

# AN12396

## EdgeLock SE05x Quick start guide with FRDM-K64F

Rev. 3.3 — 4 August 2022  
534433

Application note

### Document information

Information	Content
Keywords	EdgeLock SE05x, EdgeLock A5000, Plug & Trust middleware, FRDM-K64F
Abstract	This document explains how to get started with the EdgeLock Plug & Trust middleware using the EdgeLock SE05x/A5000 development boards and FRDM-K64F MCU board. It provides detailed instructions to run projects imported either from the FRDM-K64F SDK or the CMake-based build system included in the EdgeLock SE05x Plug & Trust middleware.



## Revision history

### Revision history

Revision number	Date	Description
1.0	2019-06-08	First document release
1.1	2019-06-20	Update of board figures
2.0	2019-11-25	Major update to incorporate details to import projects from FRDM-K64F SDK and CMake-based build system.
2.1	2019-12-17	Corrected OM-SE050ARD-E J14 jumper setting.
3.0	2020-10-27	Updated for EdgeLock SE051
3.1	2020-12-07	Updated to latest template and fixed broken links
3.2	2022-03-28	<p>Add EdgeLock SE050E and EdgeLock A5000 product variants. Update <a href="#">Table 1</a>, <a href="#">Figure 1</a>, <a href="#">Figure 2</a>, <a href="#">Figure 3</a>, <a href="#">Figure 10</a>, <a href="#">Figure 11</a>, <a href="#">Figure 15</a>, <a href="#">Figure 39</a>, <a href="#">Figure 40</a>, <a href="#">Figure 44</a>, <a href="#">Figure 55</a>, <a href="#">Figure 56</a>, <a href="#">Figure 57</a>, <a href="#">Figure 58</a>, <a href="#">Figure 59</a> and <a href="#">Figure 60</a></p> <p>Add note (step 3) in <a href="#">Section 4.5</a> Build, run and debug project example</p> <p>Add <a href="#">Section 4.6</a> Product specific build settings</p> <p>Add note in <a href="#">Section 5.6.2</a> Run EdgeLock SE05x Plug &amp; Trust middleware examples</p> <p>Add <a href="#">Section 5.7</a> Product specific CMake build settings</p> <p>Add <a href="#">Section 6</a> Binding EdgeLock SE05x to a host using Platform SCP</p>
3.3	2022-08-04	<p>Moved section "Update FRDM-K64F board with DAPLink firmware" into <a href="#">Section 10</a>.</p> <p>Update to EdgeLock SE Plug &amp; Trust Middleware version 04.02.xx.</p> <p>Update note (step 3) in <a href="#">Section 4.5</a> Build, run and debug project example</p> <p>Update <a href="#">Section 4.6</a> Product specific build settings</p> <p>Update note in <a href="#">Section 5.6.2</a> Run EdgeLock SE05x Plug &amp; Trust middleware examples</p> <p>Update <a href="#">Section 5.7</a> Product specific CMake build settings</p> <p>Update <a href="#">Section 6</a> Binding EdgeLock SE05x to a host using Platform SCP</p>

## 1 How to use this document

The EdgeLock SE05x 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. These project examples can be either:

- Imported from the MCUXpresso SDKs made available for FRDM-K64F MCU board.
- Imported from the CMake-based build system included in the EdgeLock SE05x Plug & Trust middleware package.

This document provides detailed instructions to run project examples for EdgeLock SE05x secure elements imported either from the FRDM-K64F SDK or the CMake-based build system. The FRDM-K64F SDK is recommended as it is the fastest way to import and run the project examples. The CMake-based option is provided for developers familiar with this build system or willing to run exactly the same project example on PC/Windows/Linux and embedded targets. 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. Setup your boards. [Section 3](#) describes how to setup the OM-SE05xARD boards and FRDM-K64F board;
3. Run project examples. Go to [Section 4](#) for instructions to import projects from the FRDM-K64F MCUXpresso SDK following the recommended way of working, or alternatively, go to [Section 5](#) for instructions to import projects from the CMake-based build system.

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 I<sup>2</sup>C interface. To follow the instructions provided in this document, you need an EdgeLock SE05x development board and a FRDM-K64F 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
<a href="#">OM-SE050ARD-E</a>	9354 332 66598	SE050E Arduino® compatible development kit	

**Table 1.** EdgeLock SE05x development boards. ...continued

Part number	12NC	Description	Picture
<a href="#">OM-SE050ARD-F</a>	9354 357 63598	SE050 Arduino® compatible development kit	
OM-SE050ARD	9353 832 82598	SE050F Arduino® compatible development kit	
<a href="#">OM-SE051ARD</a>	9353 991 87598	SE051 Arduino® compatible development kit	
<a href="#">OM-A5000ARD</a>	9354 243 19598	A5000 Arduino® compatible development kit	

**Note:** The pictures in this guide will show SE050E, but all boards in [Table 1](#) can be used as well with the same hardware configuration.

### FRDM-K64F MCU board ordering details

[Table 2](#) details the ordering details for the FRDM-K64F board.

**Table 2.** FRDM-K64F details

Part number	12NC	Content	Picture
<a href="#">FRDM-K64F</a>	935326293598	Freedom development platform for Kinetis K64, K63 and K24 MCUs	

## 3 Boards setup

This section explains how to prepare the OM-SE050ARD-E boards and FRDM-K64F board to run the EdgeLock SE05x Plug & Trust middleware project examples. This consists of:

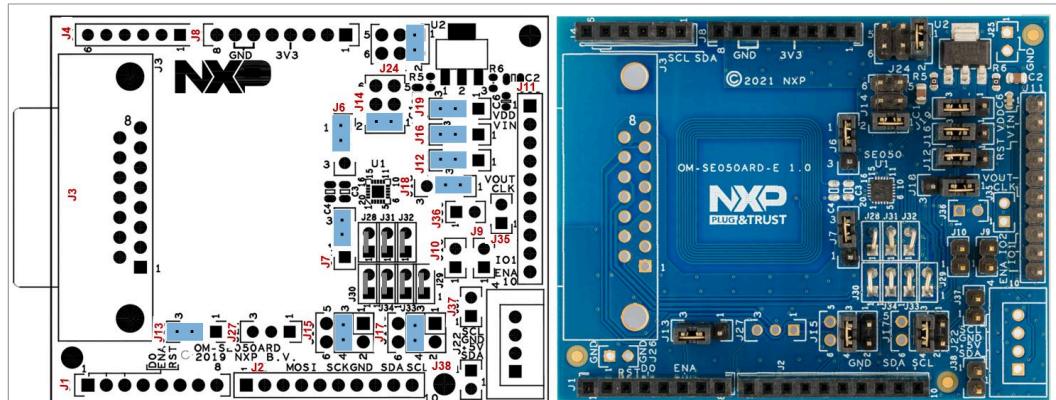
1. [OM-SE050ARD-E jumper configuration](#).
2. [OM-SE050ARD-E and FRDM-K64F board connection](#).

**Note:** If your FRDM-K64F board does not already contain the DAPLink firmware, you need to update the FRDM-K64F board as described in [Section 10](#).

### 3.1 OM-SE050ARD-E jumper configuration

The OM-SE050ARD-E boards have jumpers that allow you to configure the I<sup>2</sup>C interface of EdgeLock SE05x secure elements via the Arduino header. Configure the jumper settings as shown in [Figure 1](#) to enable this option.

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

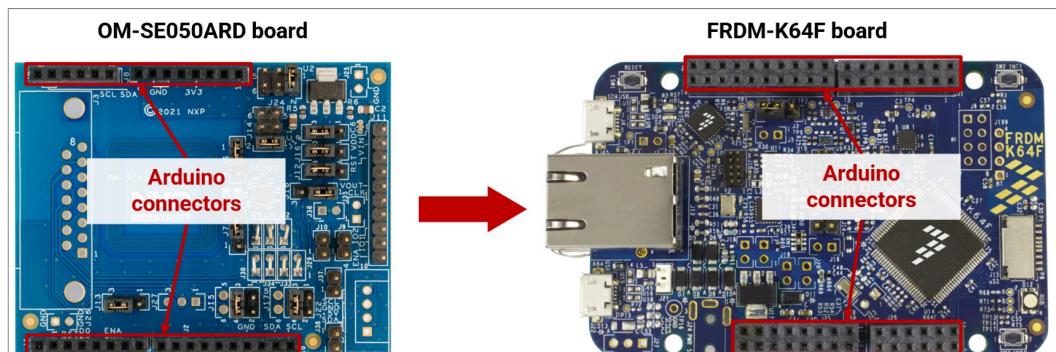


**Figure 1.** Jumper configuration for FRDM-K64F

### 3.2 OM-SE05xARD and FRDM-K64F board connection

The OM-SE05xARD boards and FRDM-K64F board can be directly connected using the Arduino connectors. The OM-SE05xARD boards come with male connectors while the FRDM-K64F board comes with female headers.

Mount any OM-SE05xARD board on top of the FRDM-K64F as shown in [Figure 2](#):



**Figure 2.** Arduino connectors of OM-SE05xARD and FRDM-K64F boards

Double check that the two boards are connected as shown in Figure 3:

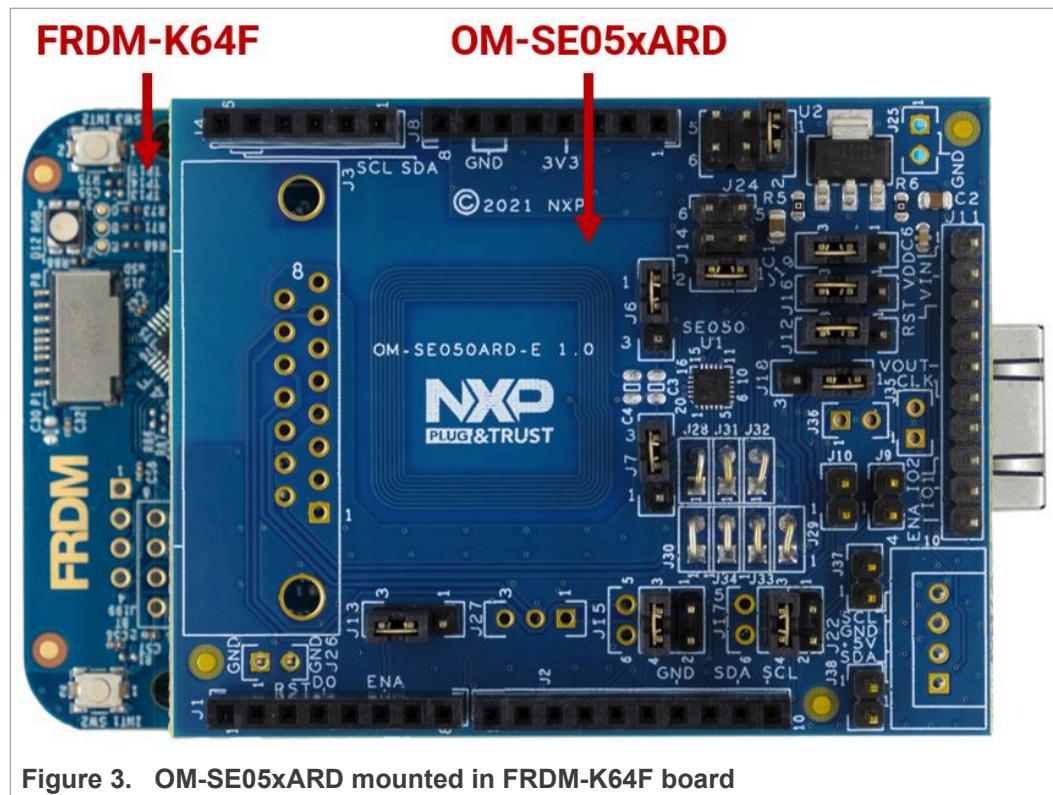


Figure 3. OM-SE05xARD mounted in FRDM-K64F board

**Note:** Refer to [Figure 1](#) for OM-SE05xARD jumper configuration.

## 4 Import project examples from FRDM-K64F SDK

This section explains how to run the example projects by importing them from the FRDM-K64F SDK. This option is the recommended one opposed to the [Section 5](#), since it implies that the MCU projects are self-contained standard MCUXpresso projects with a better debug experience.

### 4.1 Prerequisites

The following steps are required to run a project imported from the MCUXpresso SDK:

1. MCUXpresso IDE. Check [Section 7](#) for detailed installation instructions
2. TeraTerm (or an equivalent serial application). You can download and run TeraTerm installer from this [link](#).

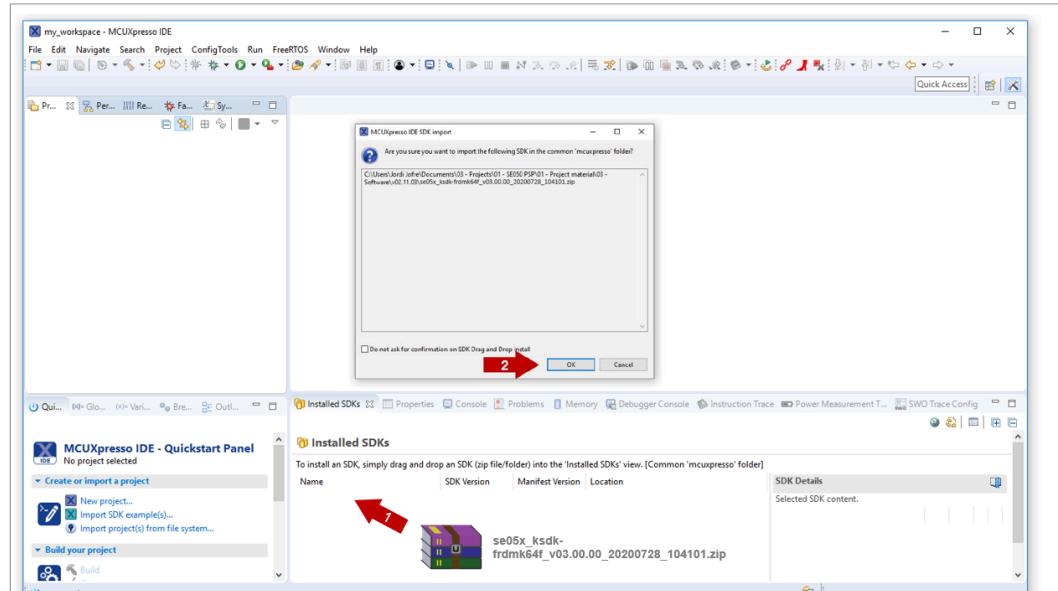
### 4.2 Download FRDM-K64F SDK

The project examples are included as part of the FRDM-K64F SDK. First, download the FRDM-K64F SDK, publicly available from the [SE050 website](#). This SDK is the recommended folder to work with, it contains the most updated files, the most complete list of project examples and guarantees the proper development of this quick start guide.

**Note:** The FRDM-K64F SDK you can download from [MCUXpresso SDK Builder website](#) may not include all the EdgeLock SE05x project examples or the latest version of them.

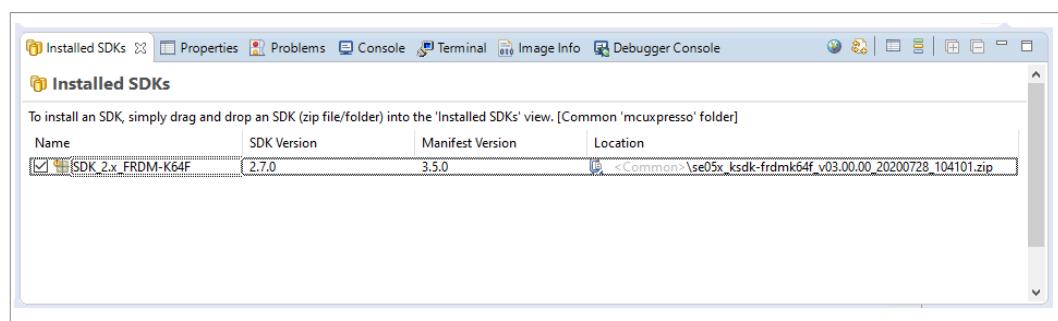
### 4.3 Install FRDM-K64F SDK

After downloading the FRDM-K64F SDK, we need to install it into our MCUXpresso workspace. To install the SDK, (1) drag and drop the FRDM-K64F SDK zip file in the **Installed SDKs** section in the bottom part of the MCUXpresso IDE and (2) click **OK** as shown in [Figure 4](#):



**Figure 4.** Import FRDM-K64F board SDK into MCUXpresso environment

If the SDK is successfully imported, you should see it listed in the **Installed SDK** window as shown in [Figure 5](#):



**Figure 5.** Imported FRDM-K64F SDK

### 4.4 Import project example in MCUXpresso

After importing the FRDM-K64F SDK in the MCUXpresso workspace, follow these instructions to import a project:

1. Click *Import SDK example(s)* in the MCUXpresso IDE quick start panel as shown in [Figure 6](#)

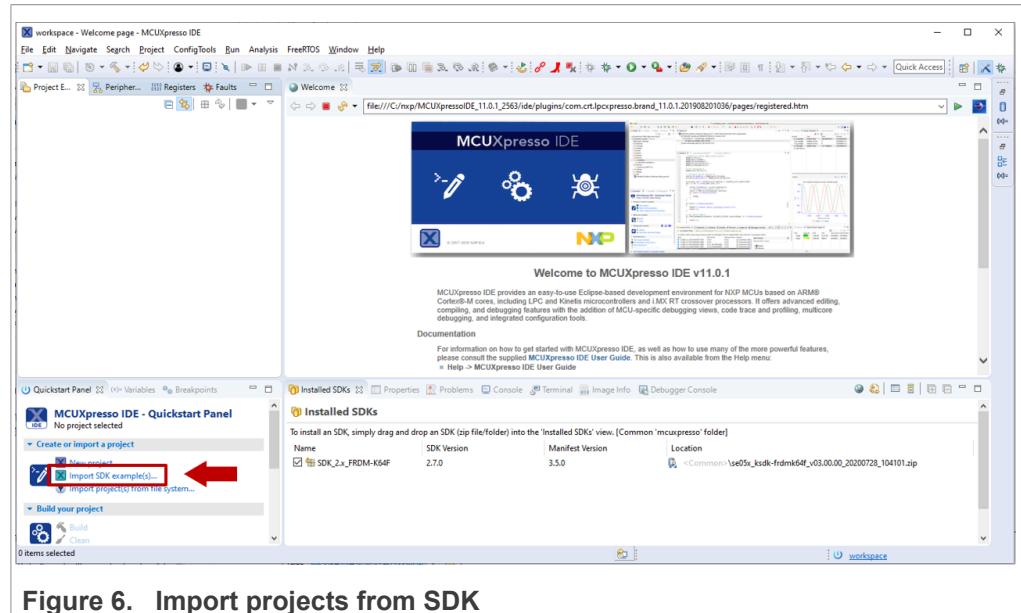
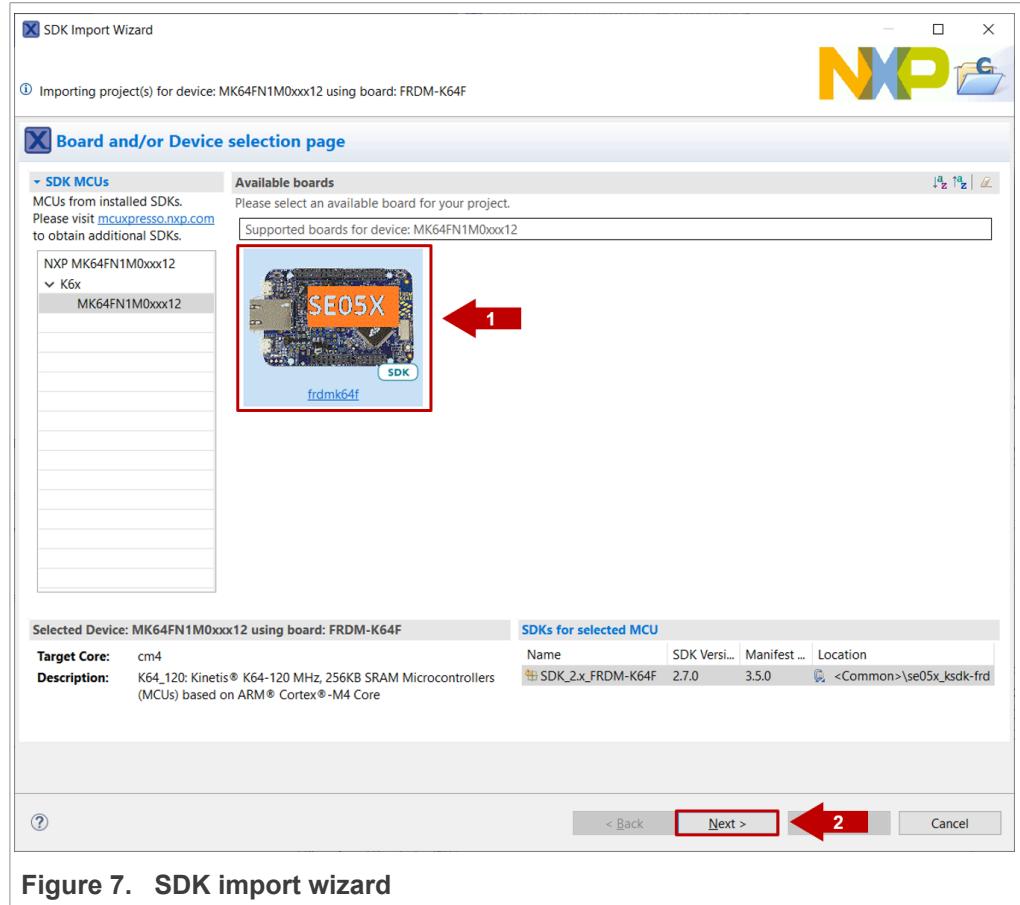


Figure 6. Import projects from SDK

2. The SDK import wizard will be opened. You should see a figure of an FRDM-K64F board with an orange label. Select the board and click *Next* button as shown in [Figure 7](#):



**Figure 7. SDK import wizard**

**Note:** If there is not an SE05x orange label on top of the board image, MCUXpresso may be recognizing a board SDK with a higher version number, downloaded from [MCUXpresso SDK Builder website](#). To access the most up-to-date and complete list of EdgeLock SE05x project examples, first you need to uninstall the SDK currently installed, and then repeat the process indicated in [Figure 4](#).

3. Under the `se_hostlib_examples` drop down list, you have the list of supported project examples for the FRDM-K64F. Select the examples you would like to import in your MCUXpresso workspace and click *Finish* button as shown in [Figure 8](#). For the

scope of this guide, you should select the `se05x_Minimal` project as an example. The same process can be done with the rest of the examples.

