AN12450

EdgeLock SE05x Quick start guide with i.MX RT1060 and i.MX RT1170

Rev. 4.2 — 4 August 2022 Application note
546942

Document information

Information	Content
Keywords	EdgeLock SE05x, EdgeLock A5000, Plug & Trust middleware, i.MX RT1060, i.MX RT1170
Abstract	This document explains how to get started with the EdgeLock SE05x Plug & Trust middleware using the EdgeLock SE05x/A5000 development boards and i.MX RT1060 or i.MX RT1170 MCU boards. It provides detailed instructions to run projects imported either from the board SDKs or the CMake-based build system included in the EdgeLock SE05x Plug & Trust middleware.

Revision history

Revision history

Revision number	Date	Description
1.0	2019-07-18	First release
2.0	2019-11-25	Major update to incorporate details to import projects from i.MX RT1060 SDK and CMake-based build system.
2.1	2019-12-17	Corrected OM-SE050ARD J14 jumper setting.
2.2	2020-01-20	Fixed broken links Section 3.
3.0	2020-10-27	Updated for EdgeLock SE051 and i.MX RT1060
3.1	2020-12-07	Updated to latest template and fixed broken links
4.0	2021-12-23	Added instructions for i.MX RT1170
4.1	2022-03-28	Add EdgeLock SE050E and EdgeLock A5000 product variants. Update Table 1, Figure 1, Figure 2, Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8 Add note (step 3) in Section 4.5 Build, run and debug project example Add Section 4.6 Product specific build settings Add note in Section 5.6.2 Run EdgeLock SE05x Plug & Trust middleware examples Add Section 5.7 Product specific CMake build settings Add Section 6 Binding EdgeLock SE05x to a host using Platform SCP
4.2	2022-08-04	Update to EdgeLock SE Plug & Trust Middleware version 04.02.xx. Update note (step 3) in Section 4.5 Build, run and debug project example Update Section 4.6 Product specific build settings Update note in Section 5.6.2 Run EdgeLock SE05x Plug & Trust middleware examples Update Section 5.7 Product specific CMake build settings Update Section 6 Binding EdgeLock SE05x to a host using Platform SCP

1 How to use this document

The Plug & Trust middleware includes a set of project examples that demonstrate the use of EdgeLock SE05x in the latest IoT security use cases. These project examples can be either:

- Imported from the MCUXpresso SDKs available for i.MX RT1060 and i.MX RT1170 MCU boards. Using the board SDKs is recommended as it is the easiest and fastest way of importing and running the project examples.
- Imported from the CMake-based build system included in the Plug & Trust middleware package. The CMake-based option is provided for developers familiar with this build system or that are willing to run exactly the same project examples on PC/Windows/ Linux and embedded targets.

This document provides detailed instructions to run EdgeLock SE05x project examples imported either from the board SDKs or the CMake-based build system.

The main body of this document should be used in this sequence:

1. Order board samples. You can find the ordering details of the boards required in this document in Section 2.
2. Setup your boards. Section 3 describes how to setup the OM-SE05xARD and your MCU board (either i.MX RT1060 board or i.MX RT1170 board).
3. Run project examples. Go to Section 4 for instructions on how to import projects from the board SDKs following the recommended way of working, or alternatively, go to Section 5 for instructions on how to import projects from the CMake-based build system.

Additional material is provided in the appendices of this document.

2 Required hardware

The EdgeLock SE05x works as an auxiliary security device attached to a host controller through an I²C interface. To follow the instructions provided in this document, you need at least an EdgeLock SE05x development board and an MCU board (either i.MX RT1060 or i.MX RT1170) acting as a host controller.

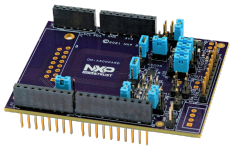
EdgeLock SE05x development boards ordering details

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

Table 1. EdgeLock SE05x development boards.

Part number	12NC	Description	Picture
OM-SE050ARD-E	9354 332 66598	SE050E Arduino® compatible development kit	

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

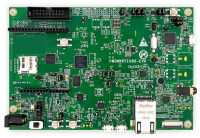
Part number	12NC	Description	Picture
OM-SE050ARD-F	9354 357 63598	SE050 Arduino® compatible development kit	
OM-SE050ARD	9353 832 82598	SE050F Arduino® compatible development kit	
OM-SE051ARD	9353 991 87598	SE051 Arduino® compatible development kit	
OM-A5000ARD	9354 243 19598	A5000 Arduino® compatible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

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

i.MX RT1060 MCU board ordering details

Table 2 provides the ordering details for the i.MX RT1060 development board.

Table 2. i.MX RT1060 evaluation kit details

Part number	12NC	Content	Picture
IMXRT1060-EVKB	935419011598	MIMXRT1060-EVK low cost evaluation kit for Cortex-M7	

i.MX RT1170 MCU board ordering details

Table 3 provides the ordering details for the i.MX RT1170 development board.

Table 3. i.MX RT1170 evaluation kit details

Part number	12NC	Content	Picture
MIMXRT1170-EVK	935378982598	MIMXRT1170-EVK low cost evaluation kit for Cortex-M7	

3 Boards setup

This section explains how to setup your i.MX RT1060/1170 and OM-SE05xARD boards to execute the Plug & Trust middleware:

1. The OM-SE05xARD board has jumpers that allow you to use the EdgeLock SE05x I^2C interface 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.*

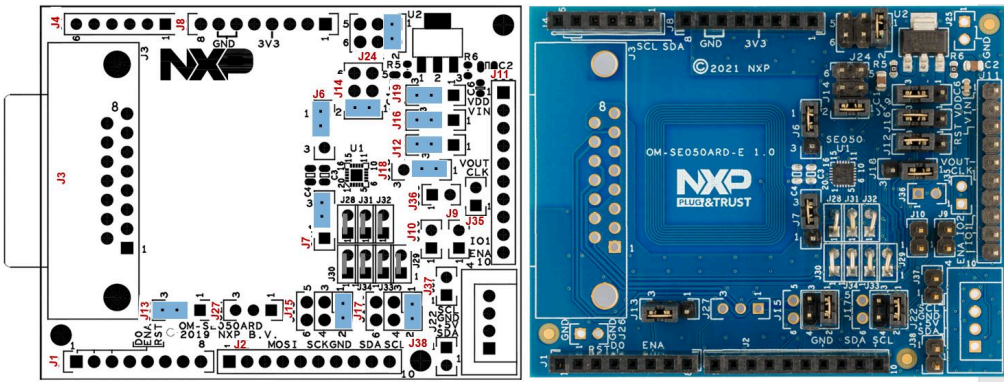


Figure 1. Jumper configuration for OM-SE05xARD

2. The OM-SE05xARD, and i.MX RT1060/1170 boards can be directly connected using the Arduino headers available in both boards. Figure 2 shows how to connect the OM-

SE05xARD board on top of the i.MX RT1060. Figure 3 shows how to connect it on top of the i.MX RT1170 board.

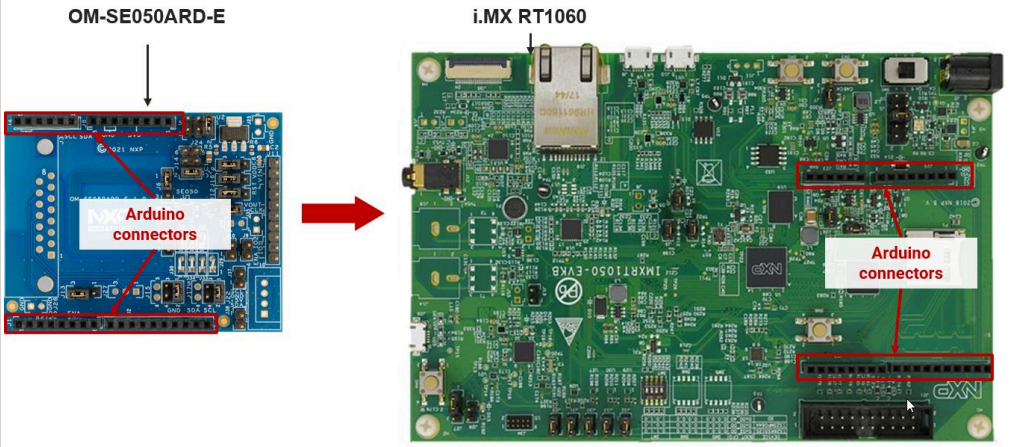


Figure 2. Plug OM-SE05xARD and i.MX RT1060 boards

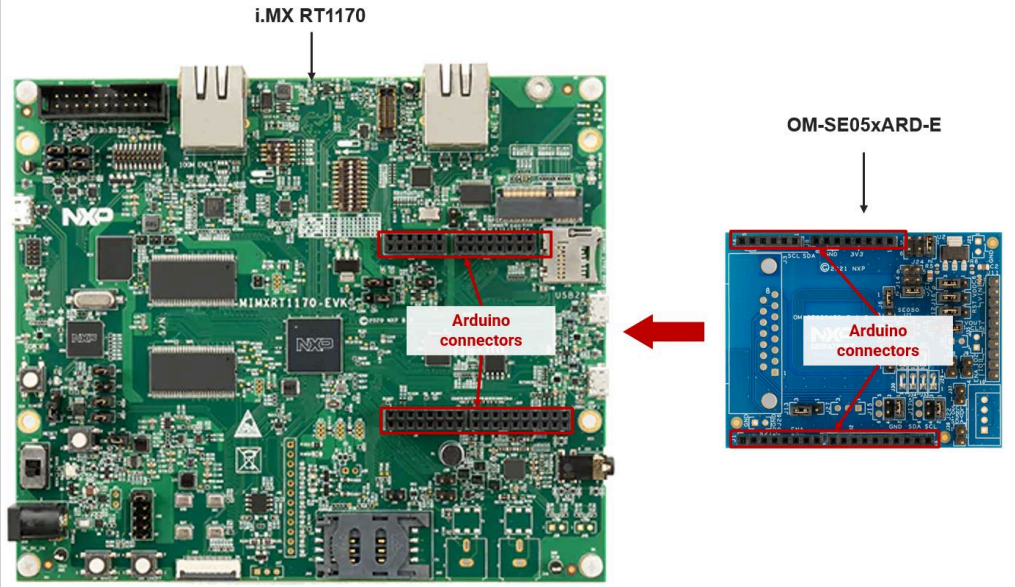


Figure 3. Plug OM-SE05xARD and i.MX RT1170 boards

3. Double check that the two boards are connected as shown in Figure 4 (for i.MX RT1060 board) or Figure 5 (for i.MX RT1170 board):

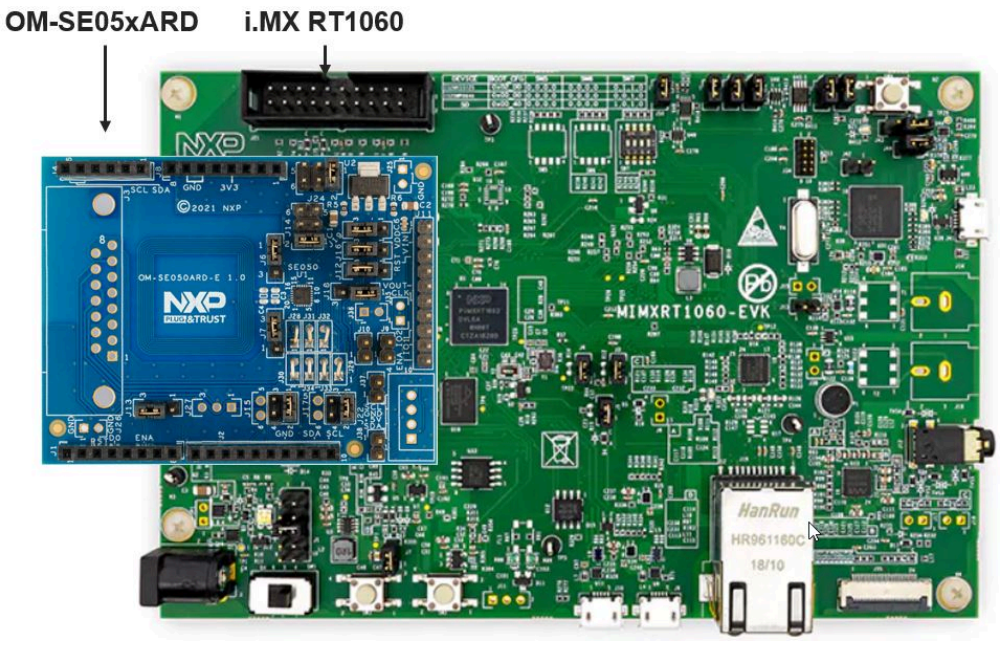


Figure 4. OM-SE05xARD mounted in i.MX RT1060 board

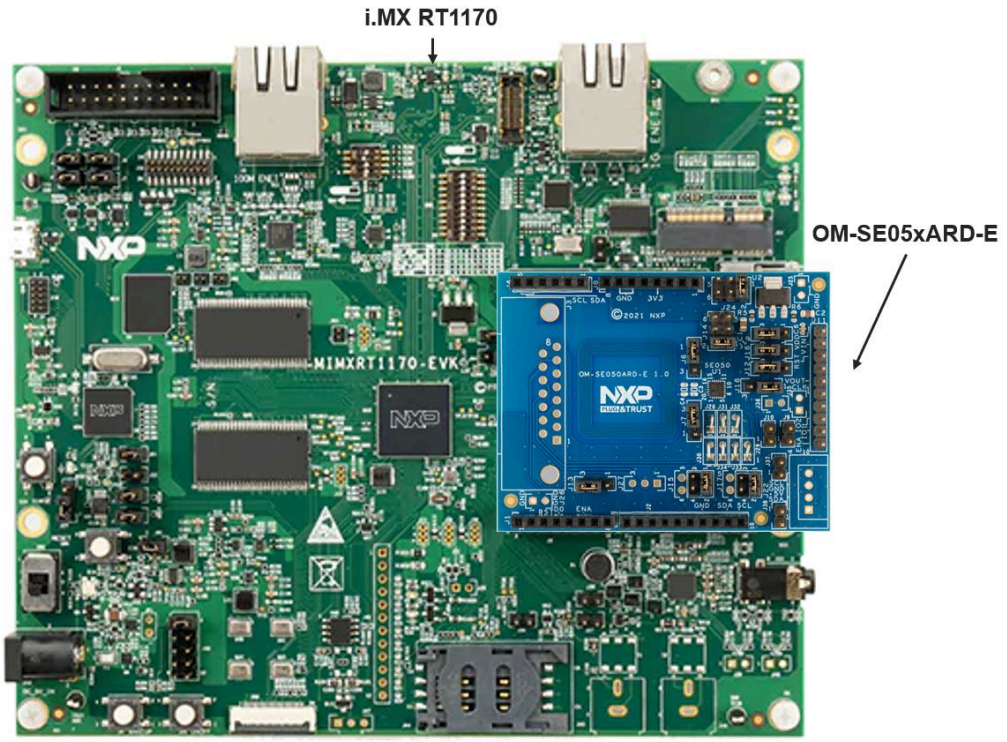


Figure 5. OM-SE05xARD mounted in i.MX RT1170 board

4. **i.MX RT1060**: configure the **J1** jumper of the i.MX RT1060 EVK board in position 5-6 as shown in Figure 6. With this setting, the board power supply and the DAP-Link debugger will function using the **J41** micro-USB connector.
   i.MX RT1170: configure the **J38** jumper of the i.MX RT1170 EVK board in position 5-6 as shown in Figure 7. With this setting, the board power supply and the DAP-Link debugger will function using the **J11** micro-USB connector.

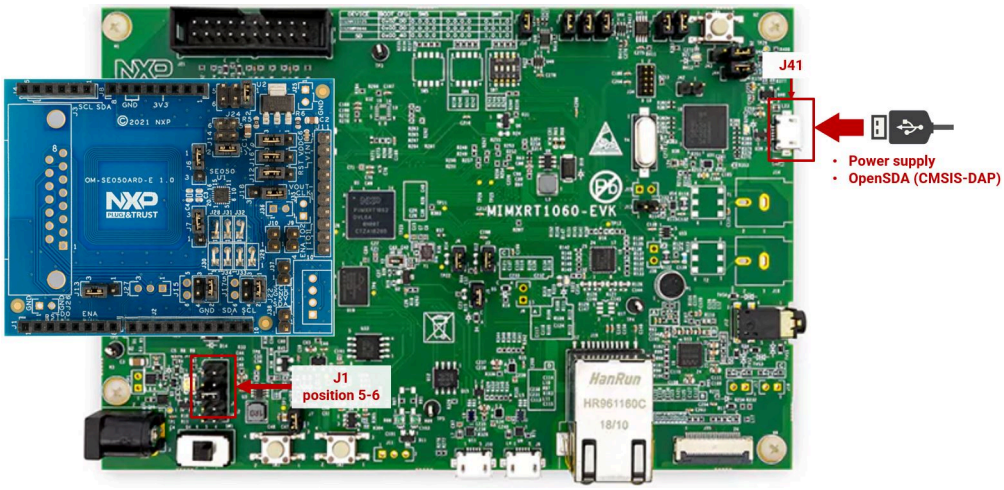


Figure 6. Jumper configuration for i.MX RT1060 EVK board

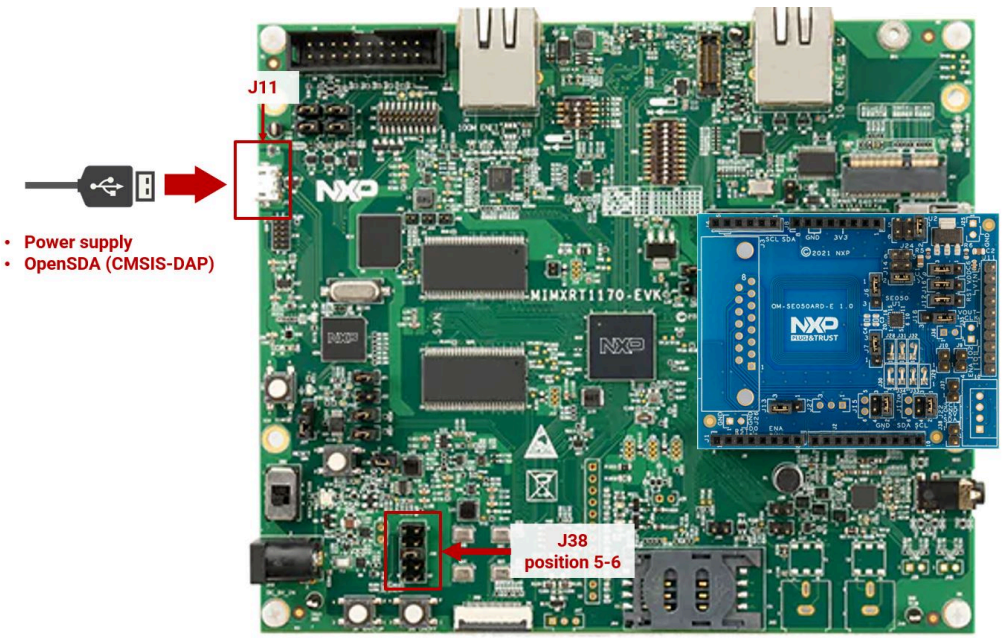


Figure 7. Jumper configuration for i.MX RT1170 EVK board

5. Check that your laptop recognizes the i.MX RT1060/1170 board by following the steps shown in Figure 8:
   a. Connect the board to your laptop using the micro-USB connector **J41** (for i.MX RT1060 board) or **J11** (for i.MX RT1170 board).
   b. Check that the serial port is recognized in the category Ports (COM & LTP). In this document, it is recognized as USB Serial Device (COM12) but this naming might change depending on your computer. Therefore, it is important that you identify which device is recognized at the moment you plug the board to the PC.

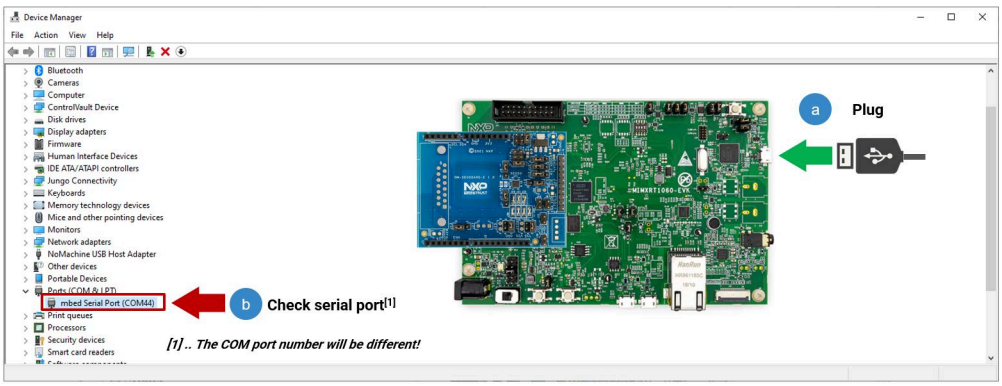


Figure 8. Identify the serial port

4 Import project examples from board SDK

This section explains how to run EdgeLock SE05x projects by importing them from the i.MX RT1060 or i.MX RT1170 SDKs. This is the most recommended option since it implies that the MCU projects are self-contained standard MCUxpresso projects providing a better debug experience. If you are an expert user or you need to compile the example for different OSs, please use the CMake build system to import and compile examples as described in Section 5.

4.1 Prerequisites

The following software tools are required to run a project imported from the board 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 the board SDK

The project examples for EdgeLock SE05x are included in the i.MX RT1060 and i.MX RT1170 SDKs. First, download the SDK for your board from the EdgeLock SE05x website. The SDKs available from EdgeLock SE05x website contain the most up-to-date and complete list of project examples to evaluate EdgeLock SE05x features.

