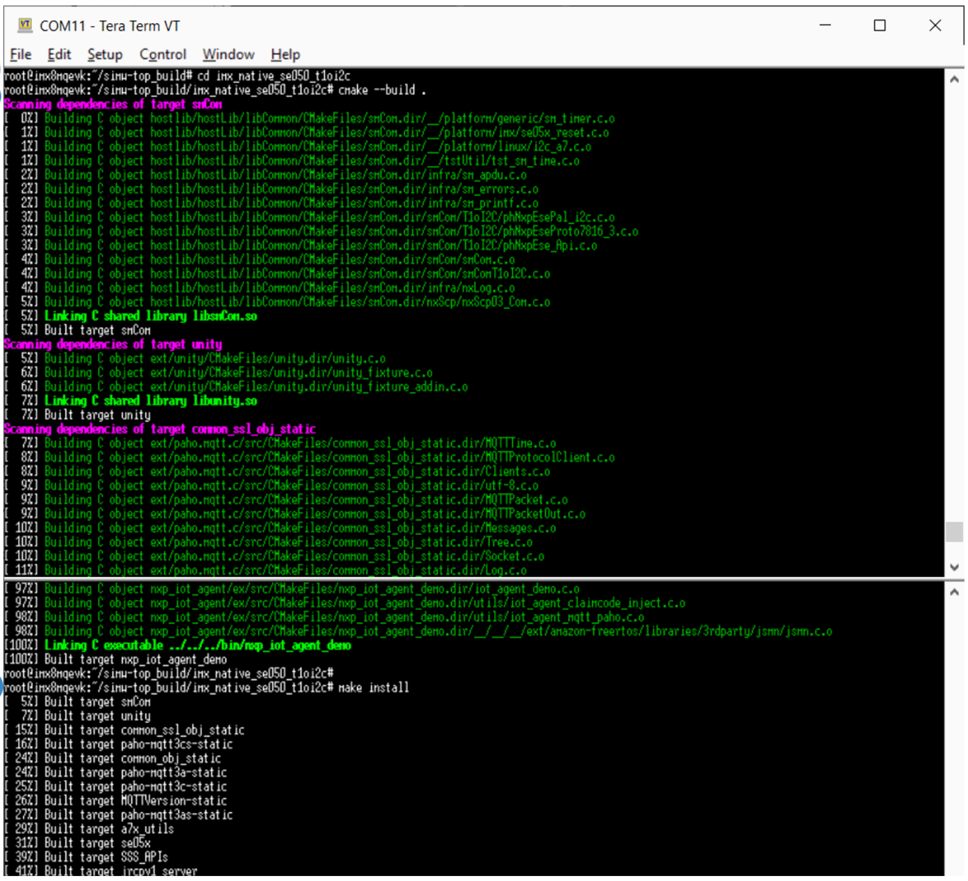


```
root@imx8mqevk:~# cd simw-top/scripts/
root@imx8mqevk:~/simw-top/scripts# python3 create_cmake_projects.py
INFO:__main__:Preprocessing /home/root/simw-top/ext/open62541/tools/schema/Opc.Ua.NodeSet2.Minimal.xml
INFO:__main__:Generating Code for Backend: open62541
INFO:__main__:NodeSet generation code successfully printed


### Native compilation on iMX Linux for A71CH using SCI2C
#cmake -DApplet=A71CH -DSE05X_Auth=None -DHostCrypto=OPENSSL -DHost=iMXLinux -DSMCOM=SCI2C -DCMAKE_BUILD_TYPE=Debug
-- The C compiler identification is GNU 8.3.0
-- The CXX compiler identification is GNU 8.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- BUILD_TYPE: Debug
-- Found OpenSSL: /usr/lib/libcrypto.so (found version "1.1.1b")
-- Found: /usr/lib/libssl.so/usr/lib/libcrypto.so
-- CMAKE_CXX_COMPILER_ID = GNU
-- CMAKE_SYSTEM_NAME = Linux
-- CMake version: 3.14.1
-- CMake system name: Linux
-- Timestamp is 2020-10-06T12:37:41Z
-- Configuring done
-- Generating done
-- Build files have been written to: /home/root/simw-top_build/imx_native_a7x_sci2c
```