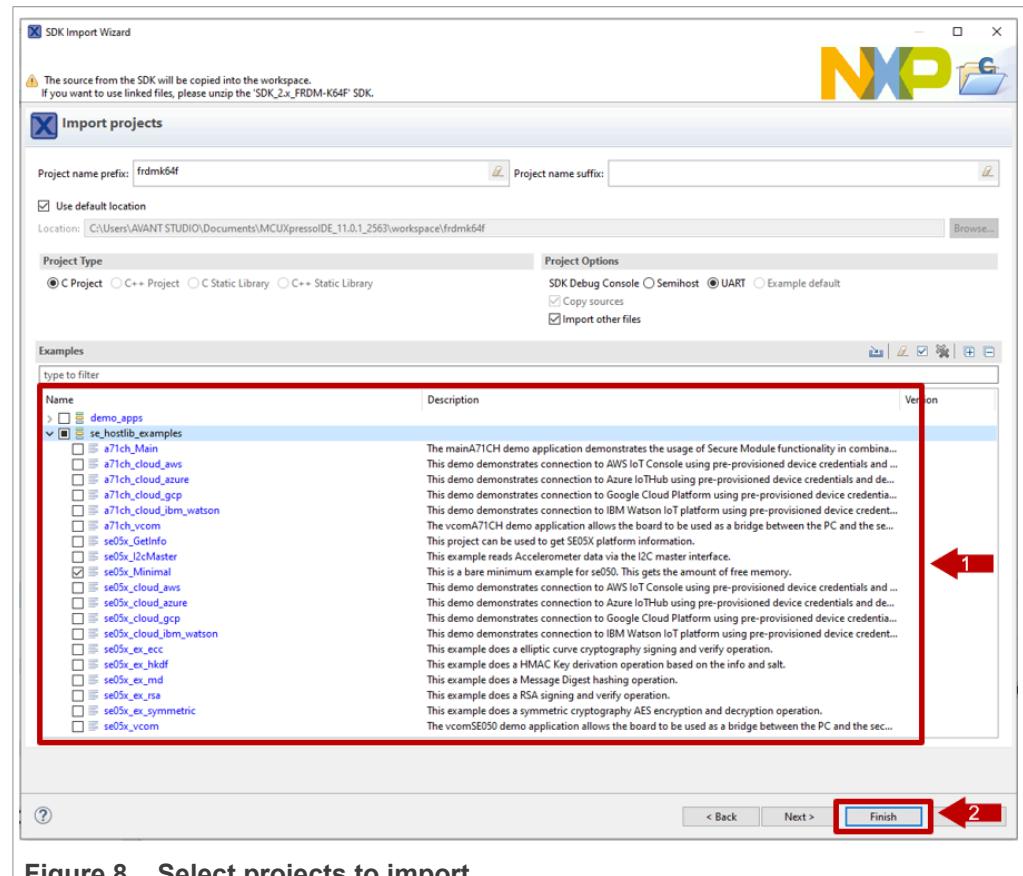


Figure 8. Select projects to import

4. The projects you selected should now be visible in your MCUXpresso workspace as shown in [Figure 9](#):

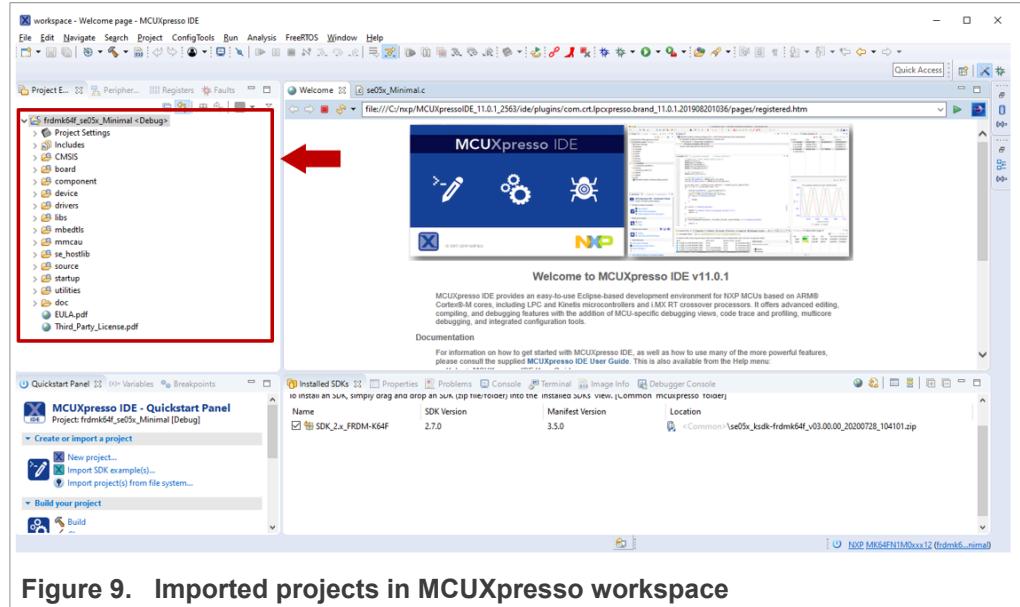


Figure 9. Imported projects in MCUXpresso workspace

## 4.5 Build, run and debug project example

After importing project examples in the MCUXpresso workspace, follow these instructions to build, run and debug a project:

1. Attach a USB cable from the computer to the K64F OpenSDA debug USB connector as shown in [Figure 10](#).

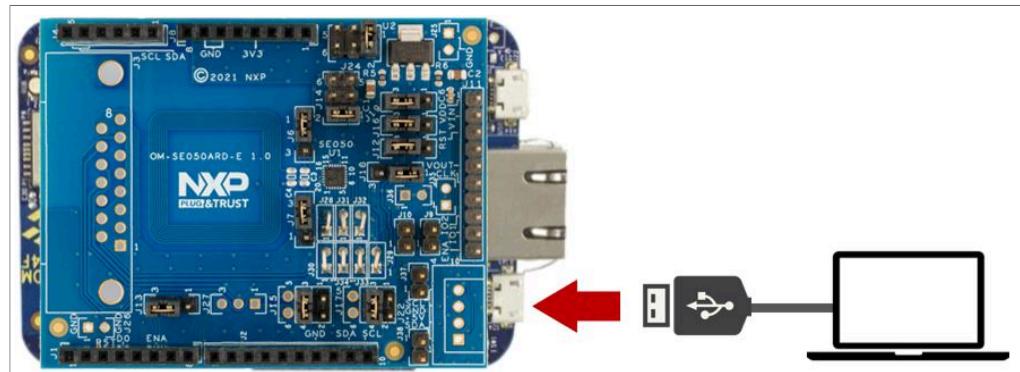


Figure 10. Connect boards to the laptop

2. Launch and setup TeraTerm application as shown in [Figure 11](#):
  - a. Click *Serial* option and select from the drop down list the COM port number assigned to your FRDM-K64F board
  - b. Go to Setup > Serial Port and configure the terminal to 115200 baud rate, 8 data bits, no parity and 1 stop bit and click OK.

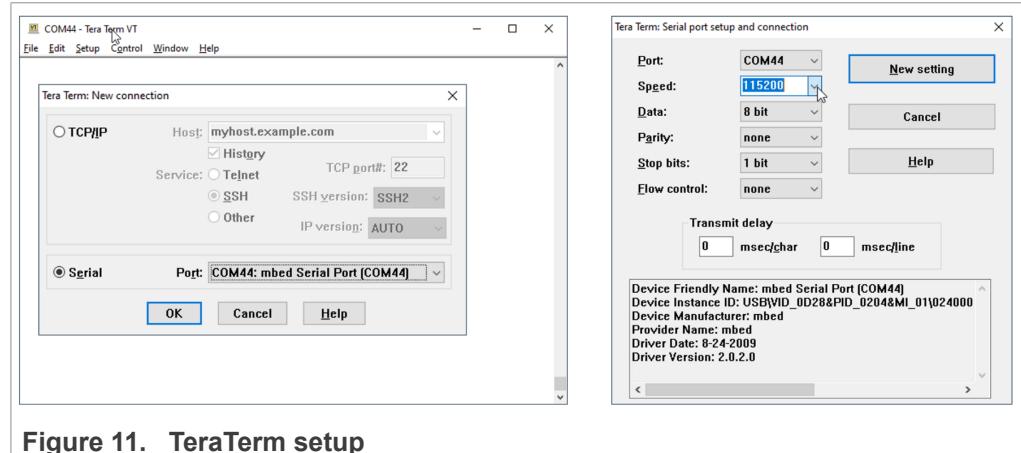


Figure 11. TeraTerm setup

3. **Note:** The default build configuration of the EdgeLock SE05x Plug & Trust middleware  $\geq$  V04.02.0x generates code for the OM-SE050ARD-E development board. You need to adapt settings in the feature header file `fsl_sss_ftr.h` 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 4.6](#).
4. Go to the MCUXpresso Quickstart Panel and click *Build* button as shown in [Figure 12](#). Wait a few seconds and check that the build process has finished successfully in the MCUXpresso console window.

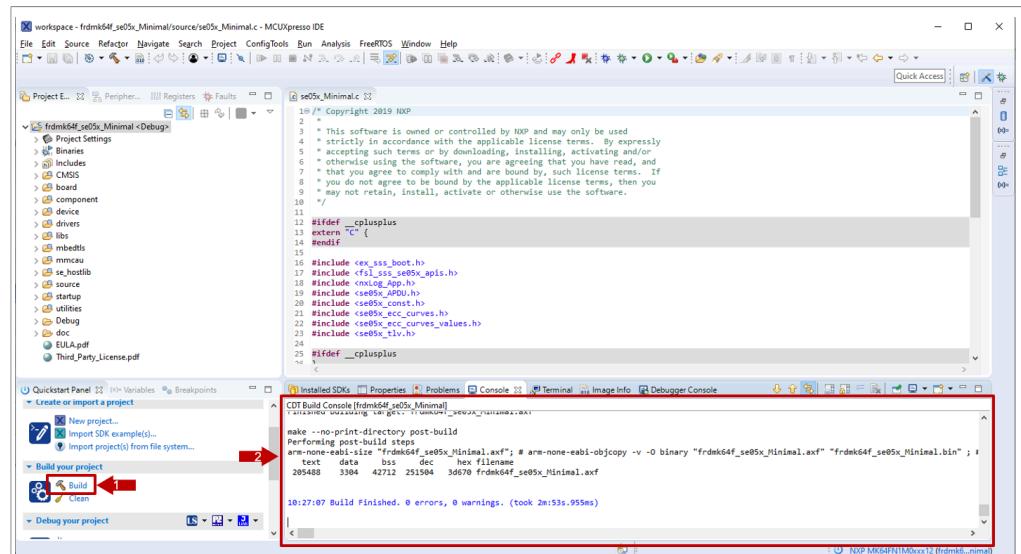
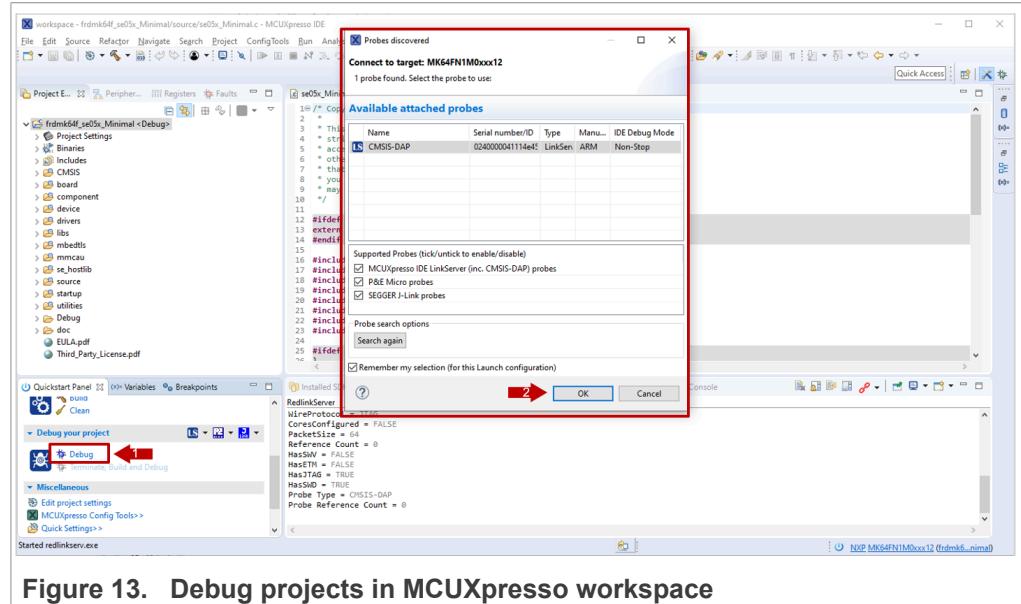


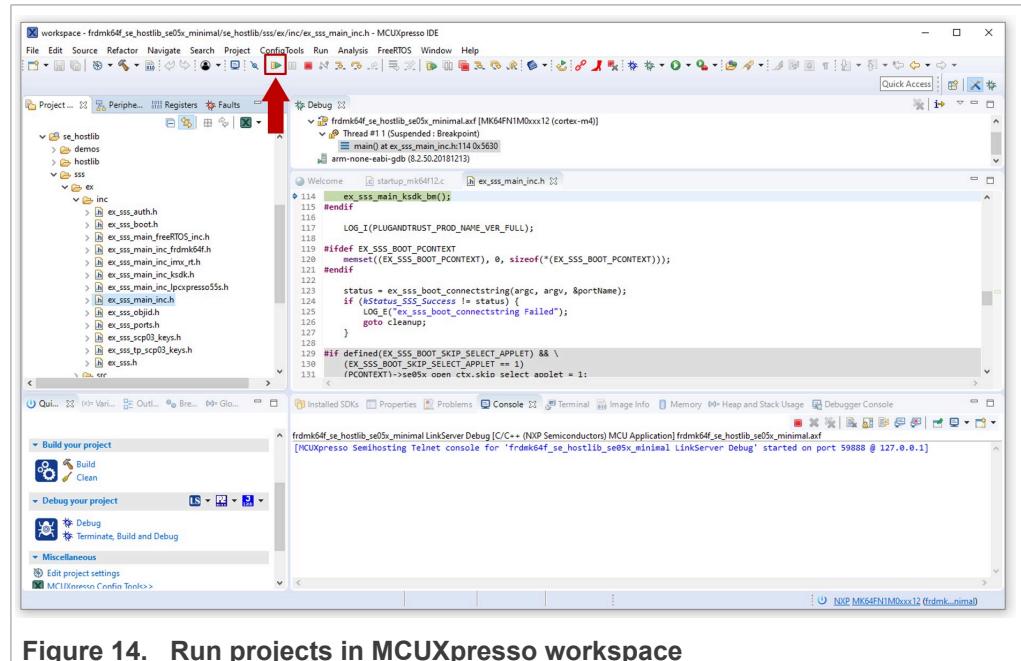
Figure 12. Build projects in MCUXpresso workspace

5. Go to the MCUXpresso Quickstart Panel and click *Debug* button as shown in [Figure 13](#). If there is more than one probe attached, you have to select the CMSIS-DAP debug probe from the list. Wait a few seconds until the project executes



**Figure 13. Debug projects in MCUXpresso workspace**

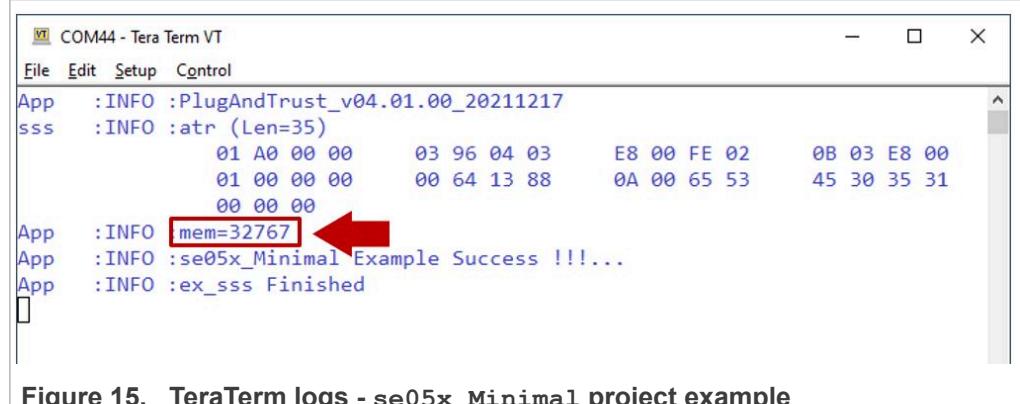
6. When it executes, it will automatically stop in a breakpoint. Click on *Resume* to allow the software to continue its execution as shown in [Figure 14](#).



**Figure 14. Run projects in MCUXpresso workspace**

7. Once the program execution begins, logs are printed on the terminal application indicating the execution status. For the `se05x_Minimal` project example, the logs

should indicate the available memory in the secure element (in this case, 20820) as can be seen in [Figure 15](#):



```
COM44 - Tera Term VT
File Edit Setup Control
App :INFO :PlugAndTrust_v04.01.00_20211217
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
```

**Figure 15. TeraTerm logs - se05x\_Minimal project example**

8. The same operation can be repeated with any of the other EdgeLock SE05x Plug & Trust middleware project examples.

#### 4.6 Product specific build settings

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

The Plug & Trust Middleware uses the feature file `fsl_sss_ftr.h` to select a dedicated EdgeLock product IC and the corresponding IoT applet or Authenticator application. The `fsl_sss_ftr.h` header file is located in the project source folder.

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 `fsl_sss_ftr.h` header file includes several compilation options to select a dedicated product variant like: `PTWM_Applet`, `PTMW_FIPS`, `PTMW_SE05X_Ver`, `PTMW_SE05X_Auth` and `PTMW_SCP`.

Select the desired value of the compilation option by setting exclusively the corresponding C-preprocessor define to 1 (enable). All other values for the same option (represented by C-preprocessor defines) must be set to 0.

**Example:** Assign the value `SE050_E` to the compilation option `PTWM_Applet`.

The screenshot shows the MCUXpresso IDE interface with the project 'FRDM\_K64 - se05x\_Minimal' open. The left sidebar displays the project structure under 'se05x\_Minimal <Debug>'. A file named 'fsl\_sss\_ftr.h' is selected and shown in the main editor window. The code in the editor is as follows:

```
19
20 /**
21 * PTMW_Applet : The Secure Element Applet
22 * You can compile host library for different Applets listed below.
23 * Please note, some of these Applets may be for NXP Internal use only.
24 */
25
26 /**
27 * Compiling without any Applet Support *
28 #define SSS_HAVE_APPLET_NONE 0
29
30 /**
31 * A71CH (ECC) *
32 #define SSS_HAVE_APPLET_A71CH 0
33
34 /**
35 * A71CL (RSA) *
36 #define SSS_HAVE_APPLET_A71CL 0
37
38 /**
39 * Similar to A71CH *
40 #define SSS_HAVE_APPLET_A71CH_SIM 0
41
42 /**
43 * SE050 Type A (ECC) *
44 #define SSS_HAVE_APPLET_SE05X_A 0
45
46 /**
47 * SE050 Type B (RSA) *
48 #define SSS_HAVE_APPLET_SE05X_B 0
49
50 /**
51 * SE050 (Super set of A + B) *
52 #define SSS_HAVE_APPLET_SE05X_C 0
53
54 /**
55 * SE050 (Similar to A71CL) *
56 #define SSS_HAVE_APPLET_SE05X_L 0
57
58 /**
59 * SE051UWB (Similar to SE05x) *
60 #define SSS_HAVE_APPLET_SE051_UWB 0
61
62 /**
63 * SE051 with SPAKE Support *
64 #define SSS_HAVE_APPLET_SE051_H 0
65
66 /**
67 * AUTH *
68 #define SSS_HAVE_APPLET_AUTH 0
69
70 /**
71 * SE050E *
72 #define SSS_HAVE_APPLET_SE050_E 1
73
```

**Figure 16. Feature file `fsl_sss_ftr.h` example: Assign the value `SE05X_E` to the CMake option `PTWM_Applet`**

The following tables show the required PTMW options to build the MCUXpresso SDK for a dedicated product variant. The SSSFTR\_SE05X\_RSA option is used to optimize the memory footprint for product variants that do not support RSA.

**Table 3. Feature file fsl\_sss\_ftr.h 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	SSS_HAVE_APPLET_SE05X_E	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_07_02	any option	SSS_HAVE_SCPCP_NONE	disabled
SE050E2	A921					or SSS_HAVE_SCPCP_SCPO3_SSS	

**Table 4. Feature file fsl\_sss\_ftr.h 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	SSS_HAVE_APPLET_SE05X_C	SSS_HAVE_FIPS_SE050	SSS_HAVE_SE05X_VER_03_xx	SSS_HAVE_SE05X_AUTH_PLATFSCP03 or SSS_HAVE_SE05X_AUTH_USERID_PLATFSCP03 or SSS_HAVE_SE05X_AUTH_AESKEY_PLATFSCP03 or SSS_HAVE_SE05X_AUTH_ECKEY_PLATFSCP03	SSS_HAVE SCP_SCPO3_SSS	enabled
SE050F2	A92A						

**Table 5. Feature file fsl\_sss\_ftr.h 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	SSS_HAVE_APPLET_SE05X_A	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_03_xx	any option	SSS_HAVE SCP_NONE or SSS_HAVE SCP_SCPO3_SSS	disabled
SE050A2	A205						
SE050B1	A202	SSS_HAVE_APPLET_SE05X_B	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_03_xx	any option	SSS_HAVE SCP_NONE or SSS_HAVE SCP_SCPO3_SSS	enabled
SE050B2	A203						
SE050C1	A200	SSS_HAVE_APPLET_SE05X_C	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_03_xx	any option	SSS_HAVE SCP_NONE or SSS_HAVE SCP_SCPO3_SSS	enabled
SE050C2	A201						
SE050 Dev Board OM-SE050ARD	A1F4						

**Table 5. Feature file fsl\_sss\_ftr.h settings for SE050 Previous Generation product variants...continued**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
SE050F2	A77E <sup>[1]</sup>	SSS_HAVE_APPLET_SE05X_C	SSS_HAVE_FIPS_SE050	SSS_HAVE_SE05X_VER_03_XX	SSS_HAVE_SE05X_AUTH_PLATFSCP03 or SSS_HAVE_SE05X_AUTH_USERID_PLATFSCP03 or SSS_HAVE_SE05X_AUTH_AESKEY_PLATFSCP03 or SSS_HAVE_SE05X_AUTH_ECKEY_PLATFSCP03	SSS_HAVE_SC_P03_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 6. Feature file fsl\_sss\_ftr.h settings for SE051 product variants**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
SE051A2	A920	SSS_HAVE_APPLET_SE05X_A	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_07_02	any option	SSS_HAVE_SC_P03_SSS	disabled
SE051C2	A8FA	SSS_HAVE_APPLET_SE05X_C	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_07_02	any option	SSS_HAVE_SC_P03_SSS	enabled
SE051W2	A739	SSS_HAVE_APPLET_SE05X_C	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_07_02	any option	SSS_HAVE_SC_P03_SSS	enabled

**Table 6. Feature file fsl\_sss\_ftr.h settings for SE051 product variants...continued**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
SE051A2	A565	SSS_HAVE_APPLET_SE05X_A	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_06_00	any option	SSS_HAVE_SC_P_NONE or SSS_HAVE_SC_P03_SSS	disabled
SE051C2	A564	SSS_HAVE_APPLET_SE05X_C	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_06_00 VER_06_00	any option	SSS_HAVE_SC_P_NONE or SSS_HAVE_SC_P03_SSS	enabled

**Table 7. Feature file fsl\_sss\_ftr.h settings for A5000 product variants**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
OM-A5000ARD	A736	SSS_HAVE_APPLET	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_07_02	any option	SSS_HAVE_SC_P_NONE or SSS_HAVE_SC_P03_SSS	disabled
A5000	A736	SSS_HAVE_APPLET_AUTH	SSS_HAVE_FIPS_NONE	SSS_HAVE_SE05X_VER_07_02	any option	SSS_HAVE_SC_P_NONE or SSS_HAVE_SC_P03_SSS	

#### 4.6.1 Example: SE050E build settings

The following images show the configuration for the SE050E development board OM-SE05ARD-E according to [Table 3](#).