Note: *The i.MX RT1060 SDK or i.MX RT1170 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 the board SDK

After downloading the SDK for your board, you need to install it in MCUXpresso. To install the SDK, (1) drag and drop the SDK zip file in the ***Installed SDKs*** section in the bottom part of the MCUXpresso IDE and (2) click ***OK*** as shown in Figure 9:

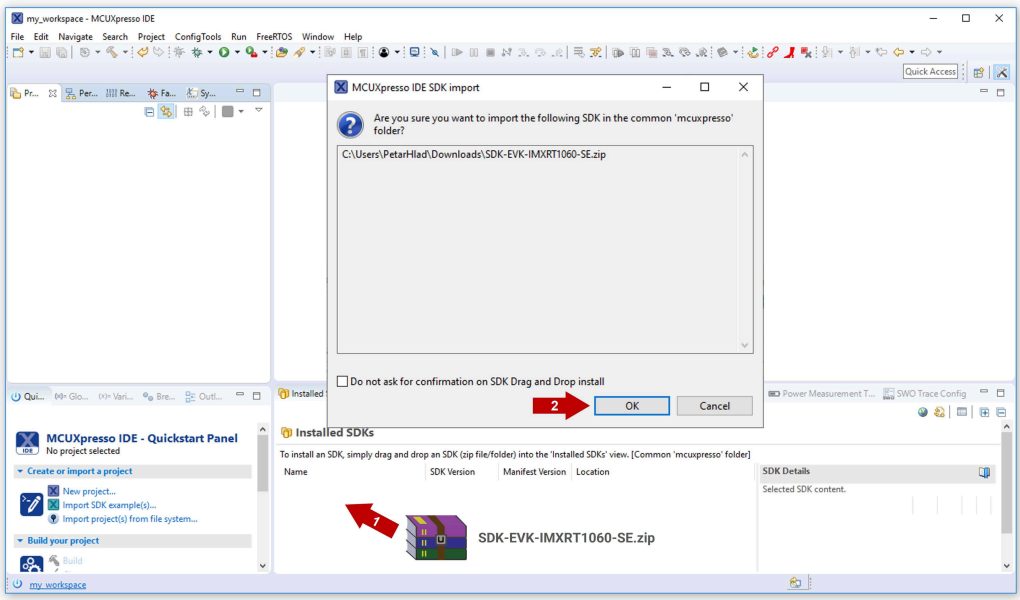


Figure 9. Import the board SDK into MCUXpresso environment

If the SDK is successfully imported, you should see it listed in the ***Installed SDK*** window as shown in Figure 10:

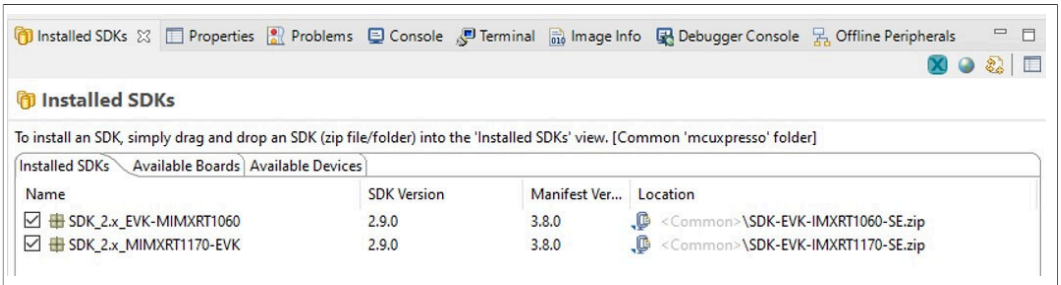


Figure 10. Imported SDKs

4.4 Import a project example in MCUXpresso

After importing the SDK for your board in the MCUXpresso workspace, follow these instructions to import an example project:

1. Click *Import SDK example(s)* in the MCUXpresso IDE quick start panel as shown in Figure 11:

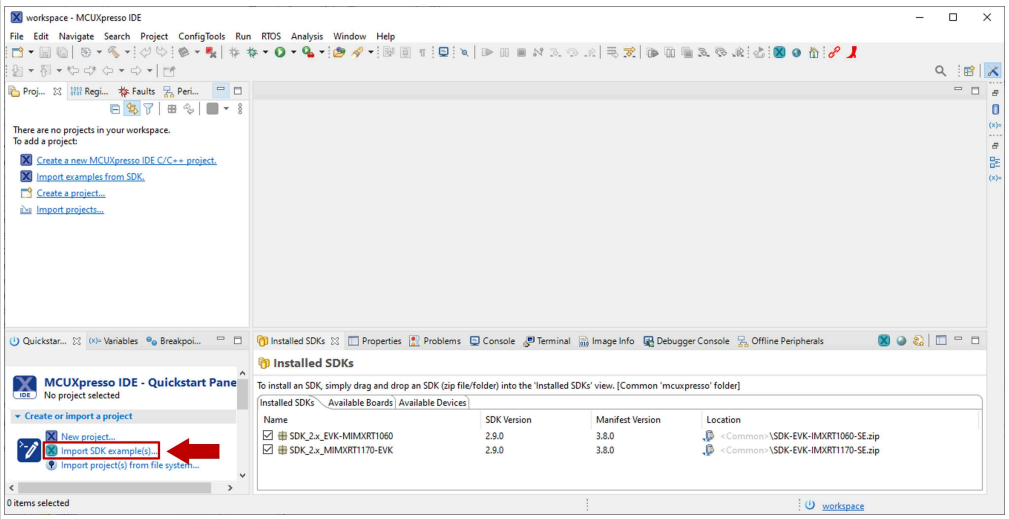


Figure 11. Import projects from SDK

2. The SDK import wizard will open. (1) You should see on the left side the list of all the SDKs imported in MCUXpresso IDE. Select the one corresponding to the board you are using (either i.MX RT1060 or i.MX RT1170). You should now see a figure of an i.MX RT1060 board (or i.MX RT1170 board in case you selected the i.MX RT1170

SDK) with an orange label. (2) Select the board and then (3) click on the *Next* button as shown in Figure 12:

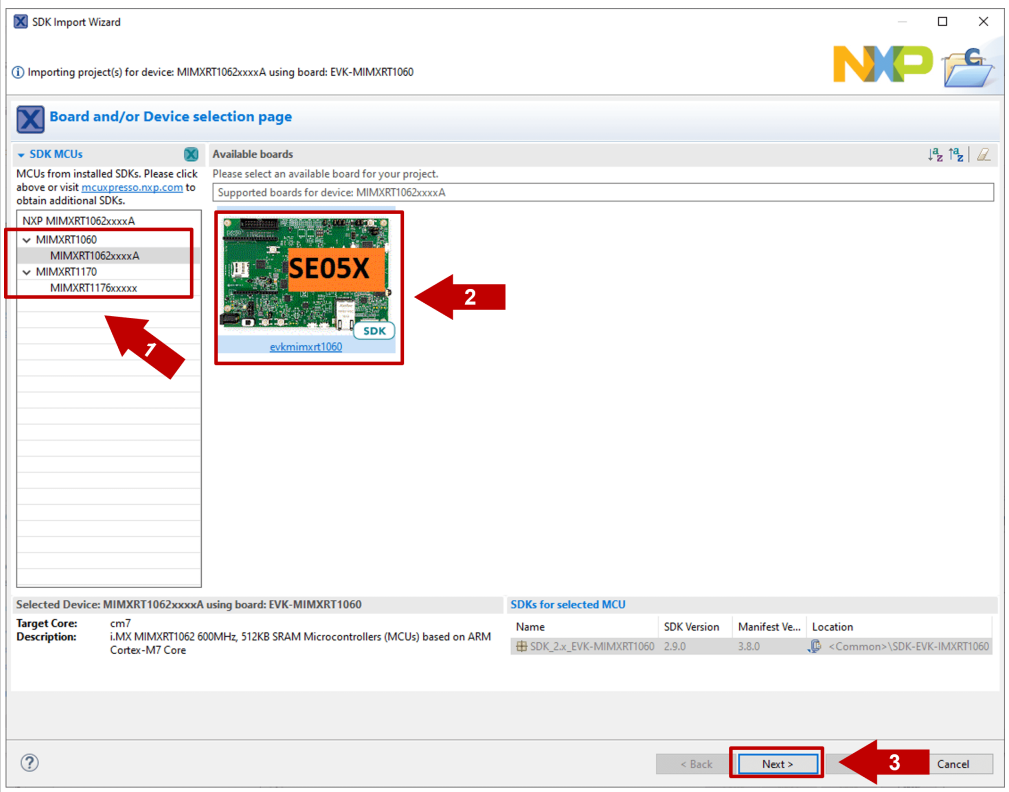


Figure 12. SDK import wizard

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 9*.*

3. Under the `se_hostlib_examples` drop down list, you have the list of supported project examples. Select the examples that you want to import in your MCUXpresso workspace and click on the *Finish* button as shown in Figure 13. In this case, we

selected the `se05x_Minimal` project as an example. The same process can be followed for importing all other examples.

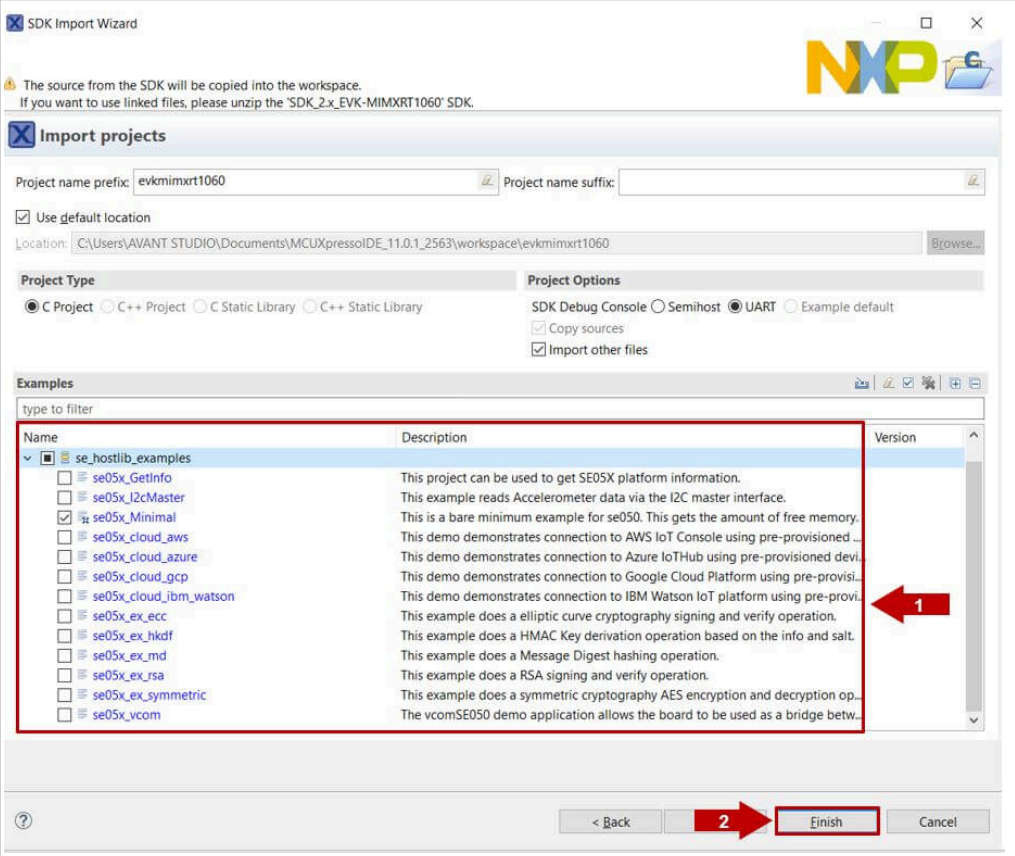


Figure 13. Select projects to import

4. The projects you selected should now be visible in your MCUXpresso workspace as shown in Figure 14:

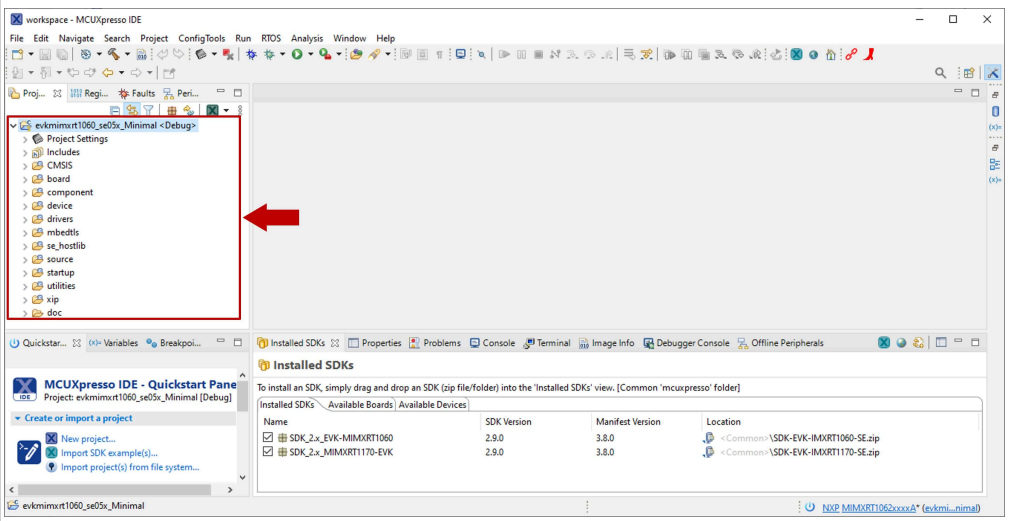


Figure 14. Imported projects in MCUXpresso workspace

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 debug USB connector of your board. Figure 15 shows the connection with i.MX RT1060 board. Figure 16 shows the connection with i.MX RT1170 board.

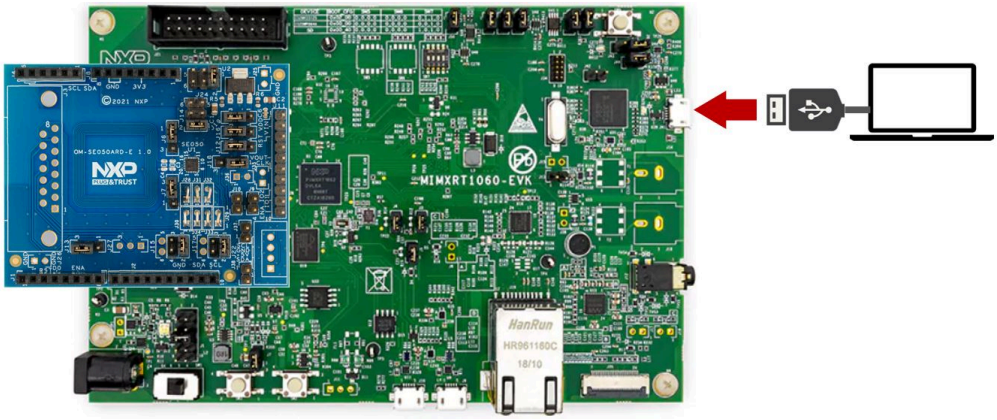


Figure 15. Connect i.MX RT1060 board to the laptop

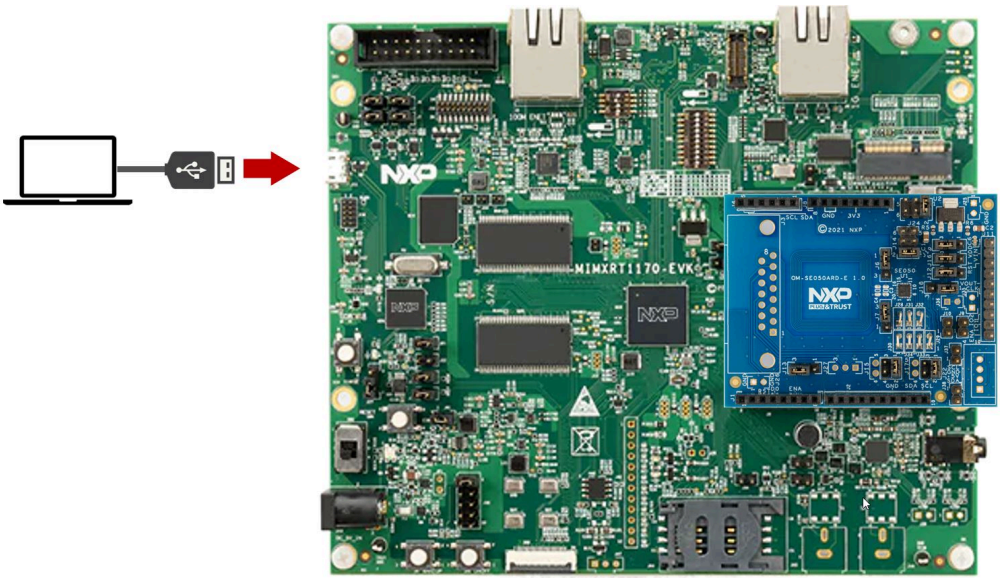


Figure 16. Connect i.MX RT1170 board to the laptop

2. Launch and configure the TeraTerm application as shown in Figure 17:
   a. Click *Serial* option and select from the drop down list the COM port number assigned to your 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.

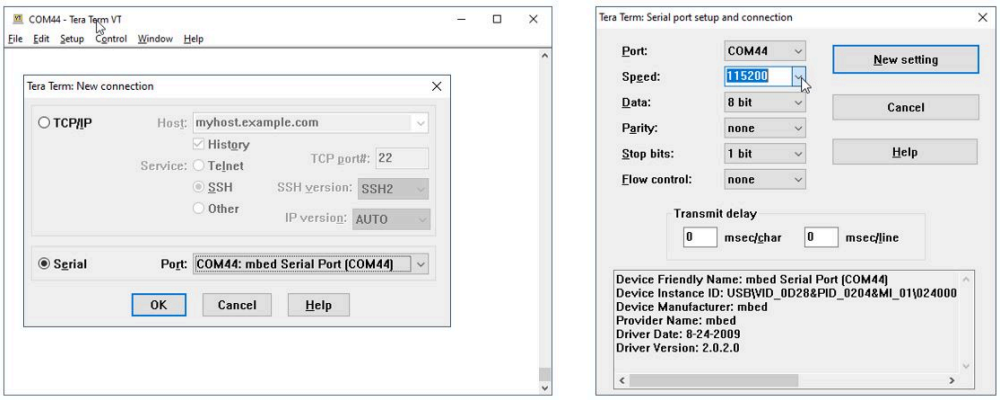


Figure 17. TeraTerm setup

3. **Note:** The default build configuration of the Plug & Trust middleware ≥ `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 on the *Build* button as shown in Figure 18. Wait a few seconds and check that the build process has finished successfully in the MCUXpresso console window.

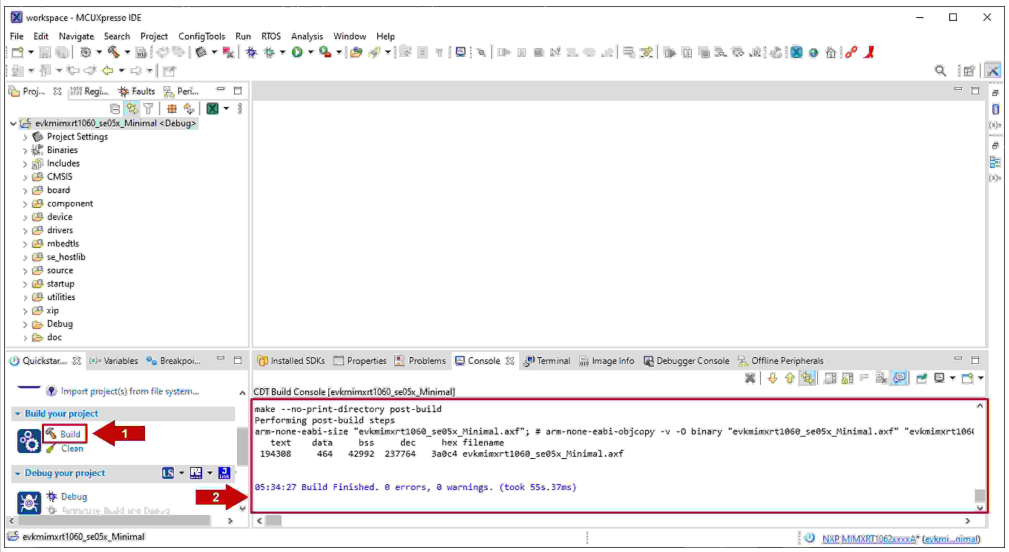


Figure 18. Build projects in MCUXpresso workspace

5. Go to the MCUXpresso Quickstart Panel and click on the *Debug* button as shown in Figure 19. 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.

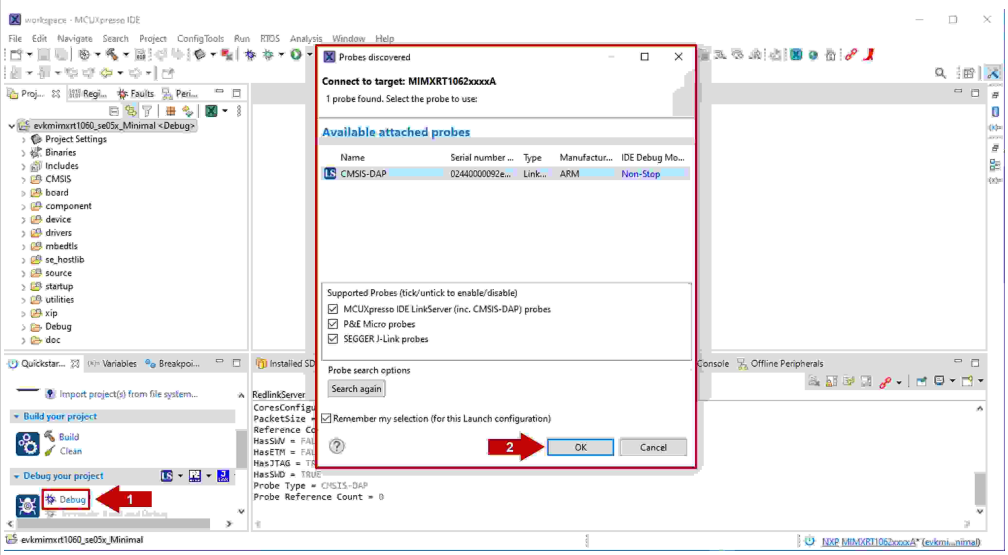


Figure 19. Debug projects in MCUXpresso workspace

6. When the example executes, it will automatically stop in a breakpoint. Click on *Resume* to allow the software to continue its execution as shown in Figure 20:

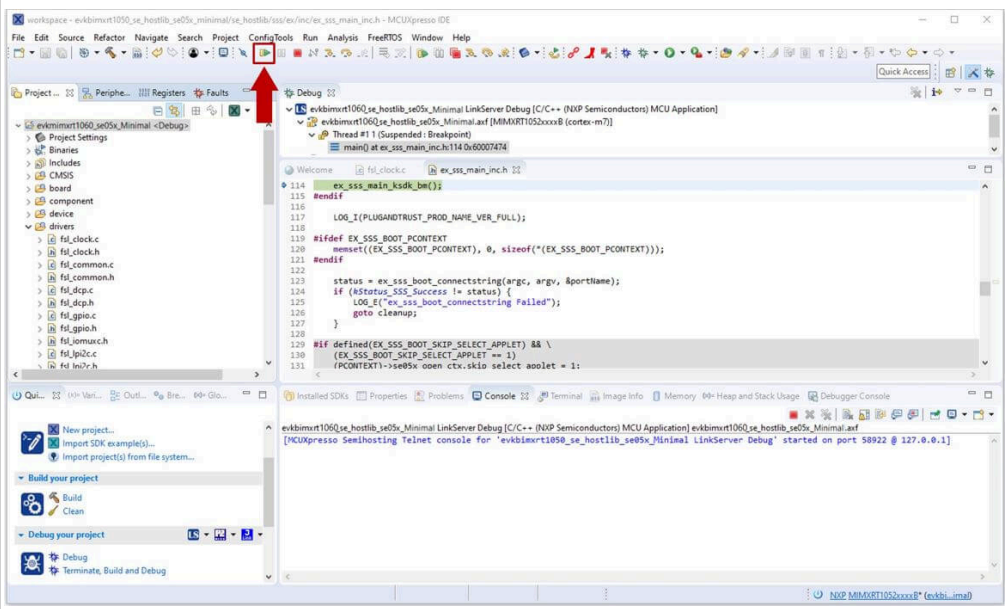


Figure 20. Run projects in MCUXpresso workspace

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, 32767) as can be seen in Figure 21.
The same operations can be repeated to run any of the other Plug & Trust middleware project examples.

