AN12542

EdgeLock SE05x Quick start guide with LPC55S69

Rev. 3.3 — 4 August 2022 **Application note**
560833

Document information

Information	Content
Keywords	EdgeLock SE05x, Plug & Trust middleware, LPC55S69
Abstract	This document explains how to get started with the EdgeLock SE05x Plug & Trust middleware using the EdgeLock SE05x development boards and LPC55S69 MCU board. It provides detailed instructions to run projects imported either from the LPC55S69 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	2019-07-24	First document release
2.0	2019-11-25	Major update to incorporate details to import projects from LPC55S69 SDK and CMakebased build system.
2.1	2019-12-17	Corrected OM-SE05xARD J14 jumper configuration.
3.0	2020-10-27	Updated for EdgeLock SE051.
3.1	2020-12-07	Updated to latest template and fixed broken URLs.
3.2	2022-03-28	Add EdgeLock SE050E and EdgeLock A5000 product variants. Update Table 1, Figure 1, Figure 2, Figure 3, Figure 4, Figure 12, Figure 17, Figure 42 and Figure 47. 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 5.6.4 Product specific CMake build settings. Add Section 6 Binding EdgeLock SE05x to a host using Platform SCP.
3.3	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 Section 5.6.2 Run EdgeLock SE05x Plug & Trust middleware examples. Update Section 5.6.4 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 product family in the latest IoT security use cases. These project examples can be either

- Imported from the MCUXpresso SDKs made available for LPC55S69 MCU board.
- Imported from the CMake-based build system included in the Plug & Trust middleware package

This document provides detailed instructions to run EdgeLock SE05x project examples imported either from the LPC55S69 SDK or the CMake-based build system. However, the LPC55S69 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 it 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 boards required in this document
2. Setup your boards. Section 3 describes how to setup the OM-SE05xARD and LPC55S69 boards.
3. Run project examples. Go to Section 4 for instructions to import projects from the LPC55S69 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 Required hardware

The EdgeLock SE05x works as an auxiliary security device attached to a host controller, communicating with through an I²C interface. To follow the instructions provided in this document, you need an EdgeLock SE05x development board and a LPC55S69 MCU board, acting as a host controller.

EdgeLock SE05x development boards ordering details

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

Table 1. EdgeLock SE05x development boards.

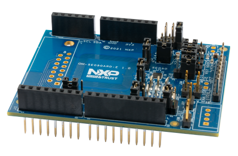
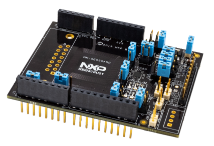
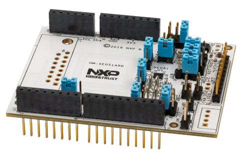
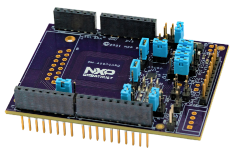
Part number	12NC	Description	Picture
OM-SE050ARD-E	9354 332 66598	SE050E Arduino® compatible development kit	

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

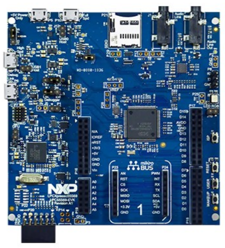
Part number	12NC	Description	Picture
OM-SE050ARD-F	9354 357 63598	SE050 Arduino® compatible development kit	
OM-SE050ARD	9353 832 82598	SE050F Arduino® compatible development kit	
OM-SE051ARD	9353 991 87598	SE051 Arduino® compatible development kit	
OM-A5000ARD	9354 243 19598	A5000 Arduino® compatible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

LPC55S69 MCU board ordering details

Table 2 details the ordering details for the LPC55S69 board.

Table 2. LPC55S69 evaluation kit details

Part number	12NC	Content	Picture
LPC55S69-EVK	935377412598	LPCXpresso55S69 Development Board	

3 Boards setup

This section explains how to setup your LPC55S69 and OM-SE05xARD board to execute the Plug & Trust middleware:

1. The OM-SE05xARD boards have jumpers that allow you to use the EdgeLock SE05x I^{2}C 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.*

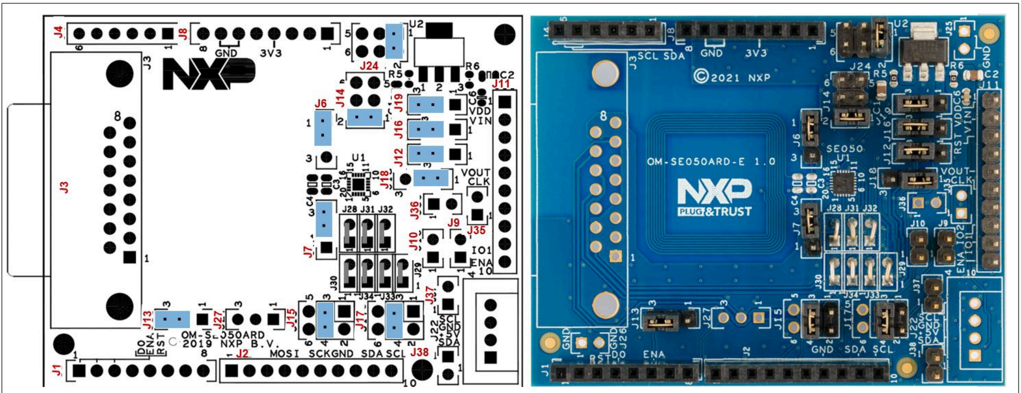


Figure 1. Jumper configuration for LPC55S69

2. The LPC55S69 board default jumper configuration must be used when running this example. For more information about the LPC55S69 board default jumper configuration settings, refer to UM11158.
3. The OM-SE05xARD and LPC55S69 boards can be directly connected using the Arduino headers present in both boards. Connect the OM-SE05xARD board on top of the LPC55S69 as shown in Figure 2. Note that OM-SE05xARD should be aligned with A5 pin in LPC55S69 P19 header and D0 pin in LPC55S69 P18 header. The two last pins in P16 and the two first pins in P18 should be left open.

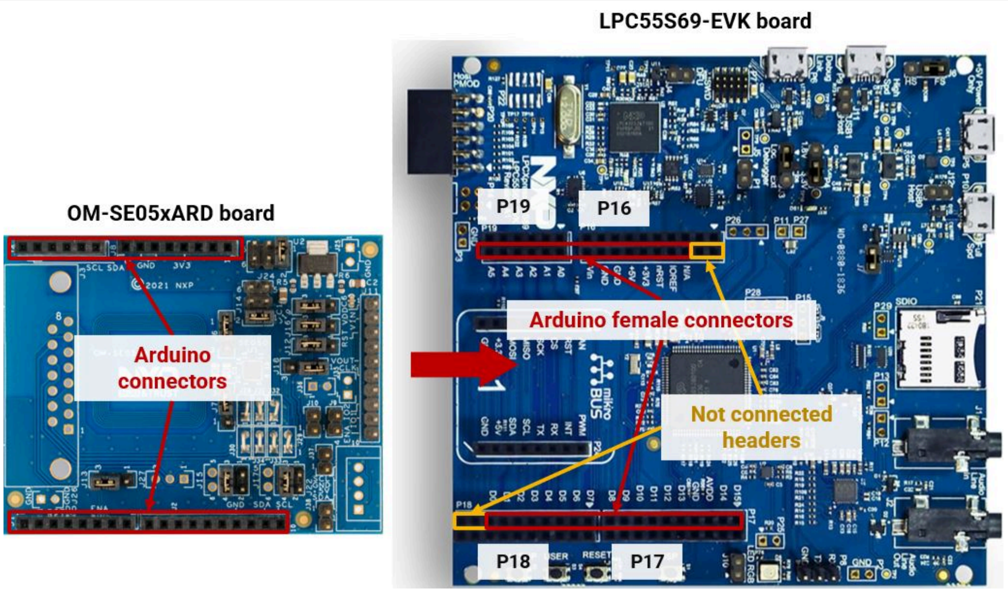


Figure 2. Arduino connectors of OM-SE05xARD and LPC55S69 boards

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

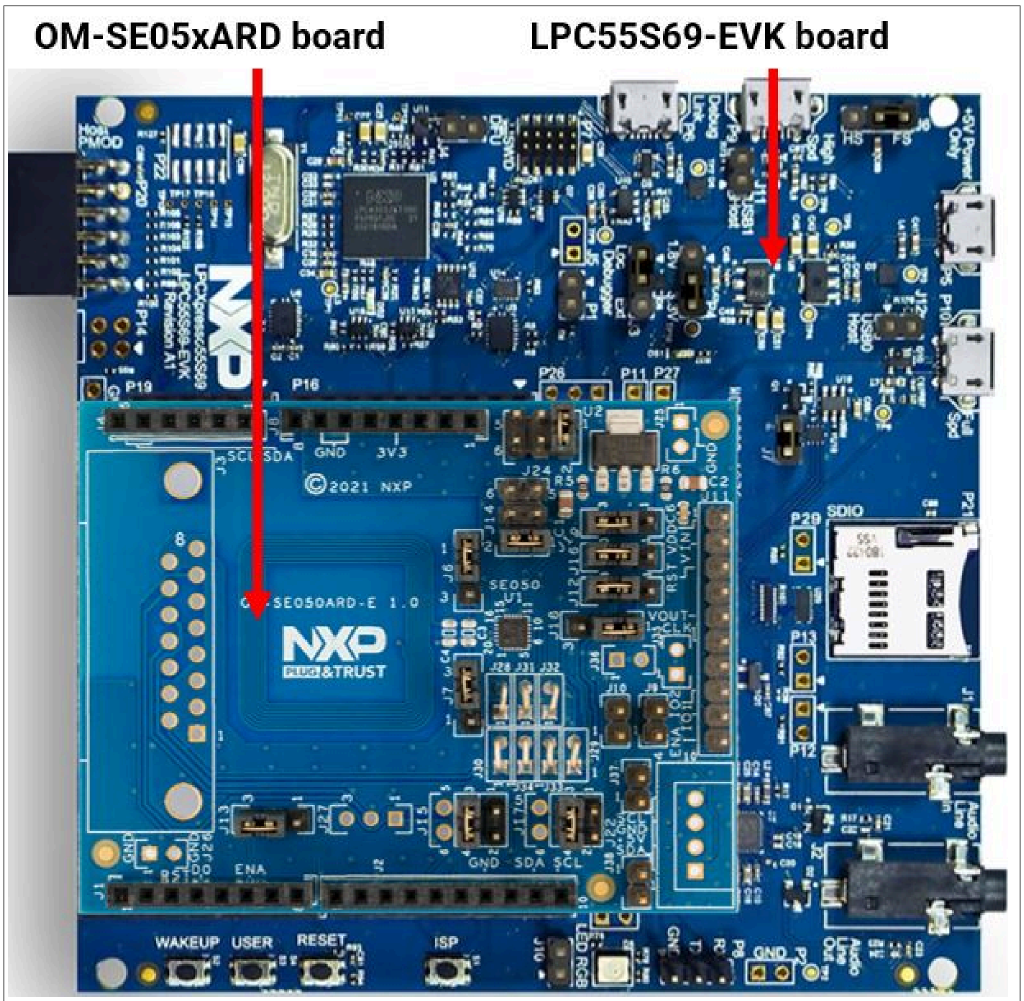


Figure 3. OM-SE05xARD mounted in LPC55S69 board

5. Check that your laptop recognizes the LPC55S69 board following the steps indicated in Figure 4
   a. Connect the board to your laptop using P6 Debug Link connector.
   b. Check that the serial port is recognized in the category Ports (COM & LTP). In this document, it is recognized as LPC-LinkII UCom Port (COM13) 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 P6 Debug Link USB port to the computer.

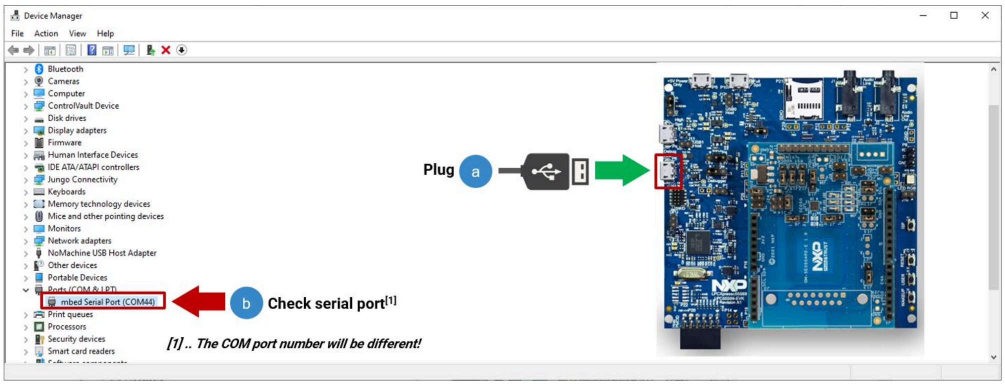


Figure 4. Check serial port

4 Import project examples from LPC55S69 SDK

This section explains how to run EdgeLock SE05x project examples by importing them from the LPC55S69 SDK. This option is the recommended one oposed 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.

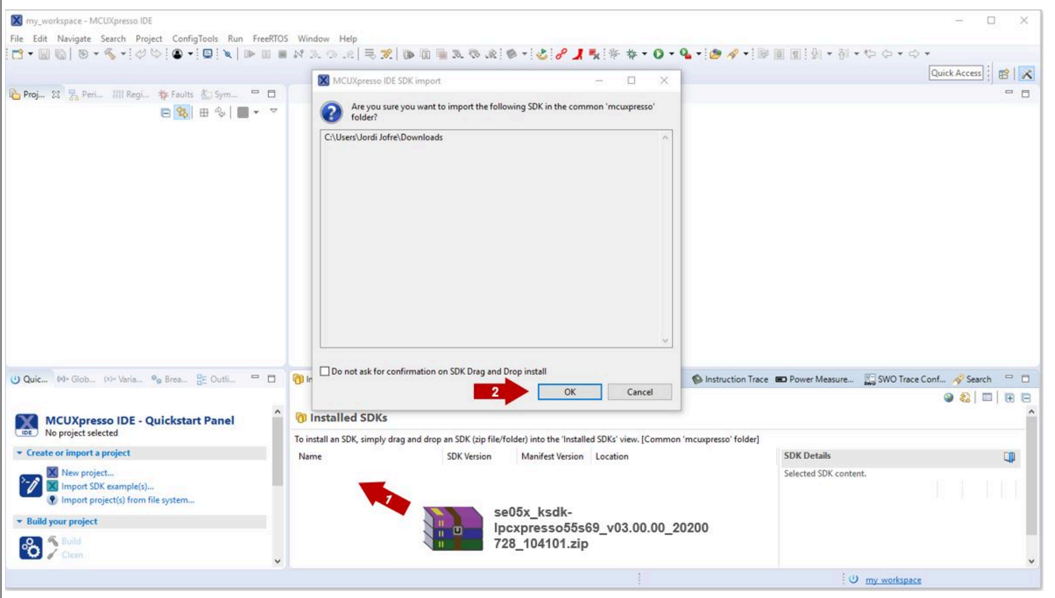
4.2 Download LPC55S69 SDK

The project examples for the EdgeLock SE05x are included as part of the LPC55S69 SDK. First, download the LPC55S69 SDK, publicly available from the NXP 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 LPC55S69 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 LPC55S69 SDK

After downloading the LPC55S69 SDK, we need to install it into the MCUXpresso workspace. To install the SDK, (1) drag and drop the LPC55S69 SDK zip file in the ***Installed SDKs*** section in the bottom part of the MCUXpresso IDE and (2) click ***OK*** as shown in Figure 5:


my_workspace - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run FreeRTOS Window Help
Quick Access
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Instruction Trace
Power Measure...
SWO Trace Conf...
Search
my_workspace
1
2
OK
Cancel

MCUXpresso IDE SDK import

Are you sure you want to import the following SDK in the common 'mcuxpresso' folder?