1. Select the Applet variant SE050E.

```

19
20 /* PTMW_Applet : The Secure Element Applet
21 *
22 * You can compile host library for different Applets listed below.
23 * Please note, some of these Applets may be for NXP Internal use only.
24 */
25
26 /** Compiling without any Applet Support */
27 #define SSS_HAVE_APPLET_NONE 0
28
29 /** A71CH (ECC) */
30 #define SSS_HAVE_APPLET_A71CH 0
31
32 /** A71CL (RSA) */
33 #define SSS_HAVE_APPLET_A71CL 0
34
35 /** Similar to A71CH */
36 #define SSS_HAVE_APPLET_A71CH_SIM 0
37
38 /** SE050 Type A (ECC) */
39 #define SSS_HAVE_APPLET_SE05X_A 0
40
41 /** SE050 Type B (RSA) */
42 #define SSS_HAVE_APPLET_SE05X_B 0
43
44 /** SE050 (Super set of A + B) */
45 #define SSS_HAVE_APPLET_SE05X_C 0
46
47 /** SE050 (Similar to A71CL) */
48 #define SSS_HAVE_APPLET_SE05X_L 0
49
50 /** SE051UWB (Similar to SE05x) */
51 #define SSS_HAVE_APPLET_SE051_UWB 0
52
53 /** SE051 with SPEAKE Support */
54 #define SSS_HAVE_APPLET_SE051_H 0
55
56 /** AUTH */
57 #define SSS_HAVE_APPLET_AUTH 0
58
59 /** SE050E */
60 #define SSS_HAVE_APPLET_SE050_E 1
61

```

Figure 17. Feature file fsl\_sss\_ftr.h - PTMW\_Applet

## 2. Select FIPS none.

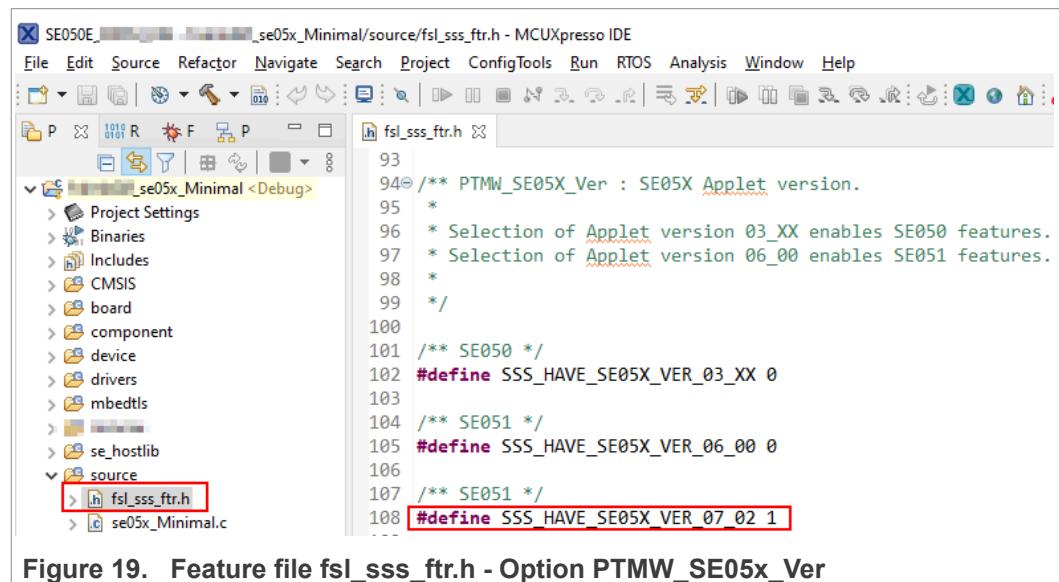
```

252 /** PTMW_FIPS : Enable or disable FIPS
253 *
254 * This selection mostly impacts tests, and generally not the actual Middleware
255 */
256
257 /** NO FIPS */
258 #define SSS_HAVE_FIPS_NONE 1
259
260 /** SE050 IC FIPS */
261 #define SSS_HAVE_FIPS_SE050 0
262
263 /** FIPS 140-2 */
264 #define SSS_HAVE_FIPS_140_2 0
265
266 /** FIPS 140-3 */
267 #define SSS_HAVE_FIPS_140_3 0

```

Figure 18. Feature file fsl\_sss\_ftr.h - Option PTMW\_FIPS

### 3. Select Applet version 7.02.



```
SE050E_..._se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
P 1010 R F P
fsl_sss_ftr.h
93
94  /** PTMW_SE05X_Ver : SE05X Applet version.
95  *
96  * Selection of Applet version 03_XX enables SE050 features.
97  * Selection of Applet version 06_00 enables SE051 features.
98  */
99 */
100 /**
101 #define SSS_HAVE_SE05X_VER_03_XX 0
102 /**
103 #define SSS_HAVE_SE05X_VER_06_00 0
104 /**
105 #define SSS_HAVE_SE05X_VER_07_02 1
106 */
107 /**
108 #define SSS_HAVE_SE05X_VER_07_02 1
```

Figure 19. Feature file fsl\_sss\_ftr.h - Option PTMW\_SE05x\_Ver

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

- Set the `#define SSS_HAVE_SE05X_AUTH_NONE` option to 1 and disable all other options by setting the flags to 0.
- Set the `#define SSS_HAVE_SCP_NONE` option to 1 and disable all other options by setting the flags to 0.

How to enable Platform SCP is described in [Section 6.3](#).

The screenshot shows the MCUXpresso IDE interface with the project "SE050E\_minimal\_se05x\_Minimal" open. The left sidebar displays the project structure, including source files like fsl\_sss\_ftr.h, se05x\_Minimal.c, and semihost\_hardfault.c. The main editor window shows the content of fsl\_sss\_ftr.h. A red box highlights the line "#define SSS\_HAVE\_SE05X\_AUTH\_NONE 1", which is part of a series of defines for different authentication methods. The code is annotated with comments explaining the various options and their usage.

```
318 //** PTMW_SE05X_Auth : SE050 Authentication
319 *
320 * This settings is used by examples to connect using various options
321 * to authenticate with the Applet.
322 * The SE05X_Auth options can be changed for KSDK Demos and Examples.
323 * To change SE05X_Auth option follow below steps.
324 * Set flag ``SSS_HAVE_SCP_SCP03_SSS`` to 1 and Reset flag ``SSS_HAVE_SCP_NONE`` to 0.
325 * To change SE05X_Auth option other than ``None`` and ``PlatfSCP03``,
326 * execute se05x_Delete_and_test_provision.exe in order to provision the Authentication Key.
327 * To change SE05X_Auth option to ``ECKey`` or ``ECKey_PlatfSCP03``,
328 * Set additional flag ``SSS_HAVE_HOSTCRYPTO_ANY`` to 1.
329 */
330
331 /** Use the default session (i.e. session less) login */
332 #define SSS_HAVE_SE05X_AUTH_NONE 1
333
334 /** Do User Authentication with UserID */
335 #define SSS_HAVE_SE05X_AUTH_USERID 0
336
337 /** Use Platform SCP for connection to SE */
338 #define SSS_HAVE_SE05X_AUTH_PLATFSCP03 0
339
340 /** Do User Authentication with AES Key
341 * Earlier this was called AppletSCP03 */
342 #define SSS_HAVE_SE05X_AUTH_AESKEY 0
343
344 /** Do User Authentication with EC Key
345 * Earlier this was called FastSCP */
346 #define SSS_HAVE_SE05X_AUTH_ECKEY 0
347
348 /** UserID and PlatfSCP03 */
349 #define SSS_HAVE_SE05X_AUTH_USERID_PLATFSCP03 0
350
351 /** AESKey and PlatfSCP03 */
352 #define SSS_HAVE_SE05X_AUTH_AESKEY_PLATFSCP03 0
353
354 /** ECKEY and PlatfSCP03 */
355 #define SSS_HAVE_SE05X_AUTH_ECKEY_PLATFSCP03 0
```

Figure 20. Feature file fsl\_sss\_ftr.h - Option PTMW\_AUTH - Plain communication

```
SE050E [ ] - se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
P S B R F P
fsf_sss_ftr.h
218
219 /** PTMW SCP : Secure Channel Protocol
220 *
221 * In case we enable secure channel to Secure Element, which interface to be used.
222 */
223
224 /**
225 #define SSS_HAVE_SCP_NONE 1
226
227 /** Use SSS Layer for SCP. Used for SE050 family. */
228 #define SSS_HAVE_SCP_SCP03_SSS 0
229
230 /** Use Host Crypto Layer for SCP03. Legacy implementation. Used for older demos of A71CH Family. */
231 #define SSS_HAVE_SCP_SCP03_HOSTCRYPTO 0
232
233 #if (( 0
234     + SSS_HAVE_SCP_NONE
235     \
```

**Figure 21. Feature file `fsl_sss_ftr.h` - Option PTMW SCP - Plain communication**

5. To reduce the EdgeLock SE05x Plug & Trust middleware memory footprint we disable RSA for the SE050E product variant.

```

412 /**
413 * =====
414 * == Feature selection/values ==
415 * =====
416
417
418 /** SE05X Secure Element : Symmetric AES */
419 #define SSSFTR_SE05X_AES 1
420
421 /** SE05X Secure Element : Elliptic Curve Cryptography */
422 #define SSSFTR_SE05X_ECC 1
423
424 /** SE05X Secure Element : RSA */
425 #define SSSFTR_SE05X_RSA 0
426
427 /** SE05X Secure Element : KEY operations : SET Key */
428 #define SSSFTR_SE05X_KEY_SET 1
429
430 /** SE05X Secure Element : KEY operations : GET Key */
431 #define SSSFTR_SE05X_KEY_GET 1
432
433 /** SE05X Secure Element : Authenticate via ECKey */
434 #define SSSFTR_SE05X_AuthECKey 1
435
436 /** SE05X Secure Element : Allow creation of user/authenticated session.
437 *
438 * If the intended deployment only uses Platform SCP
439 * Or it is a pure session less integration, this can
440 * save some code size. */
441 #define SSSFTR_SE05X_AuthSession 1
442

```

Figure 22. Feature file fsl\_sss\_ftr.h - Option SSSFTR\_SE05X\_RSA

## 5 Import project examples from CMake-based build system

This section explains how to run example projects using the CMake-based build system. Although this offers the possibility to quickly build the same example code for multiple platforms, the debug experience may be affected by MCUXpresso not being able to make use of the defines chosen in CMAKE.

### 5.1 Prerequisites

The following tools are required to run projects generated from the CMake-based build system:

1. MCUXpresso IDE. Check [Section 7](#) for detailed installation instructions.
2. CMake. Check [Section 8](#) for detailed installation instructions.
3. Python ≥ 3.7.x and ≤ 3.9.x 32-bit version. Check [Section 9](#) for detailed installation instructions.
4. TeraTerm (or an equivalent serial application). You can download and run TeraTerm installer from this [link](#).

### 5.2 Download EdgeLock SE05x Plug & Trust middleware

Follow these steps to download the EdgeLock SE05x Plug & Trust middleware in your local machine:

1. Download EdgeLock SE05x Plug & Trust middleware from the [NXP website](#)

2. Create a folder called **se05x\_middleware** in C: directory as shown in [Figure 23](#):

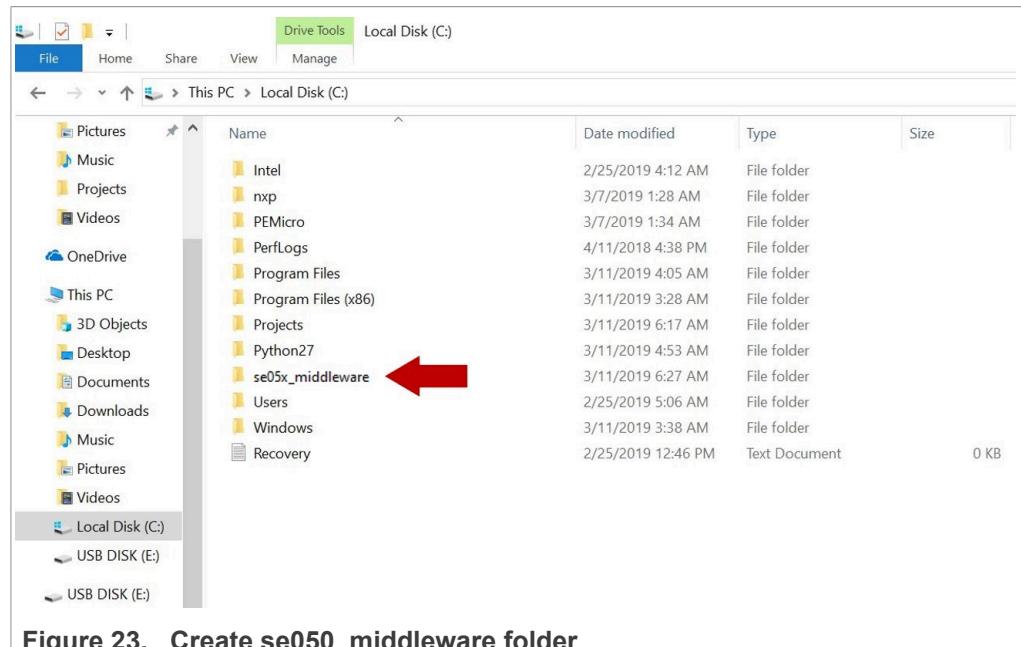


Figure 23. Create se050\_middleware folder

3. Unzip the EdgeLock SE05x Plug & Trust middleware inside the **se05x\_middleware** folder. After unzipping, you will see a folder called **simw-top** created. The contents of the **simw-top** directory should look as they appear in [Figure 24](#):

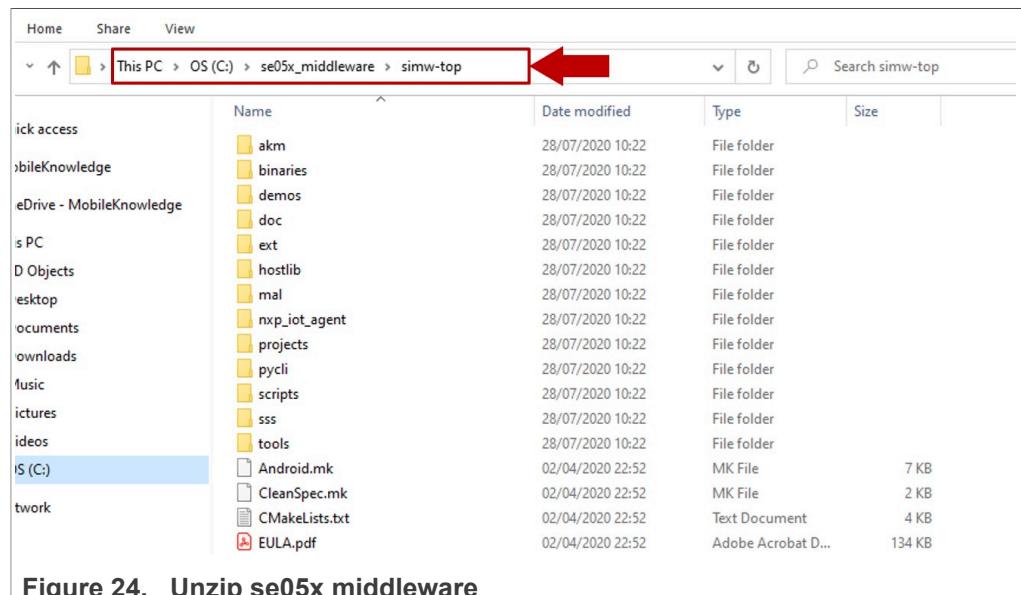


Figure 24. Unzip se05x middleware

**Note:** It is recommended to keep **se05x\_middleware** with the **shortest** path possible and **without spaces** in it. This avoids some issues that could appear when building the middleware if the path contains spaces.

### 5.3 Build EdgeLock SE05x Plug & Trust middleware project examples

The EdgeLock SE05x Plug & Trust middleware uses CMake for building the project examples into your local machine. To build EdgeLock SE05x Plug & Trust middleware, open a Command Prompt and use the following steps as shown in [Figure 25](#):

1. Go to the folder where you unzipped the SE05x middleware:  
(1) Send >> cd C:\se05x\_middleware\simw-top\scripts
2. Define the environment:  
(2) Send >> env\_setup.bat
3. Generate the EdgeLock SE05x Plug & Trust middleware project examples:  
(3) Send >> create\_cmake\_projects.py

**Note:** This command may take a few seconds to complete.

```

C:\se05x_middleware\simw-top>cd C:\se05x_middleware\simw-top\scripts
C:\se05x_middleware\simw-top\scripts>env_setup.bat
***** Visual Studio 2017 Developer Command Prompt v15.9.7 *****
** Copyright (c) 2017 Microsoft Corporation
***** [vcvarsall.bat] Environment initialized for: 'x86'
C:\se05x_middleware\simw-top\scripts>create_cmake_projects.py
Could not find "'C:\Program Files\CMakel\bin'\cmake.exe'. Assuming 'cmake.exe' is in path and running.
## Connect to Secure Element from PC
#cmake -DApplet=SE050_C -DHost=PCWindows -DHostCrypto=MBEDTLS -DCMAKE_BUILD_TYPE=Debug -A Win32
-- Selecting Windows SDK version 10.0.17763.0 to target Windows 10.0.18362.
-- BUILD_TYPE: Debug
-- CMAKE_CXX_COMPILER_ID = MSVC
-- CMAKE_SYSTEM_NAME = Windows
-- SE05X Auth - None
-- Configuring done
-- Generating done
:
C:\se05x_middleware\simw-top\scripts>

```

**Figure 25.** Generate EdgeLock SE05x Plug & Trust middleware project examples

Depending on your PC installation you may need to update the application file locations within the env\_setup.bat file.

4. Your project directory should now contain two folders: a (1) simw-top folder and a (2) simw-top\_build folder as shown in [Figure 26](#):



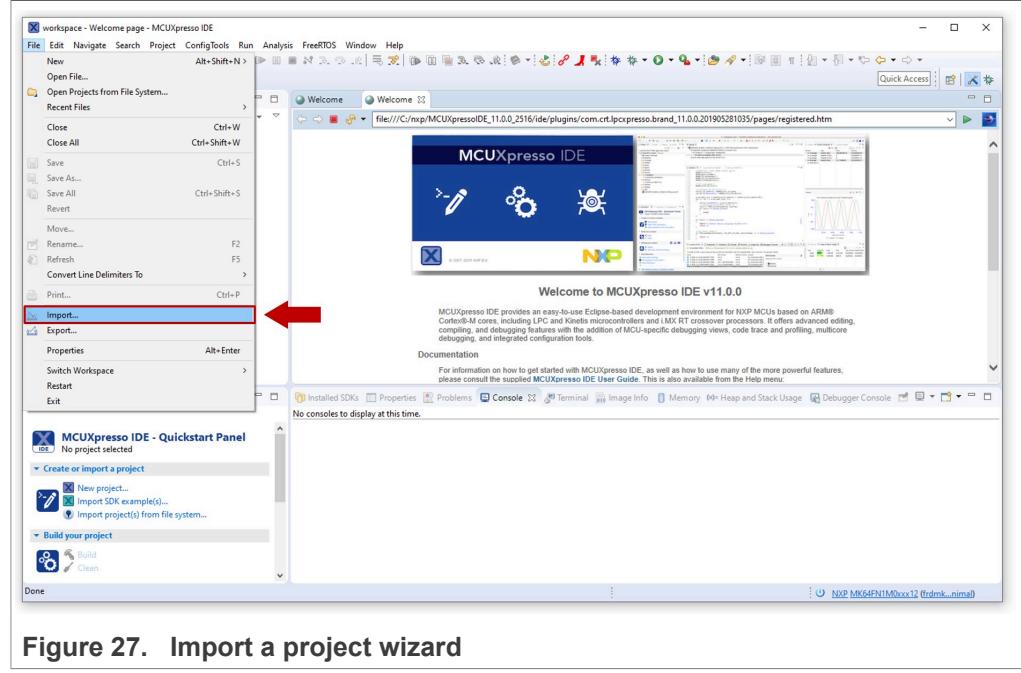
**Figure 26.** SE05x middleware project structure

### 5.4 Import *PlugAndTrustMW* project example in MCUXpresso workspace

After generating the projects in your local machine using the create\_cmake\_projects.py script, we need to import the *PlugAndTrustMW* project example in our MCUXpresso workspace. Follow these steps to import a project:

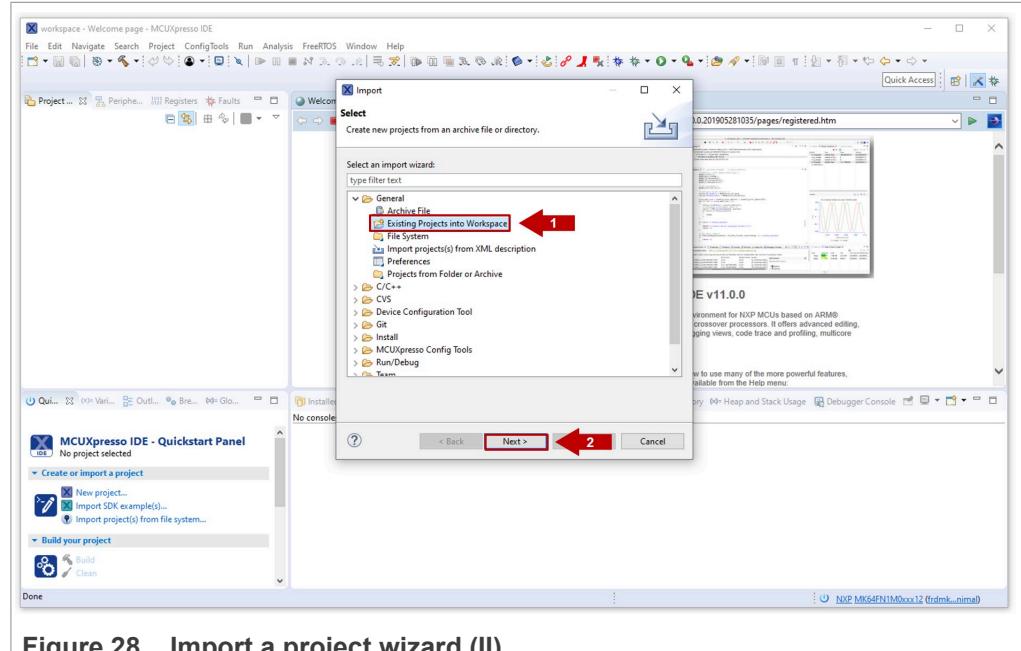
1. Go to *File* → *Import* using the top bar menu as shown in [Figure 27](#).