Figure 28. Create CMake projects for the board

3. Finally, compile and install the project as shown in Figure 29:
   (1) Navigate to the folder where the project has been generated.
   Send `cd ../../simw-top_build/imx_native_se050_t1oi2c`
   (2) Build the project. This might take some time.
   Send `cmake --build .`
   Note: The default build configuration of the Plug & Trust middleware ≤ `V04.01.0x` generates code for the OM-SE050ARD development board. You need to adapt the CMake settings in case you are using a different EdgeLock secure element

development board or a different secure element product IC. The settings are described in Section 6.
Note: *in the default CMake configuration the SCP03 protocol is not active. How to enable Platform SCP is described in* Section 7.
(3) Install the project.
Send `sudo make install`



```
root@imx8mqevk:~/simw-top_build# cd imx_native_se050_t1oi2c
root@imx8mqevk:~/simw-top_build/imx_native_se050_t1oi2c# cmake --build .
Scanning dependencies of target smCom
[  0%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/platform/generic/sm_timer.c.o
[  1%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/platform/imx/se05x_reset.c.o
[  1%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/platform/linux/i2c_a7.c.o
[  1%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/__/tstUtil/tst_sm_time.c.o
[  2%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/infra/sm_apdu.c.o
[  2%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/infra/sm_errors.c.o
[  2%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/infra/sm_printf.c.o
[  3%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/smCom/T1oI2C/phNxpEsePal_i2c.c.o
[  3%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/smCom/T1oI2C/phNxpEseProto7816_3.c.o
[  3%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/smCom/T1oI2C/phNxpEse_Api.c.o
[  4%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/smCom/smCom.c.o
[  4%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/smCom/smComT1oI2C.c.o
[  4%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/infra/nxLog.c.o
[  5%] Building C object hostlib/hostLib/libCommon/CMakeFiles/smCom.dir/nxScp/nxScp03_Com.c.o
[  5%] Linking C shared library libsmCom.so
[  5%] Built target smCom
Scanning dependencies of target unity
[  5%] Building C object ext/unity/CMakeFiles/unity.dir/unity.c.o
[  6%] Building C object ext/unity/CMakeFiles/unity.dir/unity_fixture.c.o
[  6%] Building C object ext/unity/CMakeFiles/unity.dir/unity_fixture_addin.c.o
[  7%] Linking C shared library libunity.so
[  7%] Built target unity
Scanning dependencies of target common_ssl_obj_static
[  7%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/MQTTTime.c.o
[  8%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/MQTTProtocolClient.c.o
[  8%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/Clients.c.o
[  9%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/utf-8.c.o
[  9%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/MQTTPacket.c.o
[  9%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/MQTTPacketOut.c.o
[ 10%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/Messages.c.o
[ 10%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/Tree.c.o
[ 10%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/Socket.c.o
[ 11%] Building C object ext/paho.mqtt.c/src/CMakeFiles/common_ssl_obj_static.dir/Log.c.o
```

```
[ 97%] Building C object nxp_iot_agent/ex/src/CMakeFiles/nxp_iot_agent_demo.dir/iot_agent_demo.c.o
[ 97%] Building C object nxp_iot_agent/ex/src/CMakeFiles/nxp_iot_agent_demo.dir/utils/iot_agent_claimcode_inject.c.o
[ 98%] Building C object nxp_iot_agent/ex/src/CMakeFiles/nxp_iot_agent_demo.dir/utils/iot_agent_mqtt_paho.c.o
[ 98%] Building C object nxp_iot_agent/ex/src/CMakeFiles/nxp_iot_agent_demo.dir/__/__/__/ext/amazon-freertos/libraries/3rdparty/jsmn/jsmn.c.o
[100%] Linking C executable ../../../bin/nxp_iot_agent_demo
[100%] Built target nxp_iot_agent_demo
root@imx8mqevk:~/simw-top_build/imx_native_se050_t1oi2c#
root@imx8mqevk:~/simw-top_build/imx_native_se050_t1oi2c# make install
[  5%] Built target smCom
[  7%] Built target unity
[ 15%] Built target common_ssl_obj_static
[ 16%] Built target paho-mqtt3cs-static
[ 24%] Built target common_obj_static
[ 24%] Built target paho-mqtt3a-static
[ 25%] Built target paho-mqtt3c-static
[ 26%] Built target MQTTVersion-static
[ 27%] Built target paho-mqtt3as-static
[ 29%] Built target a7x_utils
[ 31%] Built target se05x
[ 39%] Built target SSS_APIs
[ 41%] Built target ircpv1_server
```

Figure 29. Build and install the project

(4) Refresh linker cache to let it find the newly installed libraries
Send `sudo ldconfig /usr/local/lib`

11 Legal information

11.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

11.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

11.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Tables

Figures

Contents

Date of release: 12 September 2022