C:\Users\Jordi Jofre\Downloads

Do not ask for confirmation on SDK Drag and Drop install

Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

Name	SDK Version	Manifest Version	Location

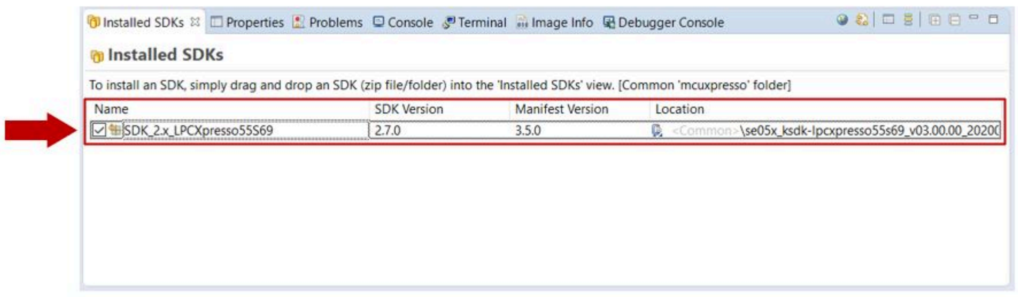
SDK Details

Selected SDK content.

se05x_ksdk-lpcxpresso55s69_v03.00.00_20200728_104101.zip

Figure 5. Import LPC55S69 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 6:



Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

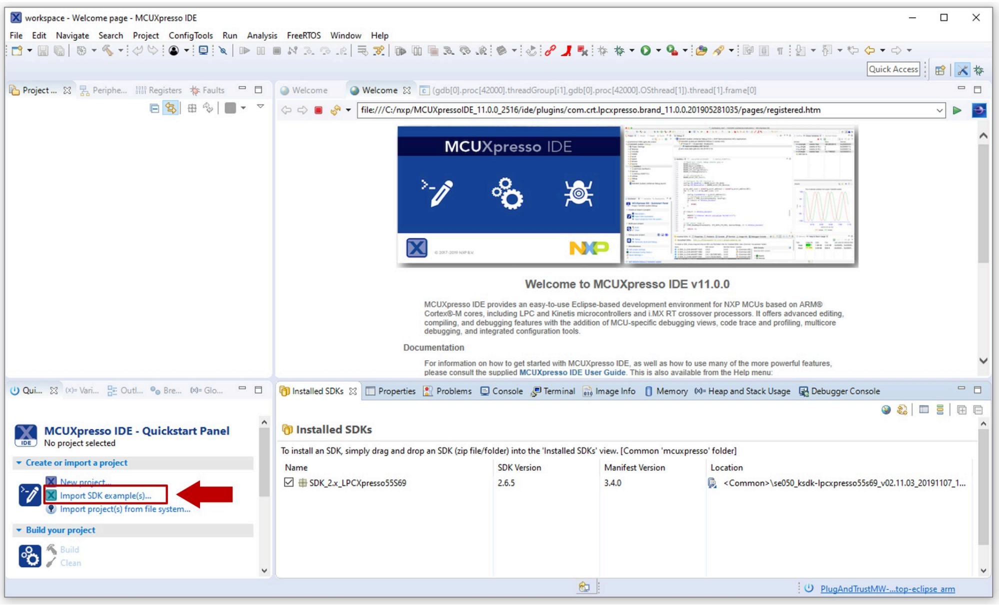
Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_LPCXpresso55S69	2.7.0	3.5.0	<Common>\se05x_ksdk-lpcxpresso55s69_v03.00.00_20200

Figure 6. Imported LPC55S69 SDK

4.4 Import project example in MCUXpresso

After importing the LPC55S69 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 7:


workspace - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Installed SDKs
Properties
Problems
Console
Terminal
Image Info
Memory
Heap and Stack Usage
Debugger Console
PlugAndTrustMW-...top-eclipse_arm

file:///C:/nxp/MCUXpressoIDE_11.0.0_2516/ide/plugins/com.crt.lpcxpresso.brand_11.0.0.201905281035/pages/registered.htm

Welcome to MCUXpresso IDE v11.0.0

MCUXpresso IDE provides an easy-to-use Eclipse-based development environment for NXP MCUs based on ARM® Cortex®-M cores, including LPC and Kinetis microcontrollers and i.MX RT crossover processors. It offers advanced editing, compiling, and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools.

Documentation

For information on how to get started with MCUXpresso IDE, as well as how to use many of the more powerful features, please consult the supplied MCUXpresso IDE User Guide. This is also available from the Help menu:

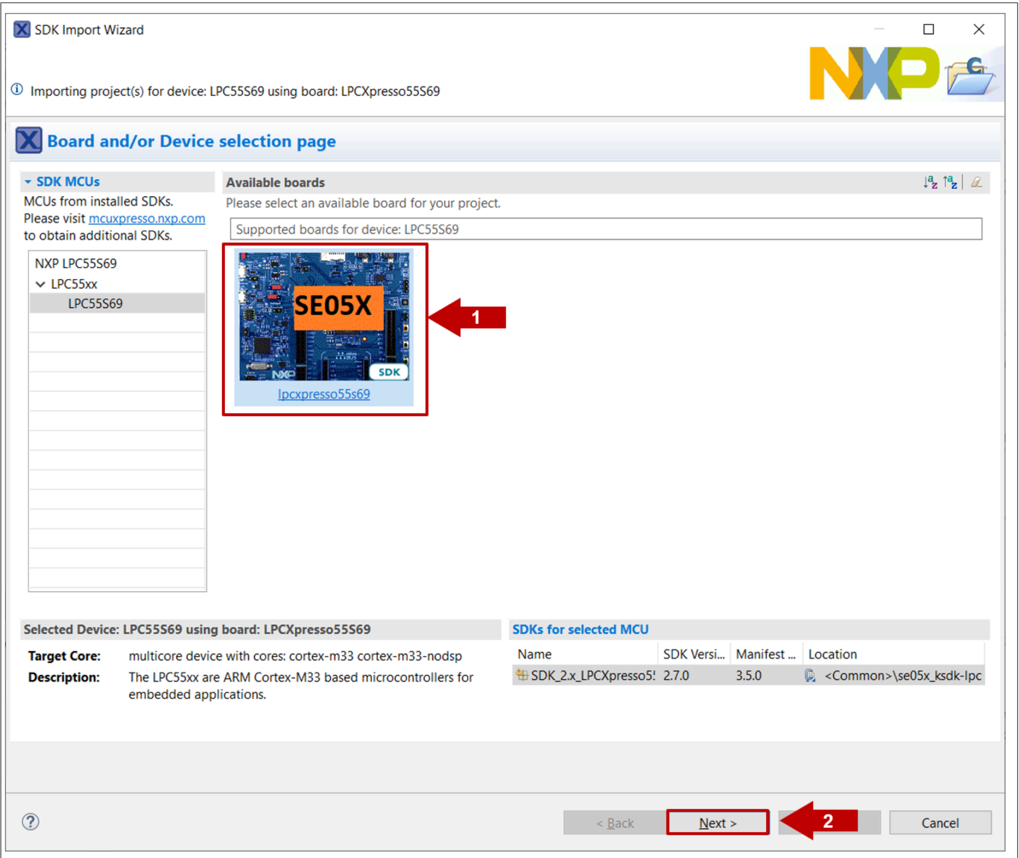
Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_LPCXpresso55S69	2.6.5	3.4.0	<Common>\se050_ksdk-lpcxpresso55s69_v02.11.03_20191107_1...

Figure 7. Import projects from SDK

2. The SDK import wizard will open. You should see a figure of an LPC55S69 board with an orange label. Select the board and click the *Next* button as shown in Figure 8:


SDK Import Wizard

Importing project(s) for device: LPC55S69 using board: LPCXpresso55S69

Board and/or Device selection page

SDK MCUs

MCUs from installed SDKs.
Please visit mcuxpresso.nxp.com
to obtain additional SDKs.

NXP LPC55S69
LPC55xx
LPC55S69

Available boards

Please select an available board for your project.

Supported boards for device: LPC55S69

lpcxpresso55s69

Selected Device: LPC55S69 using board: LPCXpresso55S69

Target Core: multicore device with cores: cortex-m33 cortex-m33-nodsp

Description: The LPC55xx are ARM Cortex-M33 based microcontrollers for embedded applications.

SDKs for selected MCU

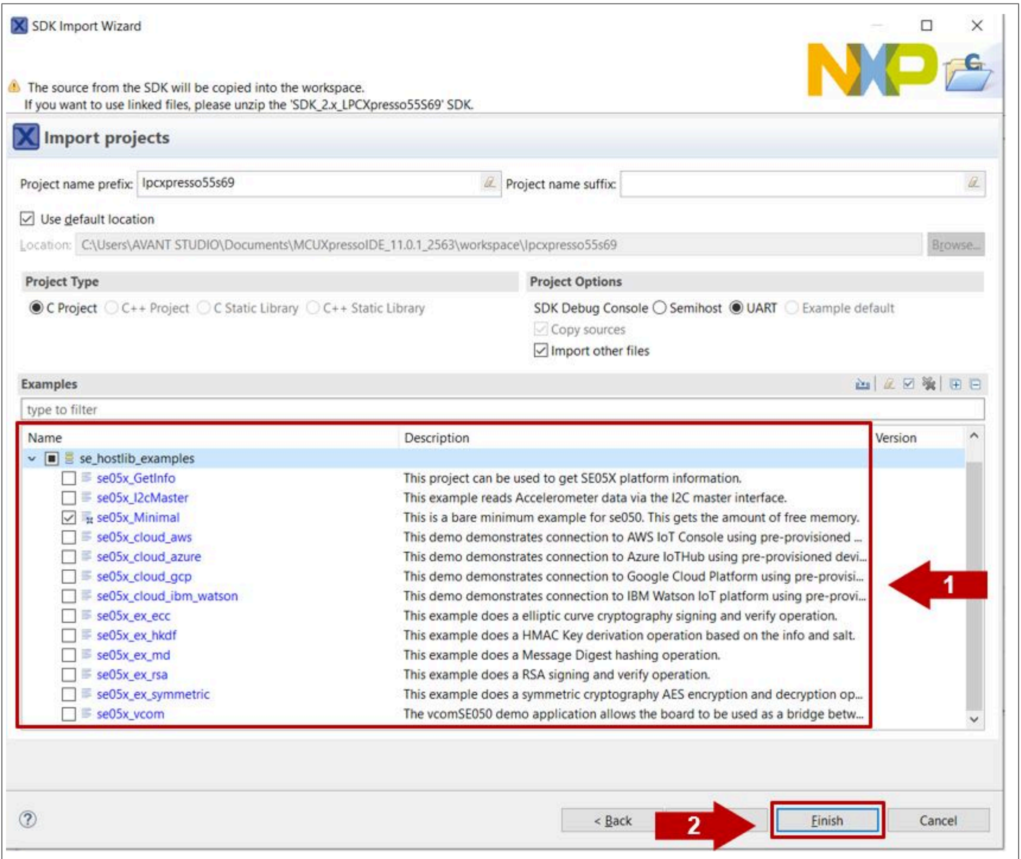
Name	SDK Versi...	Manifest ...	Location
SDK_2.x_LPCXpresso5!	2.7.0	3.5.0	<Common>\se05x_ksdk-lpc

Figure 8. 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 SE05xproject examples, first you need to uninstall the SDK currently installed, and then repeat the process indicated in* Figure 5

3. Under the `se_hostlib_examples` drop down list, you have the list of supported project examples for the LPC55S69. Select the number of examples you would like to import in your MCUXpresso workspace and click *Finish* button as shown in

Figure 9. In this case, we select the `se05x_Minimal` project as an example. The same process can be done with the rest of the examples.



The source from the SDK will be copied into the workspace.
If you want to use linked files, please unzip the 'SDK_2.x_LPCXpresso55S69' SDK.

Import projects

Project name prefix: lpcxpresso55s69 Project name suffix:

[✓] Use default location

Location: C:\Users\AVANT STUDIO\Documents\MCUXpressoIDE_11.0.1_2563\workspace\lpcxpresso55s69

Project Type

[● C Project] ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options

SDK Debug Console ○ Semihost [● UART] ○ Example default

[✓] Copy sources

[✓] Import other files

Examples

type to filter

Name	Description	Version
se_hostlib_examples		
☐ se05x_GetInfo	This project can be used to get SE05X platform information.	
☐ se05x_I2cMaster	This example reads Accelerometer data via the I2C master interface.	
✓ se05x_Minimal	This is a bare minimum example for se050. This gets the amount of free memory.	
☐ se05x_cloud_aws	This demo demonstrates connection to AWS IoT Console using pre-provisioned ...	
☐ se05x_cloud_azure	This demo demonstrates connection to Azure IoTHub using pre-provisioned devi...	
☐ se05x_cloud_gcp	This demo demonstrates connection to Google Cloud Platform using pre-provisi...	
☐ se05x_cloud_ibm_watson	This demo demonstrates connection to IBM Watson IoT platform using pre-provi...	
☐ se05x_ex_ecc	This example does a elliptic curve cryptography signing and verify operation.	
☐ se05x_ex_hkdf	This example does a HMAC Key derivation operation based on the info and salt.	
☐ se05x_ex_md	This example does a Message Digest hashing operation.	
☐ se05x_ex_rsa	This example does a RSA signing and verify operation.	
☐ se05x_ex_symmetric	This example does a symmetric cryptography AES encryption and decryption op...	
☐ se05x_vcom	The vcomSE050 demo application allows the board to be used as a bridge betw...	

Figure 9. Select projects to import

4. The projects you selected should now be visible in your MCUXpresso workspace as shown Figure 10:

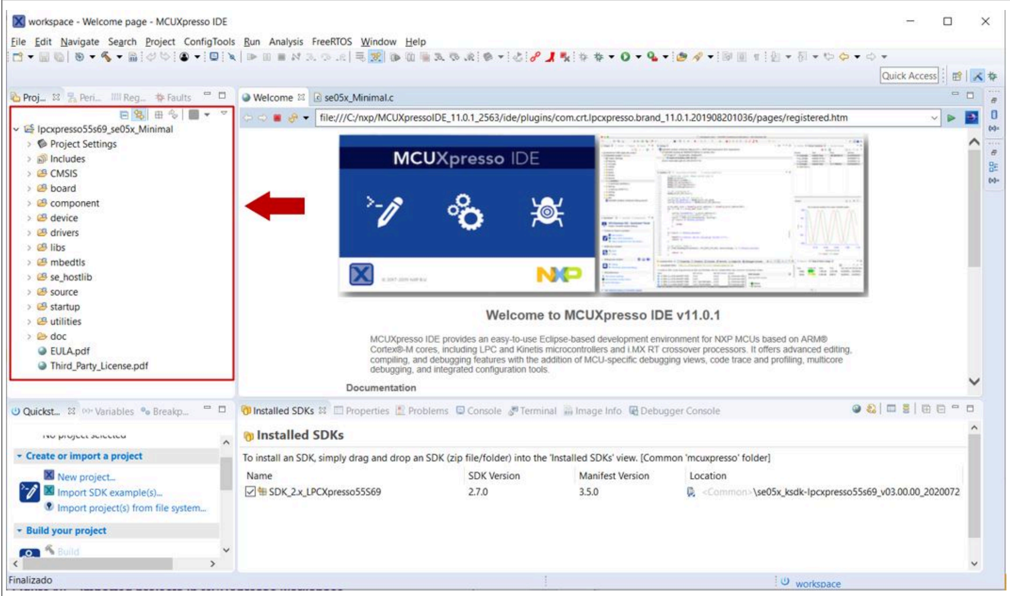

workspace - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Quick Access
Proj... Peri... Reg... Faults
lpcxpresso55s69_se05x_Minimal
Project Settings
Includes
CMSIS
board
component
device
drivers
libs
mbedtls
se_hostlib
source
startup
utilities
doc
EULA.pdf
Third_Party_License.pdf
Welcome
se05x_Minimal.c
file:///C:/nxp/MCUXpressoIDE_11.0.1_2563/ide/plugins/com.crt.lpcxpresso.brand_11.0.1.201908201036/pages/registered.htm
MCUXpresso IDE
Welcome to MCUXpresso IDE v11.0.1
MCUXpresso IDE provides an easy-to-use Eclipse-based development environment for NXP MCUs based on ARM® Cortex®-M cores, including LPC and Kinetis microcontrollers and i.MX RT crossover processors. It offers advanced editing, compiling, and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools.
Documentation
Quickst... Variables Breakp...
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Installed SDKs Properties Problems Console Terminal Image Info Debugger Console
Installed SDKs
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]
Name
SDK Version
Manifest Version
Location
SDK_2.x_LPCXpresso55S69
2.7.0
3.5.0
<Common>\se05x_ksdk-lpcxpresso55s69_v03.00.00_2020072
Finalizado
workspace