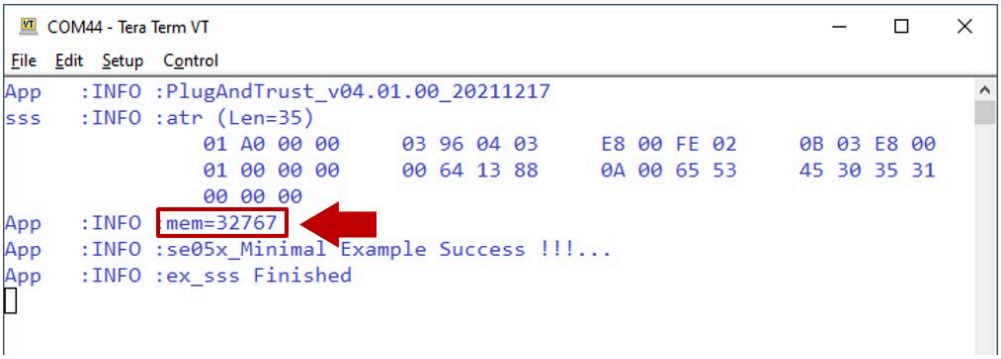


Figure 21. TeraTerm logs - `se05x_Minimal` project example

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.

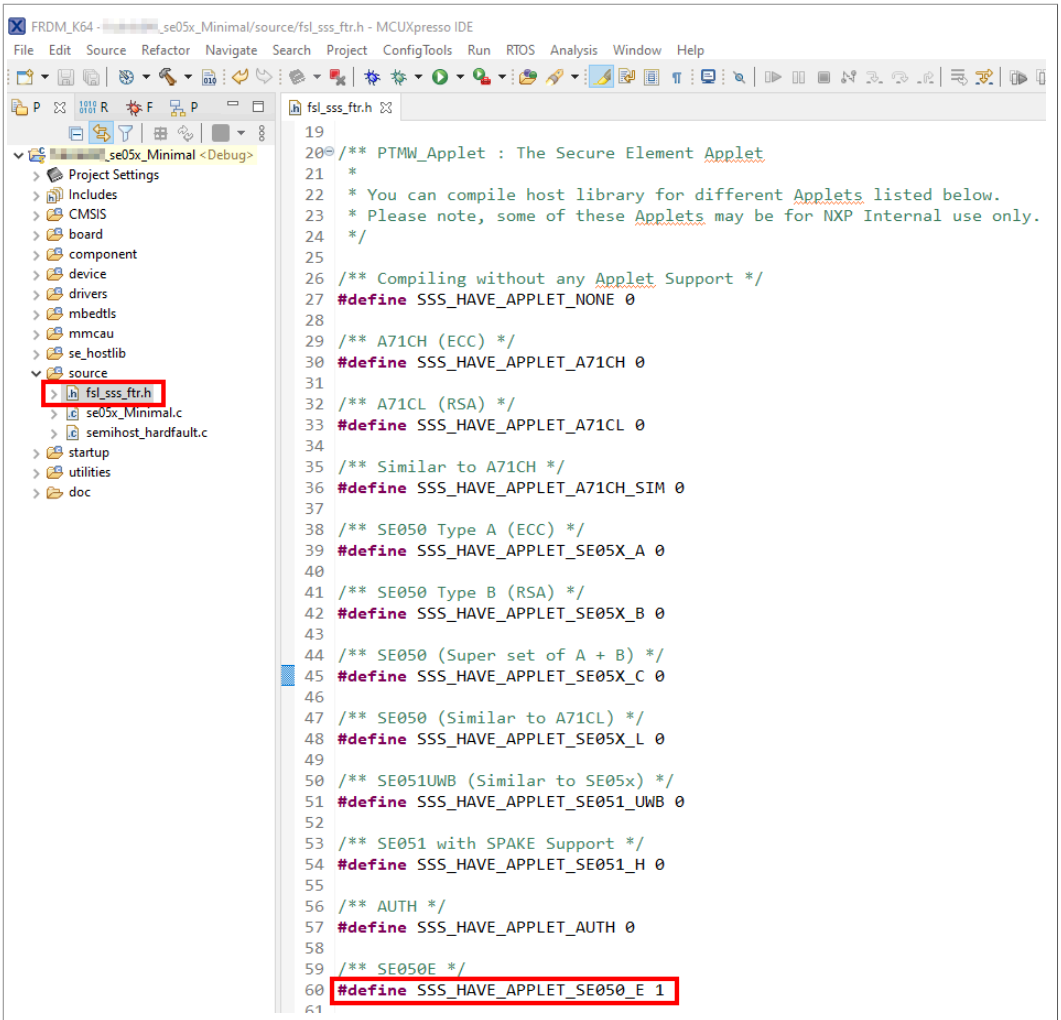


Figure 22. Feature file fsl_sss_ftr.h example: Assign the value SE05X_E to the CMake option PTWM_Applet

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 4. 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_ SCP_NONE or SSS_ HAVE_ SCP_ SCP03_ SSS	disabled
SE050E2	A921						

Table 5. 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_ SCP03_ SSS	enabled
SE050F2	A92A						

Table 6. 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_ SCP03_ 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_ SCP03_ 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_ SCP03_ SSS	enabled
SE050C2	A201						
SE050 Dev Board OM-SE050ARD	A1F4						

Table 6. 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_ SCP	SSSFTR_ SE05X_ RSA
SE050F2	A77E[1]	`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_SCP03_SSS`	enabled

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

Table 7. 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_ SCP	SSSFTR_ SE05X_ RSA
SE051A2	A920	`SSS_HAVE_APPLET_SE05X_A`	`SSS_HAVE_FIPS_NONE`	`SSS_HAVE_SE05X_VER_07_02`	any option	`SSS_HAVE_SCP_NONE` or `SSS_HAVE_SCP_SCP03_SSS`	disabled
SE051C2	A8FA	`SSS_HAVE_APPLET_SE05X_C`	`SSS_HAVE_FIPS_NONE`	`SSS_HAVE_SE05X_VER_07_02`	any option	`SSS_HAVE_SCP_NONE` or `SSS_HAVE_SCP_SCP03_SSS`	enabled
SE051W2	A739	`SSS_HAVE_APPLET_SE05X_C`	`SSS_HAVE_FIPS_NONE`	`SSS_HAVE_SE05X_VER_07_02`	any option	`SSS_HAVE_SCP_NONE` or `SSS_HAVE_SCP_SCP03_SSS`	enabled

Table 7. 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_ SCP	SSSFTR_ SE05X_ RSA
SE051A2	A565	`SSS_HAVE_APPLET_SE05X_A`	`SSS_HAVE_FIPS_NONE`	`SSS_HAVE_SE05X_VER_06_00`	any option	`SSS_HAVE_SCP_NONE` or `SSS_HAVE_SCP_SCP03_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_SCP_NONE` or `SSS_HAVE_SCP_SCP03_SSS`	enabled

Table 8. 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_ SCP	SSSFTR_ SE05X_ RSA
OM-A5000ARD	A736	`SSS_HAVE_APPLET_AUTH`	`SSS_HAVE_FIPS_NONE`	`SSS_HAVE_SE05X_VER_07_02`	any option	`SSS_HAVE_SCP_NONE` or `SSS_HAVE_SCP_SCP03_SSS`	disabled
A5000	A736						

4.6.1 Example: SE050E build settings

The following images show the configuration for the SE050E development board OM-SE05ARD-E according to Table 4.

1. Select the Applet variant SE050E.

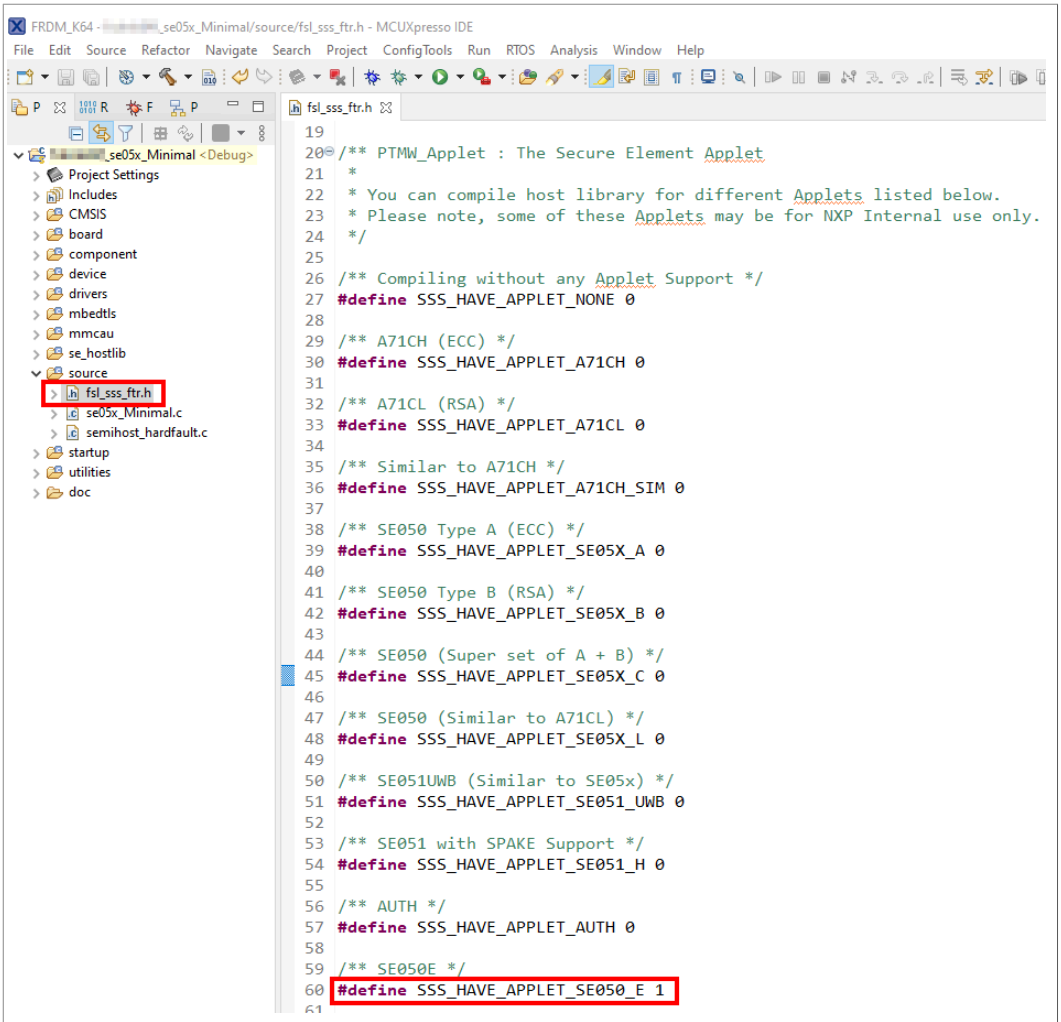


Figure 23. Feature file fsl_sss_ftr.h - PTMW_Applet

2. Select FIPS none.

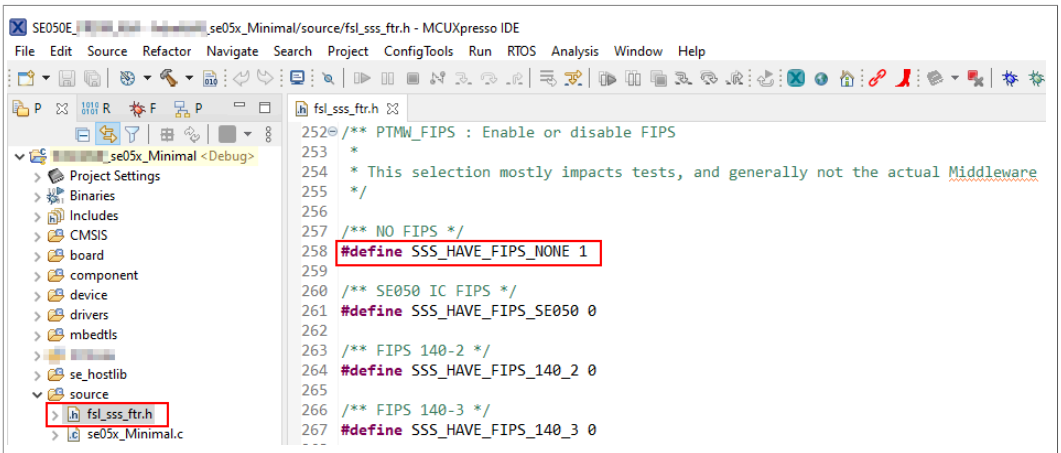


Figure 24. Feature file fsl_sss_ftr.h - Option PTMW_FIPS

3. Select Applet version 7.02.

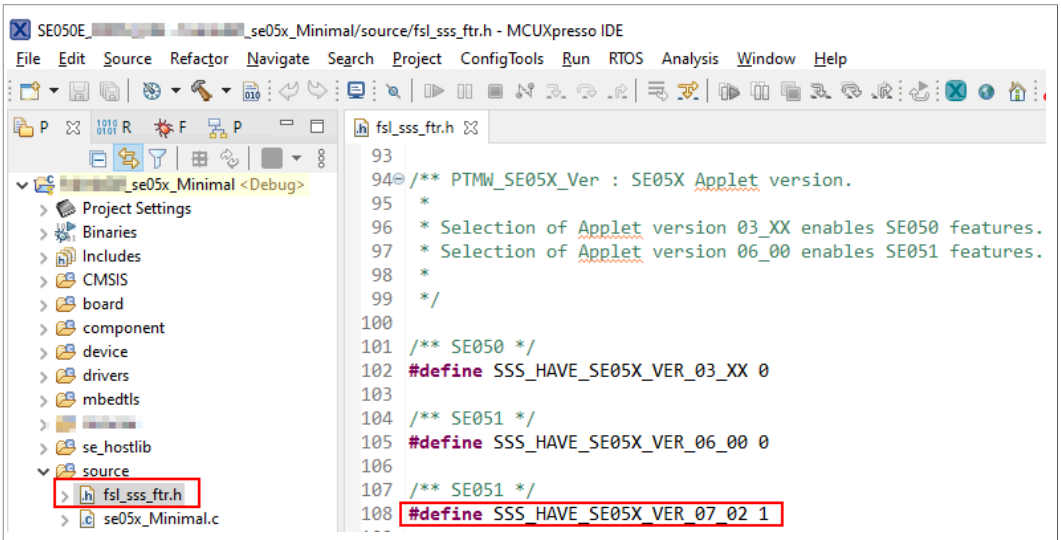


Figure 25. Feature file fsl_sss_ftr.h - Option PTMW_SE05x_Ver

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

How to enable Platform SCP is described in Section 6.3.

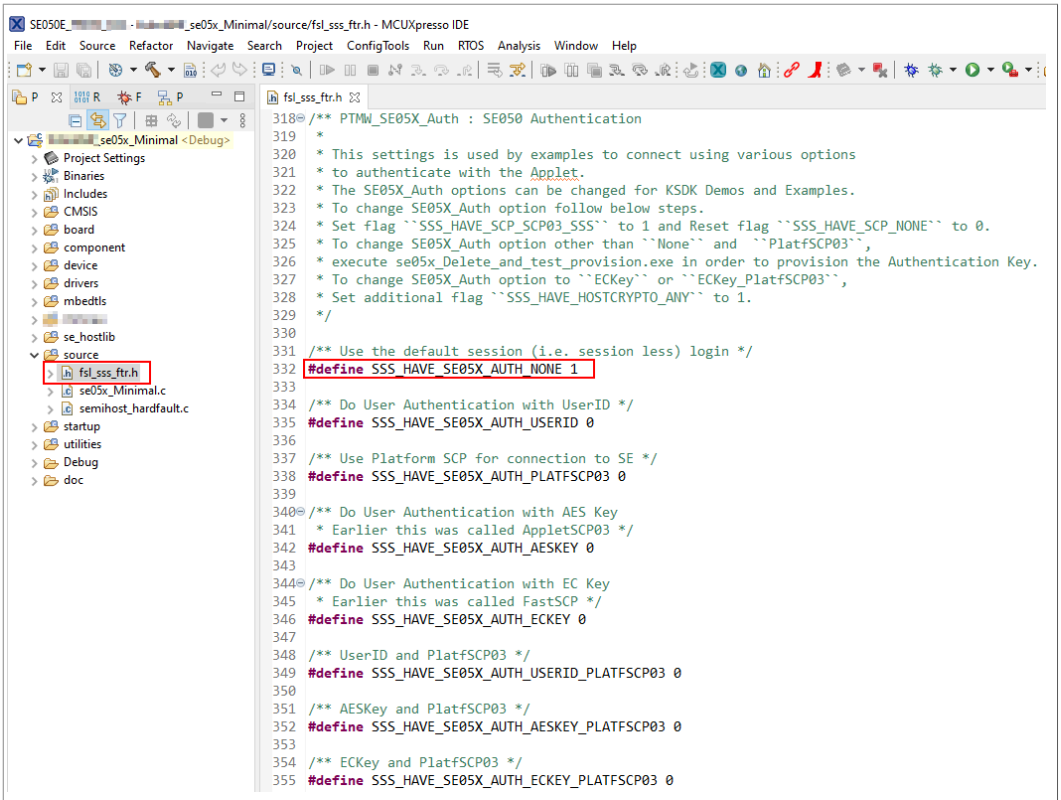


Figure 26. Feature file fsl_sss_ftr.h - Option PTMW_AUTH - Plain communication

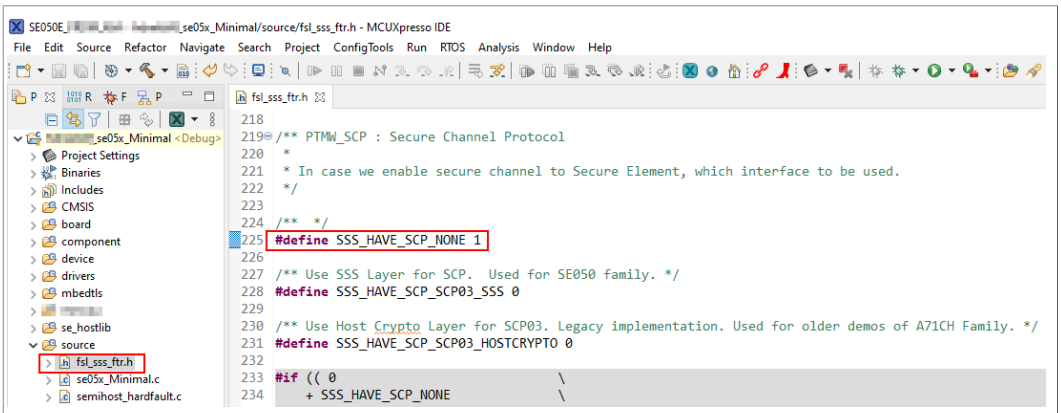


Figure 27. Feature file fsl_sss_ftr.h - Option PTMW_SCP - Plain communication

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

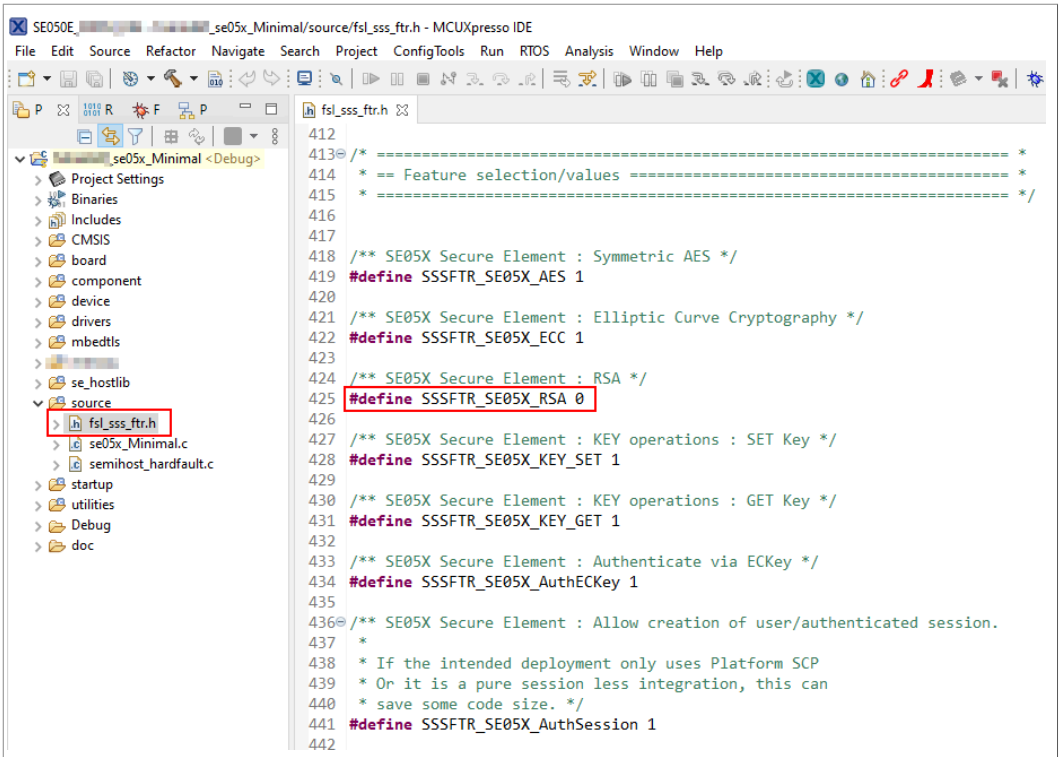


Figure 28. Feature file fsl_sss_ftr.h - Option SSSFTR_SE05X_RSA

5 Import project examples from CMake-based build system

This section explains how to run EdgeLock SE05x project examples using the CMake-based build system. Although this solution 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 software 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.
   Note: *higher Python versions may work as well.*
4. TeraTerm (or an equivalent serial application). You can download and run the TeraTerm installer from this link.