**Note:** In this case, do not use the MCUXpresso Quickstart Panel to import project.



**Figure 27. Import a project wizard**

2. In the import wizard menu, select import "**Existing Projects into Workspace**" from the **General** folder as shown in [Figure 28](#):

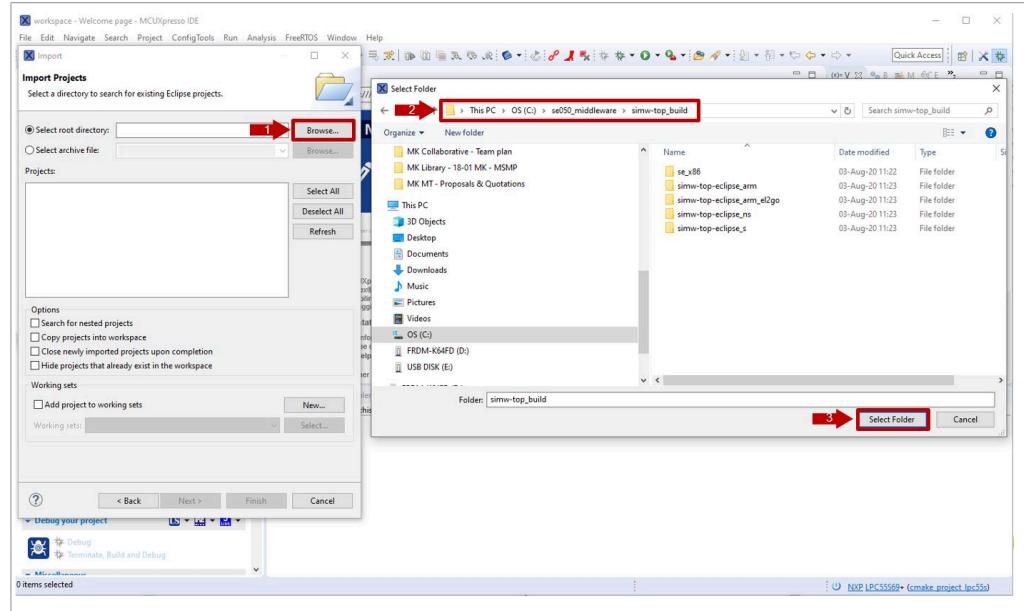


**Figure 28. Import a project wizard (II)**

3. First, we need to import EdgeLock SE05x Plug & Trust middleware project in MCUXpresso. For that, in the *Select root directory* option, browse to C:

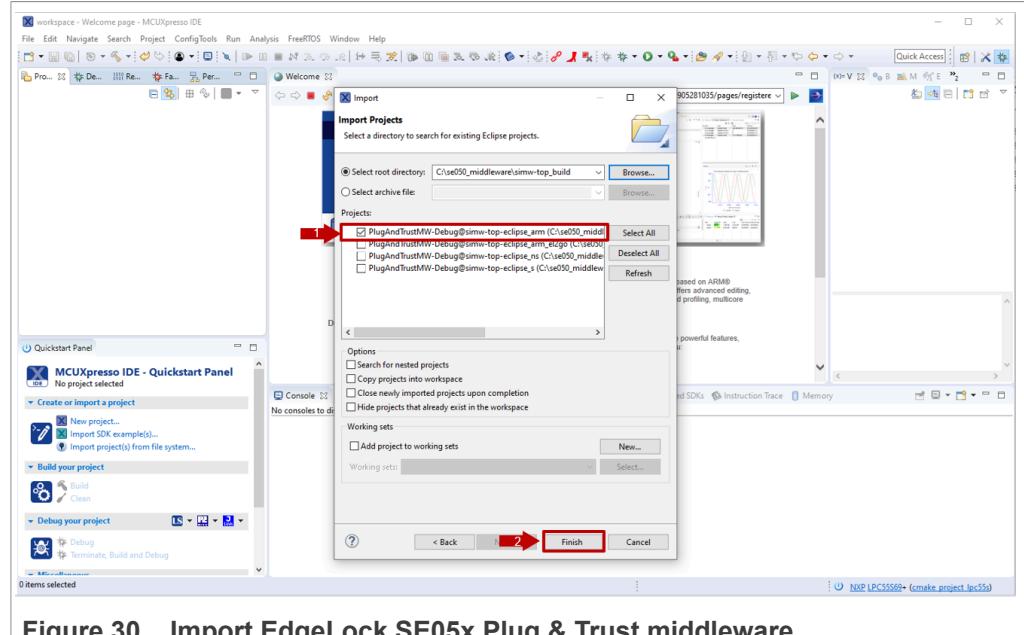
## EdgeLock SE05x Quick start guide with FRDM-K64F

`\se05x_middleware\simw-top_build` or browse the location of your EdgeLock SE05x Plug & Trust middleware directory and click *Select folder* as shown in [Figure 29](#):



**Figure 29. Select EdgeLock SE05x Plug & Trust middleware build folder**

- After selecting `C:\se05x_middleware\simw-top_build` folder, a project called `PlugAndTrustMW-Debug@simw-top-eclipse_arm` should be visible in the "projects" area. Select it and then click on the *Finish* button to import this project into your workspace as shown in [Figure 30](#):



**Figure 30. Import EdgeLock SE05x Plug & Trust middleware**

5. The *PlugAndTrustMW* project should now be imported in your workspace as shown in [Figure 31](#):

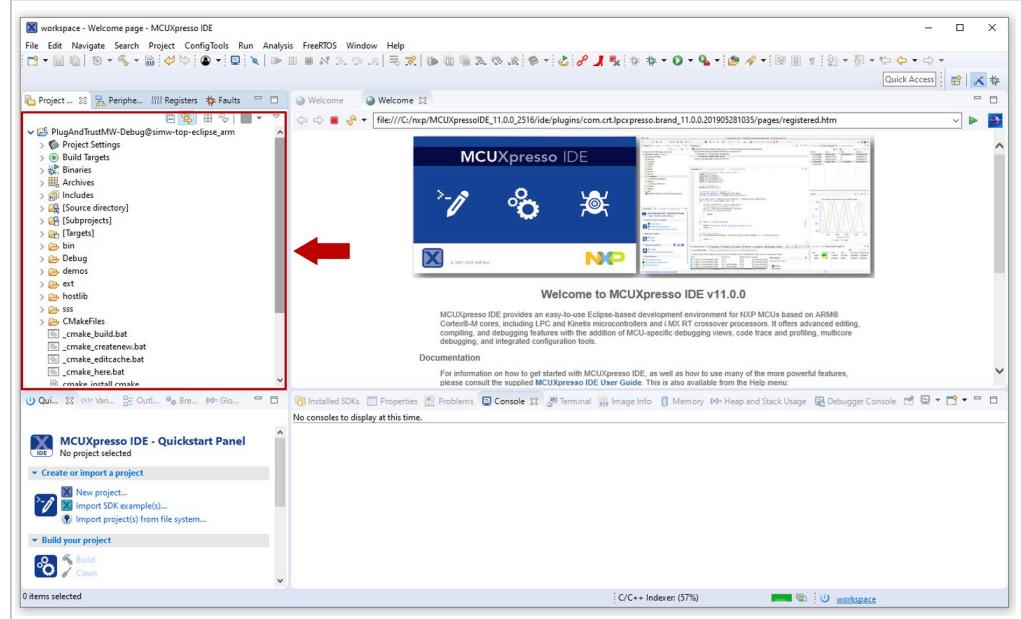


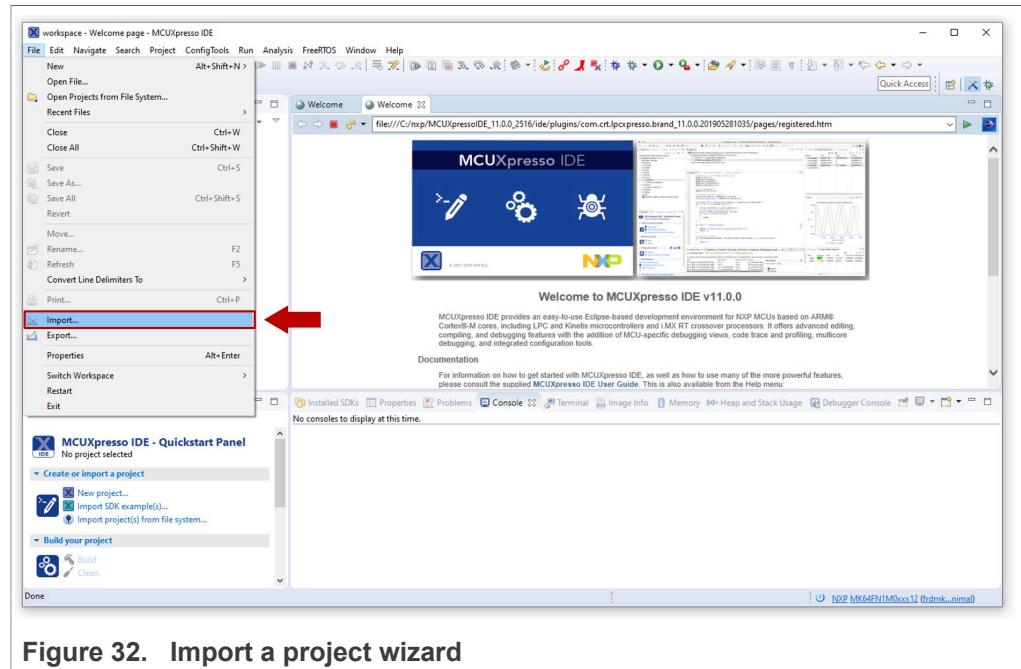
Figure 31. EdgeLock SE05x Plug & Trust middleware imported in workspace

## 5.5 Import *cmake\_projects\_frdm64f* project example in MCUXpresso workspace

After importing the *PlugAndTrustMW* project example in MCUXpresso, we need to import the *cmake\_projects\_frdm64f* project example. Follow these steps:

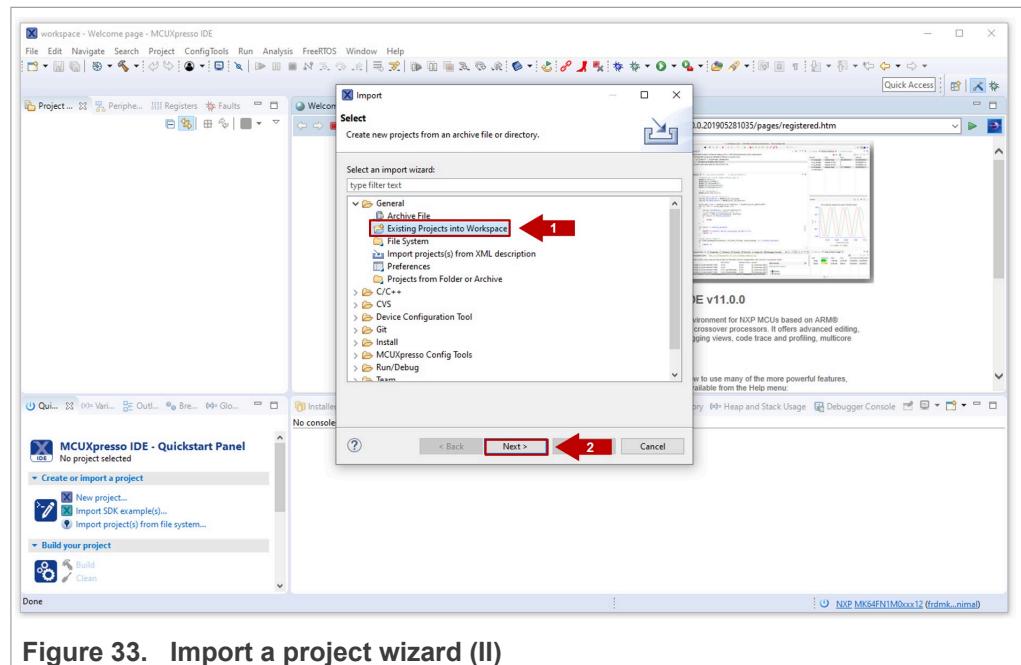
1. Go to *File* → *Import* using the top bar menu as shown in [Figure 27](#).

**Note:** In this case, do not use the MCUXpresso Quickstart Panel to import project.



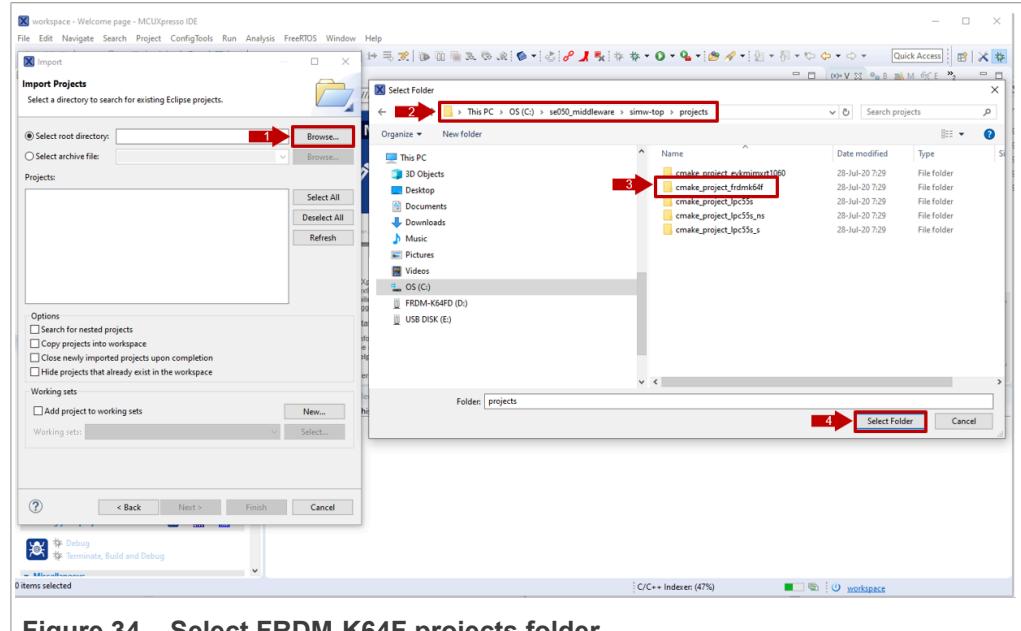
**Figure 32. Import a project wizard**

2. In the import wizard menu, select import "**Existing Projects into Workspace**" from the **General** folder as shown in [Figure 33](#):



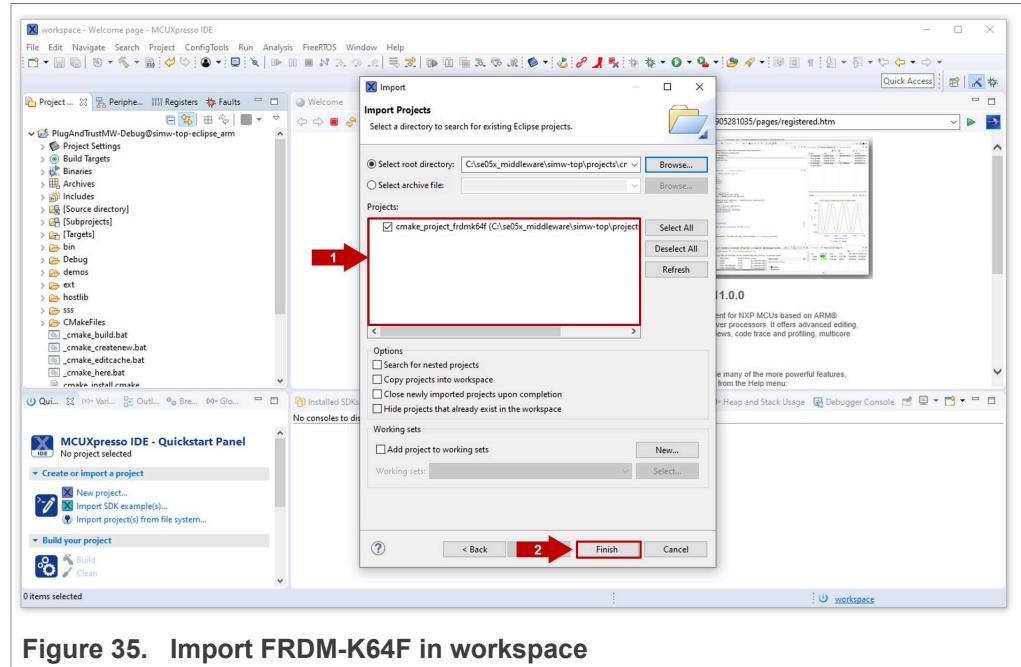
**Figure 33. Import a project wizard (II)**

3. In the *Select root directory* option, browse to `C:\se05x_middleware\simw-top\projects` or browse the location of your FRDM-K64F projects directory. Choose the `cmake_projects_frdm64f` project and click *Select folder* as shown in [Figure 34](#):



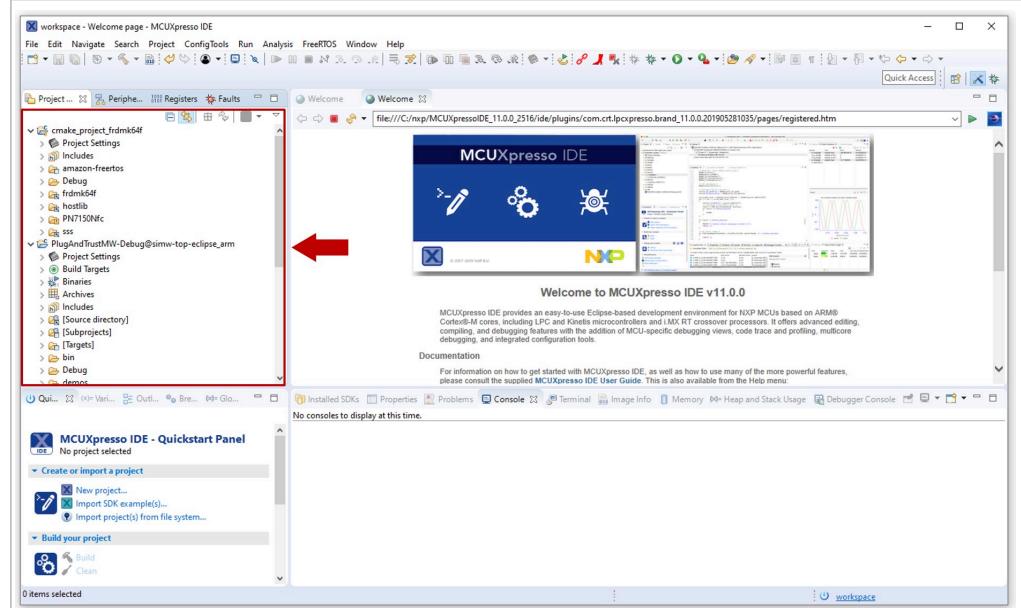
**Figure 34.** Select FRDM-K64F projects folder

4. After selecting `C:\se05x_middleware\simw-top\projects` folder, the `cmake_projects_frdm64f` project should be visible in the Projects area. Click *Finish* button to import this project into your workspace as shown in [Figure 35](#):



**Figure 35.** Import FRDM-K64F in workspace

5. Both The *PlugAndTrustMW* and *cmake\_projects\_frdm64f* projects should now be imported in your workspace as shown in [Figure 36](#):



**Figure 36. FRDM-K64F imported in workspace**

The two projects need to be imported in the same MCUXpresso workspace. The *cmake\_project\_frdm64f* project is used to compile the binary file and debug the solution while the *PlugAndTrustMW-Debug@simw-top-eclipse\_arm* project contains the source files.

**Note:** In order to be able to set breakpoints within the source code upfront, you need to navigate through the *PlugAndTrustMW-Debug@simw-top-eclipse\_arm* project files to set the breakpoints. For instance, navigating to *PlugAndTrustMW-Debug@simw-top-eclipse\_arm/[Source directory]/demos/se05x/se05x\_Minimal* directory, we can add the desired breakpoints in the project execution of the *se05x\_Minimal.c* project example.

6. Continue to [Section 5.6](#) for instructions about how to execute the project examples.

## 5.6 Run EdgeLock SE05x Plug & Trust middleware examples

This section explains how to list, edit and execute project examples using the CMake build system. It includes the following sections:

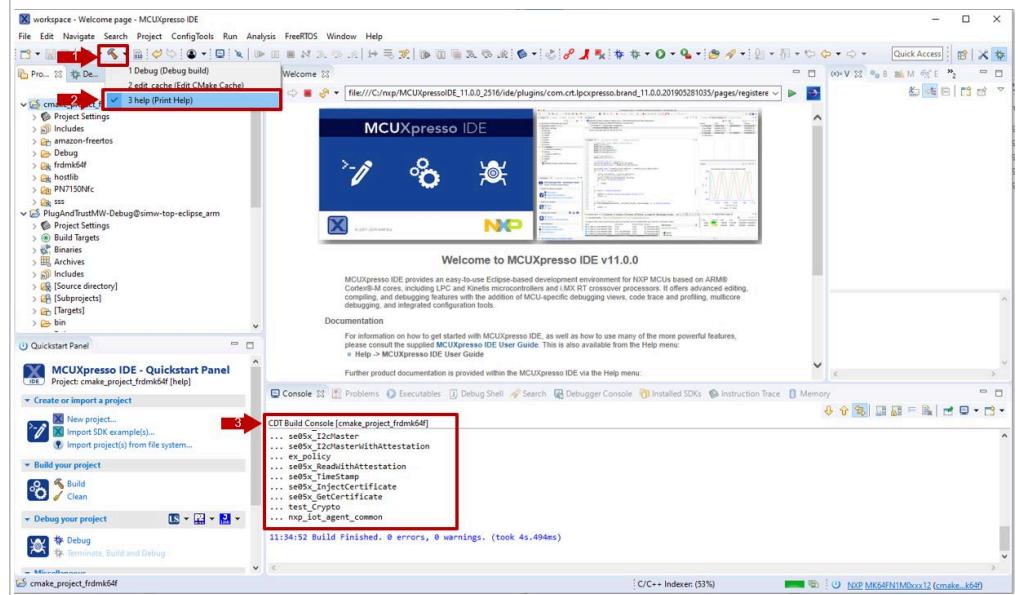
- [List the EdgeLock SE05x Plug & Trust middleware examples](#).
- [Edit EdgeLock SE05x Plug & Trust middleware example CMake options](#).
- [Execute one EdgeLock SE05x Plug & Trust middleware example](#).

### 5.6.1 List the EdgeLock SE05x Plug & Trust middleware examples

The EdgeLock SE05x Plug & Trust middleware comes with several examples used to verify atomic EdgeLock SE05x security IC features. To get the list of examples, follow these steps:

1. Select the *cmake\_project\_frdm64f* project example and click on the arrow on the "hammer" icon in the top bar menu of the MCUXpresso.

2. Select **3 help (Print help)** option. Wait a few seconds until the operation is completed.
3. The MCUXpresso console will display the list of EdgeLock SE05x Plug & Trust middleware examples which can be compiled with the currently chosen CMake settings (see [Figure 37](#)).



**Figure 37. EdgeLock SE05x Plug & Trust middleware examples**

### 5.6.2 Edit EdgeLock SE05x Plug & Trust middleware example CMake options.

The EdgeLock SE05x Plug & Trust middleware is delivered with the CMake files that include the set of directives and instructions describing the project's source files and targets. In addition, it includes the CMake configuration files used to enable or disable several features, portability and setting flags to generate the build files for your platform and native build environment.

**Note:** The default build configuration of the EdgeLock SE05x Plug & Trust middleware  $\geq V04.02.0x$  generates code for the OM-SE050ARD-E 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 5.7](#).

To edit the CMake options, follow these steps:

1. Click on the arrow on the "hammer" icon in the top bar menu of the MCUXpresso.
2. Select **2 edit\_cache (Edit CMake Cache)**.

- The CMake GUI window will open in your screen as shown in [Figure 38](#). Using this GUI, you can change the CMake options (if needed).

**Note:** In case you want to change any of the default pre-selected CMake options, you need to click on *Configure* and *Generate* buttons before closing the CMake window.

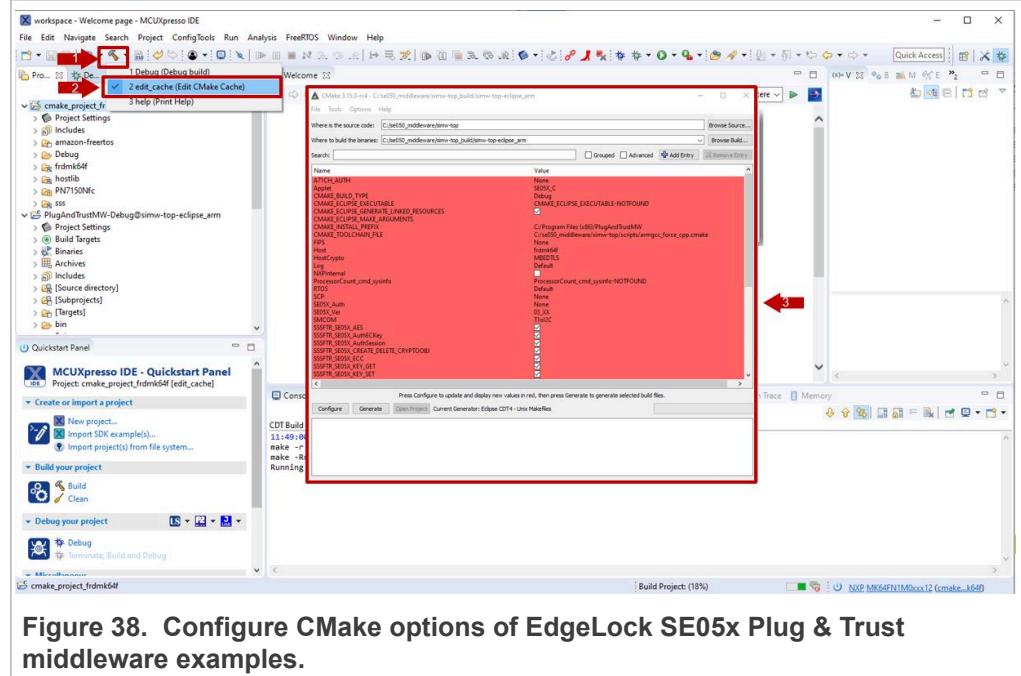


Figure 38. Configure CMake options of EdgeLock SE05x Plug & Trust middleware examples.

### 5.6.3 Build and run a EdgeLock SE05x Plug & Trust middleware project example

This section explains how to build and run the EdgeLock SE05x Plug & Trust middleware example called `se05x_Minimal`. The `se05x_Minimal` project outputs the memory left in EdgeLock SE05x security IC.

**Note:** The execution of the `se05x_Minimal` project is shown as an example. The steps detailed in this section can be replicated to run any other example included as part of the EdgeLock SE05x Plug & Trust middleware.