Figure 10. Imported projects in MCUXpresso workspace

4.5 Build, run and debug project example

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

1. Attach a USB cable from the computer to the Debug Link USB connector as shown in Figure 11.

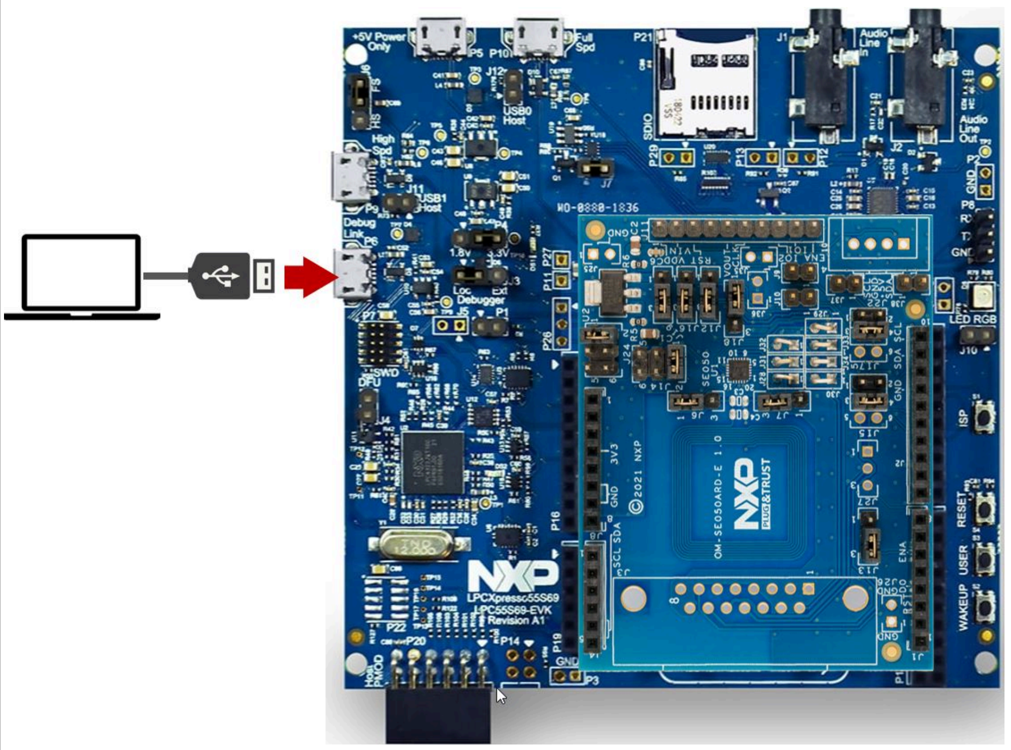
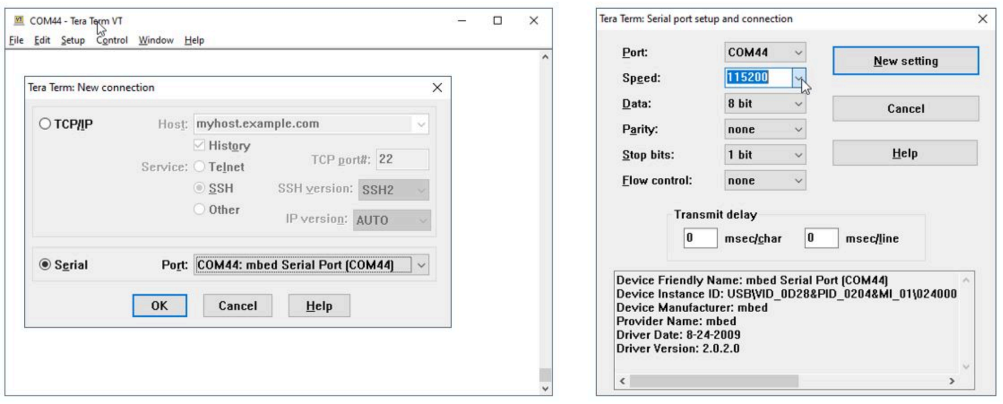


Figure 11. Connect boards to the laptop

2. Launch and setup TeraTerm application as shown in Figure 12:
   a. Click *Serial* option and select from the drop down list the COM port number assigned to your LPC55S69 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.



Tera Term: New connection

( ) TCP/IP Host: myhost.example.com
[✓] History
Service: ( ) Telnet TCP port#: 22
(•) SSH SSH version: SSH2
( ) Other IP version: AUTO

(•) Serial Port: COM44: mbed Serial Port (COM44)

Tera Term: Serial port setup and connection

Port:	COM44
Speed:	115200
Data:	8 bit
Parity:	none
Stop bits:	1 bit
Flow control:	none

Transmit delay
0 msec/char 0 msec/line

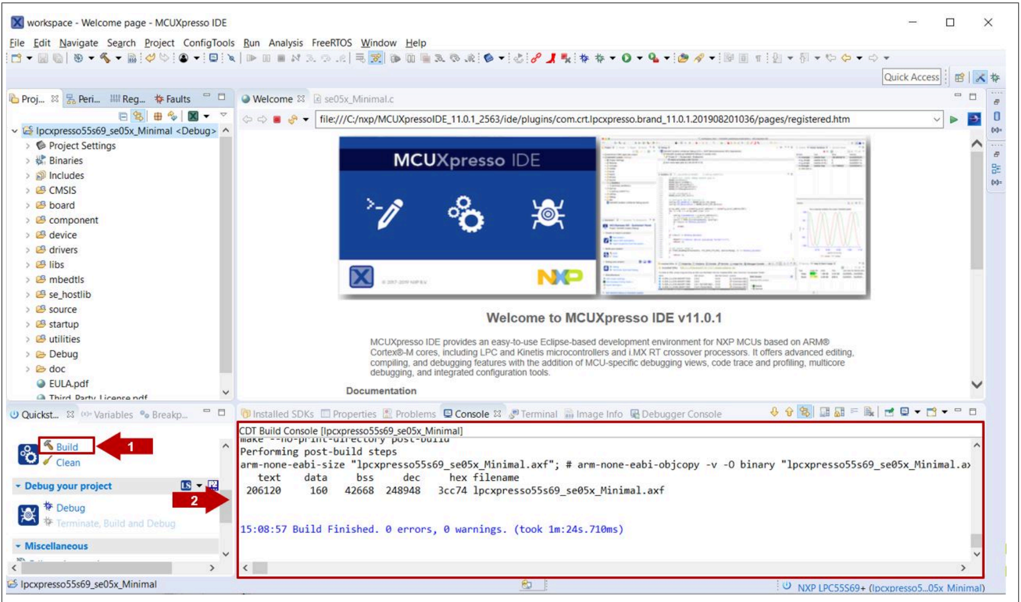
```
Device Friendly Name: mbed Serial Port (COM44)
Device Instance ID: USB\VID_0D28&PID_0204&MI_01\024000
Device Manufacturer: mbed
Provider Name: mbed
Driver Date: 8-24-2009
Driver Version: 2.0.2.0
```

Figure 12. TeraTerm setup

3. **Note:** The default build configuration of the Plug & Trust middleware ≥ `V04.02.xx` 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.

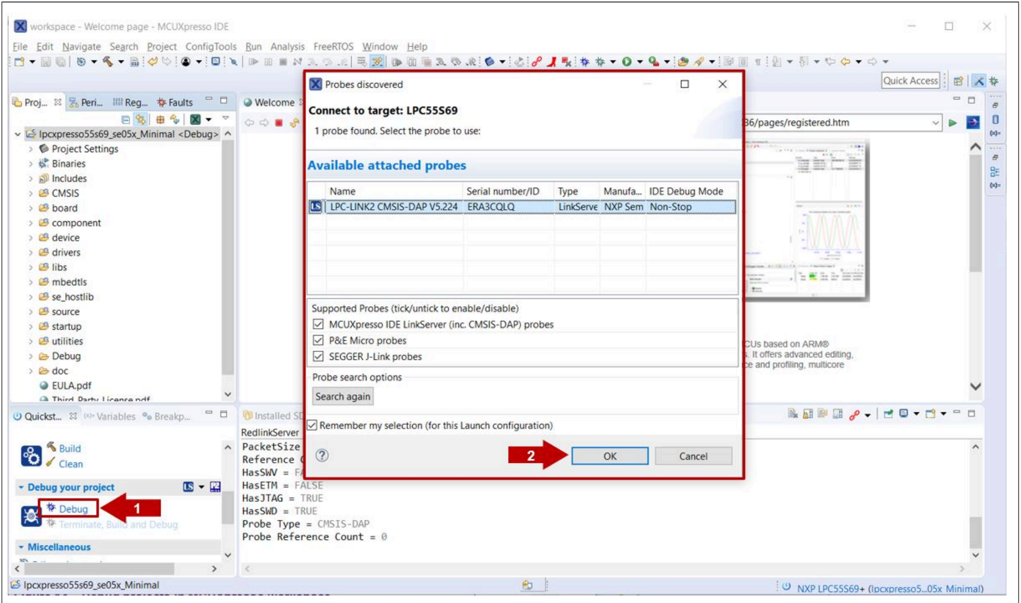
4. Go to the MCUXpresso Quickstart Panel and click the *Build* button as shown in Figure 13. Wait a few seconds and check that the build process has finished successfully in the MCUXpresso console window.


workspace - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
lpcxpresso55s69_se05x_Minimal <Debug>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
libs
mbedtls
se_hostlib
source
startup
utilities
Debug
doc
EULA.pdf
Build
Clean
Debug your project
Debug
Terminate, Build and Debug
Miscellaneous
1
2
Welcome to MCUXpresso IDE v11.0.1
Installed SDKs
Properties
Problems
Console
Terminal
Image Info
Debugger Console
NXP LPC55S69+ (lpcxpresso5...05x_Minimal)

```
CDT Build Console [lpcxpresso55s69_se05x_Minimal]
make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "lpcxpresso55s69_se05x_Minimal.axf"; # arm-none-eabi-objcopy -v -O binary "lpcxpresso55s69_se05x_Minimal.a
   text    data     bss     dec     hex filename
 206120     160   42668  248948   3cc74 lpcxpresso55s69_se05x_Minimal.axf


15:08:57 Build Finished. 0 errors, 0 warnings. (took 1m:24s.710ms)
```

Figure 13. Build projects in MCUXpresso workspace

5. Go to the MCUXpresso Quickstart Panel and click the *Debug* button as shown in Figure 14. 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.

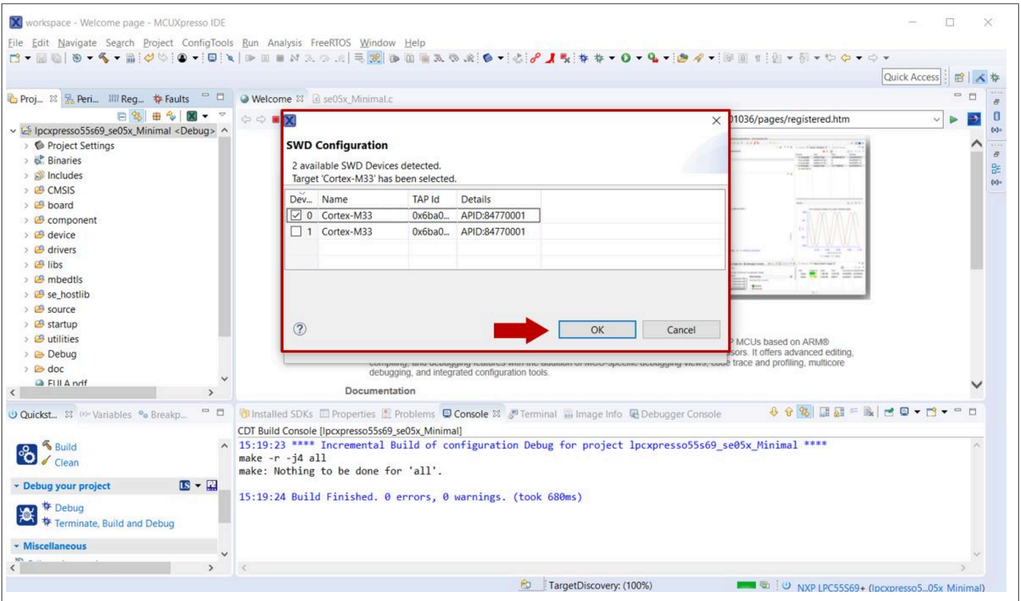

workspace - Welcome page - MCUXpresso IDE
Probes discovered
Connect to target: LPC55S69
1 probe found. Select the probe to use:
Available attached probes
Supported Probes (tick/untick to enable/disable)
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
P&E Micro probes
SEGGER J-Link probes
Probe search options
Search again
Remember my selection (for this Launch configuration)
OK
Cancel
1
2
Build
Clean
Debug your project
Debug
Miscellaneous

Name	Serial number/ID	Type	Manufa...	IDE Debug Mode
LPC-LINK2 CMSIS-DAP V5.224	ERA3CQLQ	LinkServe	NXP Sem	Non-Stop

```
RedlinkServer
PacketSize
Reference
HasSWV = FA
HasETM = FALSE
HasJTAG = TRUE
HasSWD = TRUE
Probe Type = CMSIS-DAP
Probe Reference Count = 0
```

Figure 14. Debug projects in MCUXpresso workspace

6. You might be asked to select the SWD device configuration. You can use the default one as shown in Figure 15:



SWD Configuration

2 available SWD Devices detected.
Target 'Cortex-M33' has been selected.

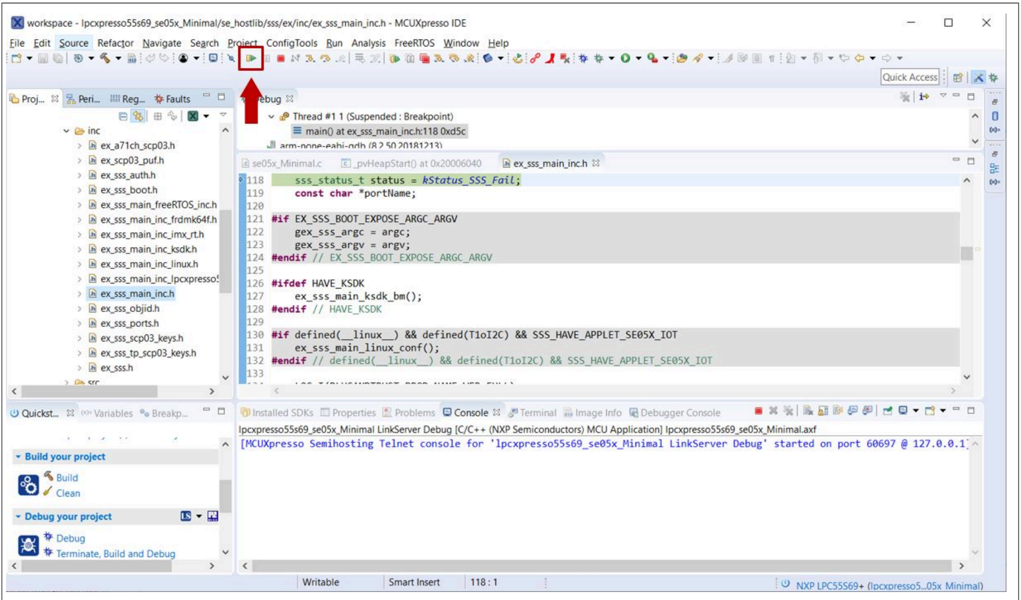
Dev...	Name	TAP Id	Details
✓ 0	Cortex-M33	0x6ba0...	APID:84770001
1	Cortex-M33	0x6ba0...	APID:84770001

CDT Build Console [lpcxpresso55s69_se05x_Minimal]

```
15:19:23 **** Incremental Build of configuration Debug for project lpcxpresso55s69_se05x_Minimal ****
make -r -j4 all
make: Nothing to be done for 'all'.

15:19:24 Build Finished. 0 errors, 0 warnings. (took 680ms)
```

Figure 15. Select SWD device configuration

7. 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 16.