5.2 Download the Plug & Trust middleware

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

1. Download Plug & Trust middleware from the NXP website.

2. Create a folder called ***se05x_middleware*** in *C:* directory as shown in Figure 29:

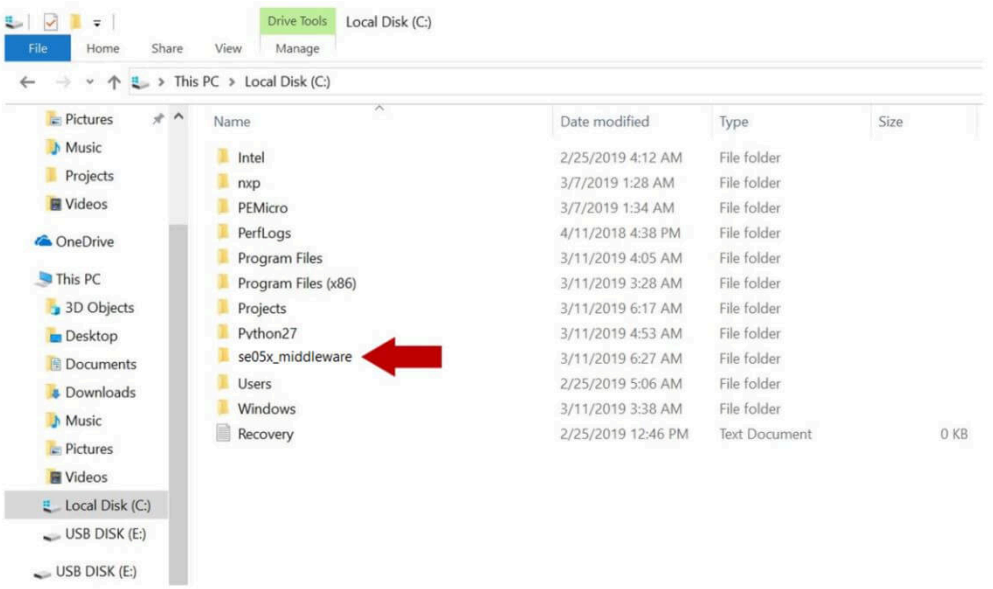


Figure 29. Create se05x_middleware folder

3. Unzip the Plug & Trust middleware inside the *se05x_middleware* folder. After unzipping, you will see a folder called ***simw-top***. The contents of the ***simw-top*** directory should look as shown in Figure 30:

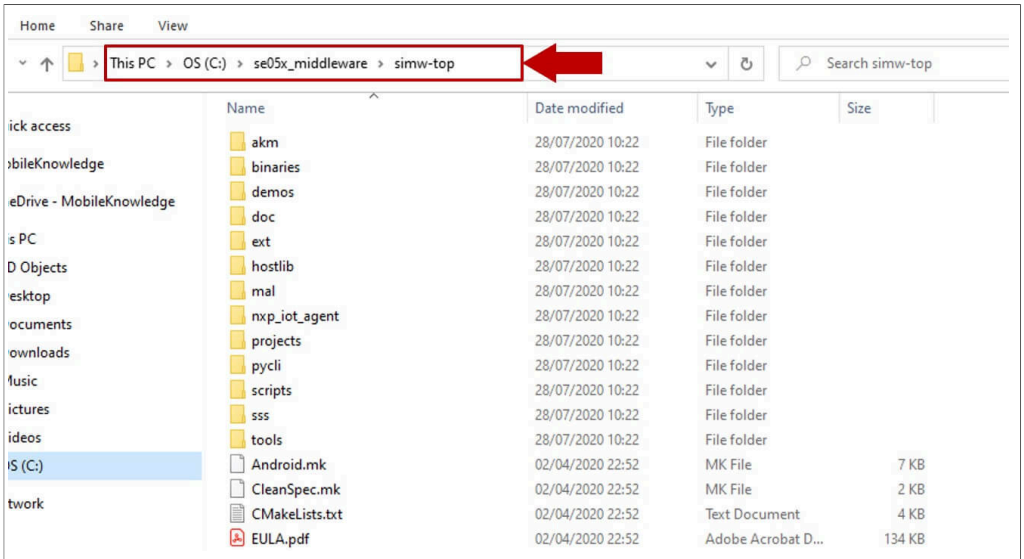


Figure 30. Unzip se05x middleware

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 the Plug & Trust middleware project examples

The Plug & Trust middleware uses CMake for building the project examples. To build the Plug & Trust middleware, open a Command Prompt and use the following steps as shown in Figure 31:

1. Go to the folder where you unzipped the Plug & Trust middleware:
   (1) Send >> `cd C:\se05x_middleware\simw-top\scripts`
2. Define the environment:
   (2) Send >> `env_setup.bat`
3. Generate the Plug & Trust middleware project examples:
   (3) Send >> `create_cmake_projects.py`
   ***Note**: This command may take a few seconds to complete.*

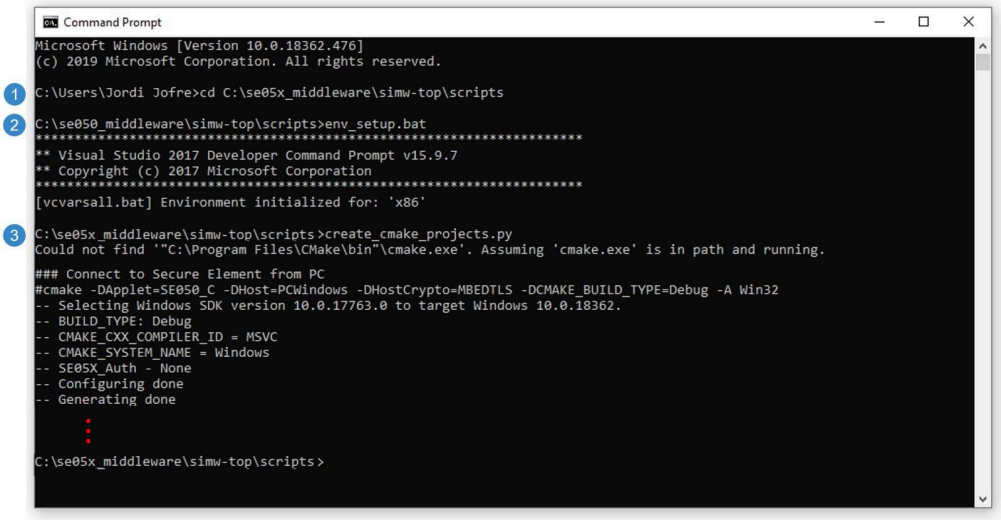


Figure 31. Generate Plug & Trust middleware project examples

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 32:

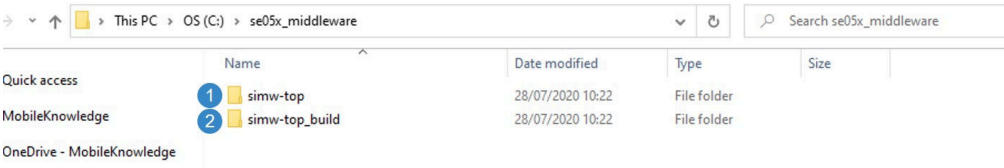


Figure 32. SE05x middleware project structure

5.4 Import *PlugAndTrustMW* project example in MCUXpresso workspace

After generating the projects in your local machine using the `create_cmake_projects.py` script, you need to import the *PlugAndTrustMW* project in your MCUXpresso workspace. Follow these steps to import the project:

1. Go to *File* → *Import* using the top bar menu as shown in Figure 33.
   ***Note**: In this case, do not use the MCUXpresso Quickstart Panel to import the project.*

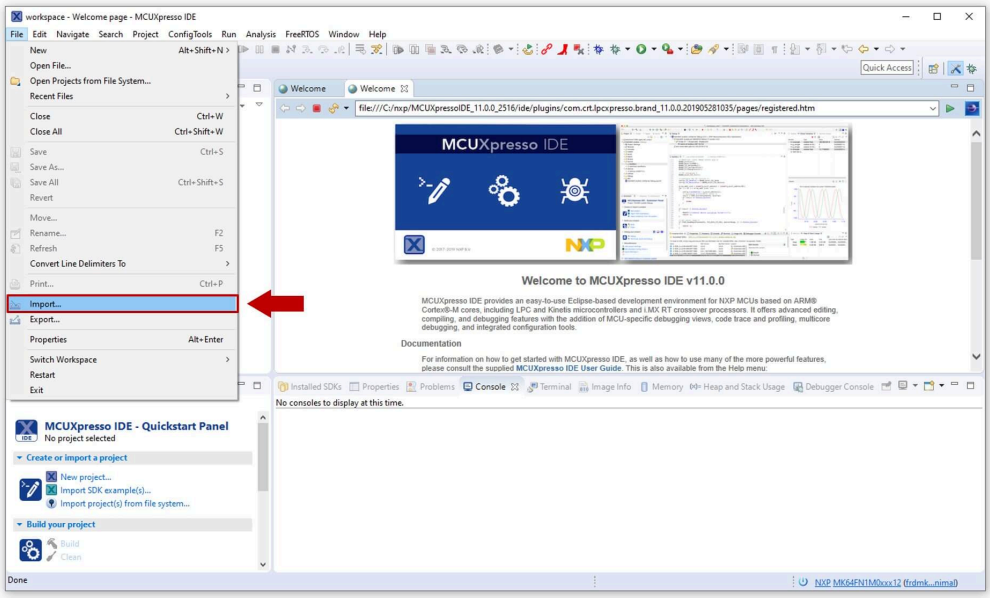


Figure 33. Import a project

2. In the import wizard menu, (1) select *Existing Projects into Workspace* from the *General* folder, then (2) click on *Next* as shown in Figure 34:

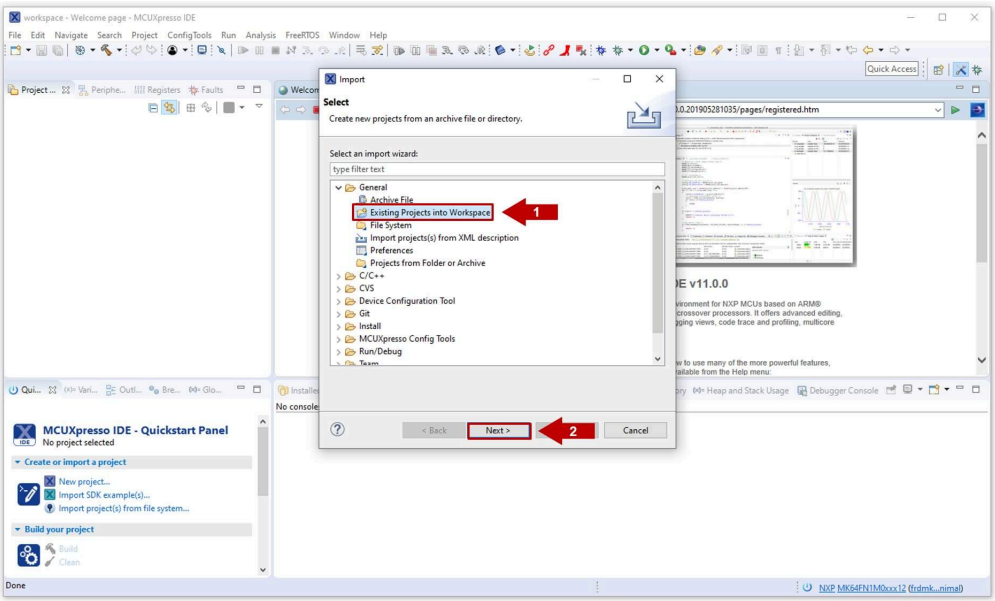


Figure 34. Import a project (II)

3. First, you need to import Plug & Trust middleware project in MCUXpresso. For that, in the *Select root directory* option, (1, 2) browse to the location of your Plug & Trust

middleware directory (in this case *C:\se05x_middleware\simw-top_build*) and (3) click on *Select folder* as shown in Figure 35:

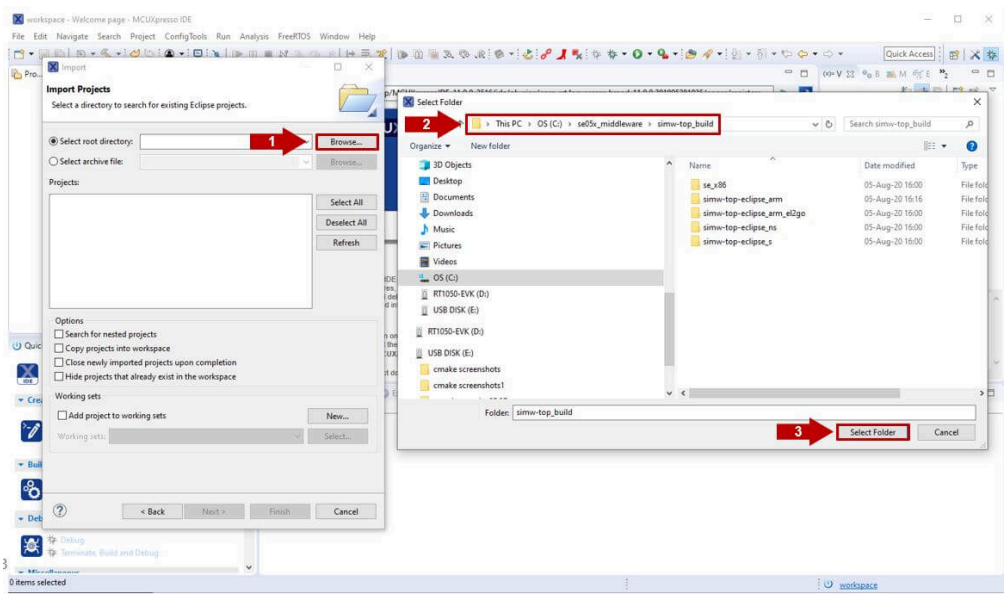


Figure 35. Select Plug & Trust middleware build folder

4. 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. (1) Select the project and (2) click on the *Finish* button to import this project into your workspace as shown in Figure 36:

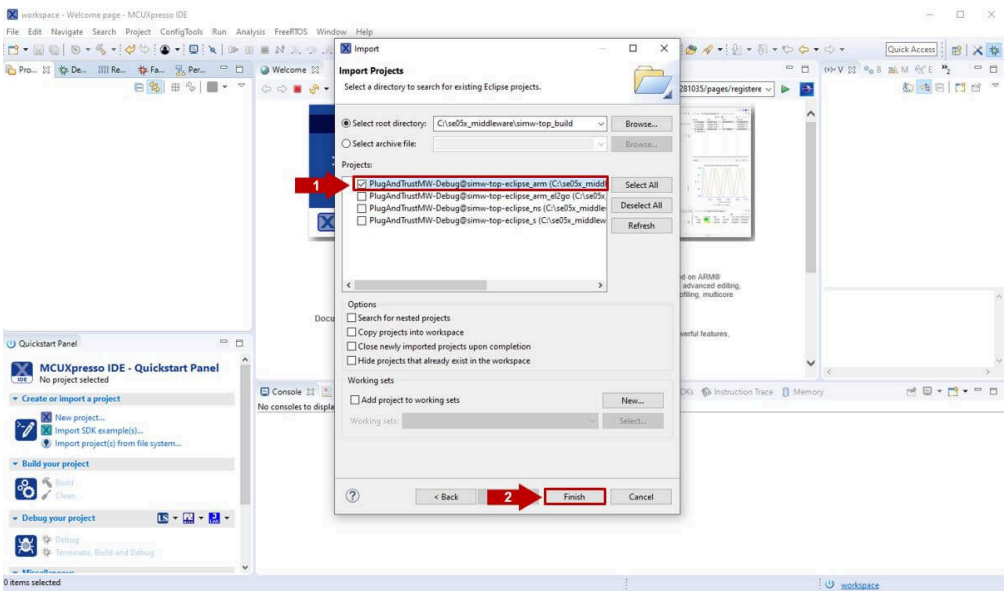


Figure 36. Import Plug & Trust middleware

5. The *PlugAndTrustMW* project should now be imported in your workspace as shown in Figure 37:

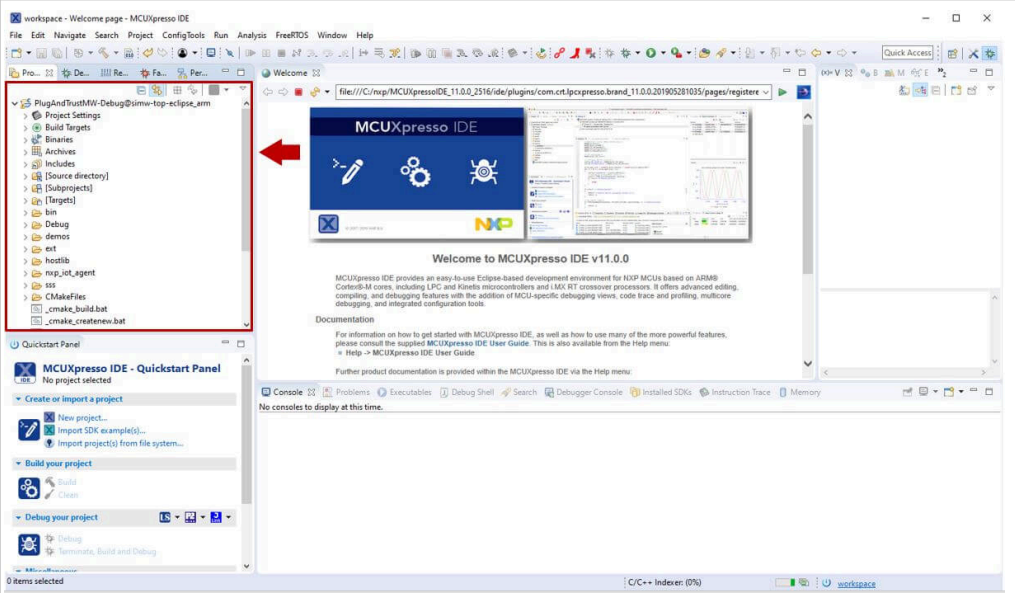


Figure 37. Plug & Trust middleware imported in workspace

5.5 Import board project example in MCUXpresso workspace

After importing the *PlugAndTrustMW* project example in MCUXpresso, you need to import the *cmake_projects_evkbimxrt1060* project (for i.MX RT1060) or the *cmake_projects_evkbimxrt1170* project (for i.MX RT1170). Follow these steps to import the projects:

1. Go to *File* → *Import* using the top bar menu as shown in Figure 38.
   ***Note**: In this case, do not use the MCUXpresso Quickstart Panel to import project.*

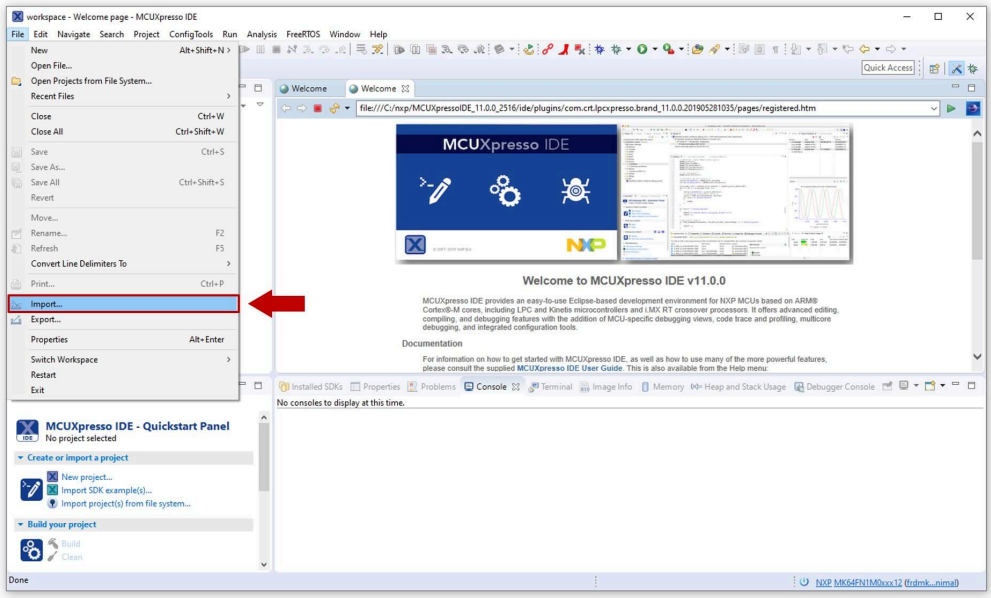


Figure 38. Import a project

2. In the import wizard menu, (1) select *Existing Projects into Workspace* from the *General* folder, then (2) click on *Next* as shown in Figure 39:

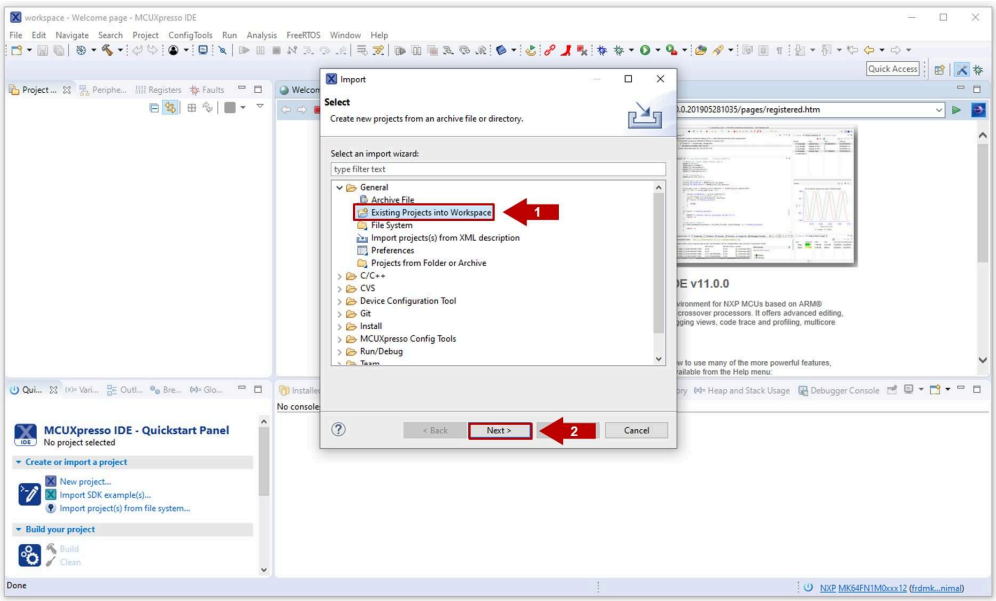


Figure 39. Import a project (II)

3. In the *Select root directory* option, (1, 2) browse to the location of your i.MX RT1060/1170 projects directory (in our case *C:\se05x_middleware\simw-top\projects*), then (3) select the *cmake_projects_evkbimxrt1060* if you are using the i.MX RT1060

board or the *cmake_projects_evkbimxrt1170* project if you are using the i.MX RT1170 board. Finally, (4) click *Select folder* as shown in Figure 40:

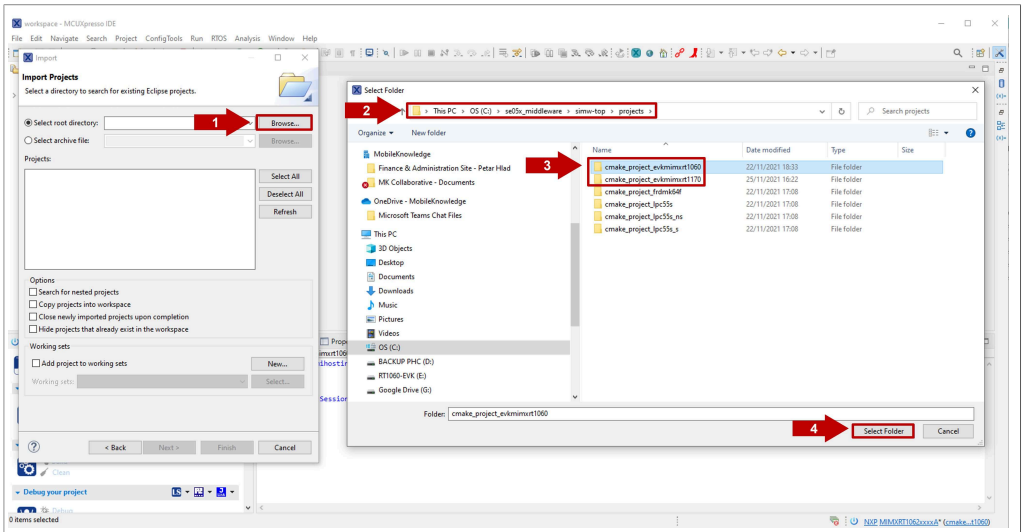


Figure 40. Select project folder for your board

4. After selecting *C:\se05x_middleware\simw-top\projects* folder, the *cmake_projects_evkbimxrt1060* (or *cmake_projects_evkbimxrt1170*) project should be visible in the *Projects* area. Click on the *Finish* button to import this project into your workspace as shown in Figure 41:

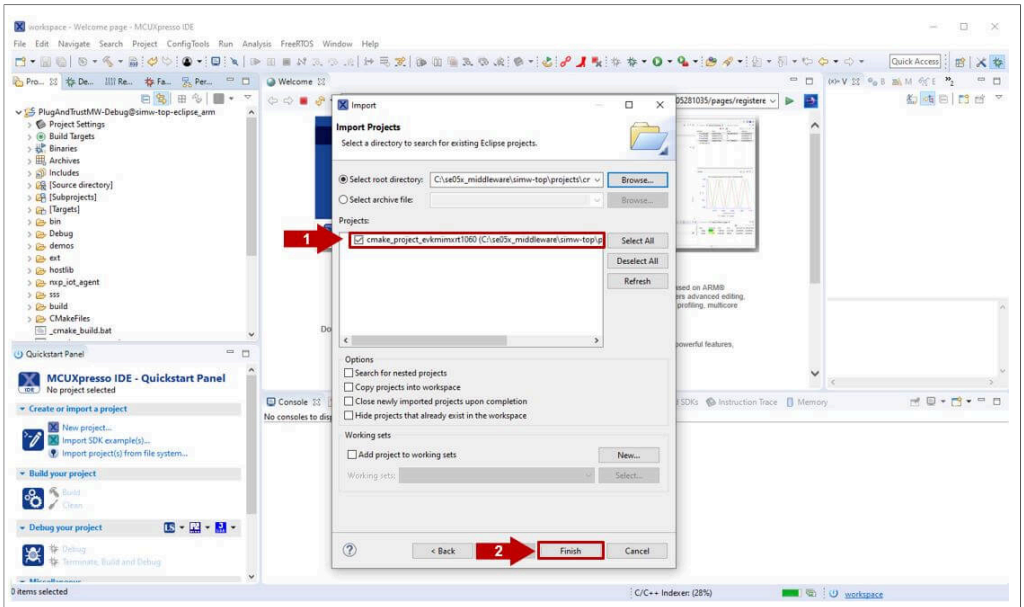


Figure 41. Import project for your board in the workspace

5. Both The *PlugAndTrustMW* and *cmake_projects_evkbimxrt1060* (or *cmake_projects_evkbimxrt1170*) projects should now be imported in your workspace as shown in Figure 42:

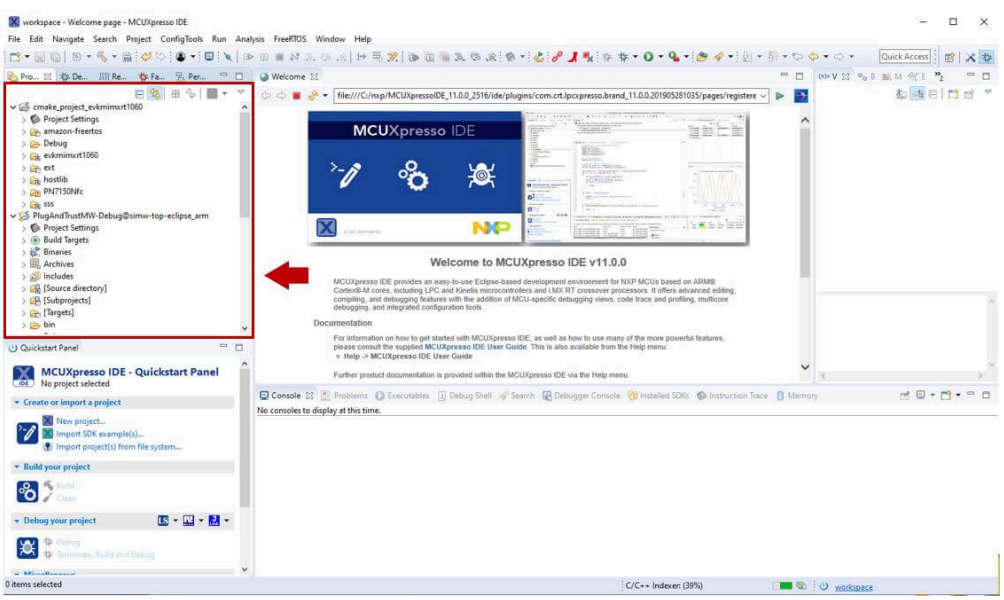


Figure 42. Board project imported in workspace

The two projects need to be imported in the same MCUXpresso workspace. The `cmake_project_evkmimxrt1060` (or `cmake_project_evkmimxrt1170`) 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, by 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 on how to execute the project examples.

5.6 Execute Plug & Trust middleware examples

This section explains how to:

- List the Plug & Trust middleware examples.
- Edit Plug & Trust middleware CMake options.
- Execute a Plug & Trust middleware example.

5.6.1 List the Plug & Trust middleware examples

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

1. Click on the arrow next to the *hammer* icon in the top bar menu of 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 Plug & Trust middleware examples which can be compiled with the currently chosen CMake settings (see Figure 43).

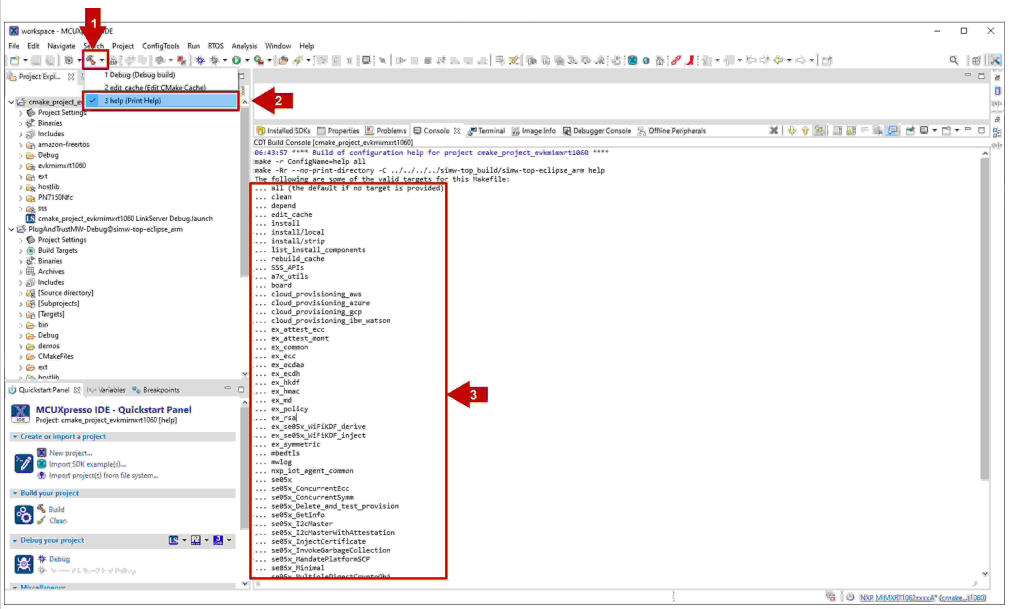


Figure 43. List Plug & Trust middleware examples

5.6.2 Edit Plug & Trust middleware example CMake options.

The 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 Plug & Trust middleware ≥ `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 the steps shown in Figure 44:

1. Click on the arrow next to the *hammer* icon in the top bar menu of MCUXpresso.
2. Select ***2 edit_cache (Edit CMake Cache)***.
3. The CMake GUI window will open in your laptop. Using this GUI, change your host platform to `evkbimxrt1060` (for i.MX RT1060 board) or to `evkbimxrt1170` (for i.MX RT1170 board).
4. Click on the **Configure** button.

5. Click on the **Generate** button to apply the configuration, then close the CMake GUI window.

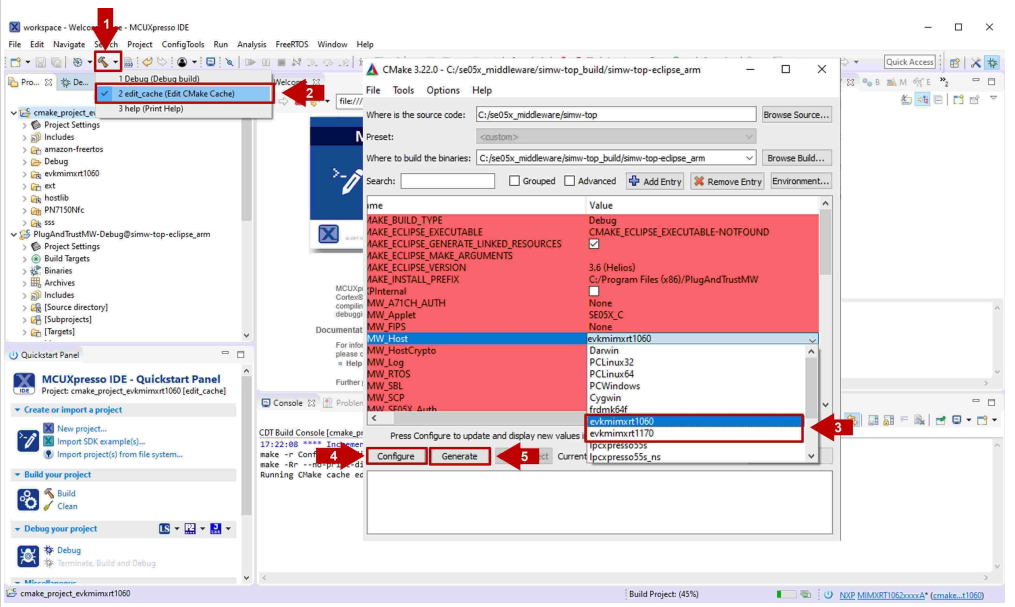


Figure 44. Configure CMake options of Plug & Trust middleware examples.

5.6.3 Build and run a Plug & Trust middleware project example

This section explains how to run the Plug & Trust middleware example called `se05x_Minimal`. The `se05x_Minimal` project outputs the memory left in the secure element.

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 Plug & Trust middleware.*

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

1. Connect the i.MX RT1060 board to your laptop as shown in Figure 45, or your i.MX RT1170 as shown in Figure 46:

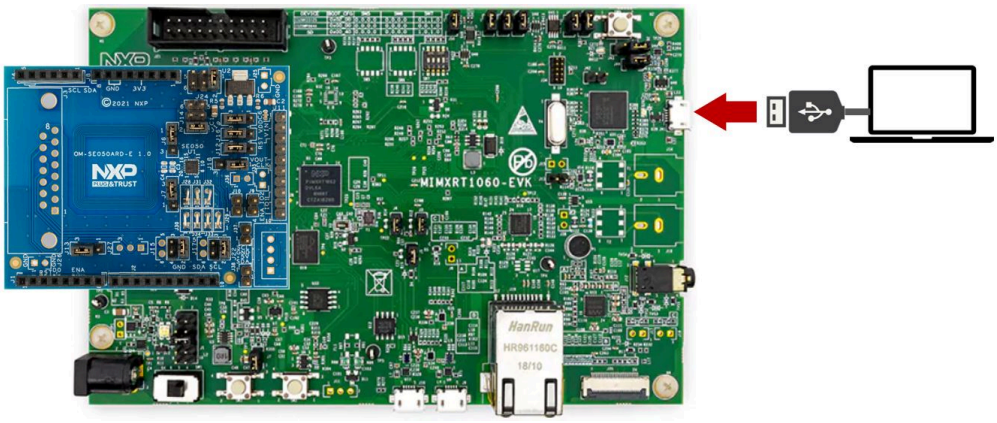


Figure 45. Connect i.MX RT1060 board to the laptop

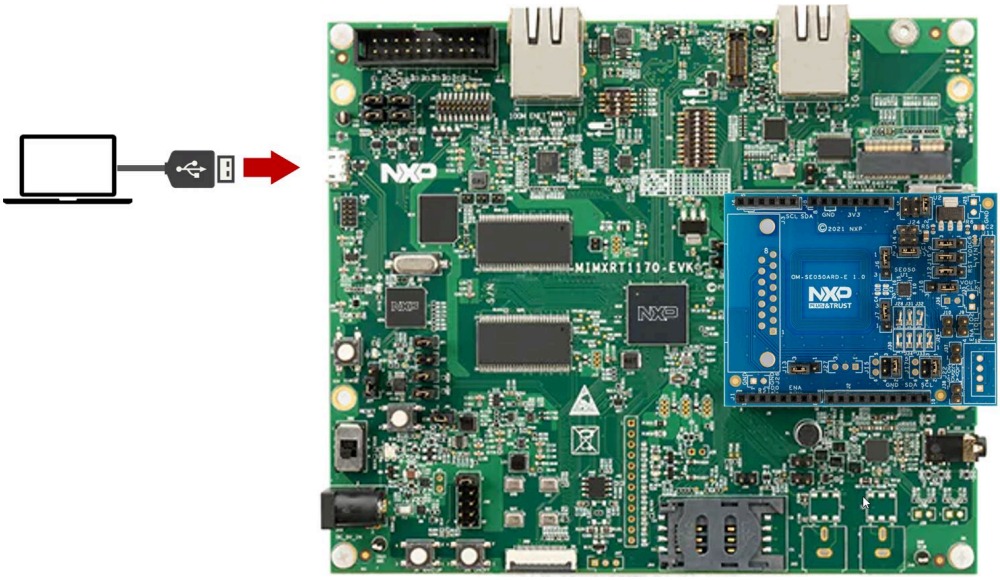


Figure 46. Connect i.MX RT1170 board to the laptop

2. Open TeraTerm. Click ***Serial*** option and select from the drop-down list the COM port number assigned to your i.MX RT1060 (or i.MX RT1170). 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 47:

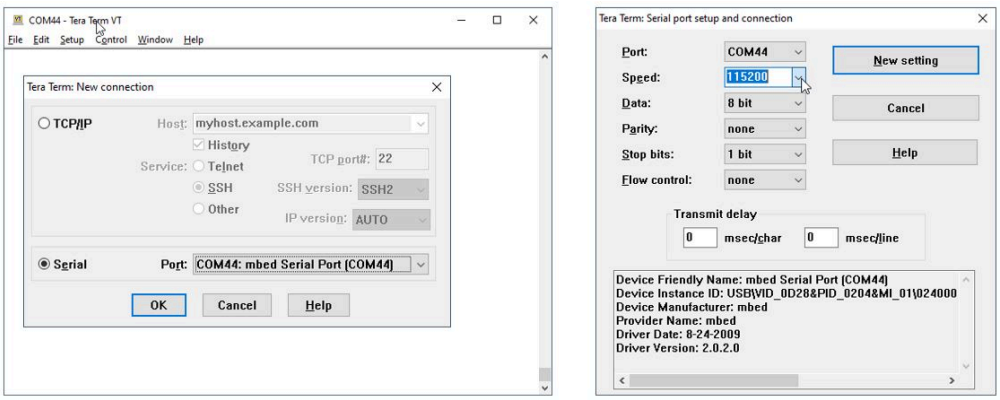


Figure 47. Configure TeraTerm

3. Select the `se05x_Minimal` as the project to be executed. For that, follow the steps shown in Figure 48:
   a. In the Project Explorer window, go to the ***Debug*** folder and open the ***Makefile*** file (under *cmake_project_evkbimxrt1060* or *cmake_project_evkbimxrt1170*).
   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 MCUXpresso.
   d. Select ***1 Debug (Debug build)***. Wait a few seconds until the build operation completes.

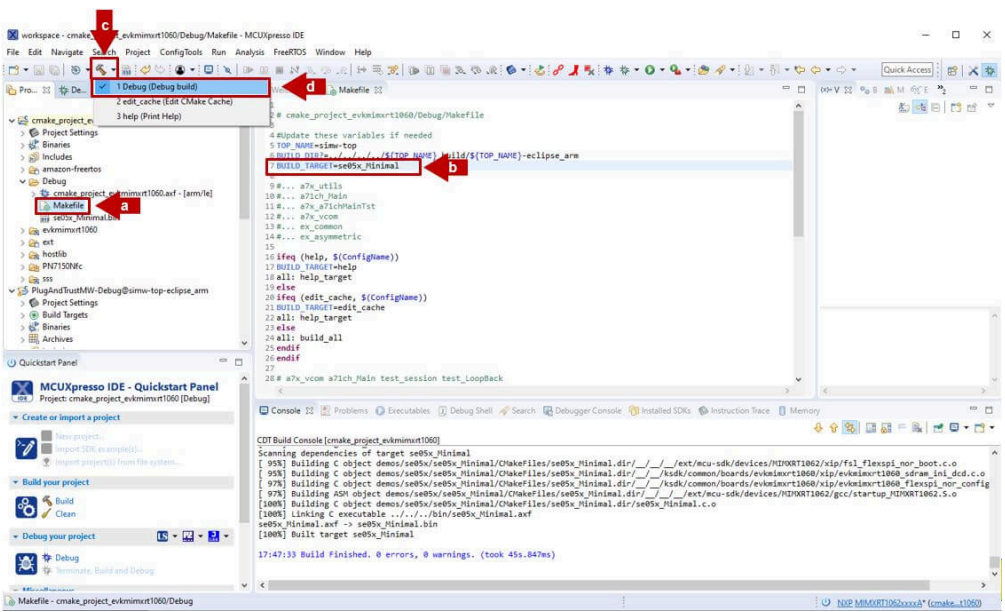


Figure 48. Debug build Plug & Trust middleware `se05x_minimal` project example

4. Go to the MCUXpresso Quickstart Panel and (1) click on the ***Debug*** button as shown in Figure 49. If there is more than one probe attached, you have to (2) select the CMSIS-DAP debug probe from the list. Wait a few seconds until the project executes:

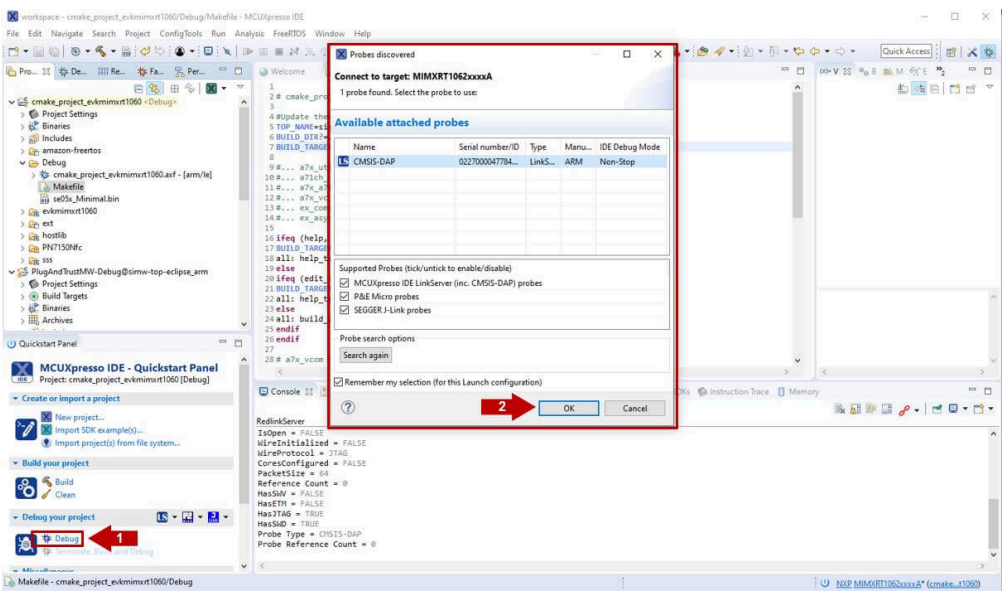


Figure 49. Debug `se05x_minimal` project example

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 50.