To execute the `se05x_Minimal` project example, follow these steps:

1. Attach a USB cable from the computer to the K64F OpenSDA debug USB connector as shown in [Figure 39](#).

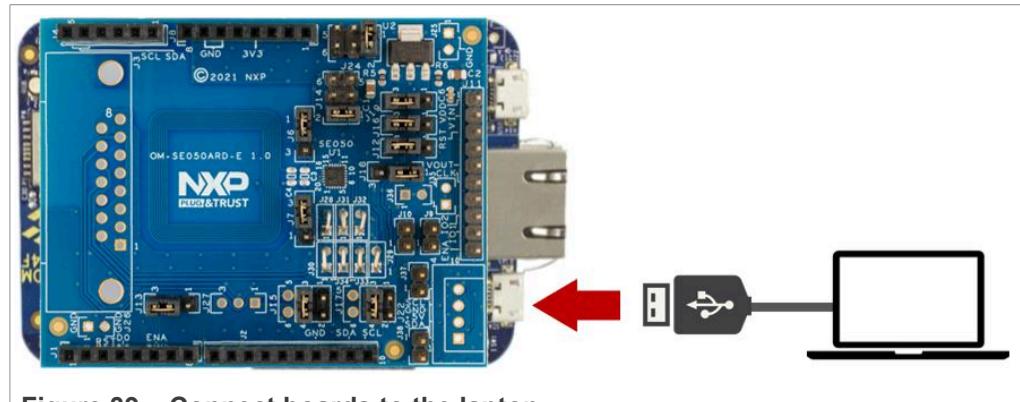


Figure 39. Connect boards to the laptop

2. Open TeraTerm. Click **Serial** option and select from the drop down list the COM port number assigned to your FRDM-K64F. Then go to Setup > Serial Port and configure the terminal to 115200 baud rate, 8 data bits, no parity and 1 stop bit and click OK as shown in [Figure 40](#):

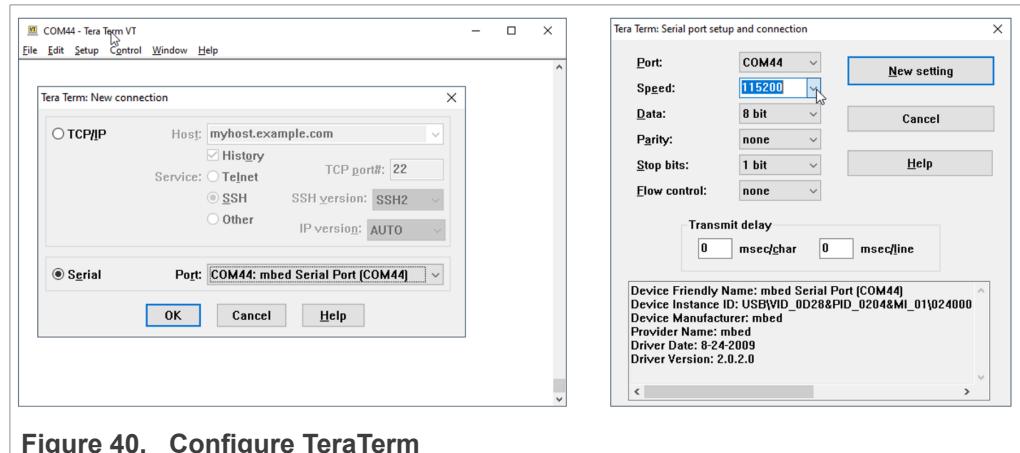
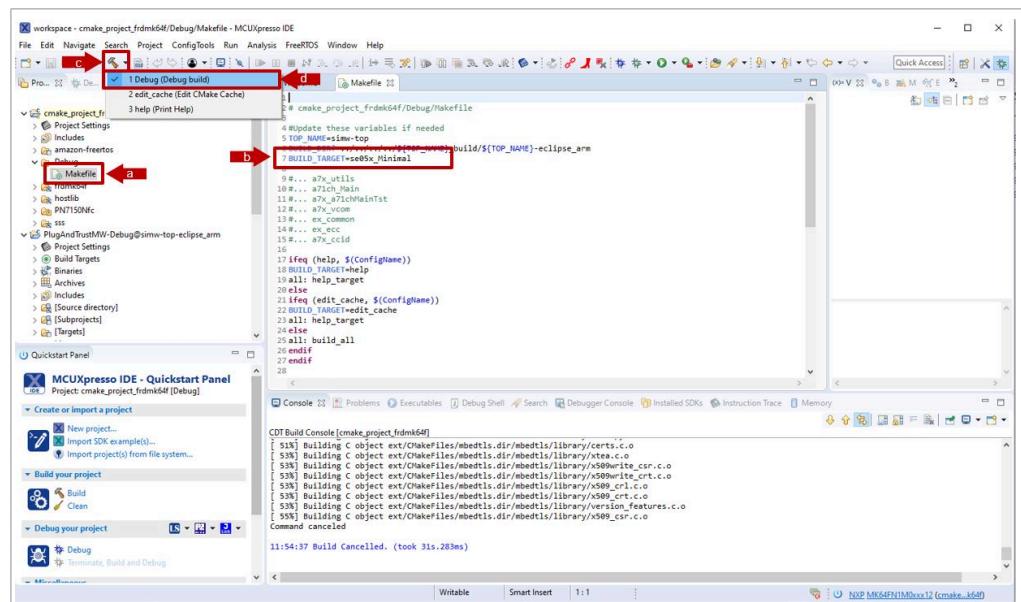


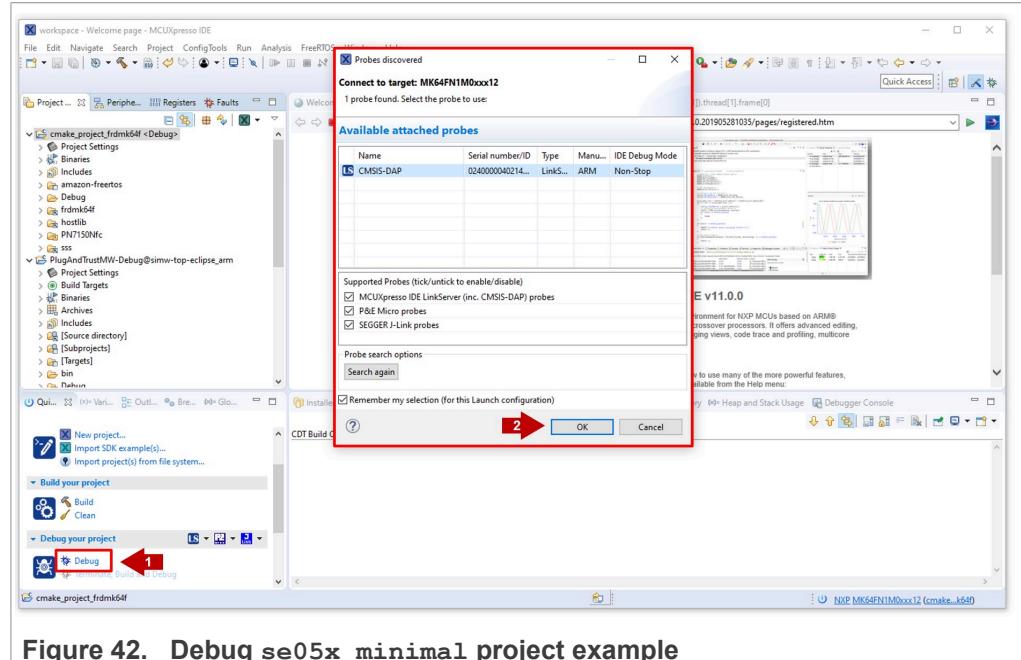
Figure 40. Configure TeraTerm

3. Select the `se05x_Minimal` as the project to be executed. For that, follow the steps shown in [Figure 41](#):
    - a. In the Project Explorer window, go to **Debug** folder and open the **Makefile** file (under `cmake_project_frdmk64f`).
    - b. The **BUILD\_TARGET** contains the name of the project to be executed. Write `se05x_Minimal` in the **BUILD\_TARGET** variable
    - c. Click on the arrow on the "hammer" icon in the top bar menu of the MCUXpresso.
    - d. Select **1 Debug (Debug build)**. Wait a few seconds until the build operation completes.



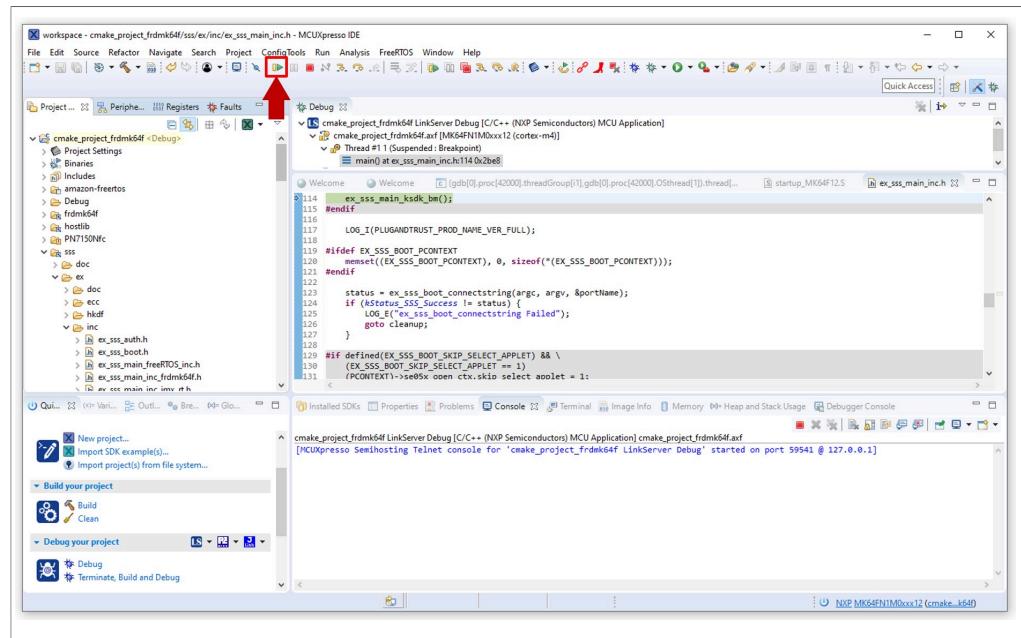
**Figure 41. Debug EdgeLock SE05x Plug & Trust middleware se05x\_minimal project example**

4. Go to the MCUXpresso Quickstart Panel and click **Debug** button as shown in [Figure 42](#). If there is more than one probe attached, you have to select the CMSIS-DAP debug probe from the list. Wait a few seconds until the project executes:



**Figure 42.** Debug se05x\_minimal project example

5. When it executes, it will automatically stop in a breakpoint. Click on **Resume** to allow the software to continue its execution as shown in [Figure 43](#).



**Figure 43.** Resume se05x\_minimal project example

6. The project example should now be running into your FRDM-K64F. If it is running successfully, the TeraTerm logs should indicate the available memory in the secure element (in this case, 20820) as can be seen in [Figure 44](#).

Figure 44 shows the TeraTerm window titled "COM44 - Tera Term VT". The log output is as follows:

```

App :INFO :PlugAndTrust_v04.01.00_20211217
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_ss Finished
  
```

7. The same operation can be repeated with any of the other EdgeLock SE05x Plug & Trust middleware project examples.

## 5.7 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 8. 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	SE05X_E	None	07_02	any option	None or SCP03_SSS	disabled
SE050E2	A921						

**Table 9. CMake Settings for SE050F product variants**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
SE050F Dev.Board OM-SE050ARD-F	A92A	SE05X_C	SE050	03_XX	PlatfSCP03 or UserID_PlatformSCP03 or AESKey_PlatformSCP03 or ECKey_PlatformSCP03	SCP03_SSS	enabled
SE050F2	A92A						

**Table 10. CMake Settings for SE050 Previous Generation product variants**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	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						
SE050C1	A200	SE05X_C	None	03_XX	any option	None or SCP03_SSS	enabled
SE050C2	A201						
SE050 Dev Board OM-SE050ARD	A1F4						
SE050F2	A77E <sup>[1]</sup>	SE05X_C	SE050	03_XX	PlatfSCP03 or UserID_PlatformSCP03 or AESKey_PlatformSCP03 or ECKey_PlatformSCP03	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 11. CMake Settings for SE051 product variants**

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
SE051A2	A920	SE05X_A	None	07_02	any option	None or SCP03_SSS	disabled

Table 11. CMake Settings for SE051 product variants...continued

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
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 12. CMake Settings for A5000 product variants

Variant	OEF ID	PTMW_Applet	PTMW_FIPS	PTMW_SE05X_Ver	PTMW_SE05X_Auth	PTMW_SC	SSSFTR_SE05X_RSA
OM-A5000ARD	A736	AUTH	None	07_02	any option	None or SCP03_SSS	disabled
A5000	A736						

### 5.7.1 Example: SE050E CMake build settings

The following images show the configuration for the SE050E development board OM-SE05ARD-E according to [Table 8](#).

- Select `SE05X_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_SC`.

How to enable Platform SCP is described in [Section 6](#).

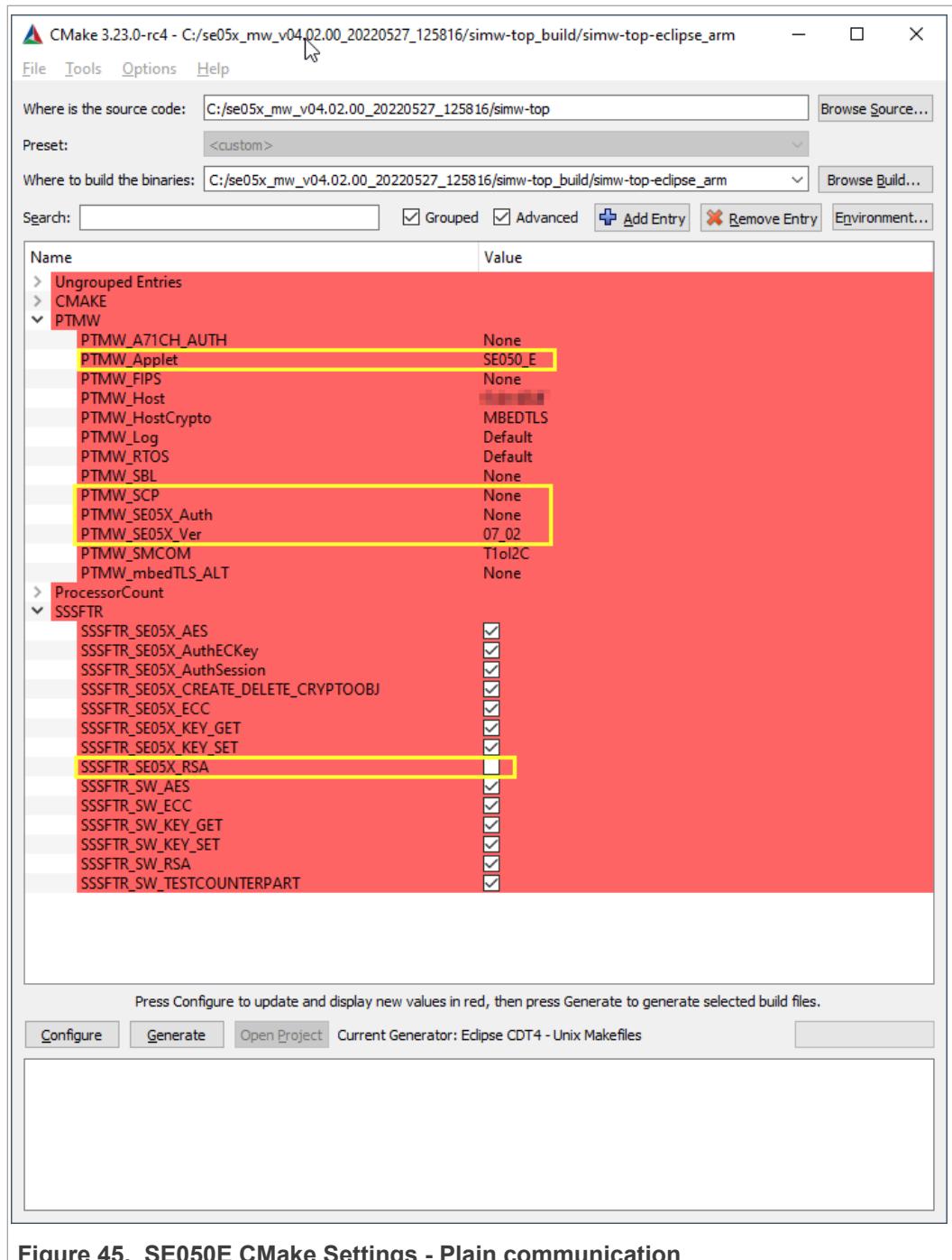


Figure 45. SE050E CMake Settings - Plain communication

## 6 Binding EdgeLock SE05x to a host 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 middleware for EdgeLock SE05x.

The following topics are discussed:

- [Section 6.1 Introduction to the Global Platform Secure Channel Protocol 03 \(SCP03\)](#)
- [Section 6.2 How to configure the Platform SCP keys in the FRDM-K64F MCUXpresso SDK](#)
- [Section 6.3 How to enable Platform SCP in the FRDM-K64F MCUXpresso SDK](#)
- [Section 6.4 How to configure the Platform SCP keys in CMake-based build system](#)
- [Section 6.5 How to enable Platform SCP in the CMake-based build system](#)

## 6.1 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 EdgeLock SE05x 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 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 Secondary 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 13. 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 EdgeLock SE05x Plug & Trust middleware host crypto.

**Table 14. 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](#).

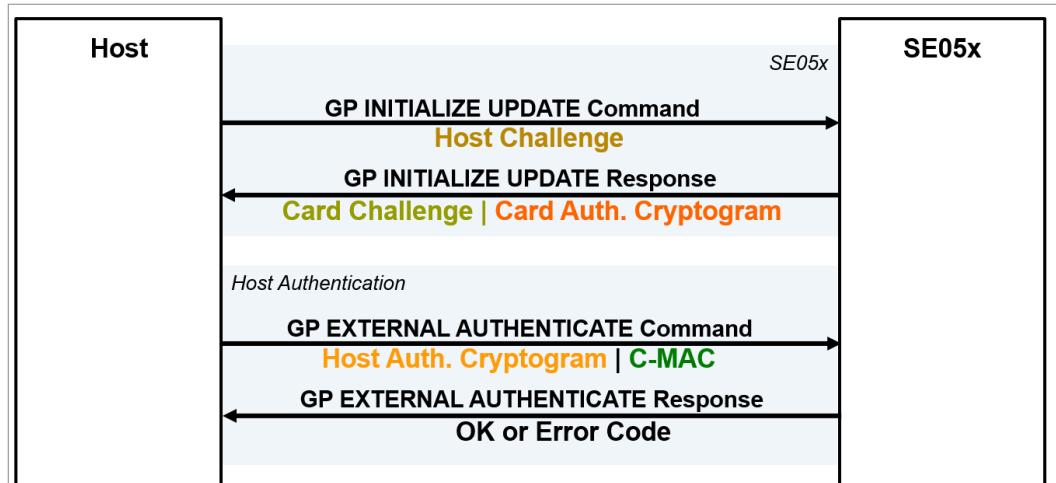


Figure 46. SPC03 mutual authentication – principle

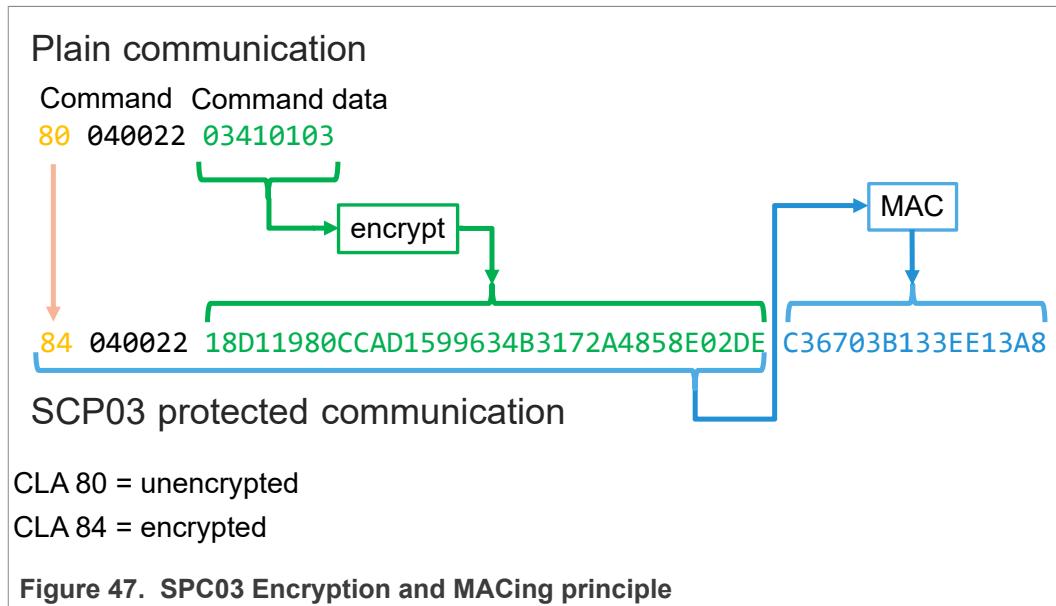


Figure 47. SPC03 Encryption and MACing principle

## 6.2 How to configure the Platform SCP keys in the FRDM-K64F MCUXpresso SDK

The product specific initial Platform SCP key values are described for the EdgeLock SE050 product variants in [AN12436](#) and for the EdgeLock SE051 variants in [AN12973](#).

The EdgeLock SE05x Plug & Trust middleware header file `ex_sss_tp_scp03_keys.h` contains the initial values of all EdgeLock SE050, EdgeLock SE051, A5000 and A71CH product variants.