Thread #1 1 (Suspended : Breakpoint)
main() at ex_sss_main_inc.h:118 0xd5c

```
    sss_status_t status = kStatus_SSS_Fail;
    const char *portName;

#if EX_SSS_BOOT_EXPOSE_ARGC_ARGV
    gex_sss_argc = argc;
    gex_sss_argv = argv;
#endif // EX_SSS_BOOT_EXPOSE_ARGC_ARGV

#ifdef HAVE_KSDK
    ex_sss_main_ksdk_bm();
#endif // HAVE_KSDK

#if defined(__linux__) && defined(T1oI2C) && SSS_HAVE_APPLET_SE05X_IOT
    ex_sss_main_linux_conf();
#endif // defined(__linux__) && defined(T1oI2C) && SSS_HAVE_APPLET_SE05X_IOT
```

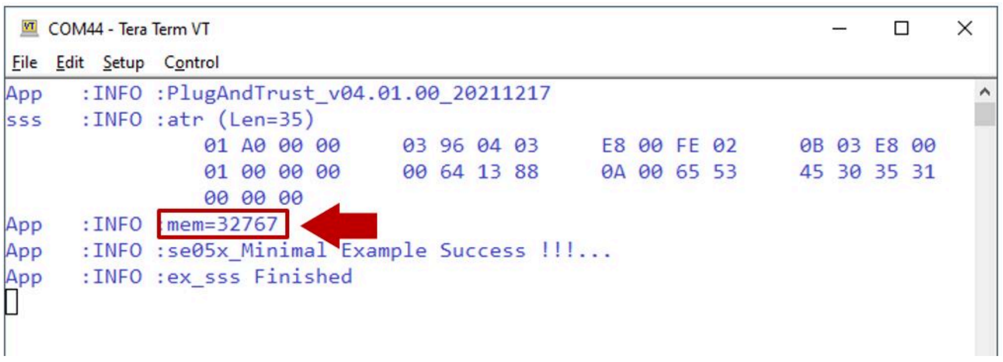
lpcxpresso55s69_se05x_Minimal LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application] lpcxpresso55s69_se05x_Minimal.axf

```
[MCUXpresso Semihosting Telnet console for 'lpcxpresso55s69_se05x_Minimal LinkServer Debug' started on port 60697 @ 127.0.0.1]
```

Figure 16. Run projects in MCUXpresso workspace

8. 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 17:



```
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 17. TeraTerm logs - `se05x_Minimal` project example

9. The same operation can be repeated with any of the other 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.

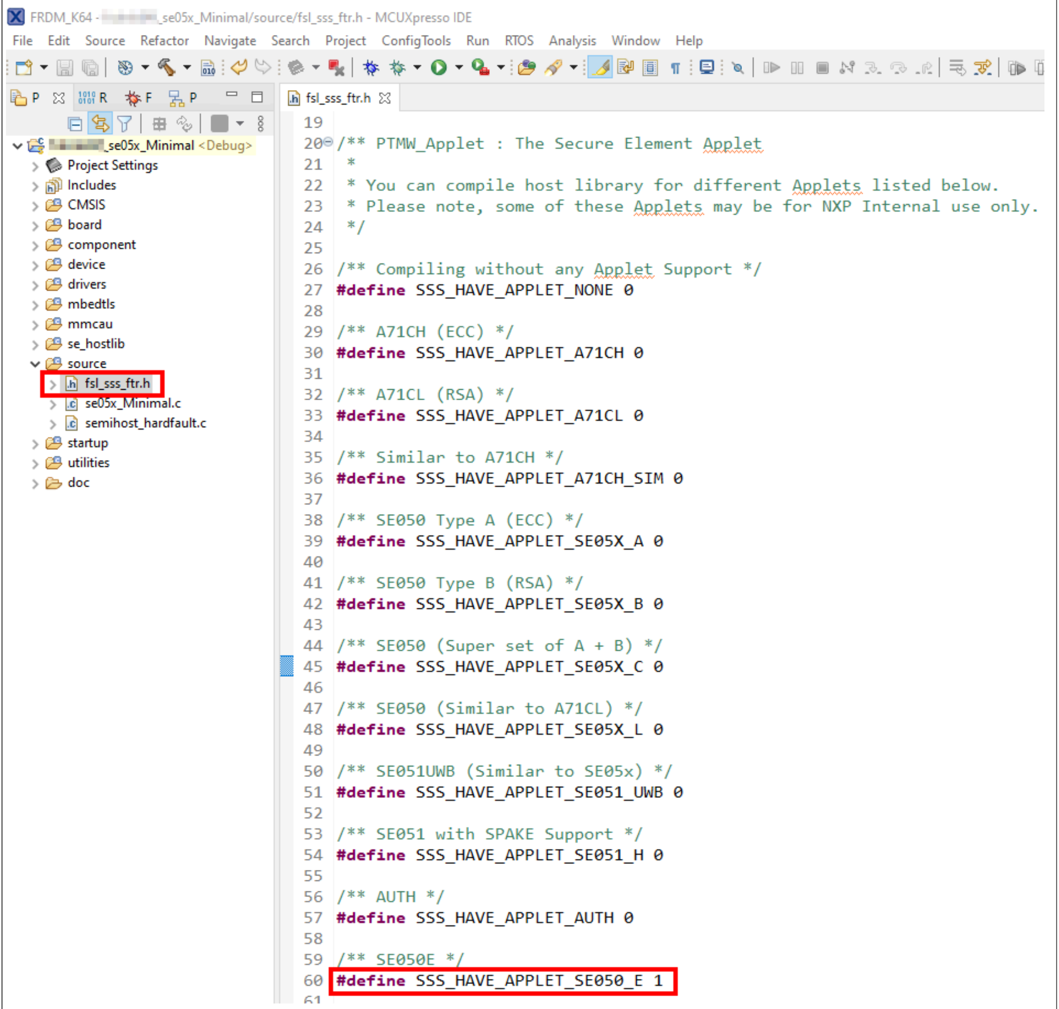
The Plug & Trust Middleware uses the feature file `fsl_sss_ftr.h` to select a dedicated EdgeLock product IC and the corresponding IoT applet or Authenticator application. The `fsl_sss_ftr.h` header file is located in the project source folder.

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.


FRDM_K64 - [redacted] se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
[redacted] se05x_Minimal <Debug>
Project Settings
Includes
CMSIS
board
component
device
drivers
mbedtls
mmcau
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
semihost_hardfault.c
startup
utilities
doc
fsl_sss_ftr.h

```
/** PTMW_Applet : The Secure Element Applet
 *
 * You can compile host library for different Applets listed below.
 * Please note, some of these Applets may be for NXP Internal use only.
 */

/** Compiling without any Applet Support */
#define SSS_HAVE_APPLET_NONE 0

/** A71CH (ECC) */
#define SSS_HAVE_APPLET_A71CH 0

/** A71CL (RSA) */
#define SSS_HAVE_APPLET_A71CL 0

/** Similar to A71CH */
#define SSS_HAVE_APPLET_A71CH_SIM 0

/** SE050 Type A (ECC) */
#define SSS_HAVE_APPLET_SE05X_A 0

/** SE050 Type B (RSA) */
#define SSS_HAVE_APPLET_SE05X_B 0

/** SE050 (Super set of A + B) */
#define SSS_HAVE_APPLET_SE05X_C 0

/** SE050 (Similar to A71CL) */
#define SSS_HAVE_APPLET_SE05X_L 0

/** SE051UWB (Similar to SE05x) */
#define SSS_HAVE_APPLET_SE051_UWB 0

/** SE051 with SPAKE Support */
#define SSS_HAVE_APPLET_SE051_H 0

/** AUTH */
#define SSS_HAVE_APPLET_AUTH 0

/** SE050E */
#define SSS_HAVE_APPLET_SE050_E 1
```

Figure 18. 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_ SCP_NONE` or `SSS_ HAVE_ SCP_ SCP03_ SSS`	disabled
SE050E2	A921						

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_SCP03_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_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 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_ 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 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_ 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 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_ 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

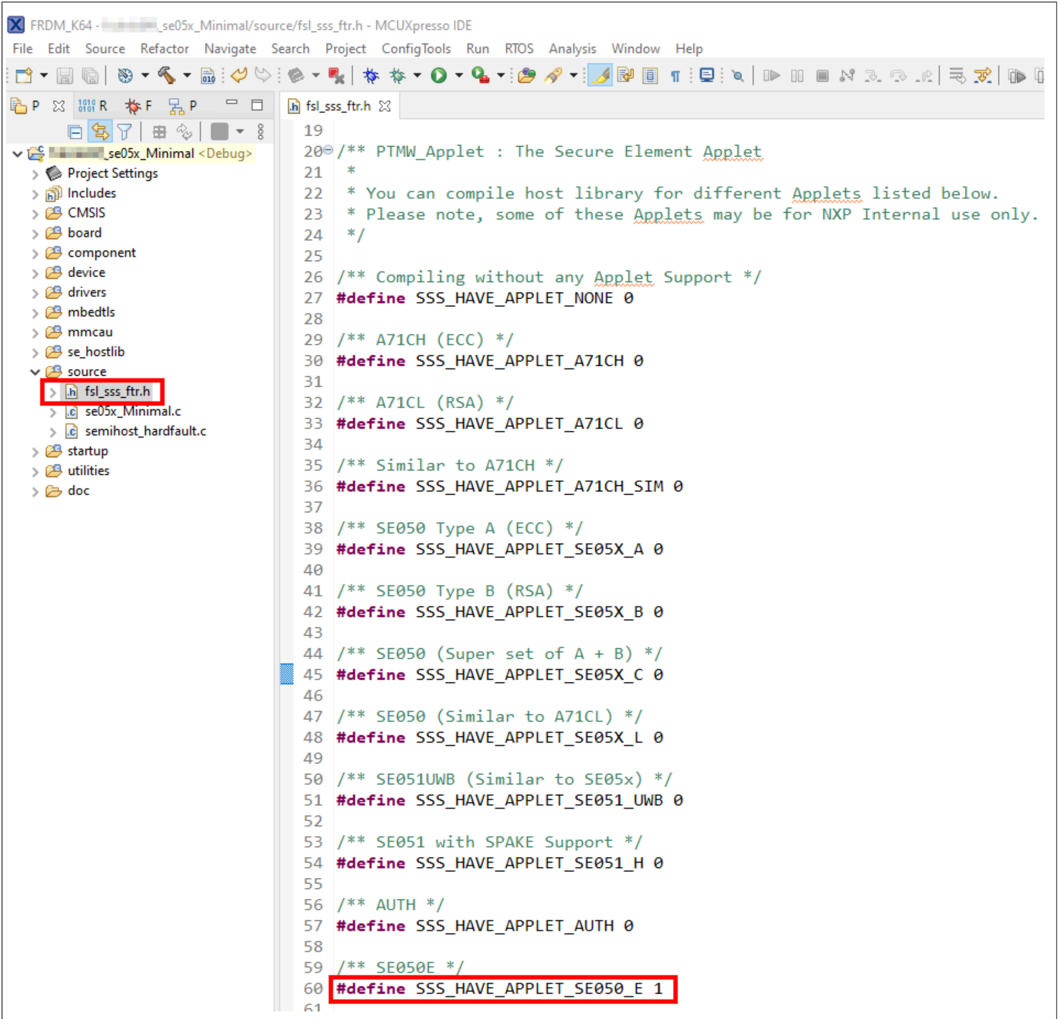
Table 7. Feature file `fsl_sss_ftr.h` settings for A5000 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ 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 3.

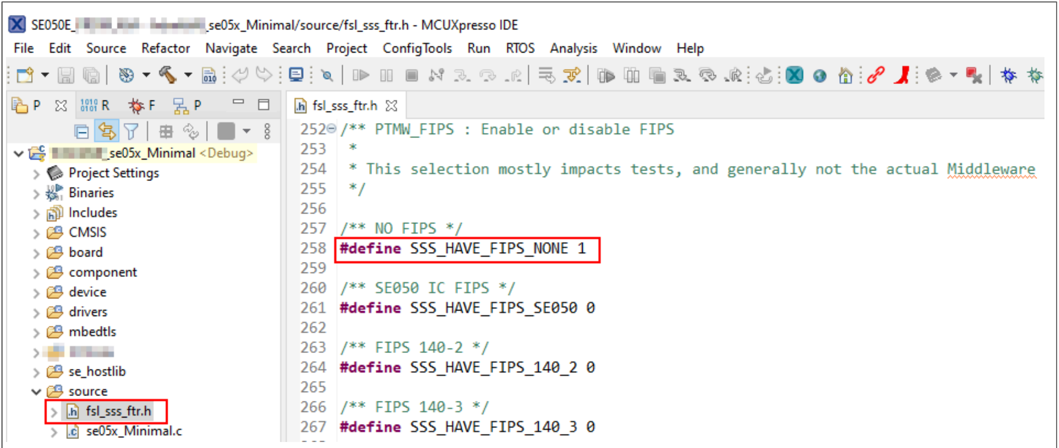
1. Select the Applet variant SE050E.


FRDM_K64 - [redacted]_se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
[redacted]_se05x_Minimal <Debug>
Project Settings
Includes
CMSIS
board
component
device
drivers
mbedtls
mmcau
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
semihost_hardfault.c
startup
utilities
doc
fsl_sss_ftr.h

```
/** PTMW_Applet : The Secure Element Applet
 *
 * You can compile host library for different Applets listed below.
 * Please note, some of these Applets may be for NXP Internal use only.
 */

/** Compiling without any Applet Support */
#define SSS_HAVE_APPLET_NONE 0

/** A71CH (ECC) */
#define SSS_HAVE_APPLET_A71CH 0

/** A71CL (RSA) */
#define SSS_HAVE_APPLET_A71CL 0

/** Similar to A71CH */
#define SSS_HAVE_APPLET_A71CH_SIM 0

/** SE050 Type A (ECC) */
#define SSS_HAVE_APPLET_SE05X_A 0

/** SE050 Type B (RSA) */
#define SSS_HAVE_APPLET_SE05X_B 0

/** SE050 (Super set of A + B) */
#define SSS_HAVE_APPLET_SE05X_C 0

/** SE050 (Similar to A71CL) */
#define SSS_HAVE_APPLET_SE05X_L 0

/** SE051UWB (Similar to SE05x) */
#define SSS_HAVE_APPLET_SE051_UWB 0

/** SE051 with SPAKE Support */
#define SSS_HAVE_APPLET_SE051_H 0

/** AUTH */
#define SSS_HAVE_APPLET_AUTH 0

/** SE050E */
#define SSS_HAVE_APPLET_SE050_E 1
```

Figure 19. Feature file fsl_sss_ftr.h - PTMW_Applet

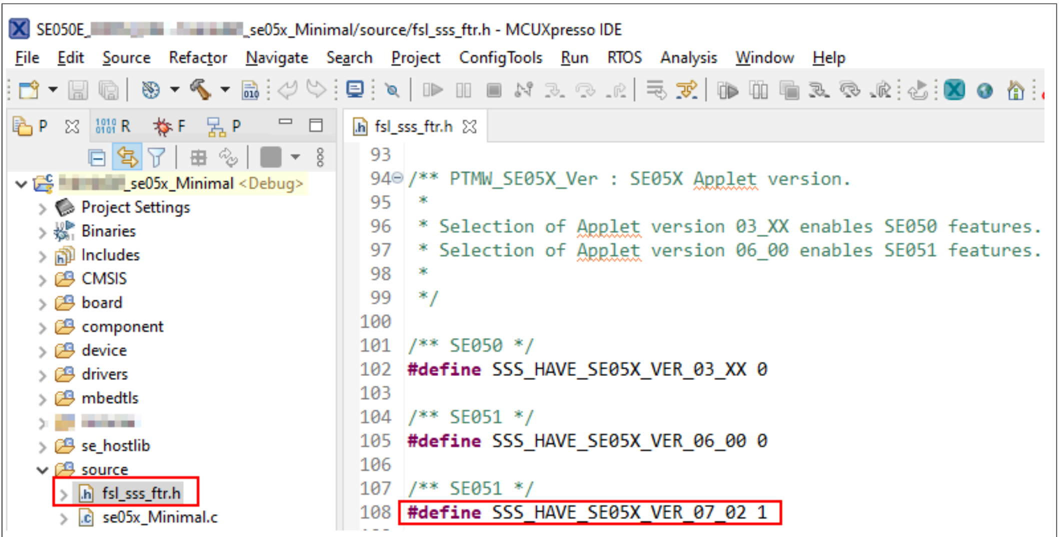
2. Select FIPS none.


SE050E_[redacted]_se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
[redacted]_se05x_Minimal <Debug>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
fsl_sss_ftr.h

```
/** PTMW_FIPS : Enable or disable FIPS
 *
 * This selection mostly impacts tests, and generally not the actual Middleware
 */