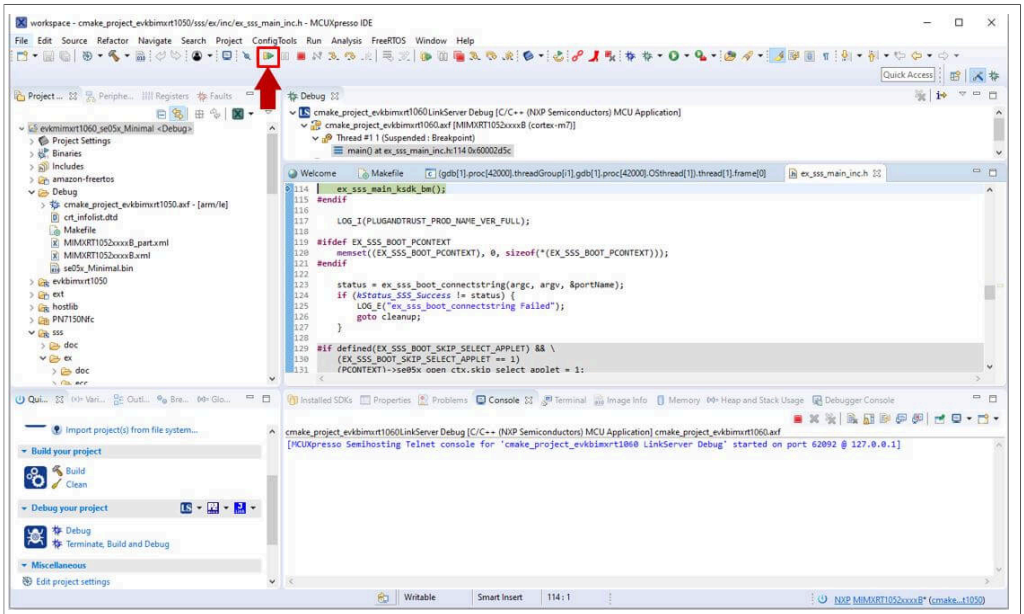


Figure 50. Resume `se05x_minimal` project example

6. The project example should now be running in your board. If it is running successfully, the TeraTerm logs should indicate the available memory in the secure element (in this case 32767), as can be seen in Figure 51.
   The same operations can be repeated to run any of the other Plug & Trust middleware project examples.

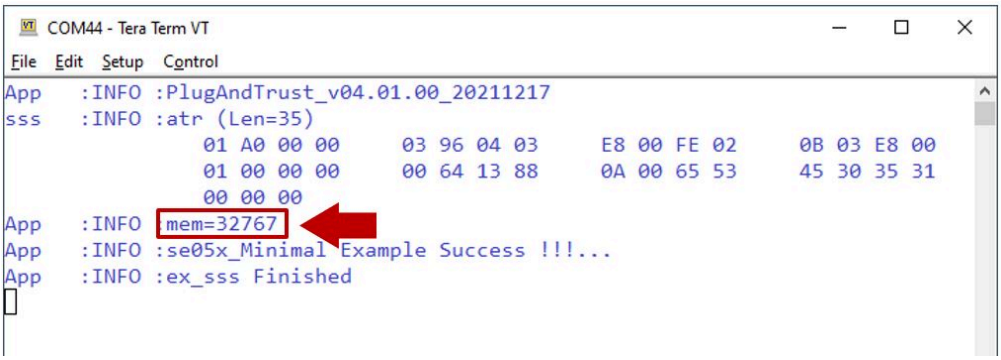


Figure 51. TeraTerm logs - `se05x_minimal` project example

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 9. 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 10. CMake Settings for SE050F product variants

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

Table 11. CMake Settings for SE050 Previous Generation product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050A1	A204	SE05X_A	None	03_XX	any option	None or SCP03_ SSS	disabled
SE050A2	A205						
SE050B1	A202	SE05X_B	None	03_XX	any option	None or SCP03_ SSS	enabled
SE050B2	A203						
SE050C1	A200	SE05X_C	None	03_XX	any option	None or SCP03_ SSS	enabled
SE050C2	A201						
SE050 Dev Board OM-SE050ARD	A1F4						
SE050F2	A77E[1]	SE05X_C	SE050	03_XX	PlatfSCP03 or UserID_PlatfSCP03 or AESKey_PlatfSCP03 or ECKey_PlatfSCP03	SCP03_ SSS	enabled

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

Table 12. CMake Settings for SE051 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE051A2	A920	SE05X_A	None	07_02	any option	None or SCP03_ SSS	disabled

Table 12. CMake Settings for SE051 product variants...*continued*

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	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 13. CMake Settings for A5000 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
OM-A5000ARD	A736	AUTH	None	07_02	any option	None or SCP03_ SSS	disabled
A5000	A736						

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 9.

- 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_SCP.

How to enable Platform SCP is described in Section 6.

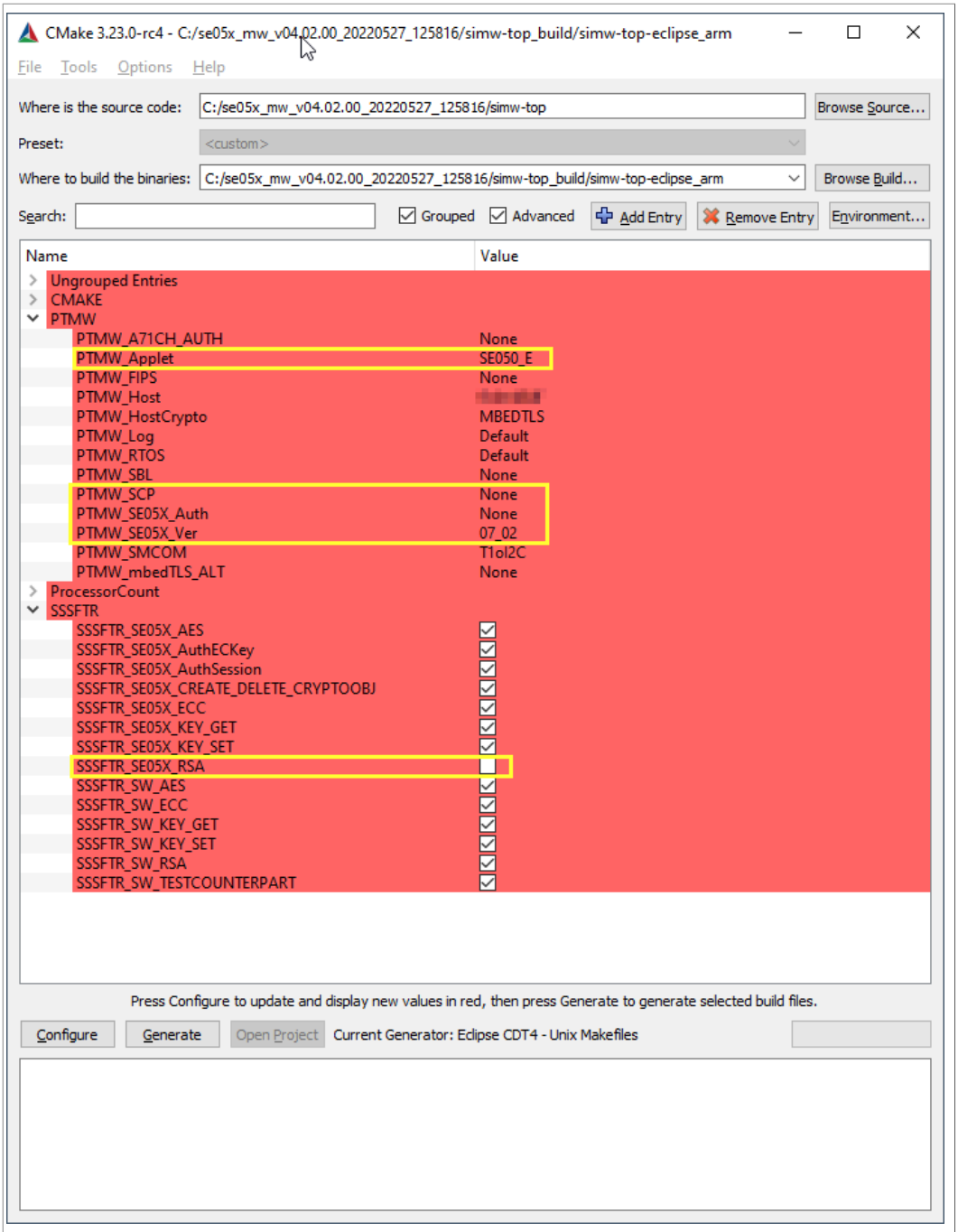


Figure 52. SE050E CMake Settings - Plain communication

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 i.MX RT1060 MCUXpresso SDK
- Section 6.3 How to enable Platform SCP in the i.MX RT1060 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 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 14. Static SCP03 keys

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

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

Table 15. 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.

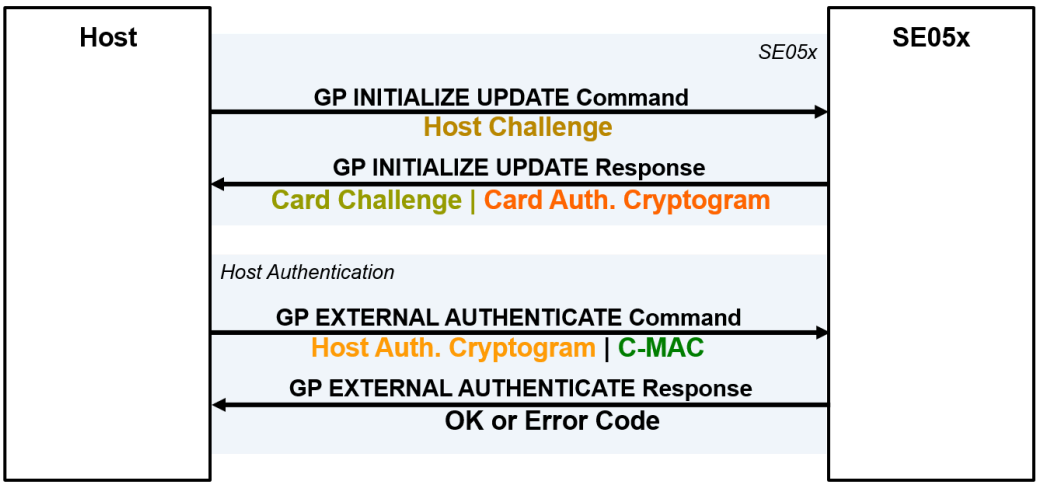


Figure 53. SPC03 mutual authentication – principle

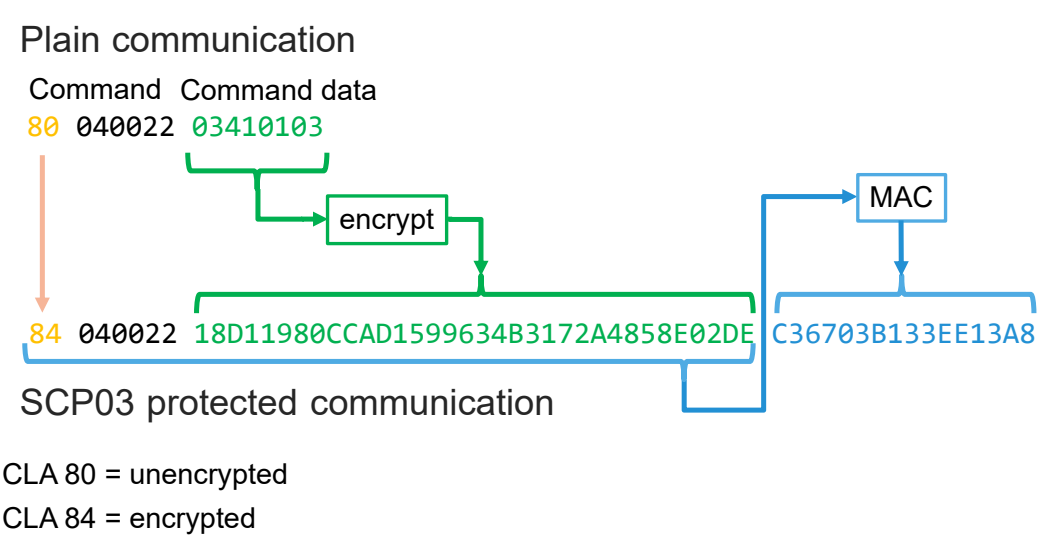


Figure 54. SPC03 Encryption and MACing principle

6.2 How to configure the Platform SCP keys in the i.MX RT1060 MCUXpresso SDK

The product specific initial Platform SCP key values are described for the EdgeLock SE05x product variants in AN12436 and for the EdgeLock SE051 variants in AN12973.

The Plug & Trust middleware header file `ex_sss_tp_scp03_keys.h` contains the initial values of all EdgeLock SE05x, 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\`

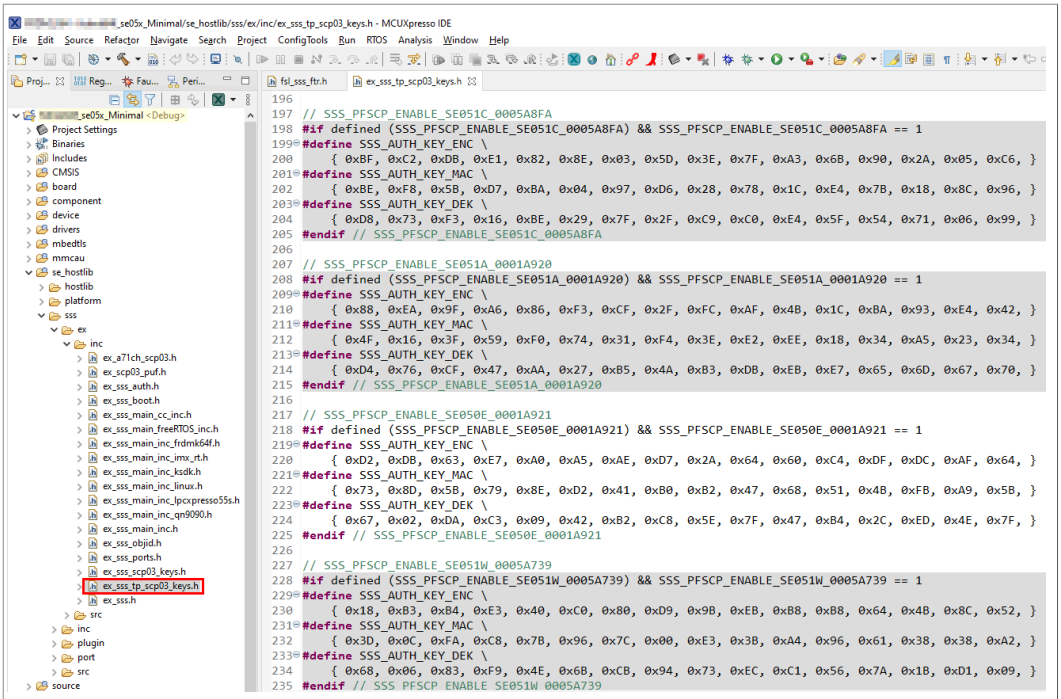


Figure 55. MCUXpresso SDK - Initial Platform SCP keys are defined in the `ex_sss_tp_scp03_keys.h` header file.

The `fsl_sss_ftr.h` header file inlcudes 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_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.

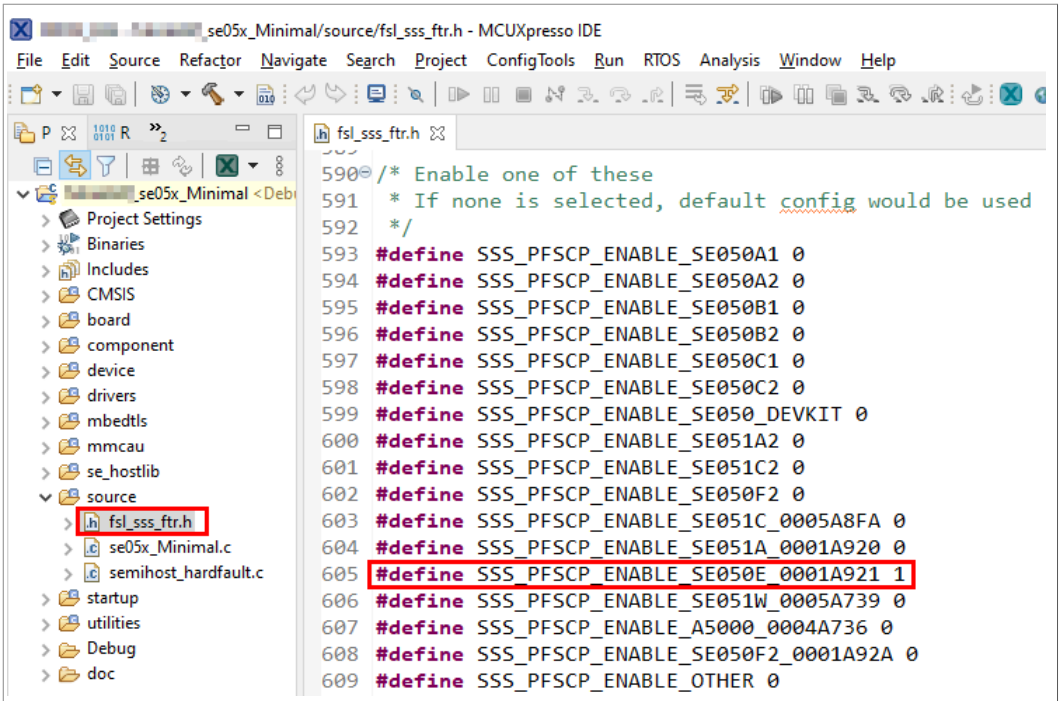


Figure 56. Select the actual Platform SCP keys in the `fsl_sss_ftr.h` header file.

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 16. 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 17. 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 18. 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 18. 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[1]	`SSS_PFSCP_ENABLE_SE050F2`

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

Table 19. 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 20. 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 Plug & Trust middleware. Section 6.3 describes how to enable Platform SCP in the Binding EdgeLock SE05x to a host MCU/MPU using Platform SCP.

6.3 How to enable Platform SCP in the i.MX RT1060 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`.

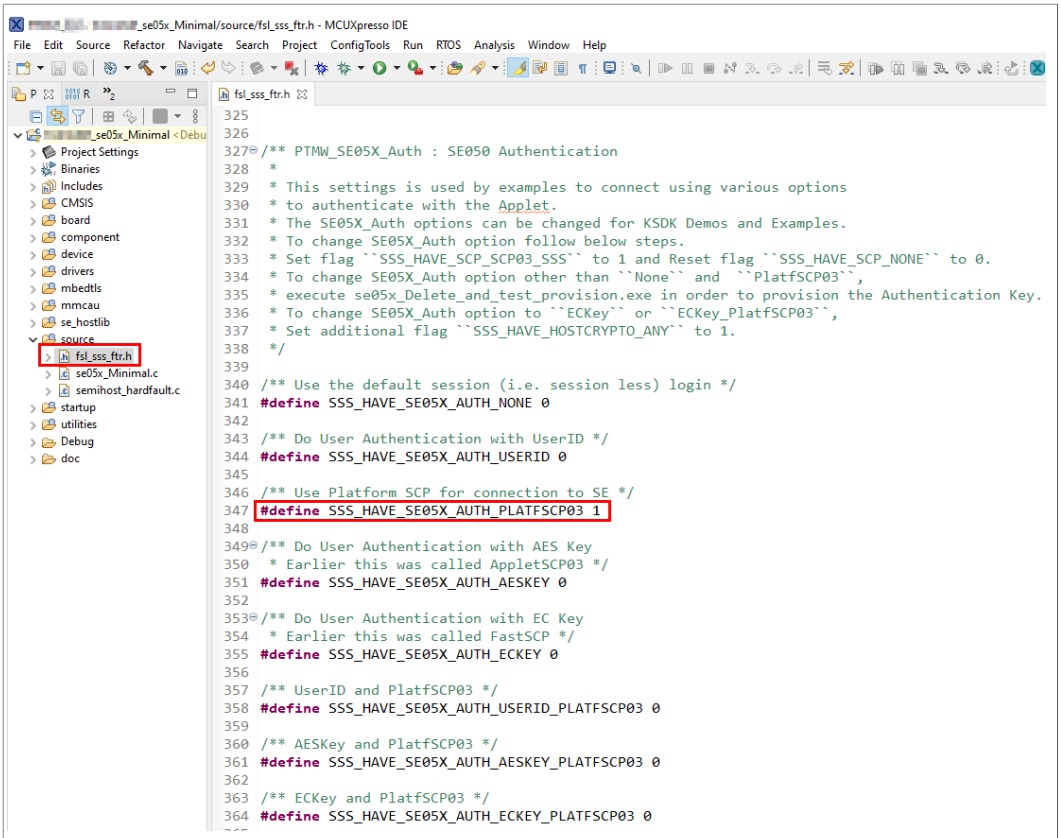


Figure 57. Feature file fsl_sss_ftr.h - Option PTMW_SE05X_Auth - PlatformSCP enabled

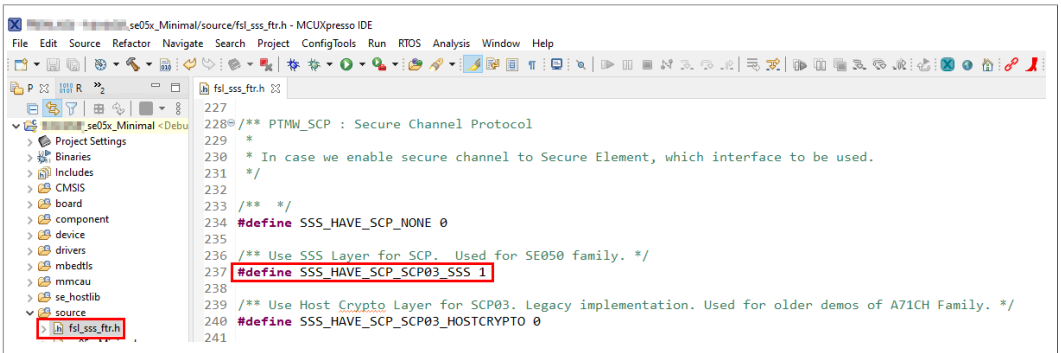


Figure 58. Feature file fsl_sss_ftr.h - Option PTMW_SCP - PlatformSCP enabled

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 SE05x product variants in AN12436 and for the EdgeLock SE051 variants in AN12973.