The `ex_sss_tp_scp03_keys.h` header file can be found in the following location:

```
.\se_hostlib\sss\ex\inc\
```

```

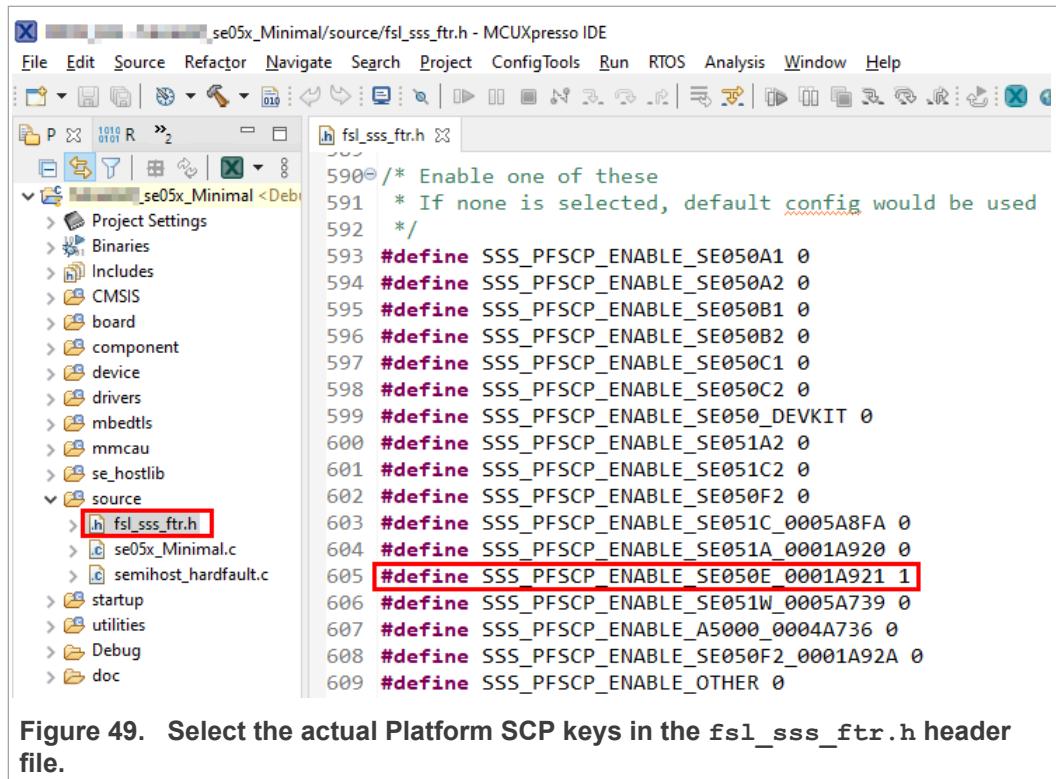
196
197 // SSS_PFSSCP_ENABLE_SE051C_0005A8FA
198 #if defined (SSS_PFSSCP_ENABLE_SE051C_0005A8FA) && SSS_PFSSCP_ENABLE_SE051C_0005A8FA == 1
199 #define SSS_AUTH_KEY_ENC \
200 { 0xBF, 0xC2, 0xD8, 0xE1, 0x82, 0x8E, 0x83, 0x5D, 0x3E, 0x7F, 0xA3, 0x6B, 0x90, 0x2A, 0x05, 0xC6, }
201 #define SSS_AUTH_KEY_MAC \
202 { 0xBE, 0xF8, 0x5B, 0xD7, 0xBA, 0x04, 0x97, 0xD6, 0x28, 0x78, 0x1C, 0xE4, 0x7B, 0x18, 0x8C, 0x96, }
203 #define SSS_AUTH_KEY_DEK \
204 { 0xDB, 0x73, 0xF3, 0x16, 0xBE, 0x29, 0x7F, 0x2F, 0xC9, 0xC0, 0xE4, 0x5F, 0x54, 0x71, 0x06, 0x99, }
205 #endif // SSS_PFSSCP_ENABLE_SE051C_0005A8FA
206
207 // SSS_PFSSCP_ENABLE_SE051A_0001A920
208 #if defined (SSS_PFSSCP_ENABLE_SE051A_0001A920) && SSS_PFSSCP_ENABLE_SE051A_0001A920 == 1
209 #define SSS_AUTH_KEY_ENC \
210 { 0x88, 0xEA, 0x9F, 0xA6, 0x86, 0xF3, 0xCF, 0x2F, 0xFC, 0xAF, 0x4B, 0x1C, 0xBA, 0x93, 0x42, }
211 #define SSS_AUTH_KEY_MAC \
212 { 0x4F, 0x16, 0x3F, 0x59, 0xF0, 0x74, 0x31, 0xF4, 0x3E, 0xE2, 0xEE, 0x18, 0x34, 0xA5, 0x23, 0x34, }
213 #define SSS_AUTH_KEY_DEK \
214 { 0xD4, 0x76, 0xCF, 0x47, 0xA4, 0x27, 0xB5, 0x4A, 0xB3, 0xDB, 0xEB, 0xE7, 0x65, 0x6D, 0x67, 0x70, }
215 #endif // SSS_PFSSCP_ENABLE_SE051A_0001A920
216
217 // SSS_PFSSCP_ENABLE_SE050E_0001A921
218 #if defined (SSS_PFSSCP_ENABLE_SE050E_0001A921) && SSS_PFSSCP_ENABLE_SE050E_0001A921 == 1
219 #define SSS_AUTH_KEY_ENC \
220 { 0xD2, 0xDB, 0x63, 0xE7, 0xA0, 0xA5, 0xAE, 0xD7, 0x2A, 0x64, 0x60, 0xC4, 0xDF, 0xDC, 0xA6, 0x64, }
221 #define SSS_AUTH_KEY_MAC \
222 { 0x73, 0x8D, 0x5B, 0x79, 0x8E, 0xD2, 0x41, 0xB0, 0xB2, 0x47, 0x68, 0x51, 0x4B, 0xFB, 0xA9, 0x5B, }
223 #define SSS_AUTH_KEY_DEK \
224 { 0x67, 0x02, 0xDA, 0xC3, 0x09, 0x42, 0xB2, 0xC8, 0x5E, 0x7F, 0x47, 0xB4, 0x2C, 0xED, 0x4E, 0x7F, }
225 #endif // SSS_PFSSCP_ENABLE_SE050E_0001A921
226
227 // SSS_PFSSCP_ENABLE_SE051W_0005A739
228 #if defined (SSS_PFSSCP_ENABLE_SE051W_0005A739) && SSS_PFSSCP_ENABLE_SE051W_0005A739 == 1
229 #define SSS_AUTH_KEY_ENC \
230 { 0x18, 0xB3, 0xB4, 0xE3, 0x40, 0xC0, 0x80, 0xD9, 0x9B, 0xEB, 0xB8, 0x64, 0x4B, 0x8C, 0x52, }
231 #define SSS_AUTH_KEY_MAC \
232 { 0x3D, 0xC, 0xFA, 0xC8, 0x7B, 0x96, 0x7C, 0x00, 0xE3, 0x3B, 0xA4, 0x96, 0x61, 0x38, 0xA2, }
233 #define SSS_AUTH_KEY_DEK \
234 { 0x6B, 0x06, 0x83, 0xF9, 0x4E, 0x6B, 0xCB, 0x94, 0x73, 0xEC, 0xC1, 0x56, 0x7A, 0x1B, 0xD1, 0x09, }
235 #endif // SSS_PFSSCP_ENABLE_SE051W_0005A739

```

**Figure 48. MCUXpresso SDK - Initial Platform SCP keys are defined in the `ex_sss_tp_scp03_keys.h` header file.**

The `fsl_sss_ftr.h` header file includes compilation options to select one of the predefined initial Platform SCP keys.

Select the desired value of the compilation option by setting exclusively the corresponding C-preprocessor define `SSS_PFSSCP_ENABLE_xx` to 1 (enable). All other values for the same option (represented by C-preprocessor defines `SSS_PFSSCP_ENABLE_xx`) must be set to 0.



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 15. 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 16. 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 17. 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

**Table 17. Platform SCP key define prefix for SE050 Previous Generation product variants...continued**

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050 Dev Board OM-SE050ARD	A1F4	SSS_PFSCP_ENABLE_SE050_DEVKIT
SE050F2	A77E <sup>[1]</sup>	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 18. 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 19. 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

In the next step it is necessary to enable Platfrom SCP in the EdgeLock SE05x Plug & Trust middleware. [Section 6.3](#) describes how to enable Platform SCP in the [Binding EdgeLock SE050 to a host MCU/MPU using Platform SCP](#).

### 6.3 How to enable Platform SCP in the FRDM-K64F MCUXpresso SDK

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

- Set exclusively the C-preprocessor define `SSS_HAVE_SE05X_AUTH_PLATFSCP03` to 1 to configure `PTMW_SE05X_Auth`.
- Set exclusively the C-preprocessor define `SSS_HAVE_SCP_SCP03_SSS` to 1 to configure `PTMW_SCP`.

```

File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
fsl_sss_ftr.h
325
326
327 /* PTMW_SE05X_Auth : SE050 Authentication
328 *
329 * This settings is used by examples to connect using various options
330 * to authenticate with the Applet.
331 * The SE05X_Auth options can be changed for KSDK Demos and Examples.
332 * To change SE05X_Auth option follow below steps.
333 * Set flag ``SSS_HAVE_SCP_SCP03_SSS`` to 1 and Reset flag ``SSS_HAVE_SCP_NONE`` to 0.
334 * To change SE05X_Auth option other than ``None`` and ``PlatformSCP03``,
335 * execute se05x_Delete_and_test_provision.exe in order to provision the Authentication Key.
336 * To change SE05X_Auth option to ``ECKey`` or ``ECKey_PlatformSCP03``,
337 * Set additional flag ``SSS_HAVE_HOSTCRYPTO_ANY`` to 1.
338 */
339
340 /** Use the default session (i.e. session less) login */
341 #define SSS_HAVE_SE05X_AUTH_NONE 0
342
343 /** Do User Authentication with UserID */
344 #define SSS_HAVE_SE05X_AUTH_USERID 0
345
346 /** Use Platform SCP for connection to SE */
347 #define SSS_HAVE_SE05X_AUTH_PLATFORMSCP03 1
348
349 /** Do User Authentication with AES Key
350 * Earlier this was called AppletSCP03 */
351 #define SSS_HAVE_SE05X_AUTH_AESKEY 0
352
353 /** Do User Authentication with EC Key
354 * Earlier this was called FastSCP */
355 #define SSS_HAVE_SE05X_AUTH_ECKEY 0
356
357 /** UserID and PlatformSCP03 */
358 #define SSS_HAVE_SE05X_AUTH_USERID_PLATFORMSCP03 0
359
360 /** AESKey and PlatformSCP03 */
361 #define SSS_HAVE_SE05X_AUTH_AESKEY_PLATFORMSCP03 0
362
363 /** ECKey and PlatformSCP03 */
364 #define SSS_HAVE_SE05X_AUTH_ECKEY_PLATFORMSCP03 0
365

```

Figure 50. Feature file fsl\_sss\_ftr.h - Option PTMW\_SE05X\_Auth - PlatformSCP enabled

```

File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
fsl_sss_ftr.h
227
228 /* PTMW_SCP : Secure Channel Protocol
229 *
230 * In case we enable secure channel to Secure Element, which interface to be used.
231 */
232
233 /**
234 #define SSS_HAVE_SCP_NONE 0
235
236 /** Use SSS Layer for SCP. Used for SE050 family. */
237 #define SSS_HAVE_SCP_SCP03_SSS 1
238
239 /** Use Host Crypto Layer for SCP03. Legacy implementation. Used for older demos of A71CH Family. */
240 #define SSS_HAVE_SCP_SCP03_HOSTCRYPTO 0
241

```

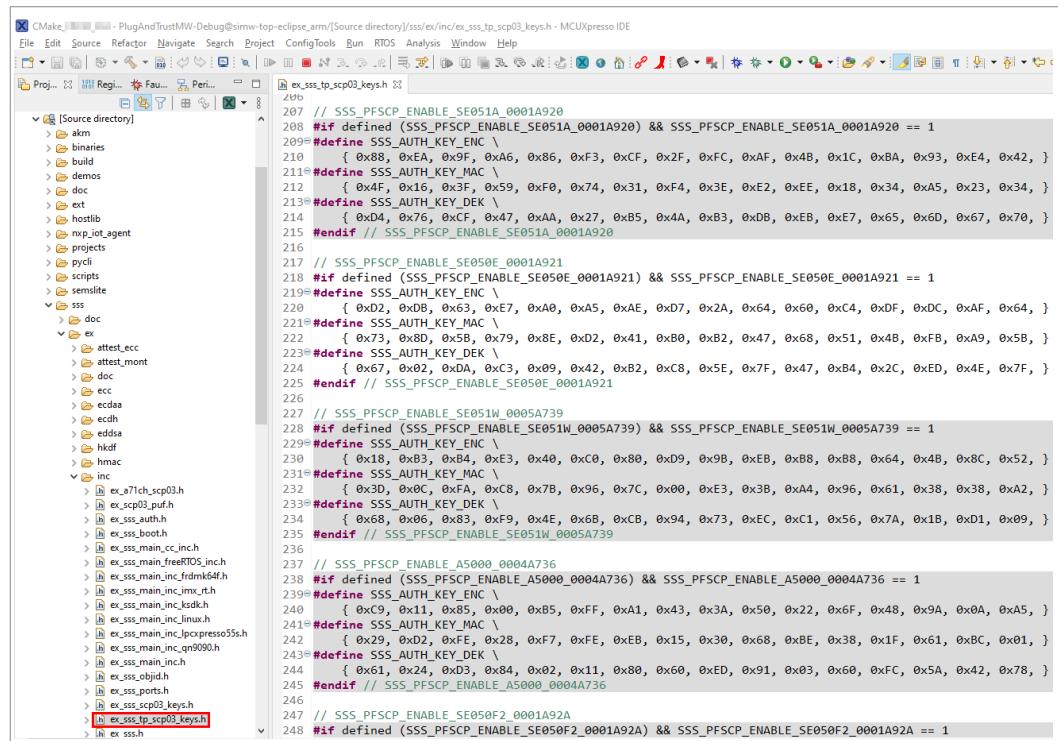
Figure 51. Feature file fsl\_sss\_ftr.h - Option PTMW\_SCP - PlatformSCP enabled

## 6.4 How to configure the Platform SCP keys in CMake-based build system

The product specific initial Platform SCP key values are described for the EdgeLock SE050 product variants in [AN12436](#) and for the EdgeLock SE051 variants in [AN12973](#).

The EdgeLock SE05x Plug & Trust middleware header file `ex_sss_tp_scp03_keys.h` contains the initial values of all EdgeLock SE050, EdgeLock SE051, A5000 and A71CH product variants.

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



```

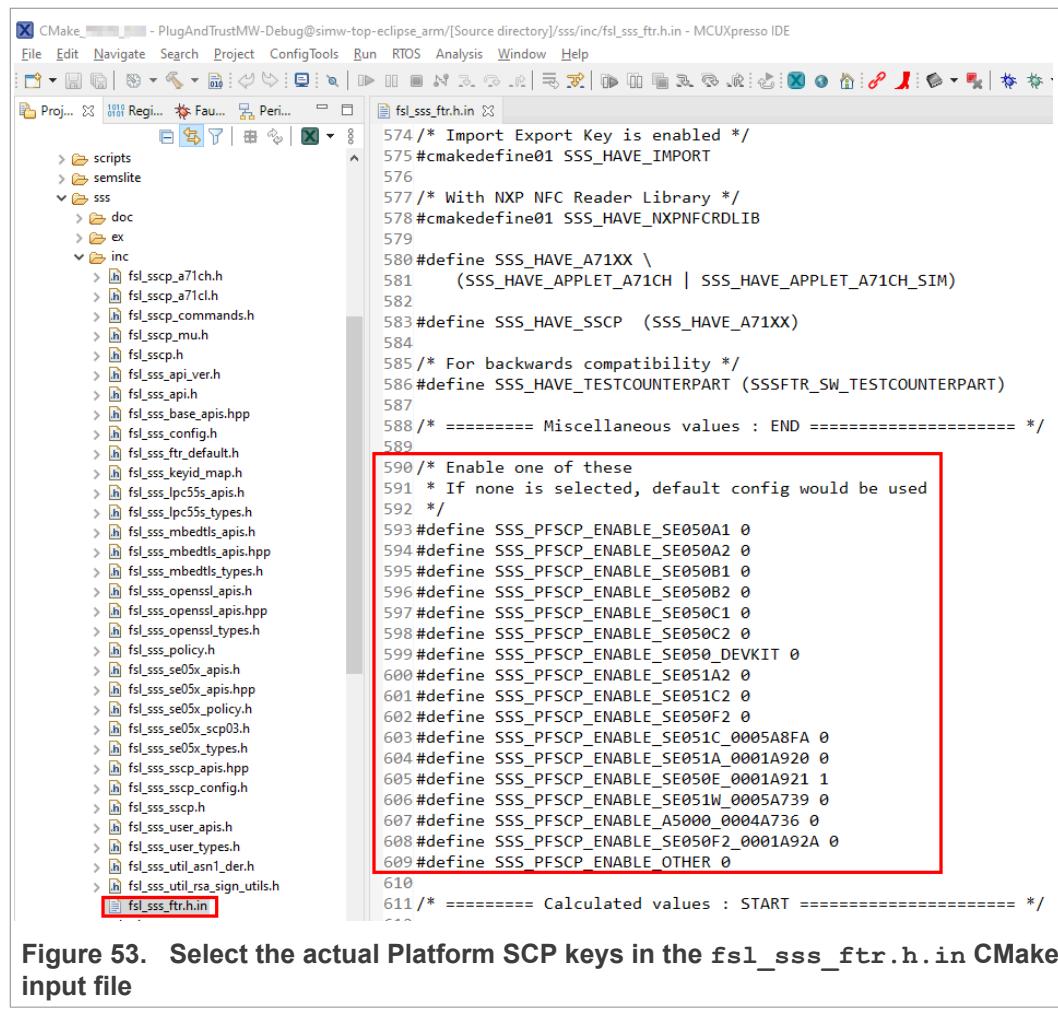
207 // SSS_PFSCP_ENABLE_SE051A_0001A920
208 #if defined (SSS_PFSCP_ENABLE_SE051A_0001A920) && SSS_PFSCP_ENABLE_SE051A_0001A920 == 1
209#define SSS_AUTH_KEY_ENC \
210 { 0x8B, 0xA4, 0x5F, 0xA6, 0x86, 0xF3, 0xCF, 0x2F, 0xFC, 0xAF, 0x4B, 0x1C, 0xBA, 0x93, 0xE4, 0x42, }
211#define SSS_AUTH_KEY_MAC \
212 { 0xAF, 0x16, 0x3F, 0x59, 0xF0, 0x74, 0x31, 0xF4, 0x3E, 0xE2, 0xEE, 0x18, 0x34, 0xA5, 0x23, 0x34, }
213#define SSS_AUTH_KEY_DEK \
214 { 0xD4, 0x76, 0xCF, 0x47, 0xAA, 0x27, 0xB5, 0x4A, 0xB3, 0xDB, 0xEB, 0xE7, 0x65, 0x6D, 0x67, 0x70, }
215#endif // SSS_PFSCP_ENABLE_SE051A_0001A920
216
217 // SSS_PFSCP_ENABLE_SE050E_0001A921
218 #if defined (SSS_PFSCP_ENABLE_SE050E_0001A921) && SSS_PFSCP_ENABLE_SE050E_0001A921 == 1
219#define SSS_AUTH_KEY_ENC \
220 { 0xD2, 0xDB, 0x63, 0xE7, 0xA0, 0xA5, 0xAE, 0xD7, 0x2A, 0x64, 0x60, 0xC4, 0xDF, 0xDC, 0xAF, 0x64, }
221#define SSS_AUTH_KEY_MAC \
222 { 0x73, 0xBD, 0x5B, 0x79, 0x8E, 0xD2, 0x41, 0xB0, 0xB2, 0x47, 0x68, 0x51, 0x4B, 0xFB, 0xA9, 0x5B, }
223#define SSS_AUTH_KEY_DEK \
224 { 0x67, 0x62, 0x6A, 0xC3, 0x09, 0x42, 0xB2, 0xC8, 0x5E, 0x7F, 0x47, 0xB4, 0x2C, 0xED, 0x4E, 0x7F, }
225#endif // SSS_PFSCP_ENABLE_SE050E_0001A921
226
227 // SSS_PFSCP_ENABLE_SE051W_0005A739
228 #if defined (SSS_PFSCP_ENABLE_SE051W_0005A739) && SSS_PFSCP_ENABLE_SE051W_0005A739 == 1
229#define SSS_AUTH_KEY_ENC \
230 { 0x18, 0xB3, 0xE4, 0x3, 0x40, 0xC0, 0x80, 0xD9, 0x9B, 0xEB, 0xB8, 0x64, 0x4B, 0x8C, 0x52, }
231#define SSS_AUTH_KEY_MAC \
232 { 0x3D, 0x6C, 0xF4, 0xC8, 0x7B, 0x96, 0x7C, 0x00, 0xE3, 0x3B, 0xA4, 0x96, 0x61, 0x38, 0x38, 0xA2, }
233#define SSS_AUTH_KEY_DEK \
234 { 0x68, 0x6, 0x83, 0xF9, 0x4E, 0x6B, 0xCB, 0x94, 0x73, 0xEC, 0xC1, 0x56, 0x7A, 0x1B, 0xD1, 0x09, }
235#endif // SSS_PFSCP_ENABLE_SE051W_0005A739
236
237 // SSS_PFSCP_ENABLE_A5000_0004A736
238 #if defined (SSS_PFSCP_ENABLE_A5000_0004A736) && SSS_PFSCP_ENABLE_A5000_0004A736 == 1
239#define SSS_AUTH_KEY_ENC \
240 { 0x9, 0x11, 0x85, 0x00, 0xB5, 0xFF, 0xA1, 0x43, 0x3A, 0x50, 0x22, 0x6F, 0x48, 0x9A, 0x0A, 0xA5, }
241#define SSS_AUTH_KEY_MAC \
242 { 0x29, 0xD2, 0xE, 0x28, 0xF7, 0xFE, 0xEB, 0x15, 0x30, 0x68, 0xBE, 0x38, 0x1F, 0x61, 0xBC, 0x01, }
243#define SSS_AUTH_KEY_DEK \
244 { 0x1, 0x24, 0xD3, 0x84, 0x02, 0x11, 0x80, 0x60, 0xED, 0x91, 0x03, 0x60, 0xFC, 0x5A, 0x42, 0x78, }
245#endif // SSS_PFSCP_ENABLE_A5000_0004A736
246
247 // SSS_PFSCP_ENABLE_SE050F2_0001A92A
248 #if defined (SSS_PFSCP_ENABLE_SE050F2_0001A92A) && SSS_PFSCP_ENABLE_SE050F2_0001A92A == 1

```

**Figure 52. MCUXpresso - Initial Platform SCP keys are defined in `ex_sss_tp_scp03_keys.h` header file**

The `fsl_sss_ftr.h.in` file includes options to select one of the predefined initial Platform SCP keys in the `ex_sss_tp_scp03_keys.h` header file. This file is located in: `.\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.



**Figure 53. Select the actual Platform SCP keys in the `fsl_sss_ftr.h.in` CMake input file**

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 overwriting 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 into 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: ..\simw-top\_build\simw-top-eclipse\_arm.

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 20. 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

**Table 20.** Platform SCP key define prefix for SE050E product variants...continued

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050E2	A921	SSS_PFSCP_ENABLE_SE050E_0001A921

**Table 21.** 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 22.** 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 <sup>[1]</sup>	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 23.** 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 24.** 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

In the next step it is necessary to enable Platfrom SCP in the EdgeLock SE05x Plug & Trust middleware. [Section 6.5](#) describes how to enable Platform SCP in the CMake-based build system.

## 6.5 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.

The following images show the configuration for the SE050E development board OM-SE05ARD-E.

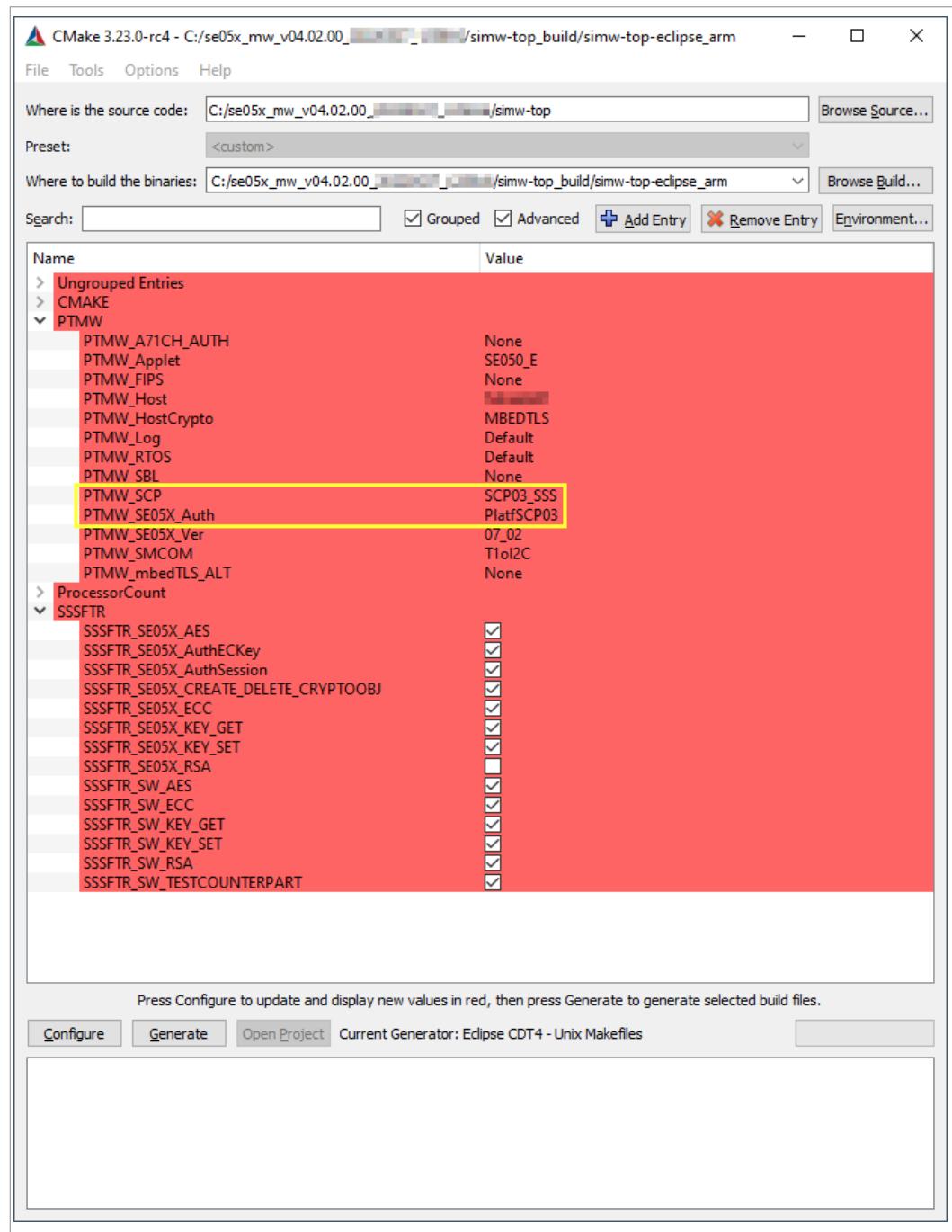


Figure 54. SE050E CMake Settings - PlatformSCP enabled

## 7 Appendix A: Install MCUXpresso IDE

MCUXpresso is a free-of-charge, code size unlimited, easy-to-use IDE for Kinetis and LPC MCUs, and i.MX RT crossover processors. To install it, do the following:

1. Go to [MCUXpresso](#) and click the download button as indicated in [Figure 55](#):

The screenshot shows the MCUXpresso IDE website. At the top, there is a navigation bar with tabs: OVERVIEW (which is selected), DOCUMENTATION, DOWNLOADS, DEVELOPMENT TOOLS, and TRAINING & SUPPORT. Above the navigation bar, there are social media links for FOLLOW, EMAIL, and GITHUB. The main content area has a sidebar on the left with 'Jump To' links: Overview & Features, Supported Devices, and System Requirements. The main content area is titled 'Overview' and contains text about the MCUXpresso IDE's features, including Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores, advanced editing, compiling, and debugging features, and support for general purpose crossover and wireless-enabled MCUs. It also mentions the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools. Below this text is a 'More ▾' link. At the bottom of the main content area are two buttons: 'USER GUIDE' and 'DOWNLOAD'. The 'DOWNLOAD' button is highlighted with a red box.