/** NO FIPS */
#define SSS_HAVE_FIPS_NONE 1

/** SE050 IC FIPS */
#define SSS_HAVE_FIPS_SE050 0

/** FIPS 140-2 */
#define SSS_HAVE_FIPS_140_2 0

/** FIPS 140-3 */
#define SSS_HAVE_FIPS_140_3 0
```

Figure 20. Feature file fsl_sss_ftr.h - Option PTMW_FIPS

3. Select Applet version 7.02.


SE050E_[redacted]_se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
P R F P
fsl_sss_ftr.h
[redacted]_se05x_Minimal <Debug>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
[redacted]
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c

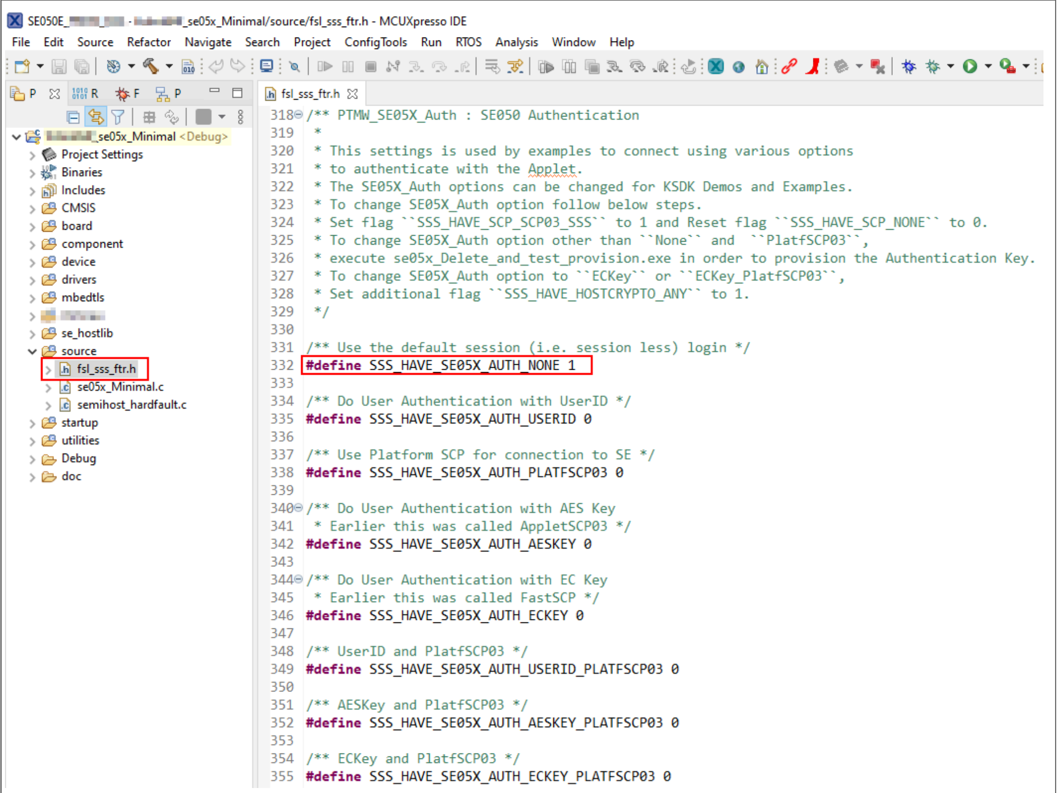
```
/** PTMW_SE05X_Ver : SE05X Applet version.
 *
 * Selection of Applet version 03_XX enables SE050 features.
 * Selection of Applet version 06_00 enables SE051 features.
 *
 */

/** SE050 */
#define SSS_HAVE_SE05X_VER_03_XX 0

/** SE051 */
#define SSS_HAVE_SE05X_VER_06_00 0

/** SE051 */
#define SSS_HAVE_SE05X_VER_07_02 1
```

Figure 21. 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:

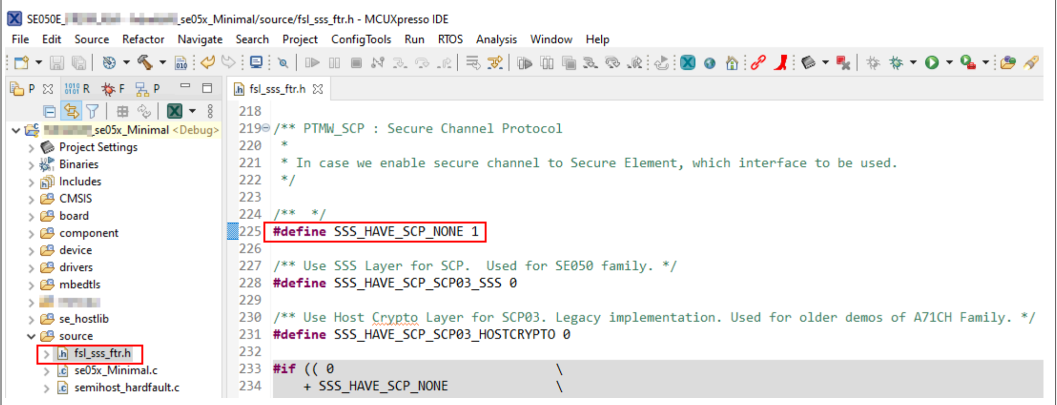
- Set the `#define SSS_HAVE_SE05X_AUTH_NONE` option to `1` and disable all other options be setting the flags to `0`.
- Set the `#define SSS_HAVE_SCP_NONE` option to `1` and disable all other options be setting the flags to `0`.

How to enable Platform SCP is described in Section 6.3.


SE050E_[redacted] - [redacted]_se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
fsl_sss_ftr.h
[redacted]_se05x_Minimal <Debug>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
[redacted]
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
semihost_hardfault.c
startup
utilities
Debug
doc

```
/** PTMW_SE05X_Auth : SE050 Authentication
 *
 * This settings is used by examples to connect using various options
 * to authenticate with the Applet.
 * The SE05X_Auth options can be changed for KSDK Demos and Examples.
 * To change SE05X_Auth option follow below steps.
 * Set flag ``SSS_HAVE_SCP_SCP03_SSS`` to 1 and Reset flag ``SSS_HAVE_SCP_NONE`` to 0.
 * To change SE05X_Auth option other than ``None`` and  ``PlatfSCP03``,
 * execute se05x_Delete_and_test_provision.exe in order to provision the Authentication Key.
 * To change SE05X_Auth option to ``ECKey`` or ``ECKey_PlatfSCP03``,
 * Set additional flag ``SSS_HAVE_HOSTCRYPTO_ANY`` to 1.
 */

/** Use the default session (i.e. session less) login */
#define SSS_HAVE_SE05X_AUTH_NONE 1

/** Do User Authentication with UserID */
#define SSS_HAVE_SE05X_AUTH_USERID 0

/** Use Platform SCP for connection to SE */
#define SSS_HAVE_SE05X_AUTH_PLATFSCP03 0

/** Do User Authentication with AES Key
 * Earlier this was called AppletSCP03 */
#define SSS_HAVE_SE05X_AUTH_AESKEY 0

/** Do User Authentication with EC Key
 * Earlier this was called FastSCP */
#define SSS_HAVE_SE05X_AUTH_ECKEY 0

/** UserID and PlatfSCP03 */
#define SSS_HAVE_SE05X_AUTH_USERID_PLATFSCP03 0

/** AESKey and PlatfSCP03 */
#define SSS_HAVE_SE05X_AUTH_AESKEY_PLATFSCP03 0

/** ECKey and PlatfSCP03 */
#define SSS_HAVE_SE05X_AUTH_ECKEY_PLATFSCP03 0
```

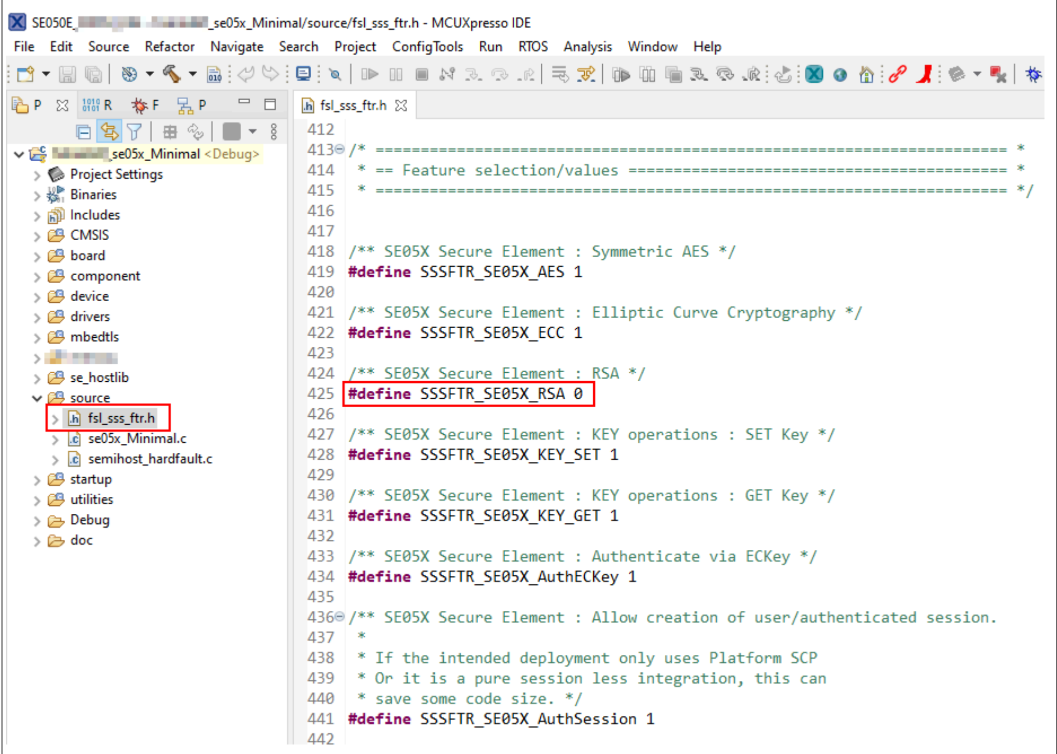
Figure 22. Feature file fsl_sss_ftr.h - Option PTMW_AUTH - Plain communication


SE050E_[redacted] - [redacted]_se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
fsl_sss_ftr.h
[redacted]_se05x_Minimal <Debug>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
[redacted]
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
semihost_hardfault.c

```

/** PTMW_SCP : Secure Channel Protocol
 *
 * In case we enable secure channel to Secure Element, which interface to be used.
 */

/**  */
#define SSS_HAVE_SCP_NONE 1

/** Use SSS Layer for SCP.  Used for SE050 family. */
#define SSS_HAVE_SCP_SCP03_SSS 0

/** Use Host Crypto Layer for SCP03. Legacy implementation. Used for older demos of A71CH Family. */
#define SSS_HAVE_SCP_SCP03_HOSTCRYPTO 0

#if (( 0                              \
    + SSS_HAVE_SCP_NONE               \
```

Figure 23. 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.


SE050E_[redacted]_se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
P R F P
fsl_sss_ftr.h
[redacted]_se05x_Minimal <Debug>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
[redacted]
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
semihost_hardfault.c
startup
utilities
Debug
doc

```
/* ====================================================================== *
 * == Feature selection/values ========================================= *
 * ====================================================================== */


/** SE05X Secure Element : Symmetric AES */
#define SSSFTR_SE05X_AES 1

/** SE05X Secure Element : Elliptic Curve Cryptography */
#define SSSFTR_SE05X_ECC 1

/** SE05X Secure Element : RSA */
#define SSSFTR_SE05X_RSA 0

/** SE05X Secure Element : KEY operations : SET Key */
#define SSSFTR_SE05X_KEY_SET 1

/** SE05X Secure Element : KEY operations : GET Key */
#define SSSFTR_SE05X_KEY_GET 1

/** SE05X Secure Element : Authenticate via ECKey */
#define SSSFTR_SE05X_AuthECKey 1

/** SE05X Secure Element : Allow creation of user/authenticated session.
 *
 * If the intended deployment only uses Platform SCP
 * Or it is a pure session less integration, this can
 * save some code size. */
#define SSSFTR_SE05X_AuthSession 1

```

Figure 24. 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 the 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:

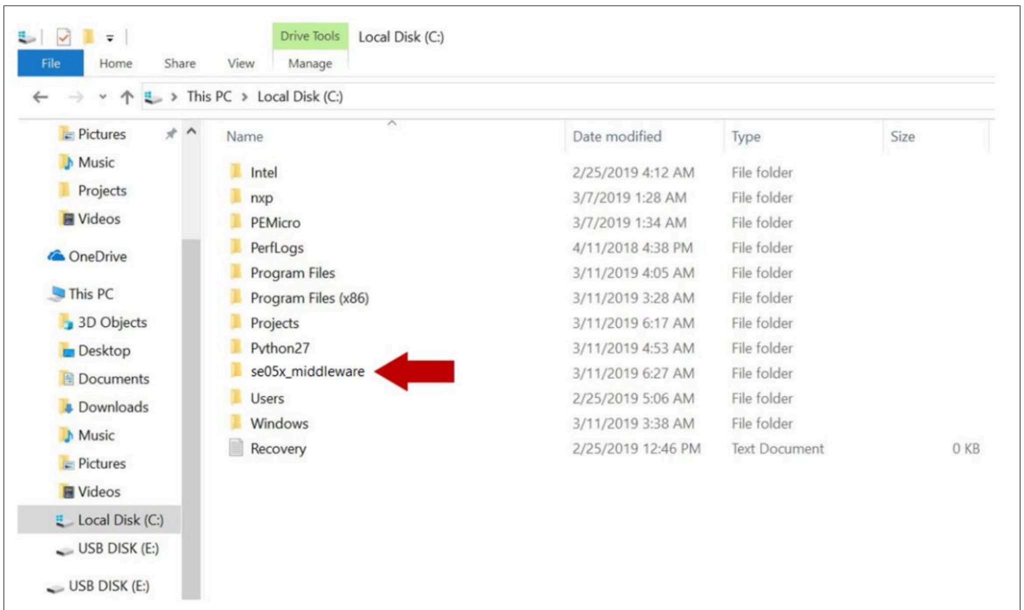
1. MCUXpresso IDE. Check Section 7 for detailed installation instructions.
2. CMake. Check Section 8 for detailed installation instructions.
3. Python ≥ 3.7.x and ≤ 3.9.x 32-bit version. Check Section 9 for detailed installation instructions.
4. TeraTerm (or an equivalent serial application). You can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

5.2 Download 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.

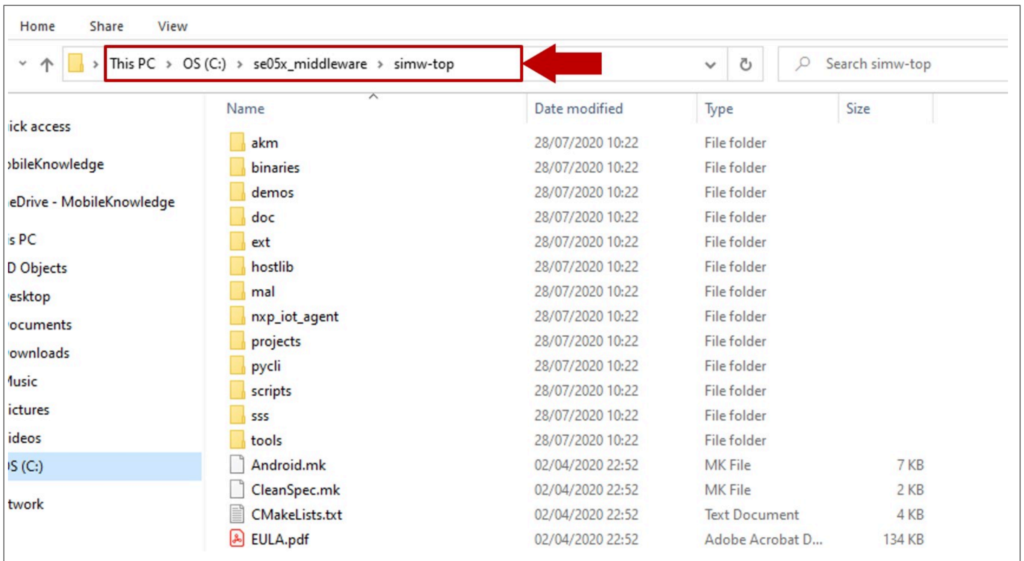
2. Create a folder called ***se05x_middleware*** in C: directory as shown in Figure 25:



Name	Date modified	Type	Size
Intel	2/25/2019 4:12 AM	File folder	
nxp	3/7/2019 1:28 AM	File folder	
PEMicro	3/7/2019 1:34 AM	File folder	
PerfLogs	4/11/2018 4:38 PM	File folder	
Program Files	3/11/2019 4:05 AM	File folder	
Program Files (x86)	3/11/2019 3:28 AM	File folder	
Projects	3/11/2019 6:17 AM	File folder	
Python27	3/11/2019 4:53 AM	File folder	
se05x_middleware	3/11/2019 6:27 AM	File folder	
Users	2/25/2019 5:06 AM	File folder	
Windows	3/11/2019 3:38 AM	File folder	
Recovery	2/25/2019 12:46 PM	Text Document	0 KB

Figure 25. 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*** created. The contents of the ***simw-top*** directory should look as they appear in Figure 26:



Name	Date modified	Type	Size
akm	28/07/2020 10:22	File folder	
binaries	28/07/2020 10:22	File folder	
demos	28/07/2020 10:22	File folder	
doc	28/07/2020 10:22	File folder	
ext	28/07/2020 10:22	File folder	
hostlib	28/07/2020 10:22	File folder	
mal	28/07/2020 10:22	File folder	
nxp_iot_agent	28/07/2020 10:22	File folder	
projects	28/07/2020 10:22	File folder	
pycli	28/07/2020 10:22	File folder	
scripts	28/07/2020 10:22	File folder	
sss	28/07/2020 10:22	File folder	
tools	28/07/2020 10:22	File folder	
Android.mk	02/04/2020 22:52	MK File	7 KB
CleanSpec.mk	02/04/2020 22:52	MK File	2 KB
CMakeLists.txt	02/04/2020 22:52	Text Document	4 KB
EULA.pdf	02/04/2020 22:52	Adobe Acrobat D...	134 KB

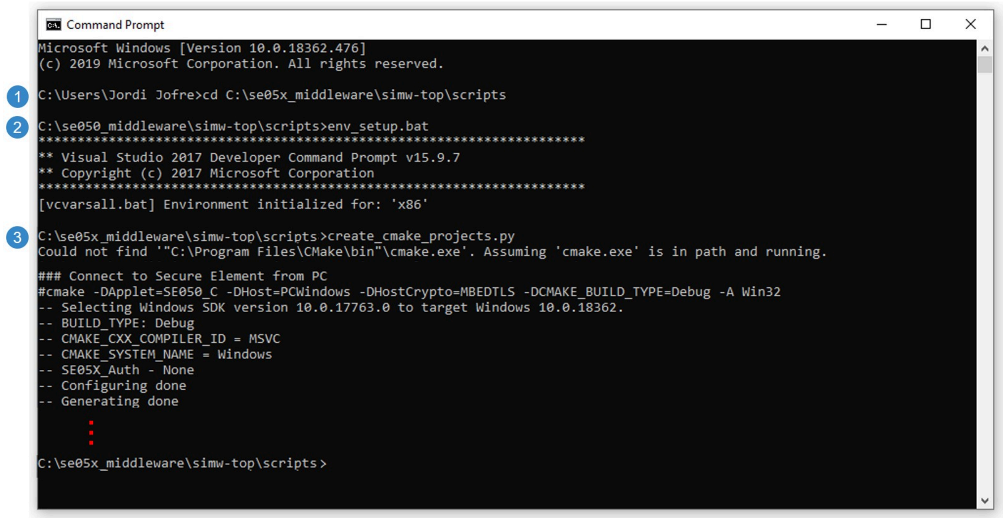
Figure 26. Unzip se05x middleware

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 Plug & Trust middleware project examples

The Plug & Trust middleware uses CMake for building the project examples into your local machine. To build Plug & Trust middleware, open a Command Prompt and follow the steps shown in Figure 27:

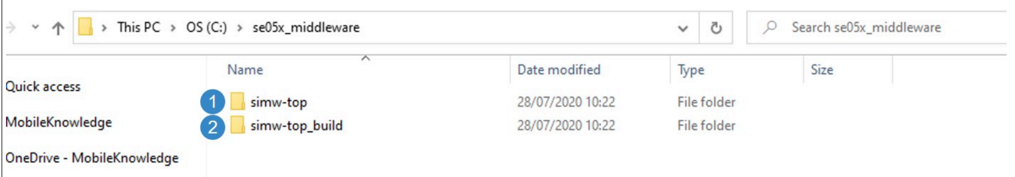
1. Go to the folder where you unzipped the SE05x middleware:
   (1) Send >> `cd C:\se05x_middleware\simw-top\scripts`
2. Define the environment:
   (2) Send >> `env_setup.bat`
3. Generate the Plug & Trust middleware project examples:
   (3) Send >> `create_cmake_projects.py`
   ***Note**: This command may take a few seconds to complete.*