The Plug & Trust middleware header file `ex_sss_tp_scp03_keys.h` contains the initial values of all EdgeLock SE05x, 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\`

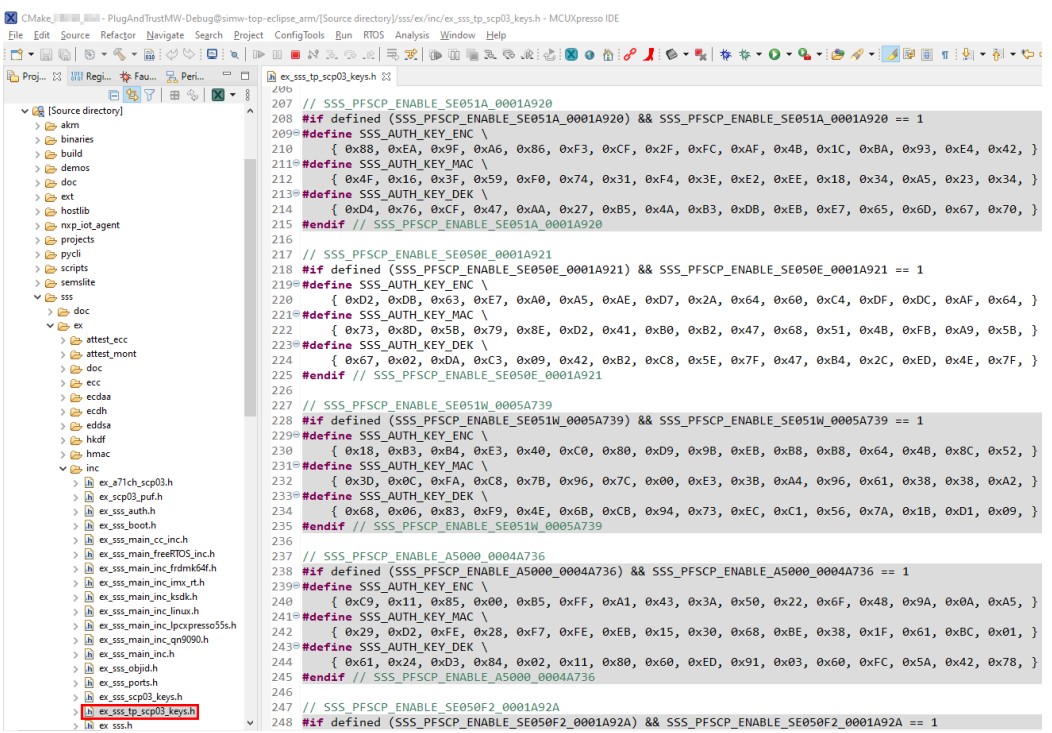


Figure 59. MCUXpresso - Initial Platform SCP keys are defined in `ex_sss_tp_scp03_keys.h` header file

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.

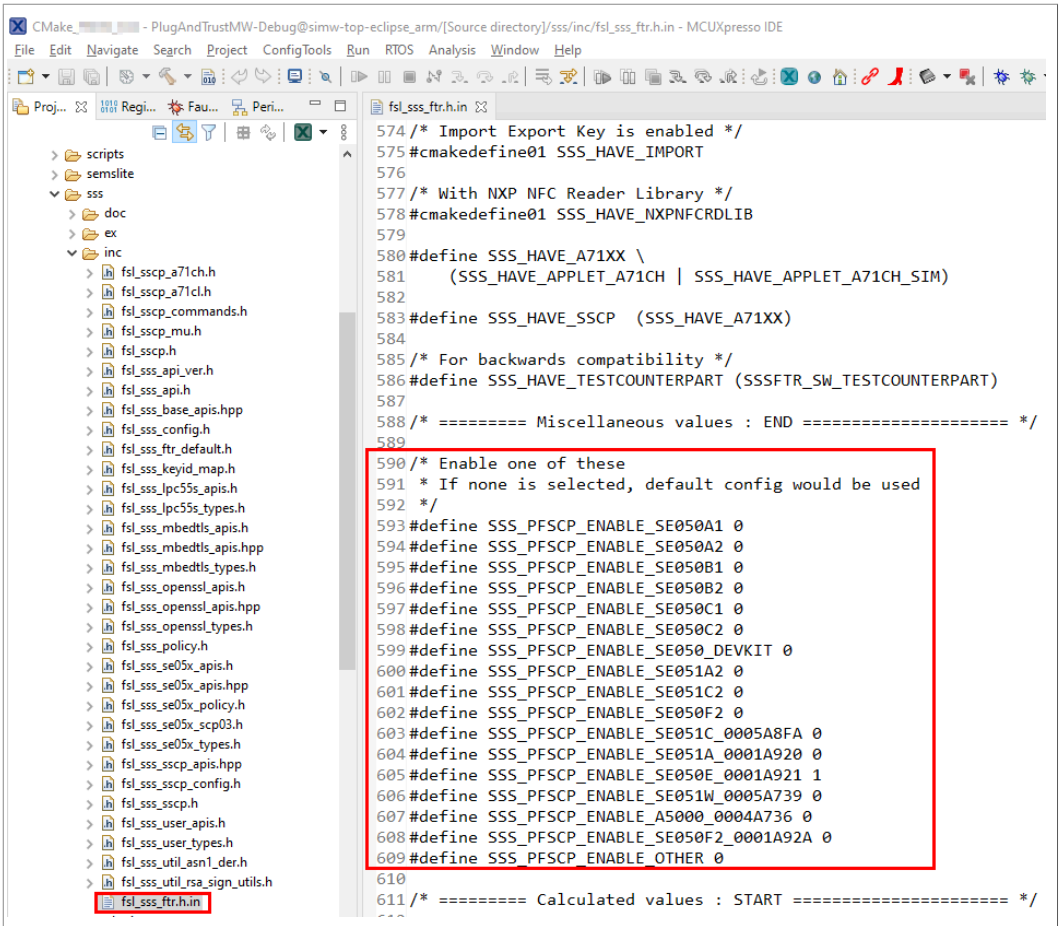


Figure 60. Select the actual Platform SCP keys in the `fsl_sss_ftr.h.in` CMake input file

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 overwritting the `fsl_sss_ftr.h` file every time CMake is invoked. CMake is using the SCP key settings of the `fsl_sss_ftr.h.in` file as input to generate the the `fsl_sss_ftr.h` file. You do not have to manually edit the `fsl_sss_ftr.h` feature file. Selections from CMake edit cache automatically updates 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 21. 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 21. 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 22. 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 23. Platform SCP key define prefix for SE050 Previous Generation product variants

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

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

Table 24. 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 25. 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 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.

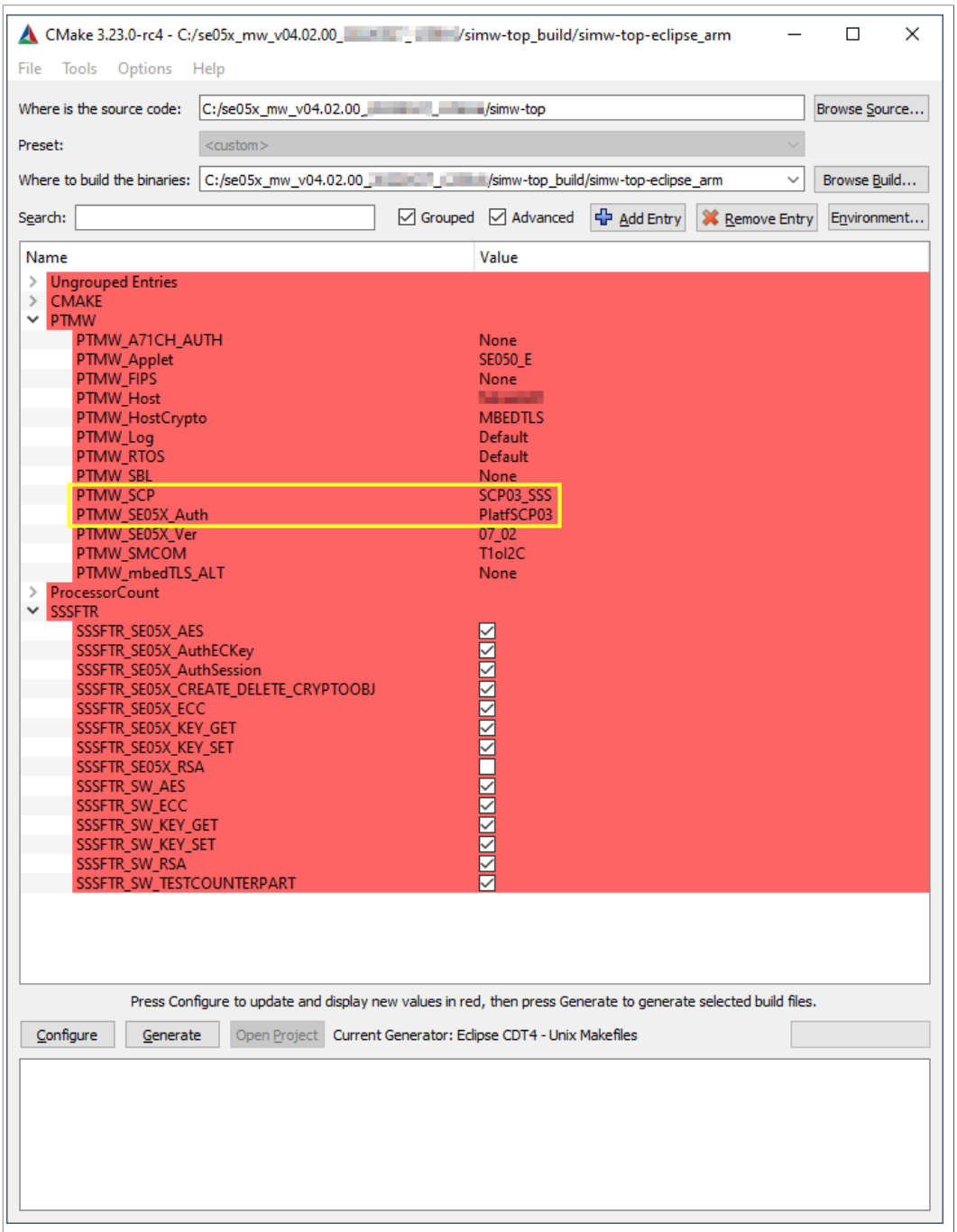


Figure 61. SE050E CMake Settings - PlatformSCP enabled

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 62:

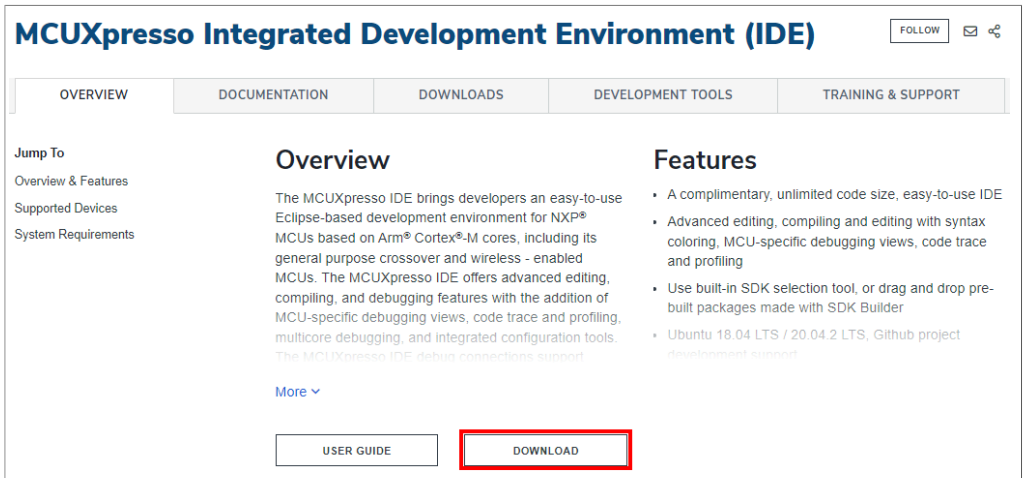


Figure 62. Go to MCUXpresso website

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 63:

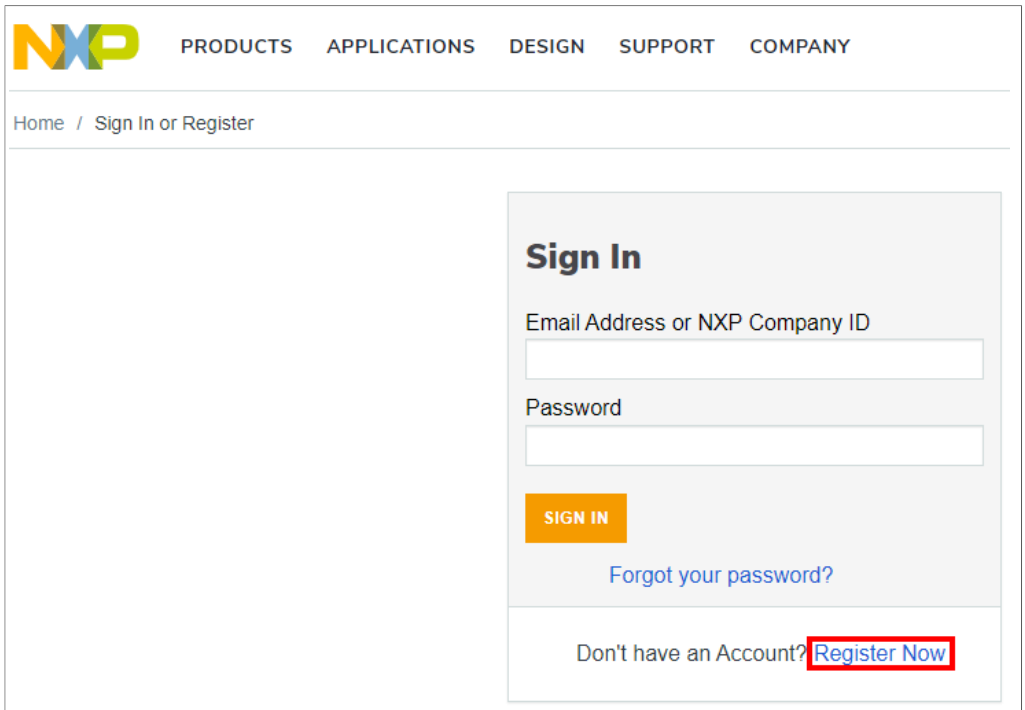


Figure 63. Register your NXP account

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 64:

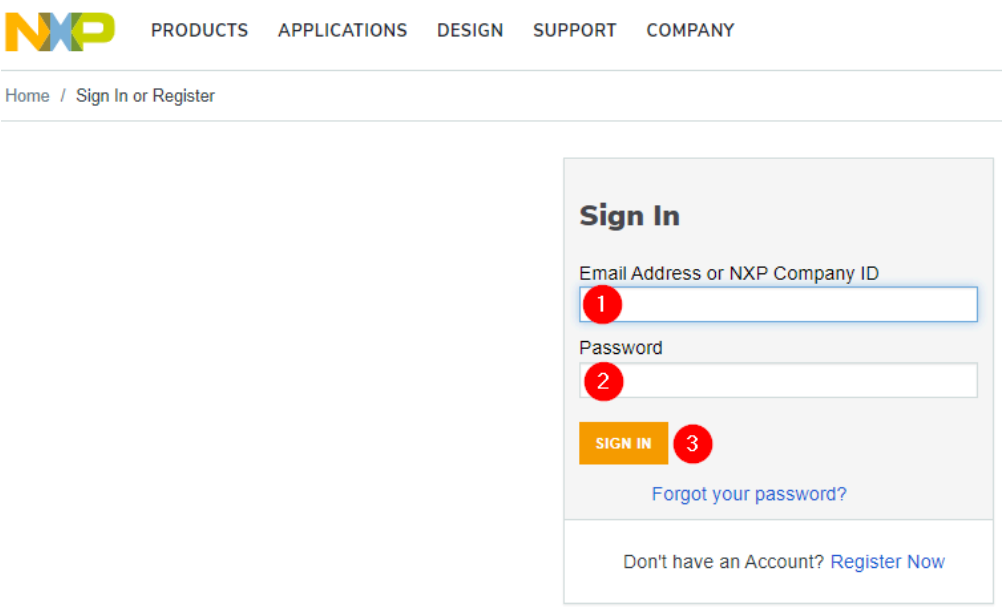


Figure 64. Sign-in in NXP website

4. Click on MCUXpresso IDE as shown in Figure 65:

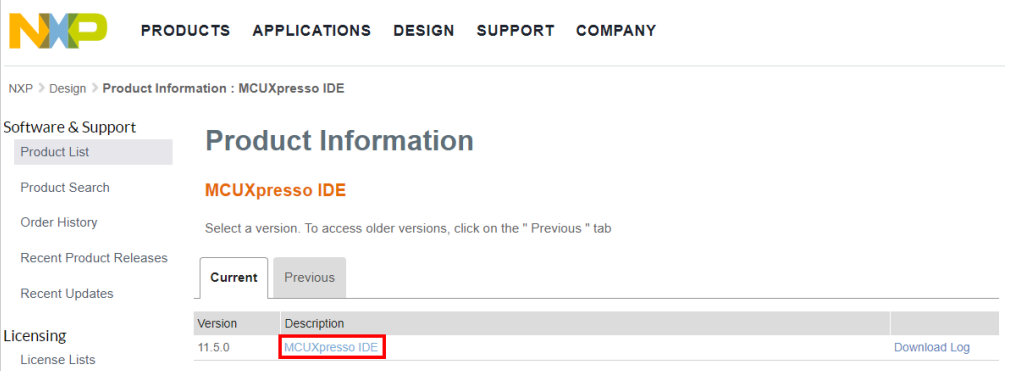


Figure 65. Select MCUXpresso

5. Accept software terms and conditions as shown in Figure 66:

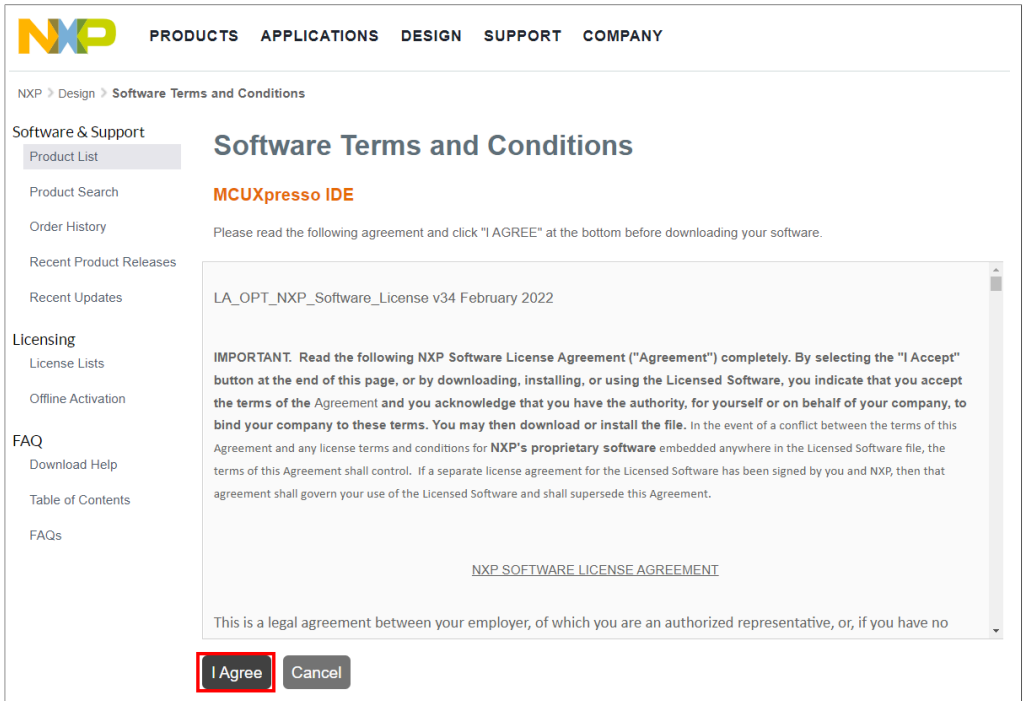


Figure 66. Accept software terms and conditions

6. Select your MCUXpresso product version and click on the corresponding ***File Name*** to start the download as shown in Figure 67:

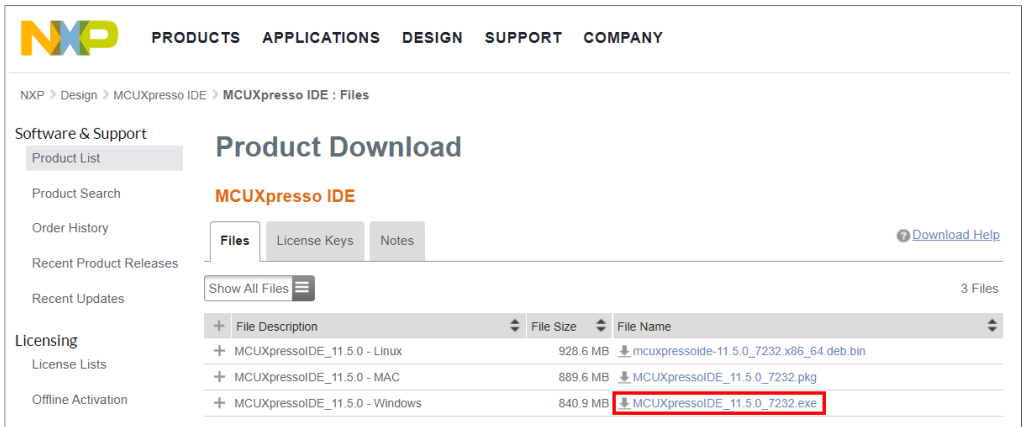


Figure 67. Download MCUXpresso

7. 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 68, Figure 69, Figure 70 and Figure 71:

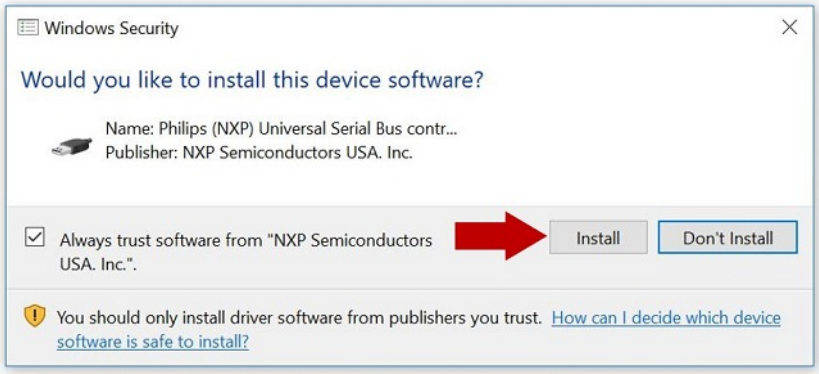


Figure 68. Install MCUXpresso required drivers I

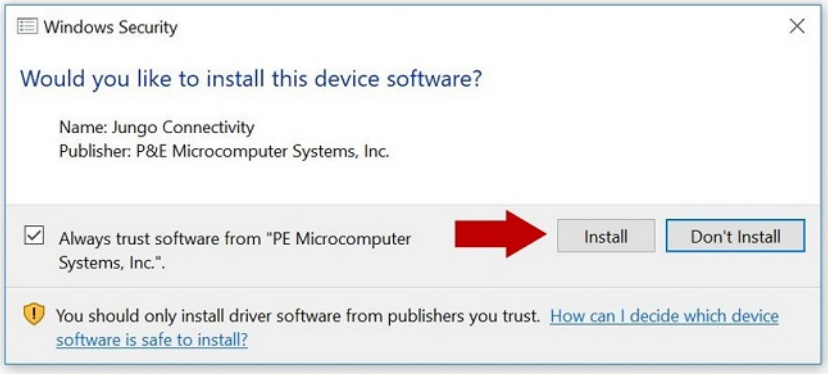


Figure 69. Install MCUXpresso required drivers II

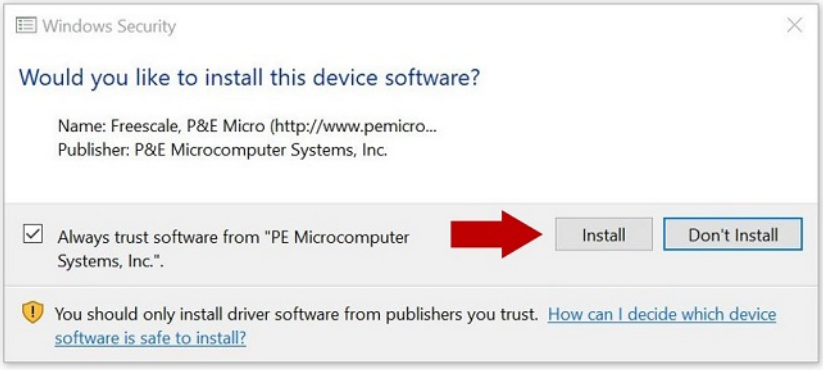


Figure 70. Install MCUXpresso required drivers III

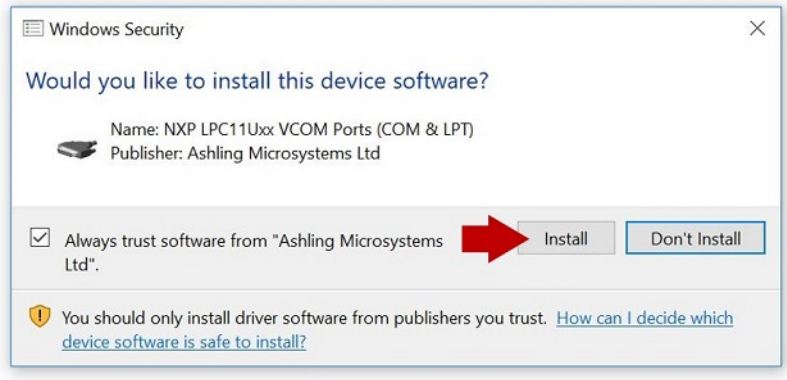


Figure 71. Install MCUXpresso required drivers IV

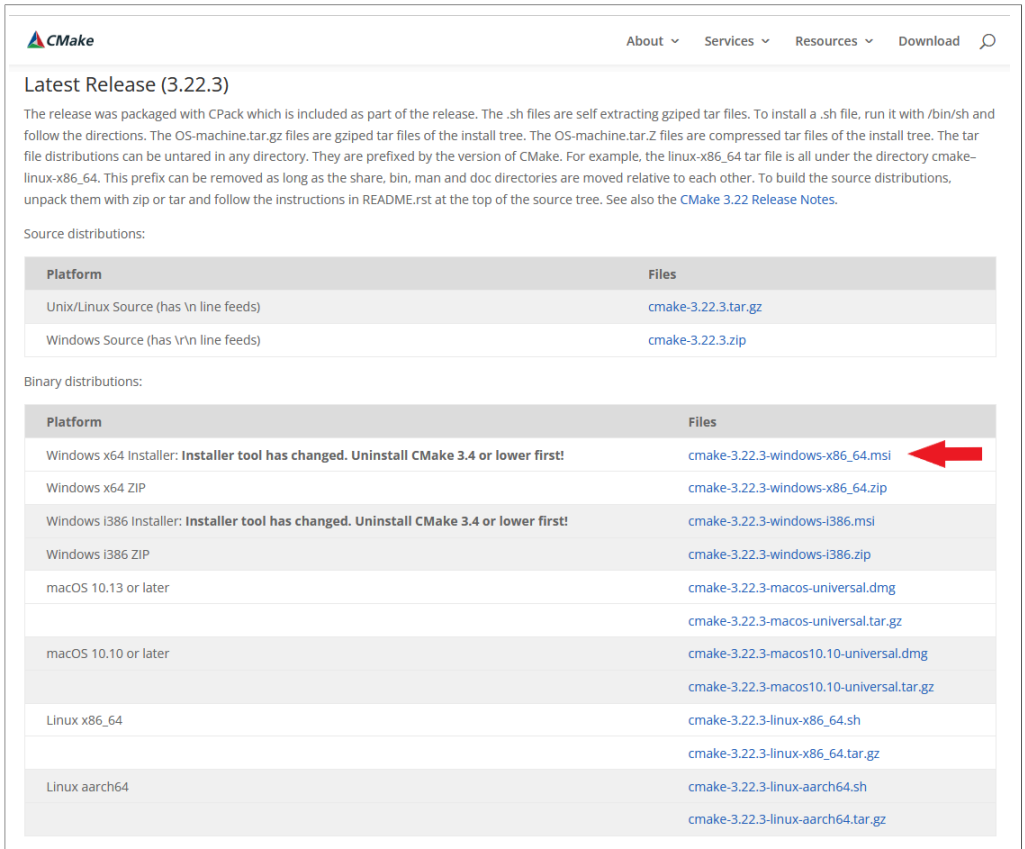
8 Appendix B: Install CMake

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 72:



CMake

About Services Resources Download

Latest Release (3.22.3)

The release was packaged with CPack which is included as part of the release. The .sh files are self extracting gziped tar files. To install a .sh file, run it with /bin/sh and follow the directions. The OS-machine.tar.gz files are gziped tar files of the install tree. The OS-machine.tar.Z files are compressed tar files of the install tree. The tar file distributions can be untared in any directory. They are prefixed by the version of CMake. For example, the linux-x86_64 tar file is all under the directory cmake–linux-x86_64. This prefix can be removed as long as the share, bin, man and doc directories are moved relative to each other. To build the source distributions, unpack them with zip or tar and follow the instructions in README.rst at the top of the source tree. See also the CMake 3.22 Release Notes.

Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.22.3.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.22.3.zip

Binary distributions:

Platform	Files
Windows x64 Installer: **Installer tool has changed. Uninstall CMake 3.4 or lower first!**	cmake-3.22.3-windows-x86_64.msi
Windows x64 ZIP	cmake-3.22.3-windows-x86_64.zip
Windows i386 Installer: **Installer tool has changed. Uninstall CMake 3.4 or lower first!**	cmake-3.22.3-windows-i386.msi
Windows i386 ZIP	cmake-3.22.3-windows-i386.zip
macOS 10.13 or later	cmake-3.22.3-macos-universal.dmg
	cmake-3.22.3-macos-universal.tar.gz
macOS 10.10 or later	cmake-3.22.3-macos10.10-universal.dmg
	cmake-3.22.3-macos10.10-universal.tar.gz
Linux x86_64	cmake-3.22.3-linux-x86_64.sh
	cmake-3.22.3-linux-x86_64.tar.gz
Linux aarch64	cmake-3.22.3-linux-aarch64.sh
	cmake-3.22.3-linux-aarch64.tar.gz

Figure 72. Download CMake

3. Double click on the downloaded installer file. Windows Defender SmartScreen might pop-up the wizard shown in Figure 73:

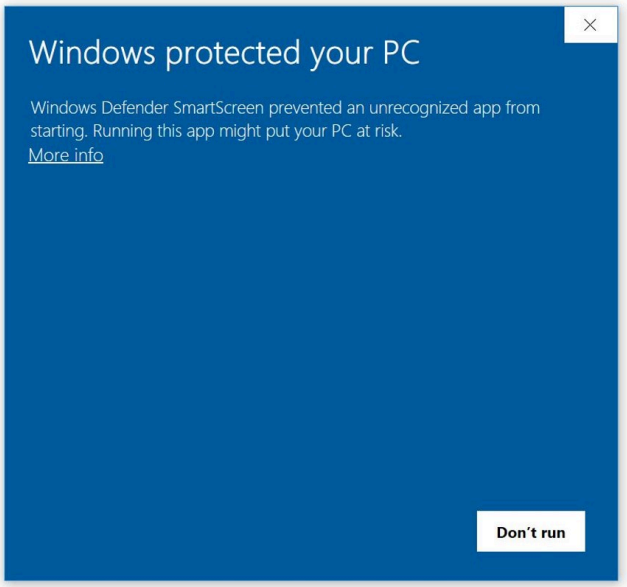


Figure 73. Execute CMake installer

4. If this is your case: Click (1) on ***More info*** and then (2) click on ***Run anyway*** as shown in Figure 74:

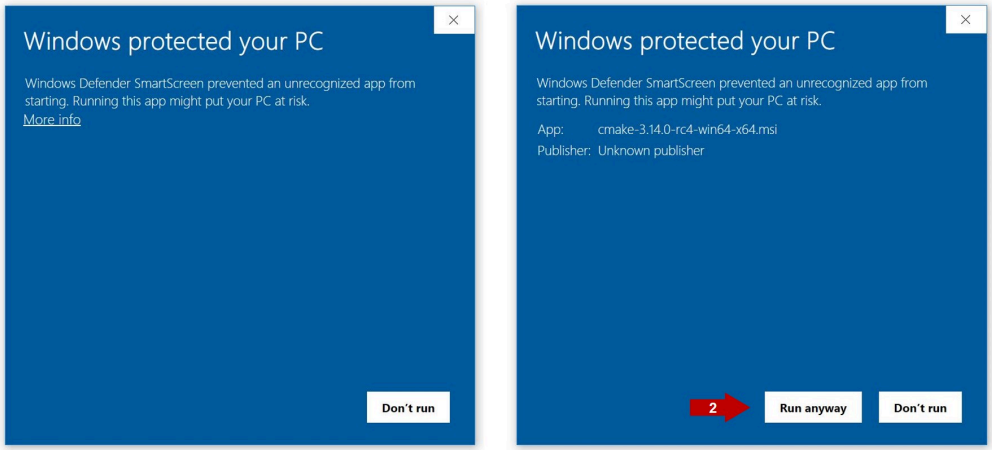


Figure 74. Run the CMake installer (II)

5. The CMake installation wizard will open. Click (1) ***Next*** and (2) **accept** the End-User License Agreement as shown in Figure 75:

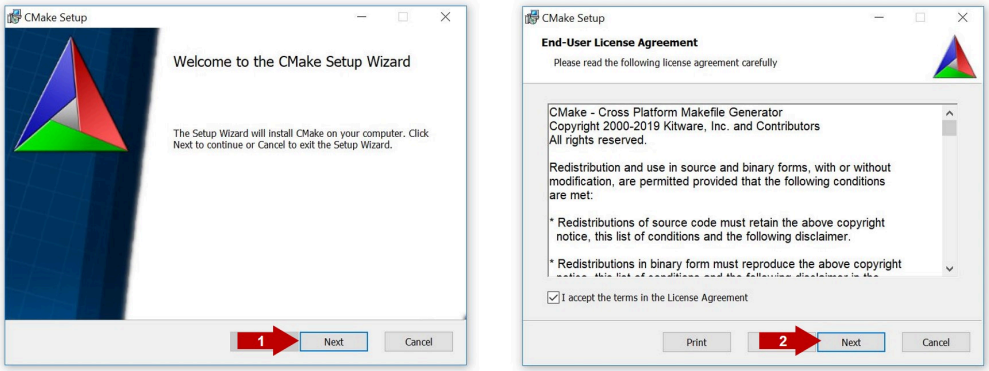


Figure 75. CMake installation wizard

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 76:

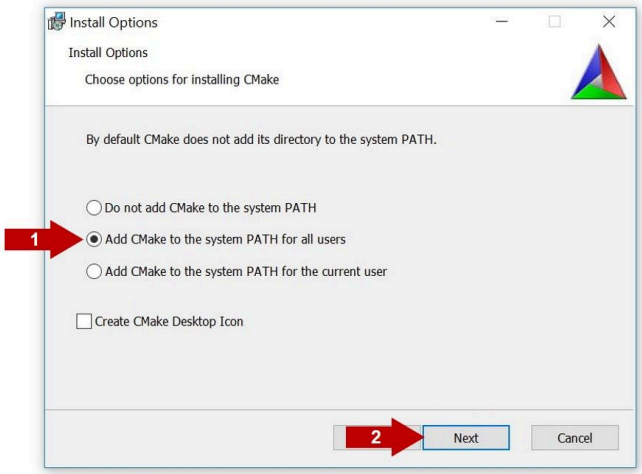


Figure 76. Add CMake path

7. Select a destination folder, (1) click ***Next*** and then (2) click ***Install*** as shown in Figure 77:

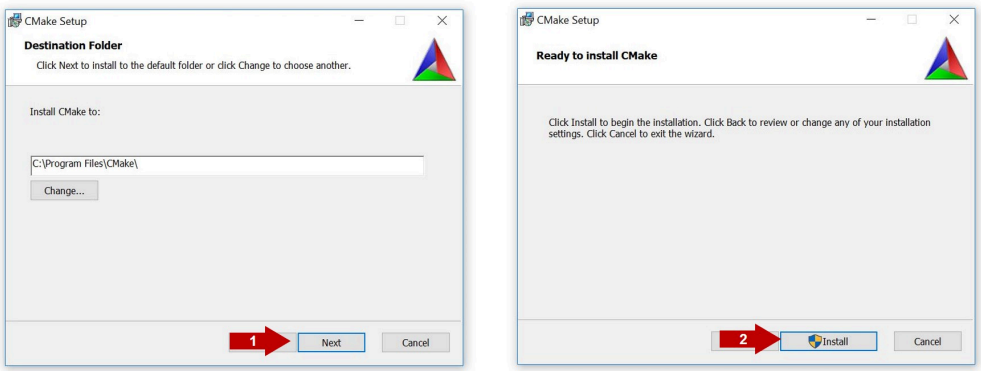


Figure 77. Install CMake

8. Wait a few seconds until the installation is completed and click ***Finish*** as shown in Figure 78:

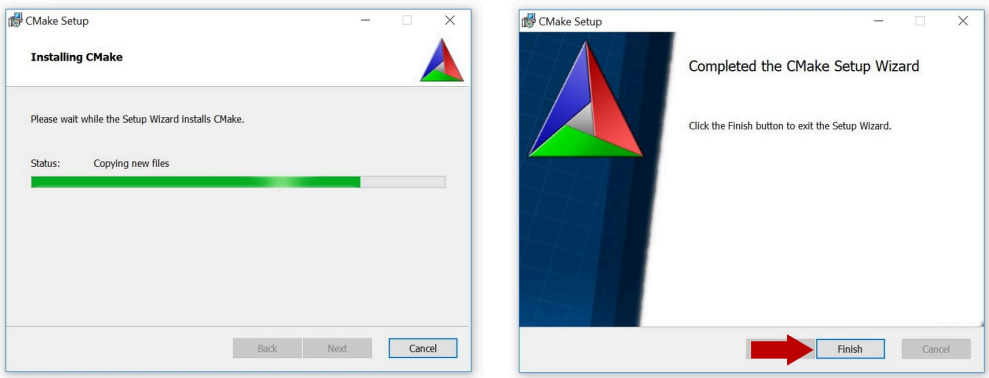


Figure 78. Complete CMake installation

9 Appendix C: Install Python

This section explains how to install Python ≥ 3.7.x and ≤ 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	MD5 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		c1d2af96d9f3564f57f35cfc3c1006eb	7671509	SIG
Windows embeddable package (64-bit)	Windows		b8e8bfba8e56edcd654d15e3bdc2e29a	8509821	SIG
Windows help file	Windows		784020441c1a25289483d3d8771a8215	9284044	SIG
Windows installer (32-bit)	Windows		457d648dc8a71b6bc32da30a7805c55b	27767040	SIG
Windows installer (64-bit)	Windows	Recommended	747ac35ae667f4ec1ee3b001e9b7dbc6	28909456	SIG

Figure 79. 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 80:

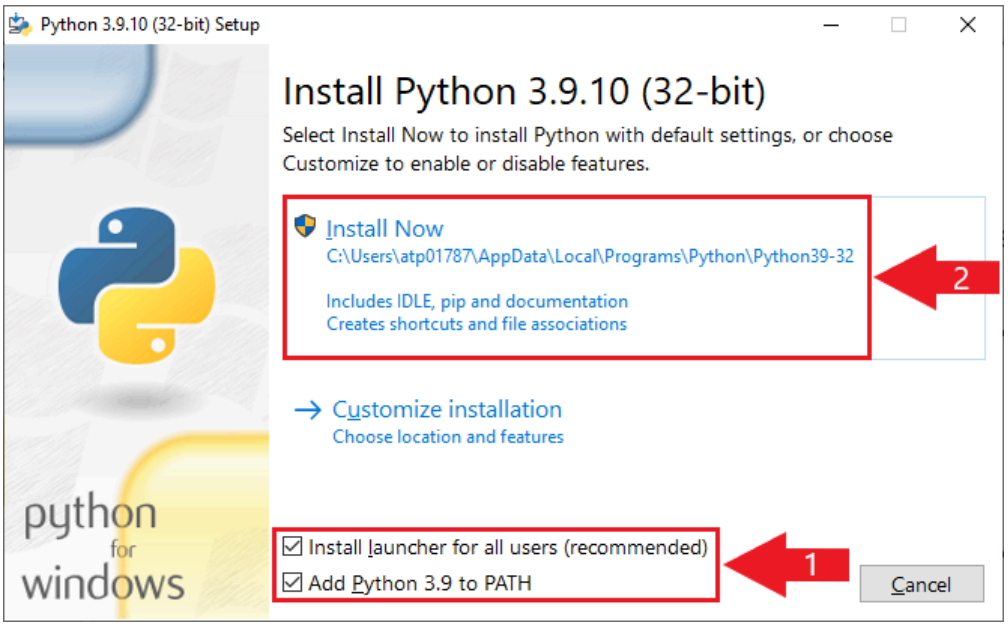


Figure 80. Install Python 3.9.x 32 bit for Windows

3. Wait a few seconds until the installation is completed as indicated in Figure 81

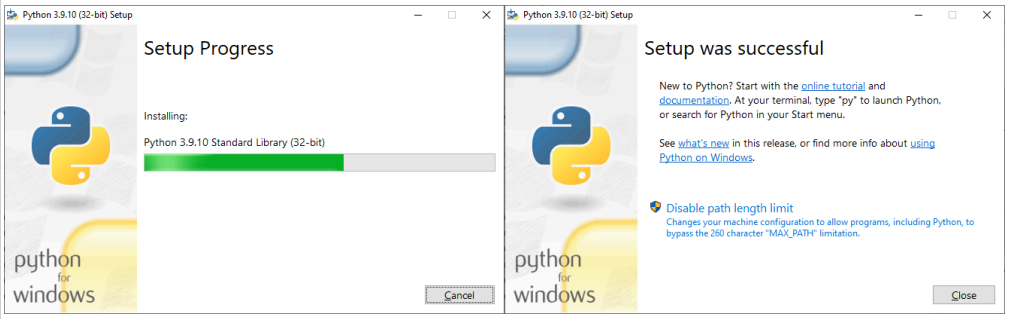


Figure 81. Python 3.9.x 32 bit installation completed

10 Legal information

10.1 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.

10.2 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.

10.3 Trademarks

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

Tables

Figures

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 4 August 2022
Document identifier: AN12450
Document number: 546942