Figure 55. Go to MCUXpresso website

2. You will be asked to sign-in with your account at the NXP website. If you do not have an account, click on **Register Now** as shown in [Figure 56](#):

The screenshot shows the NXP website. At the top, there is a navigation bar with links: PRODUCTS, APPLICATIONS, DESIGN, SUPPORT, and COMPANY. Below the navigation bar, there is a breadcrumb trail: Home / Sign In or Register. The main content area is titled 'Sign In' and contains fields for 'Email Address or NXP Company ID' and 'Password', both with placeholder text. There is a large orange 'SIGN IN' button. Below the password field is a link 'Forgot your password?'. At the bottom of the sign-in form is a link 'Don't have an Account? [Register Now](#)'.

Figure 56. Register your NXP account

3. If you already have an account, you can directly type your (1) email address, (2) password and (3) click sign-in button as shown in [Figure 57](#):

Home / Sign In or Register

### Sign In

Email Address or NXP Company ID  
1

Password  
2

SIGN IN  
3

Forgot your password?

Don't have an Account? [Register Now](#)

Figure 57. Sign-in in NXP website

4. Click on MCUXpresso IDE as shown in [Figure 58](#):

NXP > Design > Product Information : MCUXpresso IDE

### Product Information

MCUXpresso IDE

Select a version. To access older versions, click on the "Previous" tab

Version	Description
11.5.0	MCUXpresso IDE

Download Log

Figure 58. Select MCUXpresso

- Accept software terms and conditions as shown in [Figure 59](#):

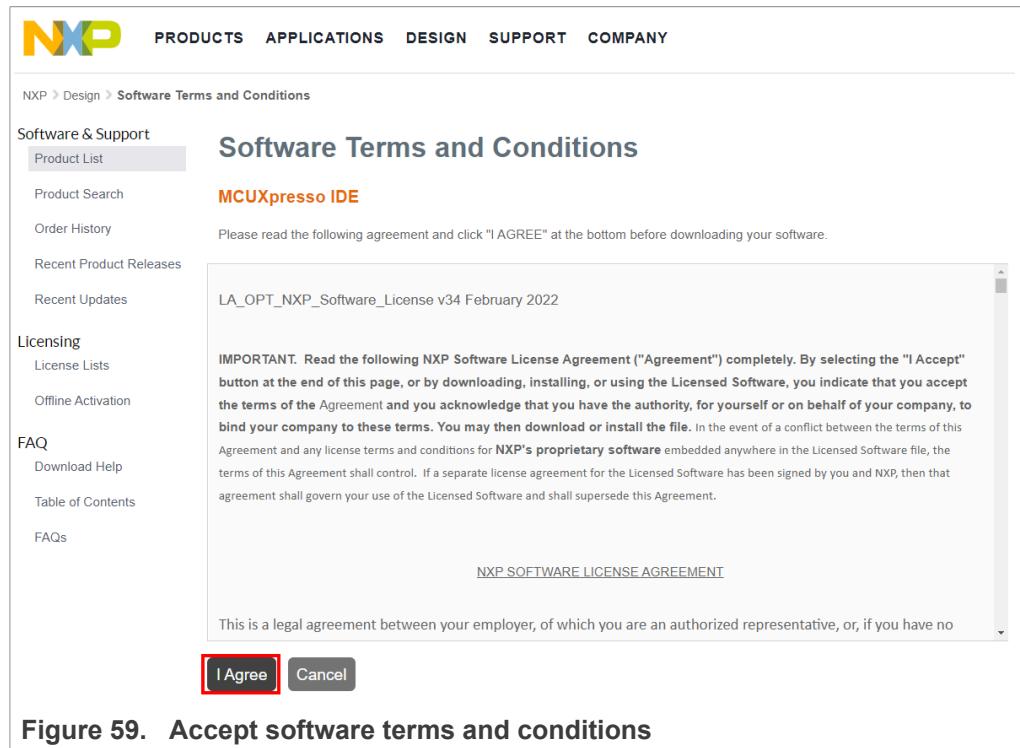


Figure 59. Accept software terms and conditions

- Select your MCUXpresso product version and click on the corresponding **File Name** to start the download as shown in [Figure 60](#):

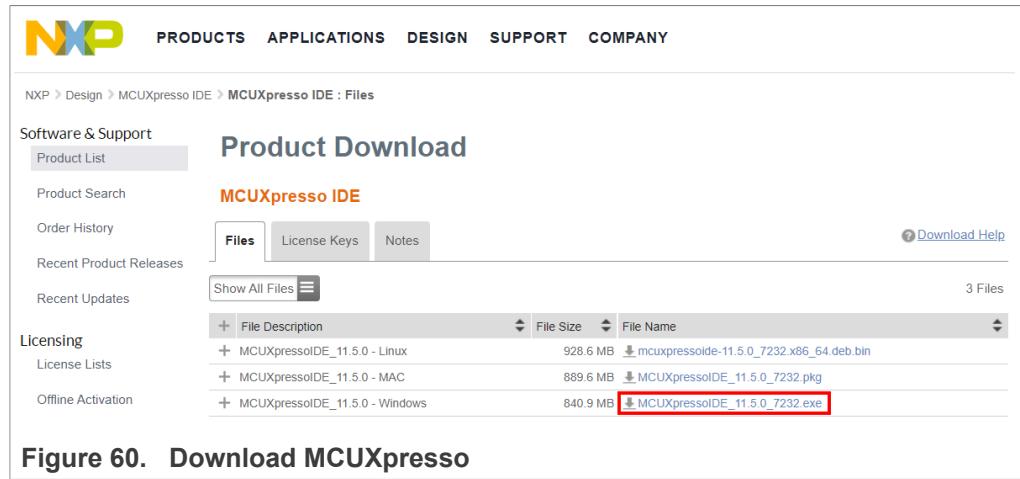


Figure 60. Download MCUXpresso

- Double click on the installer file and follow the setup wizard until MCUXpresso installation is completed. Please, make sure you allow the installation of the additional

drivers required by MCUXpresso during the installation process as shown in [Figure 61](#), [Figure 62](#), [Figure 63](#) and [Figure 64](#):



Figure 61. Install MCUXpresso required drivers I



Figure 62. Install MCUXpresso required drivers II

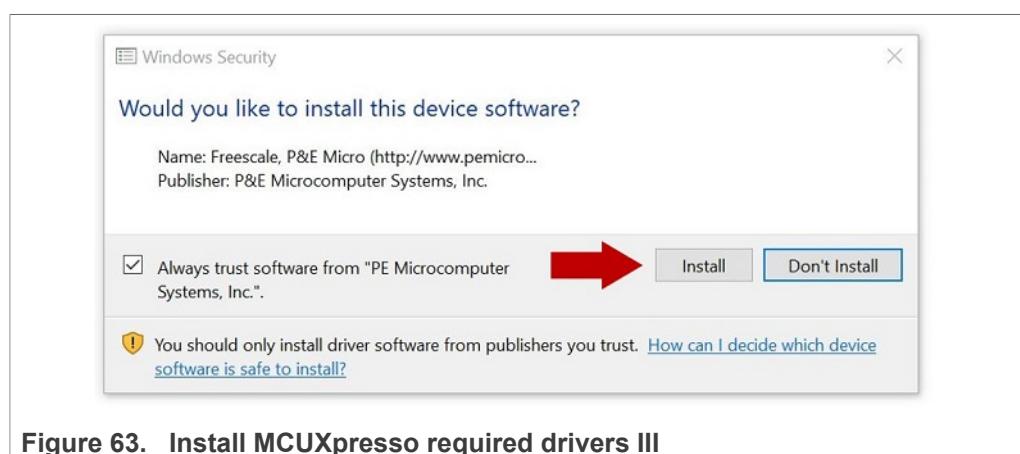
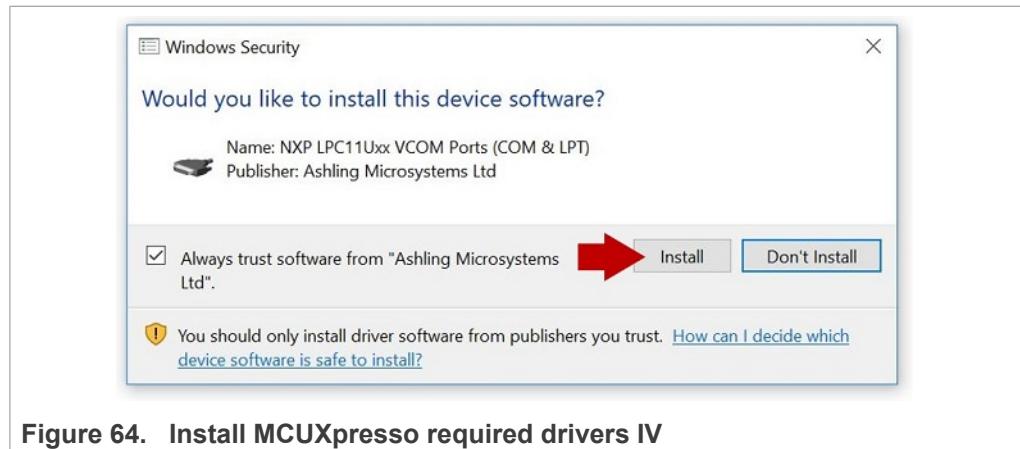


Figure 63. Install MCUXpresso required drivers III



## 8 Appendix B: Install CMake

CMake is an open-source, cross-platform family of tools that helps you build C/C++ projects on multiple platforms using a compiler-independent method. It has minimal dependencies, requiring only a C++ compiler on its own build system. SE05x middleware leverages on CMake to generate native makefiles and workspaces that can be used in the compiler environment of your choice.

To install CMake:

1. Go to CMake downloads page: <https://cmake.org/download/>

2. Scroll down and select your binary distribution. For this guide, the binary distribution is Windows as shown in [Figure 65](#):

The screenshot shows the CMake download page for version 3.22.3. It includes sections for Source distributions and Binary distributions. A red arrow points to the 'cmake-3.22.3-windows-x86\_64.msi' file under the Windows x64 Installer section.

Platform	Files
Unix/Linux Source (has \n line feeds)	<a href="#">cmake-3.22.3.tar.gz</a>
Windows Source (has \\r\\n line feeds)	<a href="#">cmake-3.22.3.zip</a>

Platform	Files
Windows x64 Installer: <b>Installer tool has changed. Uninstall CMake 3.4 or lower first!</b>	<a href="#">cmake-3.22.3-windows-x86_64.msi</a> <span style="color: red;">(Red arrow points here)</span>
Windows x64 ZIP	<a href="#">cmake-3.22.3-windows-x86_64.zip</a>
Windows i386 Installer: <b>Installer tool has changed. Uninstall CMake 3.4 or lower first!</b>	<a href="#">cmake-3.22.3-windows-i386.msi</a>
Windows i386 ZIP	<a href="#">cmake-3.22.3-windows-i386.zip</a>
macOS 10.13 or later	<a href="#">cmake-3.22.3-macos-universal.dmg</a>
	<a href="#">cmake-3.22.3-macos-universal.tar.gz</a>
macOS 10.10 or later	<a href="#">cmake-3.22.3-macos10.10-universal.dmg</a>
	<a href="#">cmake-3.22.3-macos10.10-universal.tar.gz</a>
Linux x86_64	<a href="#">cmake-3.22.3-linux-x86_64.sh</a>
	<a href="#">cmake-3.22.3-linux-x86_64.tar.gz</a>
Linux aarch64	<a href="#">cmake-3.22.3-linux-aarch64.sh</a>
	<a href="#">cmake-3.22.3-linux-aarch64.tar.gz</a>

Figure 65. Download CMake

3. Double click on the downloaded installer file. Windows Defender SmartScreen might pop-up the wizard shown in [Figure 66](#):



Figure 66. Execute CMake installer

4. If this is your case: Click (1) on **More info** and then (2) click on **Run anyway** as shown in [Figure 67](#):

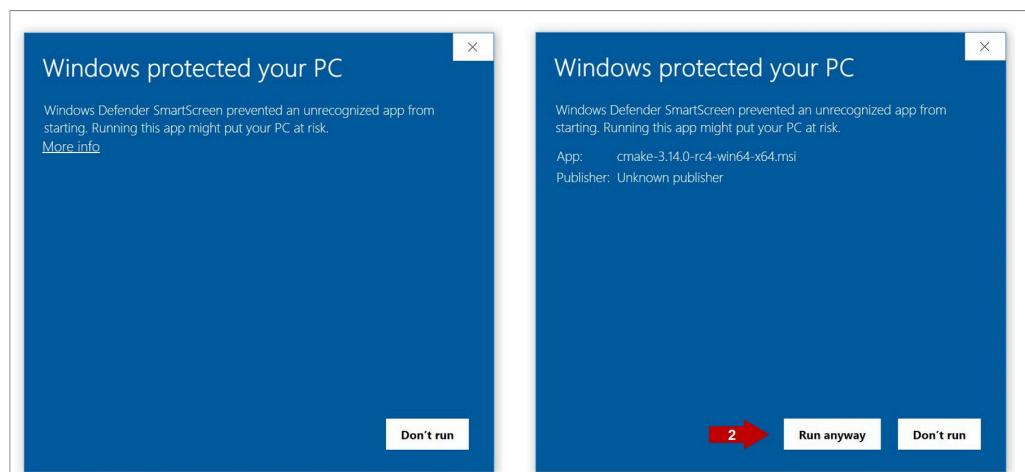


Figure 67. Run the CMake installer (II)

5. The CMake installation wizard will open. Click (1) **Next** and (2) **accept** the End-User License Agreement as shown in [Figure 68](#):

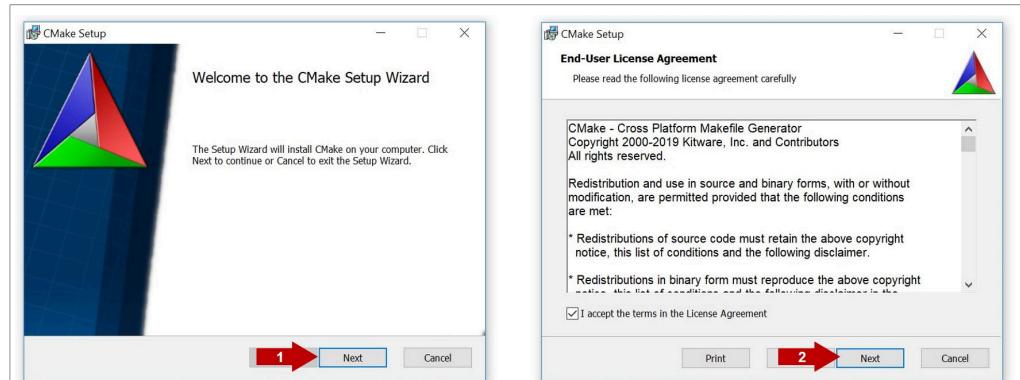


Figure 68. CMake installation wizard

6. As part of the CMake setup, (1) **Add Cmake to the system PATH for all users** and (2) click **Next** as shown in [Figure 69](#):

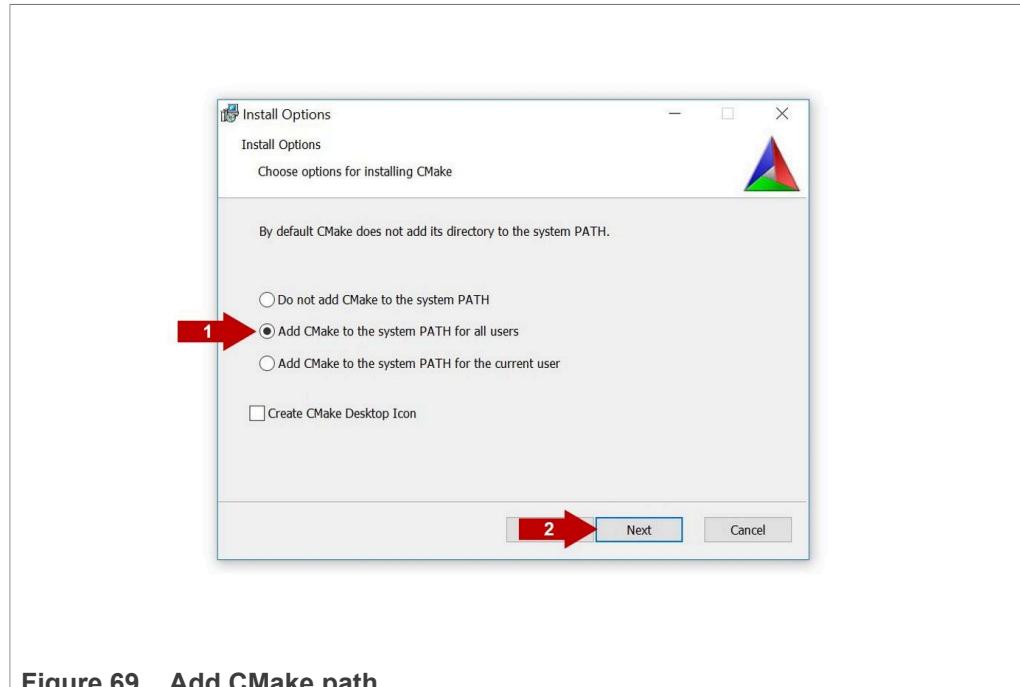


Figure 69. Add CMake path

7. Select a destination folder, (1) click **Next** and then (2) click **Install** as shown in [Figure 70](#):

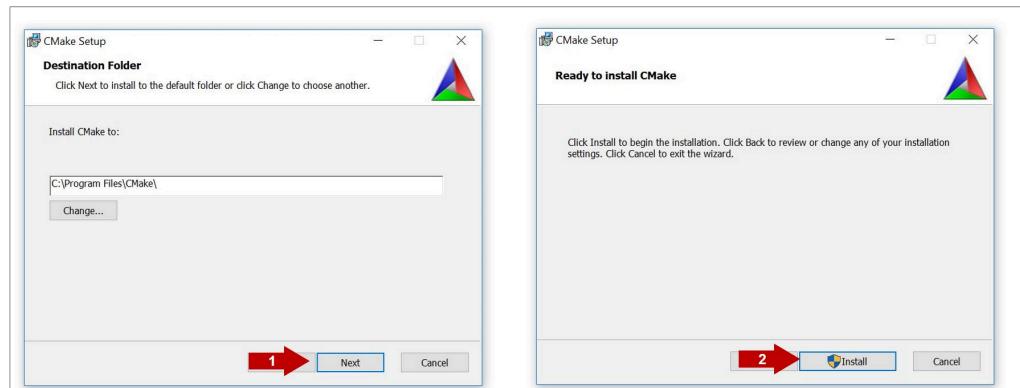


Figure 70. Install CMake

8. Wait a few seconds until the installation is completed and click **Finish** as shown in [Figure 71](#):

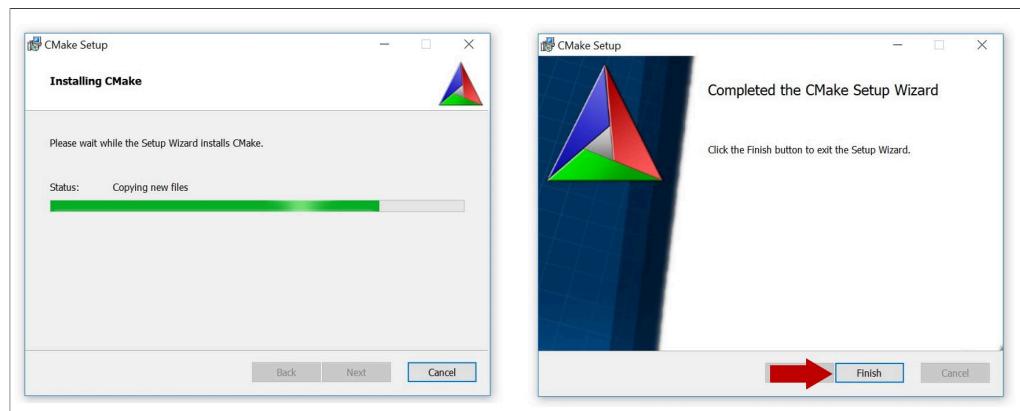


Figure 71. Complete CMake installation

## 9 Appendix C: Install Python

This section explains how to install Python  $\geq 3.7.x$  and  $\leq 3.9.x$  32-bit version, but the same procedure can be applied for more recent versions. Follow these steps to install Python in your local machine:

1. Go to <https://www.python.org/downloads> and download **Python ≥ 3.7.x and ≤ 3.9 32-bit version**. Make sure you download the Python 32 bit version.

**Files**

Version	Operating System	Description	MDS Sum	File Size	GPG
Gzipped source tarball	Source release		1440acb71471e2394befdb30b1a958d1	25800844	SIG
XZ compressed source tarball	Source release		e754c4b2276750fd5b4785a1b443683a	19154136	SIG
macOS 64-bit Intel-only installer	macOS	for macOS 10.9 and later, deprecated	2714cb9e6241cf7e2f9022714a55d27a	30395760	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	c2393ab11a423d817501b8566ab5da9f	38217233	SIG
Windows embeddable package (32-bit)	Windows		c1d2af96d9f3564f57f35fcfc3c1006eb	7671509	SIG
Windows embeddable package (64-bit)	Windows		b8e8bfba8e56edcd654d15e3bcd2e29a	8509821	SIG
Windows help file	Windows		784020441c1a25289483d3d8771a8215	9284044	SIG
<b>Windows installer (32-bit)</b>	Windows		<b>457d648dc8a71b6bc32da30a7805c55b</b>	<b>27767040</b>	<b>SIG</b>
Windows installer (64-bit)	Windows	Recommended	747ac35ae667f4ec1ee3b001e9b7dbc6	28909456	SIG

Figure 72. Download Python 3.9.x 32 bit version

2. Double click on the downloaded installer file. Select the "*Install launcher for all users*" and "*Add Python 3.7 to Path*" options and click *Install Now* as indicated in Figure 73:

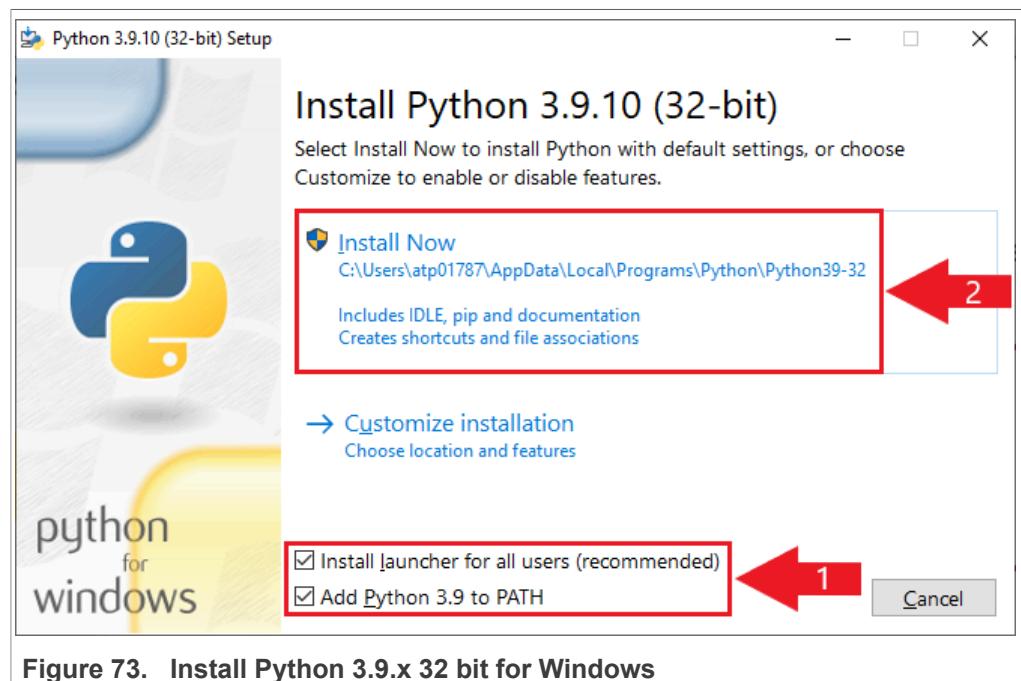
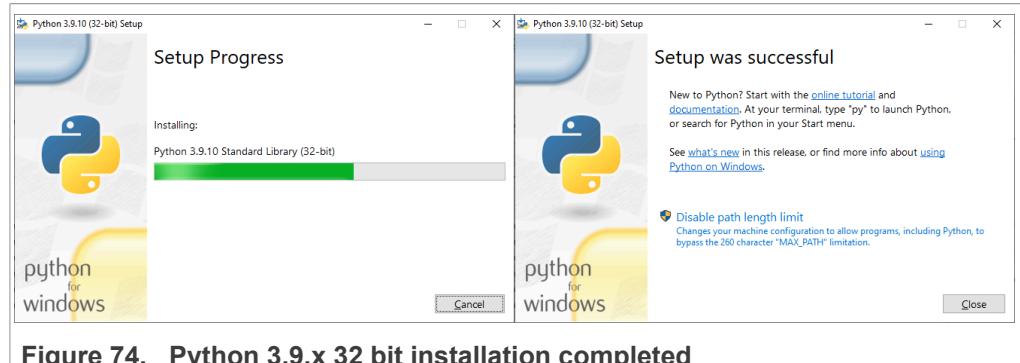


Figure 73. Install Python 3.9.x 32 bit for Windows

3. Wait a few seconds until the installation is completed as indicated in [Figure 74](#)



## 10 Appendix D: Update FRDM-K64F board with DAPLink firmware

Arm Mbed DAPLink is an open-source software project that enables programming and debugging application software running on Arm Cortex CPUs. DAPLink runs an open-source bootloader and enables developers with drag-and-drop programming, a serial port and CMSIS-DAP based debugging.