```
Microsoft Windows [Version 10.0.18362.476]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Jordi Jofre>cd C:\se05x_middleware\simw-top\scripts

C:\se050_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\CMake\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 27. Generate Plug & Trust middleware project examples

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 28:



This PC > OS (C:) > se05x_middleware
Search se05x_middleware
Quick access
MobileKnowledge
OneDrive - MobileKnowledge
1
2

Name	Date modified	Type	Size
simw-top	28/07/2020 10:22	File folder	
simw-top_build	28/07/2020 10:22	File folder	

Figure 28. 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 29.
   Note: In this case, do not use the MCUXpresso Quickstart Panel to import project.

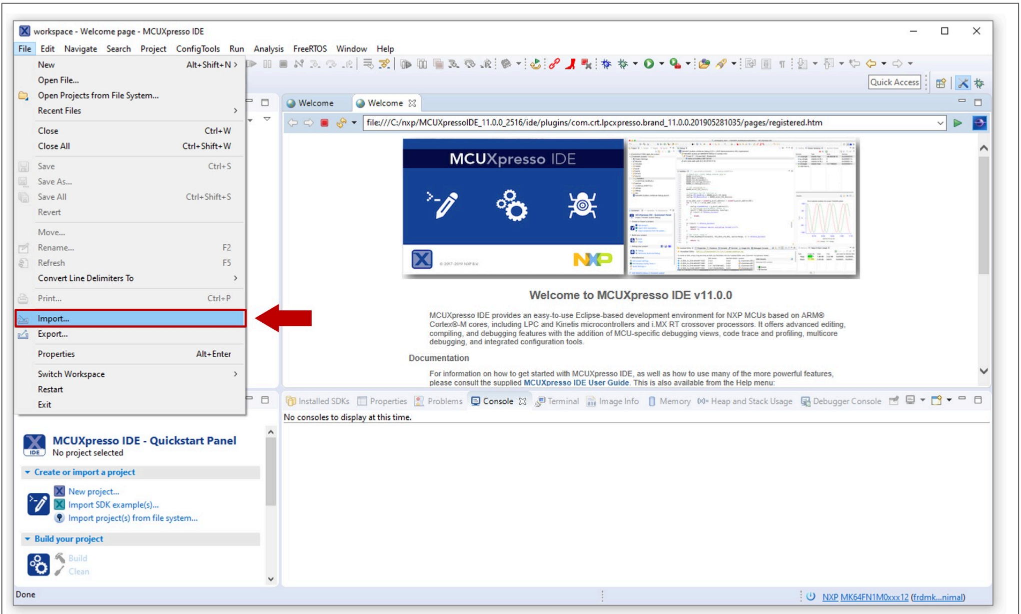


Figure 29. Import a project wizard

2. In the import wizard menu, select import ***"Existing Projects into Workspace"*** from the *General* folder as shown in Figure 30:

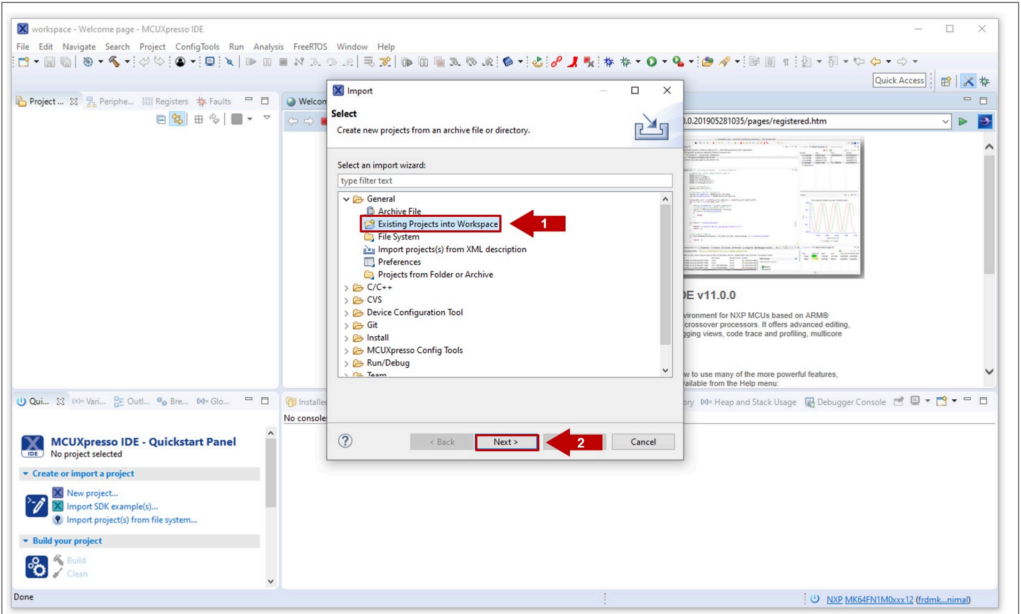

Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Archive File
Existing Projects into Workspace
1
File System
Import projects(s) from XML description
Preferences
Projects from Folder or Archive
C/C++
CVS
Device Configuration Tool
Git
Install
MCUXpresso Config Tools
Run/Debug
Team
< Back
Next >
2
Cancel
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Done

Figure 30. Import a project wizard (II)

3. First, we need to import Plug & Trust middleware project in MCUXpresso. For that, in the *Select root directory* option, browse to *C:\se05x_middleware\simw-top_build* or

browse the location of your Plug & Trust middleware directory and click *Select folder* as shown in Figure 31:

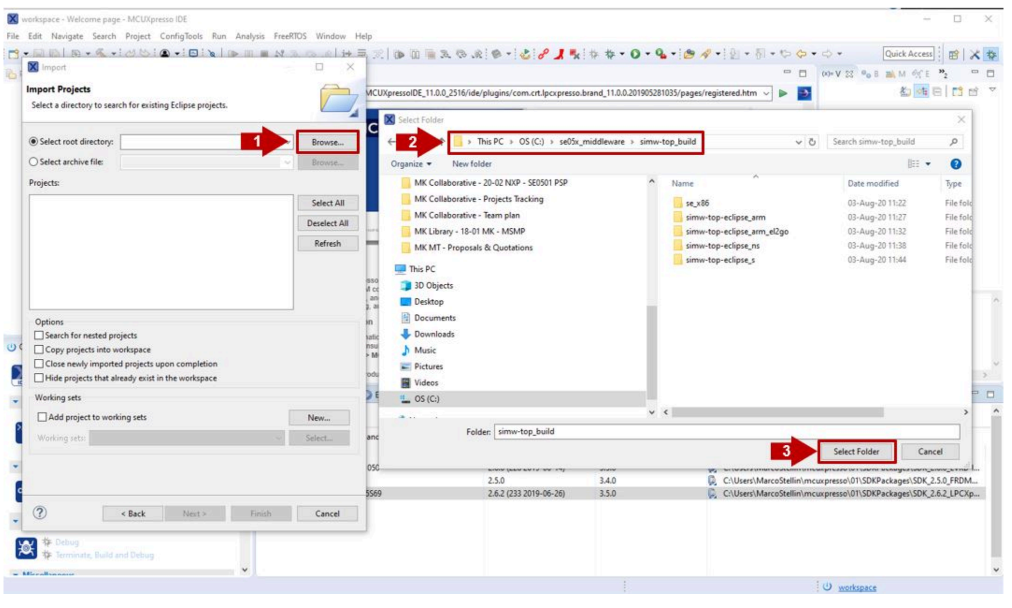

Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Projects:
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Close newly imported projects upon completion
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
< Back
Next >
Finish
Cancel
Select Folder
This PC > OS (C:) > se05x_middleware > simw-top_build
Organize
New folder
se_x86
simw-top-eclipse_arm
simw-top-eclipse_arm_el2go
simw-top-eclipse_ns
simw-top-eclipse_s
Folder: simw-top_build
Select Folder
Cancel
1
2
3

Figure 31. 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. Select it and click on the *Finish* button to import this project into your workspace as shown in Figure 32:

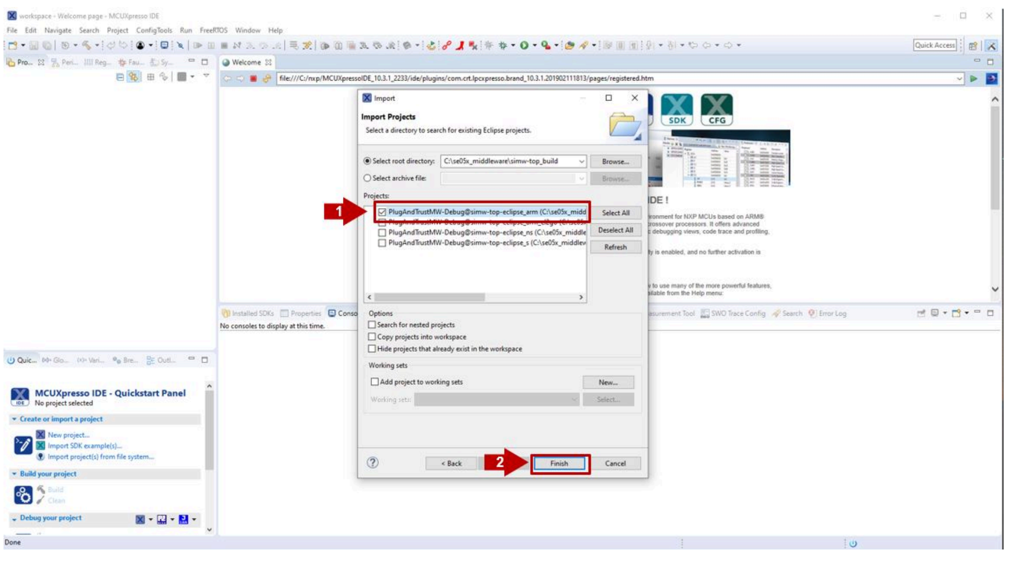

Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory: C:\se05x_middleware\simw-top_build
Browse...
Select archive file:
Projects:
PlugAndTrustMW-Debug@simw-top-eclipse_arm (C:\se05x_midd
PlugAndTrustMW-Debug@simw-top-eclipse_ns (C:\se05x_middle
PlugAndTrustMW-Debug@simw-top-eclipse_s (C:\se05x_middlew
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
< Back
Finish
Cancel
MCUXpresso IDE - Quickstart Panel
1
2

Figure 32. Import Plug & Trust middleware

5. The *PlugAndTrustMW* project should now be imported in your workspace as shown in Figure 33:

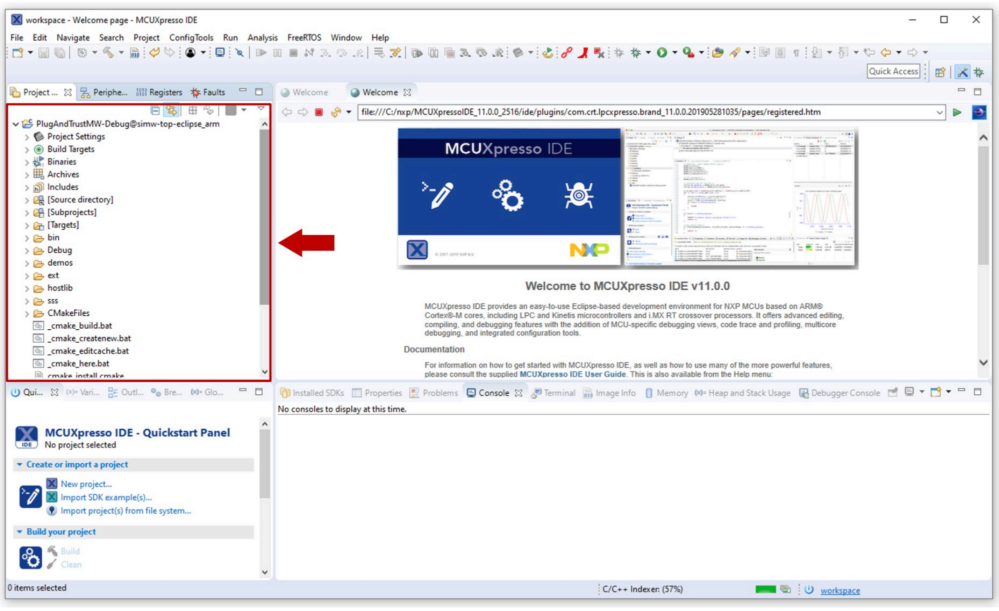

workspace - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Quick Access
Project ... Periphe... Registers Faults
PlugAndTrustMW-Debug@simw-top-eclipse_arm
Project Settings
Build Targets
Binaries
Archives
Includes
[Source directory]
[Subprojects]
[Targets]
bin
Debug
demos
ext
hostlib
sss
CMakeFiles
_cmake_build.bat
_cmake_createnew.bat
_cmake_editcache.bat
_cmake_here.bat
Welcome
file:///C:/nxp/MCUXpressoIDE_11.0.0_2516/ide/plugins/com.crt.lpcxpresso.brand_11.0.0.201905281035/pages/registered.htm
MCUXpresso IDE
Welcome to MCUXpresso IDE v11.0.0
MCUXpresso IDE provides an easy-to-use Eclipse-based development environment for NXP MCUs based on ARM® Cortex®-M cores, including LPC and Kinetis microcontrollers and i.MX RT crossover processors. It offers advanced editing, compiling, and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools.
Documentation
For information on how to get started with MCUXpresso IDE, as well as how to use many of the more powerful features, please consult the supplied MCUXpresso IDE User Guide. This is also available from the Help menu:
Qui... Vari... Outl... Bre... Glo...
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Installed SDKs Properties Problems Console Terminal Image Info Memory Heap and Stack Usage Debugger Console
No consoles to display at this time.
0 items selected
C/C++ Indexer: (57%)
workspace

Figure 33. Plug & Trust middleware imported in workspace

5.5 Import *cmake_projects_lpc55s* project example in MCUXpresso workspace

After importing the *PlugAndTrustMW* project example in MCUXpresso, we need to import the *cmake_projects_lpc55s* project example. Follow these steps:

1. Go to *File* → *Import* using the top bar menu as shown in Figure 34.
   Note: In this case, do not use the MCUXpresso Quickstart Panel to import project.