**Note:** To debug MCUXpresso project examples, we need to flash FRDM-K64F with DAPLink firmware. If your FRDM-K64F board already includes DAPLink firmware, you can skip these steps.

To flash DAPLink firmware, follow these steps:

1. Go to [NXP OpenSDA](#) site

2. Scroll down and select FRDM-K64F board from the ***Download - OpenSDA bootloader and application*** drop down list as indicated in [Figure 75](#):

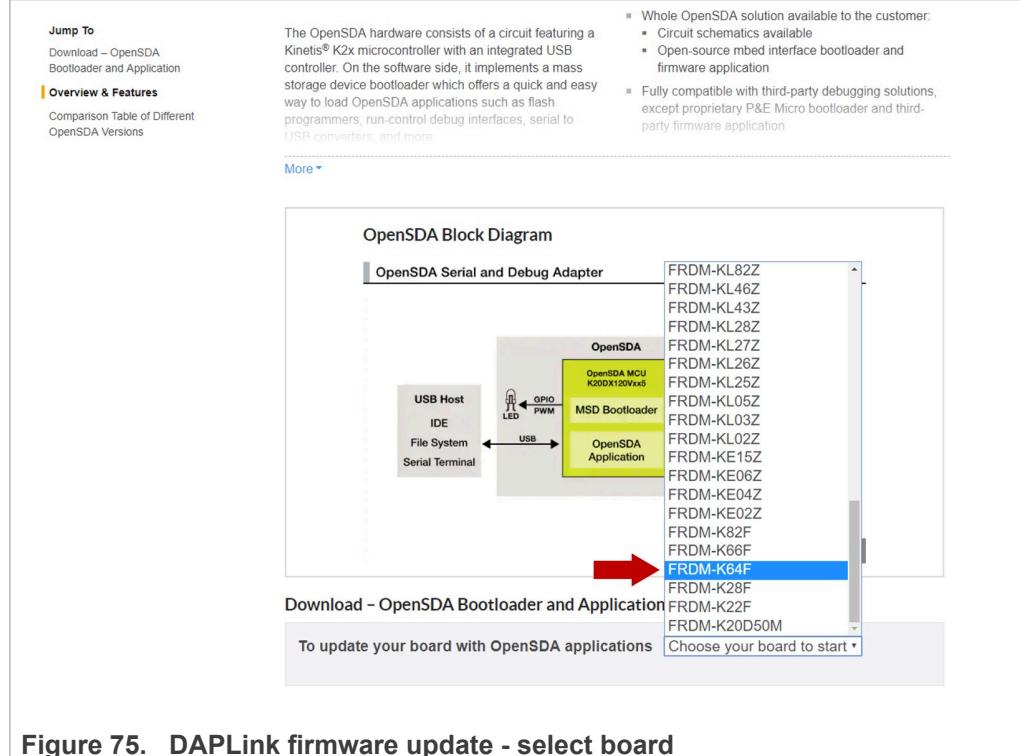


Figure 75. DAPLink firmware update - select board

3. Download the latest DAPLink firmware version as shown in [Figure 76](#):

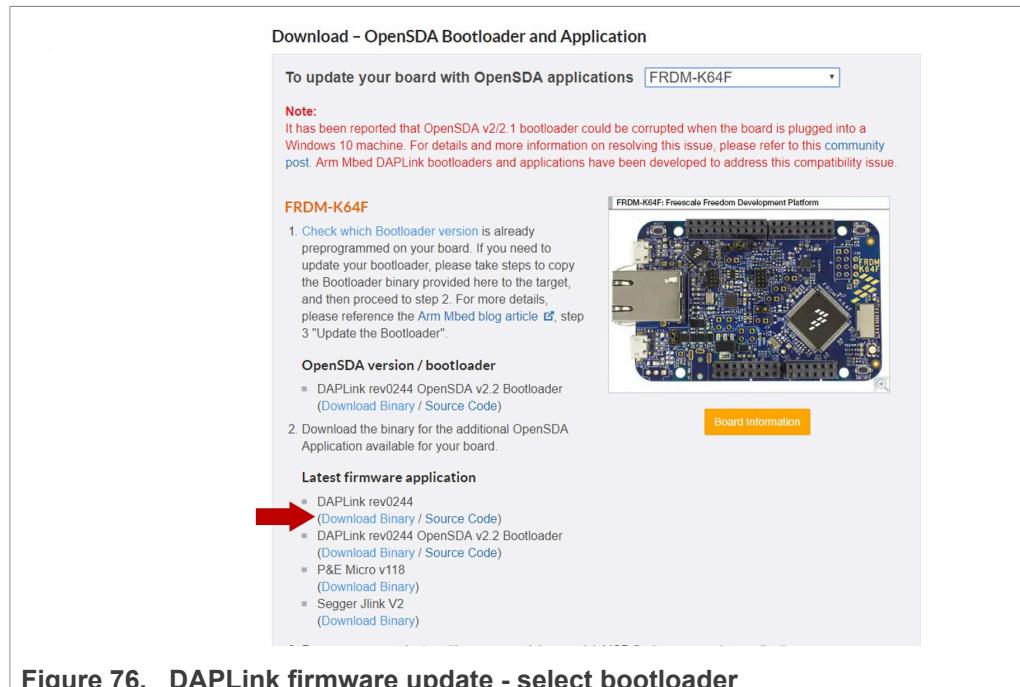


Figure 76. DAPLink firmware update - select bootloader

4. Start the board's bootloader mode. To do so, (1) keep reset button pressed while (2) connecting the USB cable to the SDA USB port and release it after 1s ([Figure 77](#)):

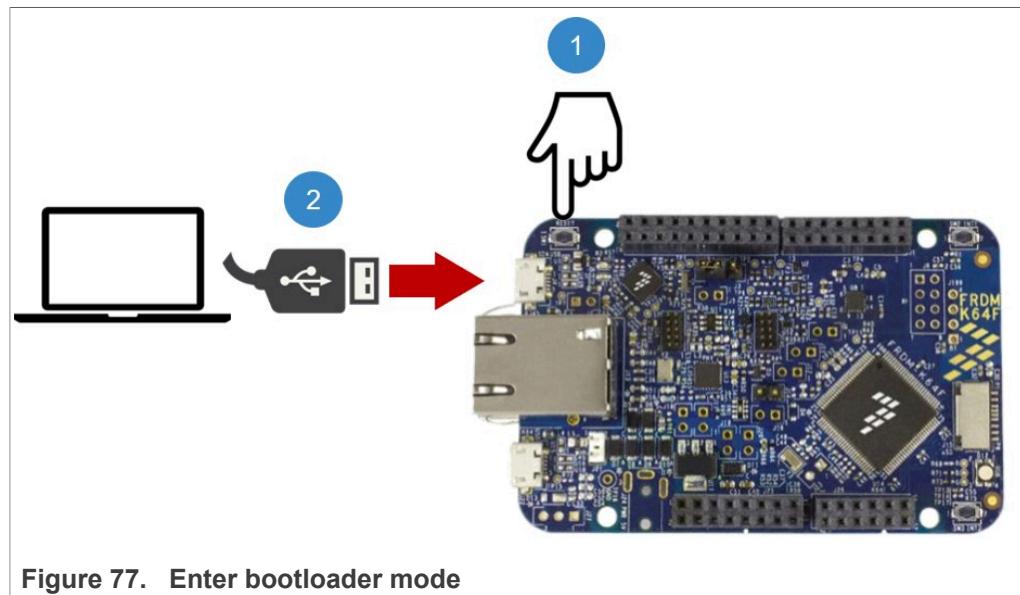


Figure 77. Enter bootloader mode

5. Drag and drop or copy and paste the binary file into the BOOTLOADER drive from your computer file explorer as shown in [Figure 78](#). The FRDM-K64F will automatically un-mount after the drag and drop operation.

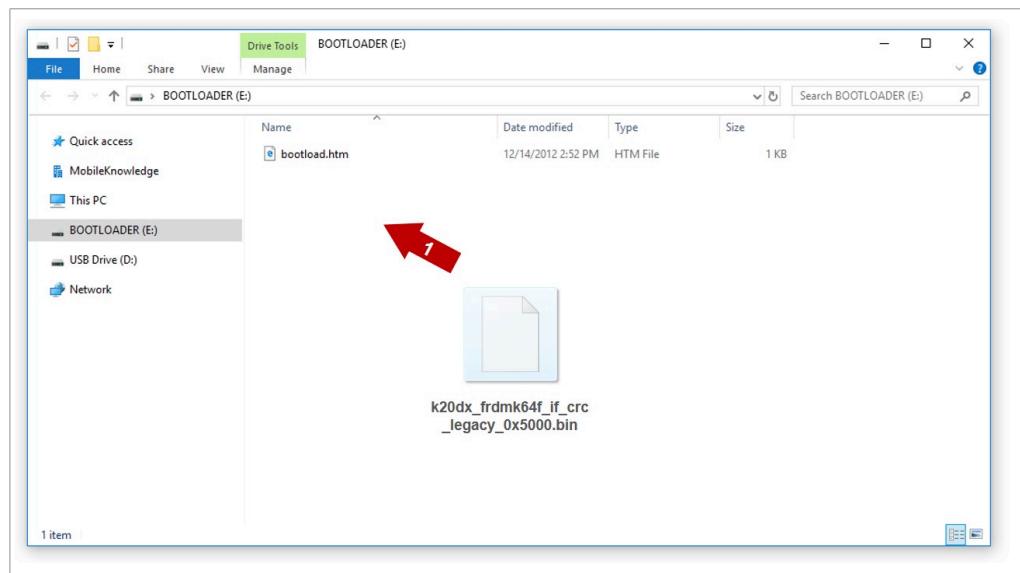


Figure 78. Enter bootloader mode

6. Un-plug and re-plug the USB cable from the SDA USB port **without** keeping reset button pressed.

7. Check the category Ports (COM & LPT) from your computer Device Manager to ensure that new devices have been properly detected and their driver correctly installed by your computer OS.

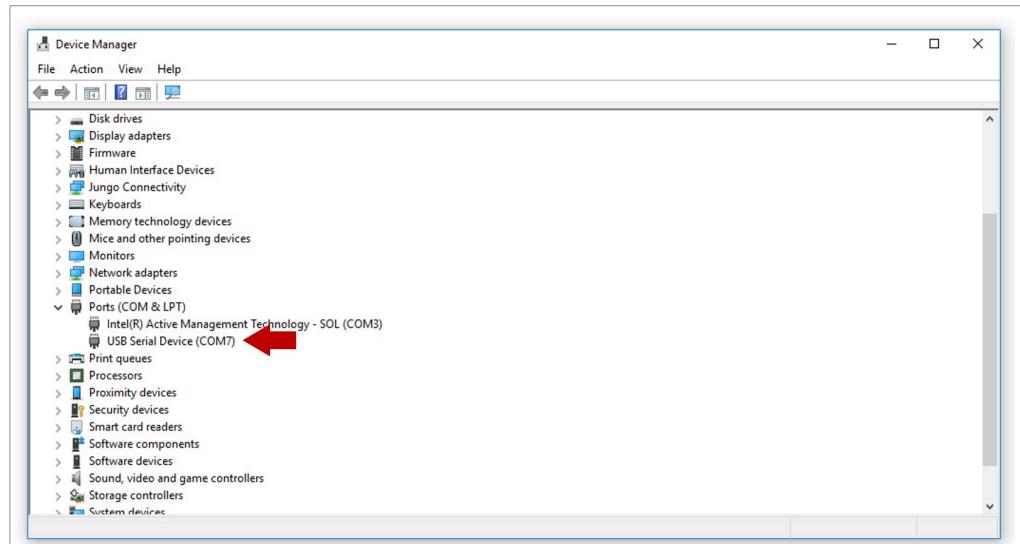


Figure 79. Enter bootloader mode

**Note:** In case the device does not show up in your Device Manager, please download the latest bootloader version, as shown in [Figure 76](#), or check / exchange the USB cables used.

## 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

Tab. 1.	EdgeLock SE05x development boards .....	3
Tab. 2.	FRDM-K64F details .....	4
Tab. 3.	Feature file fsl_sss_ftr.h settings for SE050E product variants .....	15
Tab. 4.	Feature file fsl_sss_ftr.h settings for SE050F product variants .....	16
Tab. 5.	Feature file fsl_sss_ftr.h settings for SE050 Previous Generation product variants .....	16
Tab. 6.	Feature file fsl_sss_ftr.h settings for SE051 product variants .....	17
Tab. 7.	Feature file fsl_sss_ftr.h settings for A5000 product variants .....	18
Tab. 8.	CMake Settings for SE050E product variants .....	36
Tab. 9.	CMake Settings for SE050F product variants .....	37
Tab. 10.	CMake Settings for SE050 Previous Generation product variants .....	37
Tab. 11.	CMake Settings for SE051 product variants ....	37
Tab. 12.	CMake Settings for A5000 product variants .....	38
Tab. 13.	Static SCP03 keys .....	41
Tab. 14.	SCP03 session keys .....	41
Tab. 15.	Platform SCP key define prefix for SE050E product variants .....	44
Tab. 16.	Platform SCP key define prefix for SE050F product variants .....	44
Tab. 17.	Platform SCP key define prefix for SE050 Previous Generation product variants .....	44
Tab. 18.	Platform SCP key define prefix for SE051 product variants .....	45
Tab. 19.	Platform SCP key define prefix for A5000 product variants .....	45
Tab. 20.	Platform SCP key define prefix for SE050E product variants .....	48
Tab. 21.	Platform SCP key define prefix for SE050F product variants .....	49
Tab. 22.	Platform SCP key define prefix for SE050 Previous Generation product variants .....	49
Tab. 23.	Platform SCP key define prefix for SE051 product variants .....	49
Tab. 24.	Platform SCP key define prefix for A5000 product variants .....	49

## Figures

Fig. 1.	Jumper configuration for FRDM-K64F .....	5
Fig. 2.	Arduino connectors of OM-SE05xARD and FRDM-K64F boards .....	5
Fig. 3.	OM-SE05xARD mounted in FRDM-K64F board .....	6
Fig. 4.	Import FRDM-K64F board SDK into MCUXpresso environment .....	7
Fig. 5.	Imported FRDM-K64F SDK .....	7
Fig. 6.	Import projects from SDK .....	8
Fig. 7.	SDK import wizard .....	9
Fig. 8.	Select projects to import .....	10
Fig. 9.	Imported projects in MCUXpresso workspace .....	11
Fig. 10.	Connect boards to the laptop .....	11
Fig. 11.	TeraTerm setup .....	12
Fig. 12.	Build projects in MCUXpresso workspace .....	12
Fig. 13.	Debug projects in MCUXpresso workspace .....	13
Fig. 14.	Run projects in MCUXpresso workspace .....	13
Fig. 15.	TeraTerm logs - se05x_Minimal project example .....	14
Fig. 16.	Feature file fsl_sss_ftr.h example: Assign the value SE05X_E to the CMake option PTWM_Applet .....	15
Fig. 17.	Feature file fsl_sss_ftr.h - PTMW_Applet .....	19
Fig. 18.	Feature file fsl_sss_ftr.h - Option PTMW_ FIPS .....	19
Fig. 19.	Feature file fsl_sss_ftr.h - Option PTMW_ SE05x_Ver .....	20
Fig. 20.	Feature file fsl_sss_ftr.h - Option PTMW_ AUTH - Plain communication .....	21
Fig. 21.	Feature file fsl_sss_ftr.h - Option PTMW_ SCP - Plain communication .....	21
Fig. 22.	Feature file fsl_sss_ftr.h - Option SSSFTR_ SE05X_RSA .....	22
Fig. 23.	Create se050_middleware folder .....	23
Fig. 24.	Unzip se05x middleware .....	23
Fig. 25.	Generate EdgeLock SE05x Plug & Trust middleware project examples .....	24
Fig. 26.	SE05x middleware project structure .....	24
Fig. 27.	Import a project wizard .....	25
Fig. 28.	Import a project wizard (II) .....	25
Fig. 29.	Select EdgeLock SE05x Plug & Trust middleware build folder .....	26
Fig. 30.	Import EdgeLock SE05x Plug & Trust middleware .....	26
Fig. 31.	EdgeLock SE05x Plug & Trust middleware imported in workspace .....	27
Fig. 32.	Import a project wizard .....	28
Fig. 33.	Import a project wizard (II) .....	28
Fig. 34.	Select FRDM-K64F projects folder .....	29
Fig. 35.	Import FRDM-K64F in workspace .....	29
Fig. 36.	FRDM-K64F imported in workspace .....	30
Fig. 37.	EdgeLock SE05x Plug & Trust middleware examples .....	31
Fig. 38.	Configure CMake options of EdgeLock SE05x Plug & Trust middleware examples .....	32
Fig. 39.	Connect boards to the laptop .....	33
Fig. 40.	Configure TeraTerm .....	33
Fig. 41.	Debug EdgeLock SE05x Plug & Trust middleware se05x_minimal project example .....	34

Fig. 42.	Debug se05x_minimal project example .....	35	Fig. 56.	Register your NXP account .....	51
Fig. 43.	Resume se05x_minimal project example .....	35	Fig. 57.	Sign-in in NXP website .....	52
Fig. 44.	TeraTerm logs - se05x_minimal project example .....	36	Fig. 58.	Select MCUXpresso .....	52
Fig. 45.	SE050E CMake Settings - Plain communication .....	39	Fig. 59.	Accept software terms and conditions .....	53
Fig. 46.	SPC03 mutual authentication – principle .....	42	Fig. 60.	Download MCUXpresso .....	53
Fig. 47.	SPC03 Encryption and MACing principle .....	42	Fig. 61.	Install MCUXpresso required drivers I .....	54
Fig. 48.	MCUXpresso SDK - Initial Platform SCP keys are defined in the ex_sss_tp_scp03_keys.h header file. ....	43	Fig. 62.	Install MCUXpresso required drivers II .....	54
Fig. 49.	Select the actual Platform SCP keys in the fsl_sss_ftr.h header file. ....	44	Fig. 63.	Install MCUXpresso required drivers III .....	54
Fig. 50.	Feature file fsl_sss_ftr.h - Option PTMW_SE05X_Auth - PlatformSCP enabled .....	46	Fig. 64.	Install MCUXpresso required drivers IV .....	55
Fig. 51.	Feature file fsl_sss_ftr.h - Option PTMW_SC - PlatformSCP enabled .....	46	Fig. 65.	Download CMake .....	56
Fig. 52.	MCUXpresso - Initial Platform SCP keys are defined in ex_sss_tp_scp03_keys.h header file .....	47	Fig. 66.	Execute CMake installer .....	57
Fig. 53.	Select the actual Platform SCP keys in the fsl_sss_ftr.h.in CMake input file .....	48	Fig. 67.	Run the CMake installer (II) .....	57
Fig. 54.	SE050E CMake Settings - PlatformSCP enabled .....	50	Fig. 68.	CMake installation wizard .....	58
Fig. 55.	Go to MCUXpresso website .....	51	Fig. 69.	Add CMake path .....	58
			Fig. 70.	Install CMake .....	59
			Fig. 71.	Complete CMake installation .....	59
			Fig. 72.	Download Python 3.9.x 32 bit version .....	60
			Fig. 73.	Install Python 3.9.x 32 bit for Windows .....	60
			Fig. 74.	Python 3.9.x 32 bit installation completed .....	61
			Fig. 75.	DAPLink firmware update - select board .....	62
			Fig. 76.	DAPLink firmware update - select bootloader .....	62
			Fig. 77.	Enter bootloader mode .....	63
			Fig. 78.	Enter bootloader mode .....	63
			Fig. 79.	Enter bootloader mode .....	64

## Contents

1	<b>How to use this document .....</b>	3
2	<b>Hardware required .....</b>	3
3	<b>Boards setup .....</b>	4
3.1	OM-SE050ARD-E jumper configuration .....	5
3.2	OM-SE05xARD and FRDM-K64F board connection .....	5
4	<b>Import project examples from FRDM-K64F</b>	
	<b>SDK .....</b>	6
4.1	Prerequisites .....	6
4.2	Download FRDM-K64F SDK .....	6
4.3	Install FRDM-K64F SDK .....	7
4.4	Import project example in MCUXpresso .....	7
4.5	Build, run and debug project example .....	11
4.6	Product specific build settings .....	14
4.6.1	Example: SE050E build settings .....	18
5	<b>Import project examples from CMake-based build system .....</b>	22
5.1	Prerequisites .....	22
5.2	Download EdgeLock SE05x Plug & Trust middleware .....	22
5.3	Build EdgeLock SE05x Plug & Trust middleware project examples .....	24
5.4	Import PlugAndTrustMW project example in MCUXpresso workspace .....	24
5.5	Import cmake_projects_frdm64f project example in MCUXpresso workspace .....	27
5.6	Run EdgeLock SE05x Plug & Trust middleware examples .....	30
5.6.1	List the EdgeLock SE05x Plug & Trust middleware examples .....	30
5.6.2	Edit EdgeLock SE05x Plug & Trust middleware example CMake options .....	31
5.6.3	Build and run a EdgeLock SE05x Plug & Trust middleware project example .....	32
5.7	Product specific CMake build settings .....	36
5.7.1	Example: SE050E CMake build settings .....	38
6	<b>Binding EdgeLock SE05x to a host using Platform SCP .....</b>	39
6.1	Introduction to the Global Platform Secure Channel Protocol 03 (SCP03) .....	40
6.2	How to configure the Platform SCP keys in the FRDM-K64F MCUXpresso SDK .....	42
6.3	How to enable Platform SCP in the FRDM-K64F MCUXpresso SDK .....	45
6.4	How to configure the Platform SCP keys in CMake-based build system .....	46
6.5	How to enable Platform SCP in the CMake-based build system .....	49
7	<b>Appendix A: Install MCUXpresso IDE .....</b>	51
8	<b>Appendix B: Install CMake .....</b>	55
9	<b>Appendix C: Install Python .....</b>	59
10	<b>Appendix D: Update FRDM-K64F board with DAPLink firmware .....</b>	61

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.