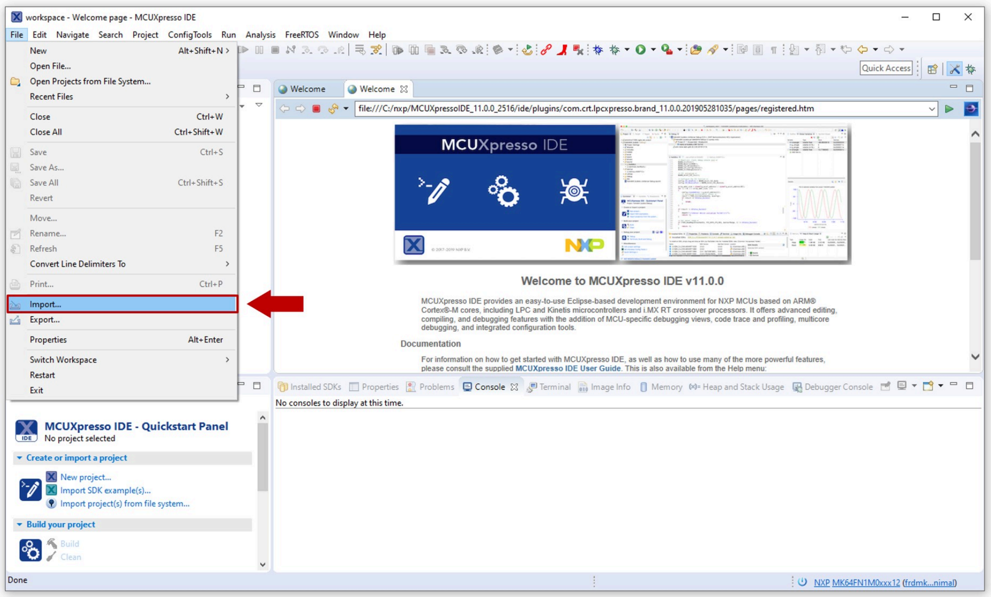


Figure 34. Import a project wizard

2. In the import wizard menu, select import ***"Existing Projects into Workspace"*** from the *General* folder as shown in Figure 35:

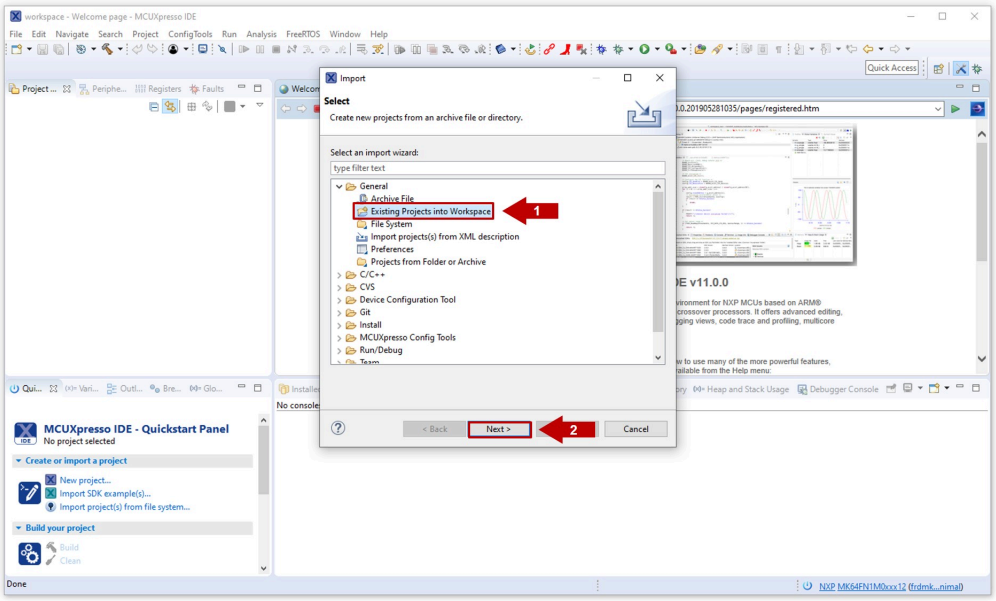

Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
General
Archive File
Existing Projects into Workspace
File System
Import projects(s) from XML description
Preferences
Projects from Folder or Archive
C/C++
CVS
Device Configuration Tool
Git
Install
MCUXpresso Config Tools
Run/Debug
1
< Back
Next >
2
Cancel

Figure 35. 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 LPC55S69 projects directory. Choose the *cmake_projects_lpc55s* project and click *Select folder* as shown in Figure 36:

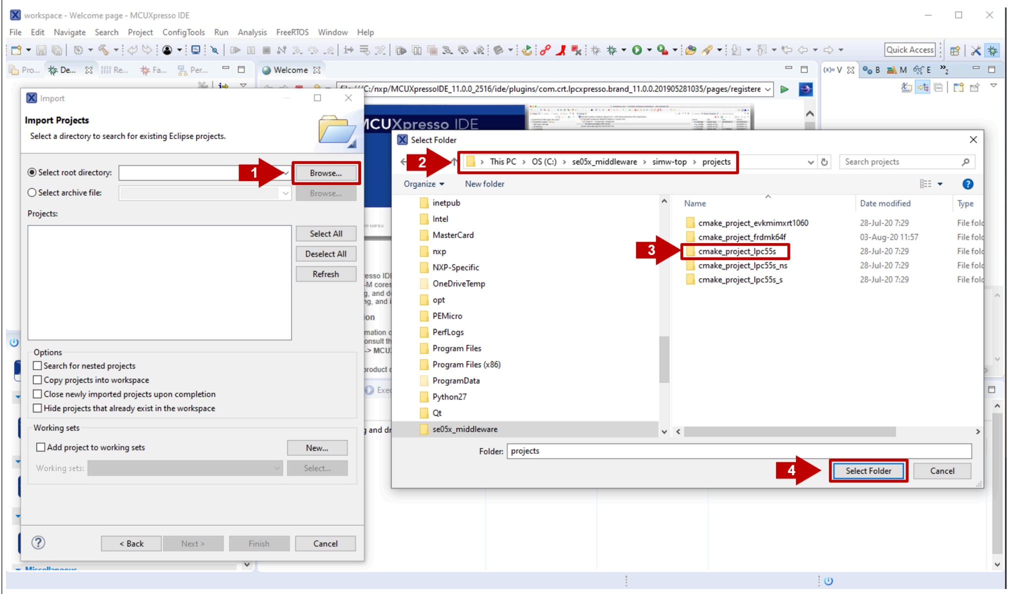


Figure 36. Select LPC55S69 projects folder

4. After selecting *C:\se05x_middleware\simw-top\projects* folder, the *cmake_projects_lpc55s* project should be visible in the Projects area. Click *Finish* button to import this project into your workspace as shown in Figure 37:

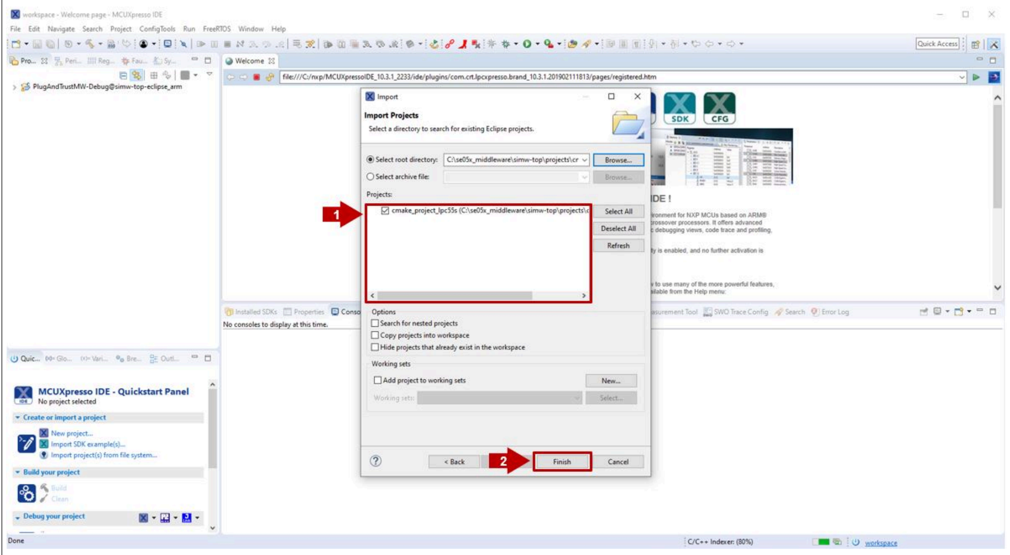

Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory: C:\se05x_middleware\simw-top\projects\cr
Select archive file:
Browse...
Projects:
cmake_project_lpc55s (C:\se05x_middleware\simw-top\projects\
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
< Back
Finish
Cancel
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Debug your project
1
2

Figure 37. Import LPC55S69 in workspace

5. Both The *PlugAndTrustMW* and *cmake_projects_lpc55s* projects should now be imported in your workspace as shown in Figure 38:

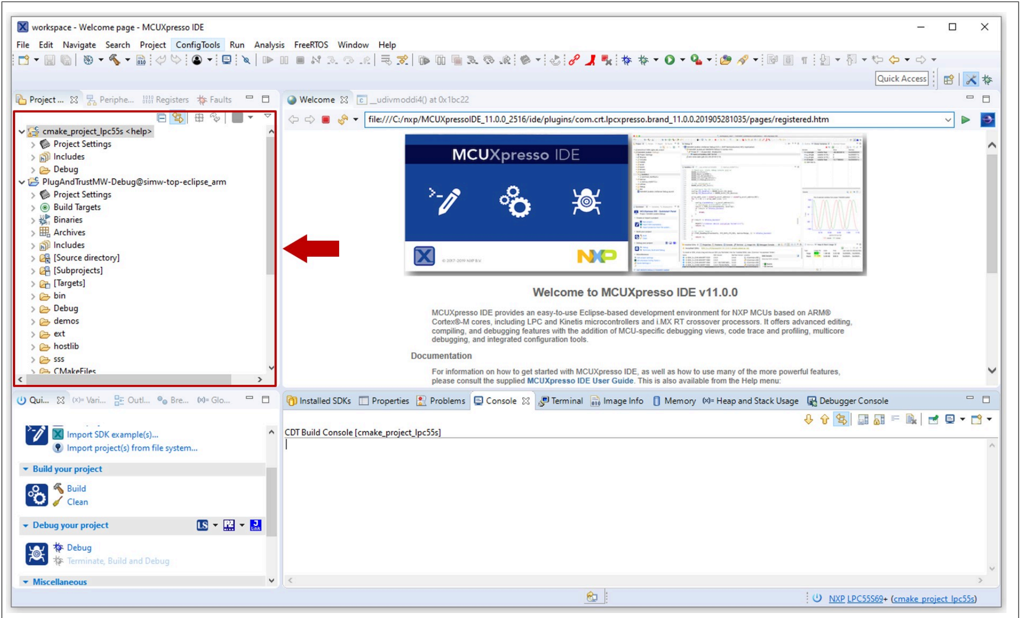

workspace - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
cmake_project_lpc55s <help>
Project Settings
Includes
Debug
PlugAndTrustMW-Debug@simw-top-eclipse_arm
Build Targets
Binaries
Archives
[Source directory]
[Subprojects]
[Targets]
bin
demos
ext
hostlib
sss
CMakeFiles
Welcome to MCUXpresso IDE v11.0.0
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug
Miscellaneous
CDT Build Console [cmake_project_lpc55s]

Figure 38. LPC55S69 imported in workspace

The two projects need to be imported in the same MCUXpresso workspace. The `cmake_project_lpc55s` 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 Execute Plug & Trust middleware examples

This section explains how to:

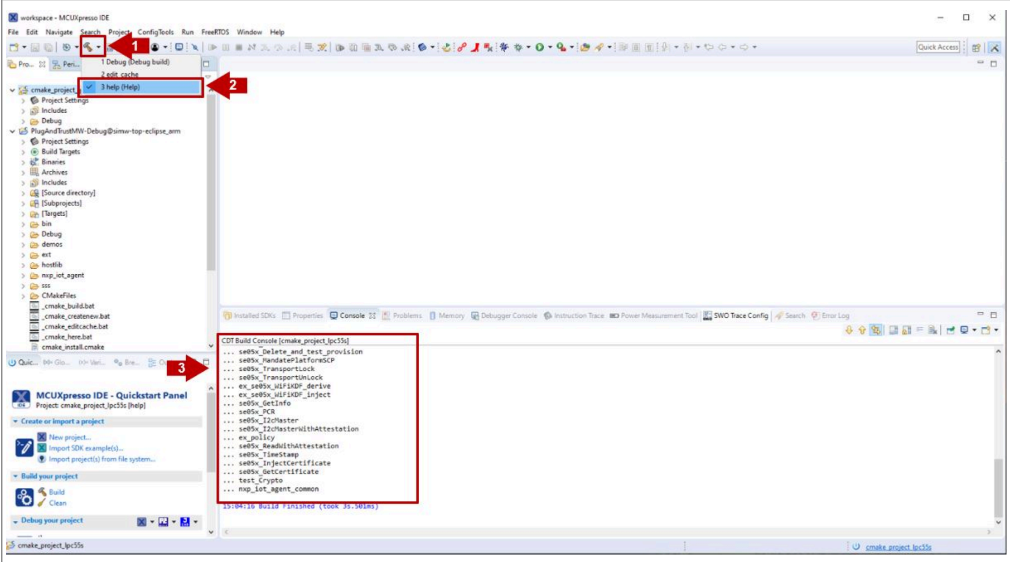
- List the Plug & Trust middleware test examples.
- Edit Plug & Trust middleware test example CMake options.
- Execute one Plug & Trust middleware test example.

5.6.1 List the Plug & Trust middleware examples

The Plug & Trust middleware comes with several test examples used to verify atomic SE050 security IC features. To get the list of examples, follow these steps:

1. 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 Plug & Trust middleware examples which can be compiled with the currently chosen CMake settings (see Figure 39).



```
CDT Build Console [cmake_project_lpc55s]
... se05x_Delete_and_test_provision
... se05x_MandatePlatformSCP
... se05x_TransportLock
... se05x_TransportUnLock
... ex_se05x_WiFiKDF_derive
... ex_se05x_WiFiKDF_inject
... se05x_GetInfo
... se05x_PCR
... se05x_I2cMaster
... se05x_I2cMasterWithAttestation
... ex_policy
... se05x_ReadWithAttestation
... se05x_TimeStamp
... se05x_InjectCertificate
... se05x_GetCertificate
... test_Crypto
... nxp_iot_agent_common
15:04:16 Build Finished (took 3s.501ms)
```

Figure 39. Plug & Trust middleware examples

5.6.2 Edit Plug & Trust middleware example CMake options.

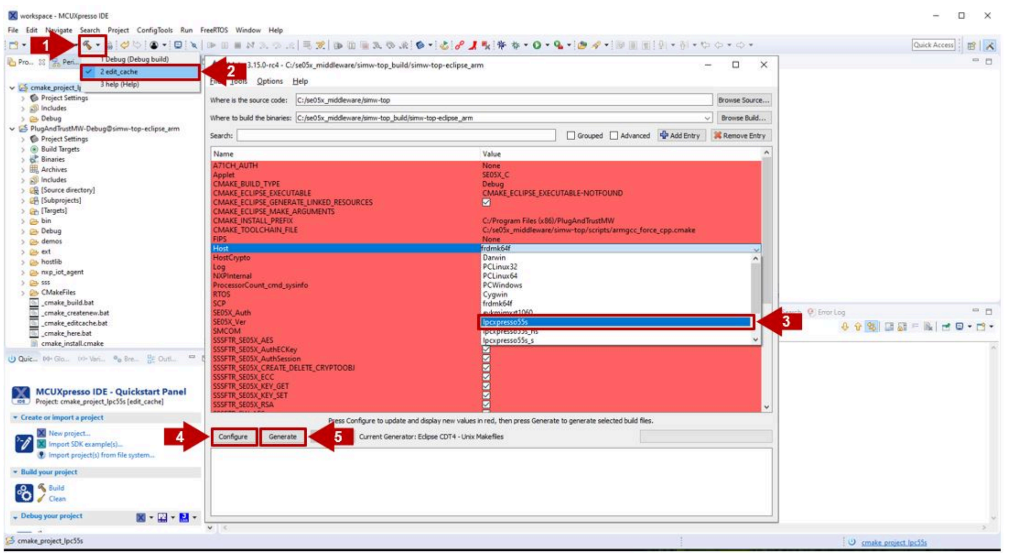
The Plug & Trust middleware is delivered with the CMake files that include the set of directives and instructions describing the project's source files and targets. In addition, it includes the CMake configuration files used to enable or disable several features, portability and setting flags to generate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

Note: The default build configuration of the Plug & Trust middleware ≥ `V04.02.xx` 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 5.6.4.

To edit the CMake options, follow these steps indicated in Figure 40:

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)***.
3. The CMake GUI window will open in your laptop. Using this GUI, change your host platform to `lpcxpresso55s`
4. Click **Configure** button

5. Click **Generate** button and close the CMake GUI window.



3.15.0-rc4 - C:/se05x_middleware/simw-top_build/simw-top-eclipse_arm

Where is the source code: C:/se05x_middleware/simw-top

Where to build the binaries: C:/se05x_middleware/simw-top_build/simw-top-eclipse_arm

Name	Value
A71CH_AUTH	None
Applet	SE05X_C
CMAKE_BUILD_TYPE	Debug
CMAKE_ECLIPSE_EXECUTABLE	CMAKE_ECLIPSE_EXECUTABLE-NOTFOUND
CMAKE_ECLIPSE_GENERATE_LINKED_RESOURCES	✓
CMAKE_ECLIPSE_MAKE_ARGUMENTS	
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/PlugAndTrustMW
CMAKE_TOOLCHAIN_FILE	C:/se05x_middleware/simw-top/scripts/armgcc_force_cpp.cmake
FIPS	None
Host	frdmk64f
HostCrypto	Darwin
Log	PCLinux32
NXPInternal	PCLinux64
ProcessorCount_cmd_sysinfo	PCWindows
RTOS	Cygwin
SCP	frdmk64f
SE05X_Auth	
SE05X_Ver	lpcxpresso55s
SMCOM	
SSSFTR_SE05X_AES	lpcxpresso55s_s
SSSFTR_SE05X_AuthECKey	✓
SSSFTR_SE05X_AuthSession	✓
SSSFTR_SE05X_CREATE_DELETE_CRYPTOOBJ	✓
SSSFTR_SE05X_ECC	✓
SSSFTR_SE05X_KEY_GET	✓
SSSFTR_SE05X_KEY_SET	✓
SSSFTR_SE05X_RSA	✓

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Eclipse CDT4 - Unix Makefiles

Figure 40. 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 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 Plug & Trust middleware.*

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

1. Connect the LPC55S69 board to your laptop as shown in Figure 41.

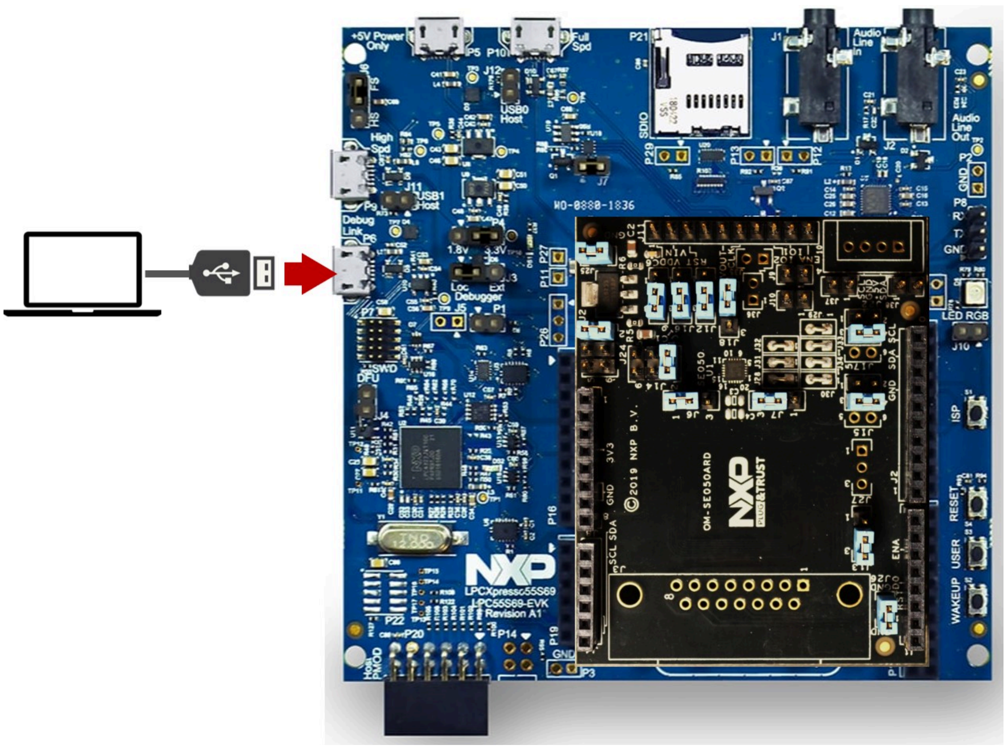
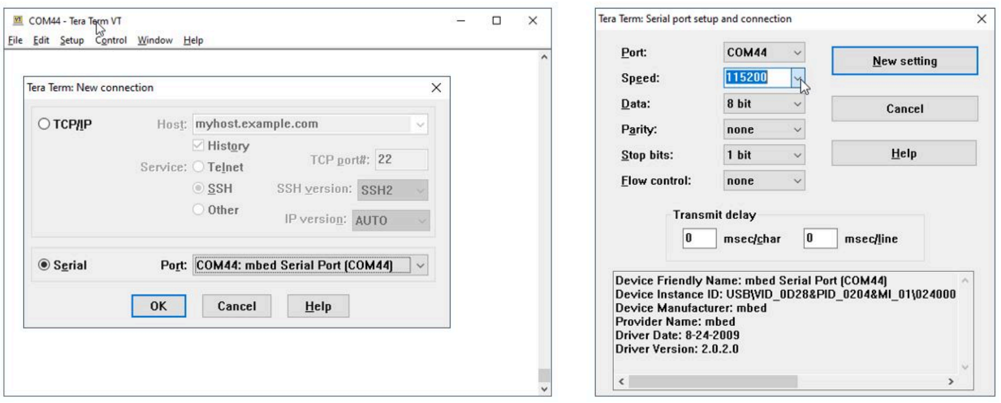


Figure 41. Connect boards to the laptop

2. Open TeraTerm. Click ***Serial*** option and select from the drop-down list the COM port number assigned to your LPC55S69. 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 42:


COM44 - Tera Term VT
File Edit Setup Control Window Help
Tera Term: New connection
OK
Cancel
Help
Tera Term: Serial port setup and connection
New setting
Cancel
Help

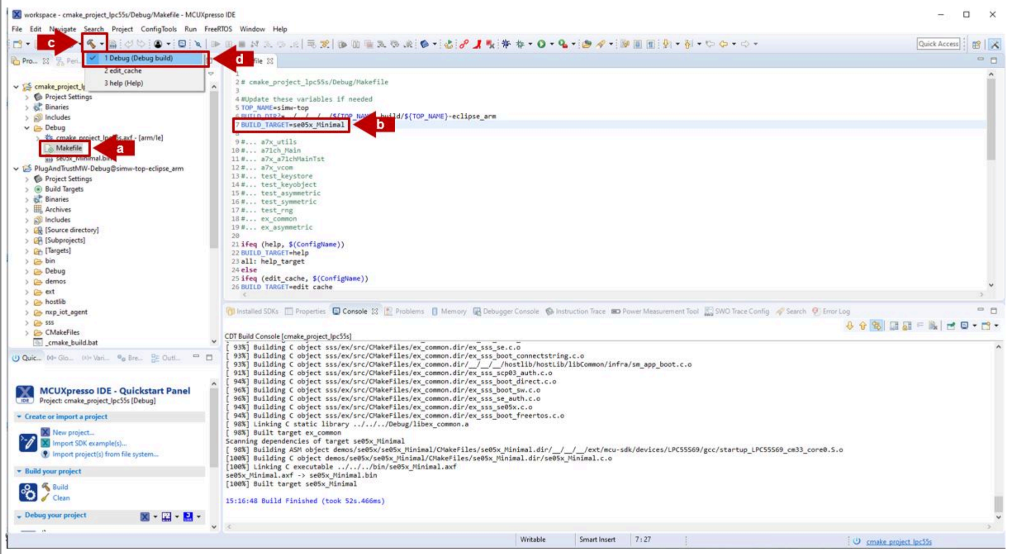
TCP/IP Host: myhost.example.com
History
Service: Telnet TCP port#: 22
SSH SSH version: SSH2
Other IP version: AUTO
Serial Port: COM44: mbed Serial Port (COM44)

Port: COM44
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line

```
Device Friendly Name: mbed Serial Port (COM44)
Device Instance ID: USB\VID_0D28&PID_0204&MI_01\024000
Device Manufacturer: mbed
Provider Name: mbed
Driver Date: 8-24-2009
Driver Version: 2.0.2.0
```

Figure 42. Configure TeraTerm

3. Select the `se05x_Minimal` as the project to be executed. For that, follow the steps shown in Figure 43:
   a. In the Project Explorer window, go to ***Debug*** folder and open the ***Makefile*** file (under cmake_project_lpc55s)..
   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.



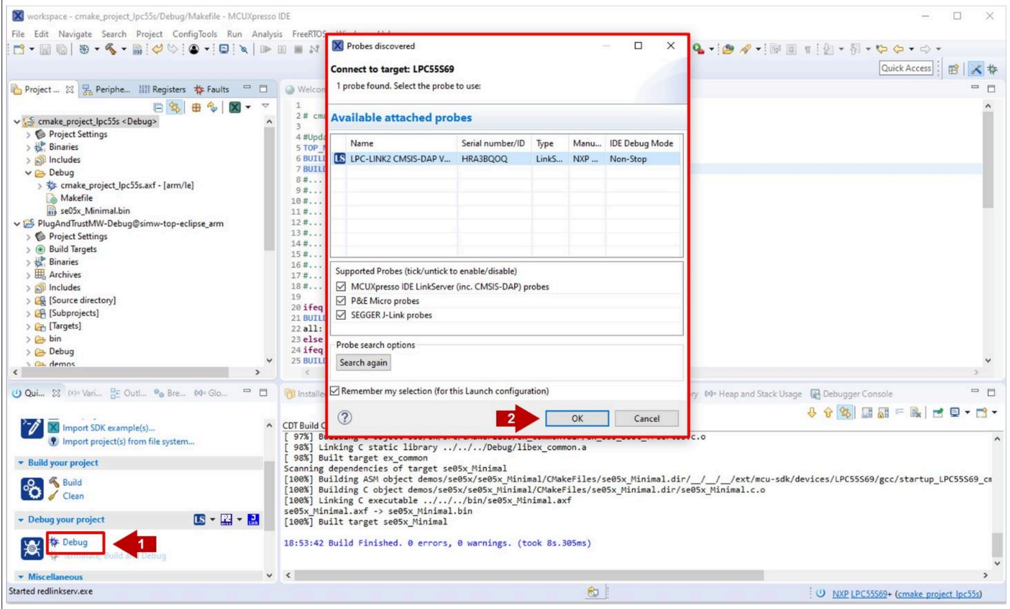
```
# cmake_project_lpc55s/Debug/Makefile

#Update these variables if needed
TOP_NAME=simw-top
BUILD_DIR?=../../../${TOP_NAME}/build/${TOP_NAME}-eclipse_arm
BUILD_TARGET=se05x_Minimal

#... a7x_utils
#... a71ch_Main
#... a7x_a71chMainTst
#... a7x_vcom
#... test_keystore
#... test_keyobject
#... test_asymmetric
#... test_symmetric
#... test_rng
#... ex_common
#... ex_asymmetric

ifeq (help, $(ConfigName))
BUILD_TARGET=help
all: help_target
else
ifeq (edit_cache, $(ConfigName))
BUILD_TARGET=edit_cache
```

CDT Build Console [cmake_project_lpc55s]

```
[ 93%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_se.c.o
[ 93%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_boot_connectstring.c.o
[ 93%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/__/__/__/hostlib/hostLib/libCommon/infra/sm_app_boot.c.o
[ 91%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_scp03_auth.c.o
[ 94%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_boot_direct.c.o
[ 96%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_boot_sw.c.o
[ 96%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_se_auth.c.o
[ 94%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_se05x.c.o
[ 94%] Building C object sss/ex/src/CMakeFiles/ex_common.dir/ex_sss_boot_freertos.c.o
[ 98%] Linking C static library ../../../Debug/libex_common.a
[ 98%] Built target ex_common
Scanning dependencies of target se05x_Minimal
[ 98%] Building ASM object demos/se05x/se05x_Minimal/CMakeFiles/se05x_Minimal.dir/__/__/__/ext/mcu-sdk/devices/LPC55S69/gcc/startup_LPC55S69_cm33_core0.S.o
[100%] Building C object demos/se05x/se05x_Minimal/CMakeFiles/se05x_Minimal.dir/se05x_Minimal.c.o
[100%] Linking C executable ../../../bin/se05x_Minimal.axf
se05x_Minimal.axf -> se05x_Minimal.bin
[100%] Built target se05x_Minimal

15:16:48 Build Finished (took 52s.466ms)
```

Figure 43. Debug build Plug & Trust middleware `se05x_minimal` project example

4. Go to the MCUXpresso Quickstart Panel and click ***Debug*** button as shown in Figure 44. 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:



Connect to target: LPC55S69

1 probe found. Select the probe to use:

Available attached probes

Name	Serial number/ID	Type	Manu...	IDE Debug Mode
LPC-LINK2 CMSIS-DAP V...	HRA3BQOQ	LinkS...	NXP ...	Non-Stop

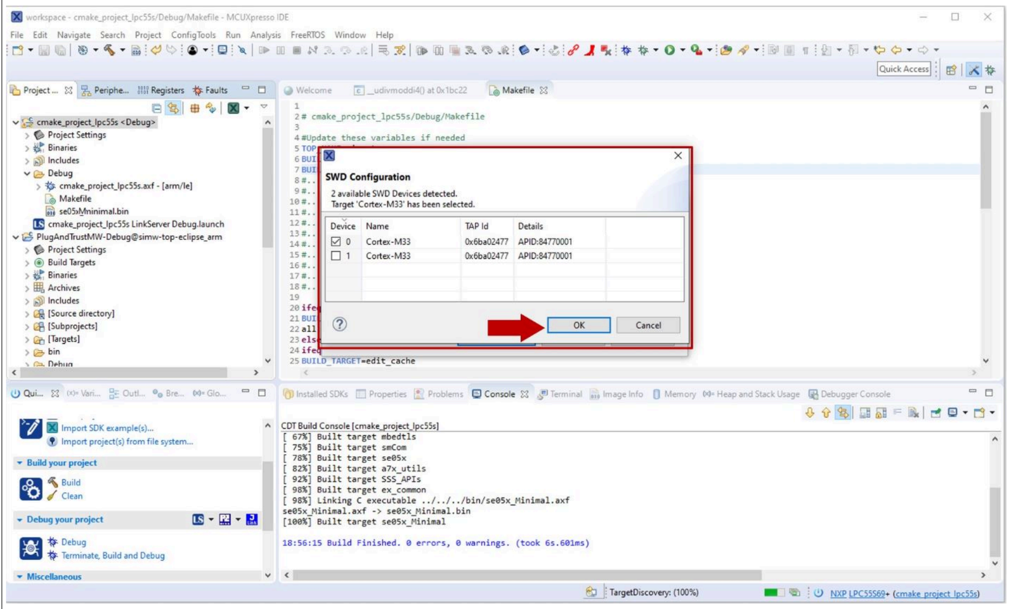
Supported Probes (tick/untick to enable/disable)
- ✓ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- ✓ P&E Micro probes
- ✓ SEGGER J-Link probes

✓ Remember my selection (for this Launch configuration)

```
[ 98%] Linking C static library ../../../Debug/libex_common.a
[ 98%] Built target ex_common
Scanning dependencies of target se05x_Minimal
[100%] Building ASM object demos/se05x/se05x_Minimal/CMakeFiles/se05x_Minimal.dir/__/__/__/ext/mcu-sdk/devices/LPC55S69/gcc/startup_LPC55S69_cm
[100%] Building C object demos/se05x/se05x_Minimal/CMakeFiles/se05x_Minimal.dir/se05x_Minimal.c.o
[100%] Linking C executable ../../../bin/se05x_Minimal.axf
se05x_Minimal.axf -> se05x_Minimal.bin
[100%] Built target se05x_Minimal

18:53:42 Build Finished. 0 errors, 0 warnings. (took 8s.305ms)
```

Figure 44. Debug `se05x_minimal` project example

5. You may be asked to select the SWD configuration. You can use the default one and click **OK** as shown in Figure 45:



SWD Configuration

2 available SWD Devices detected.
Target 'Cortex-M33' has been selected.

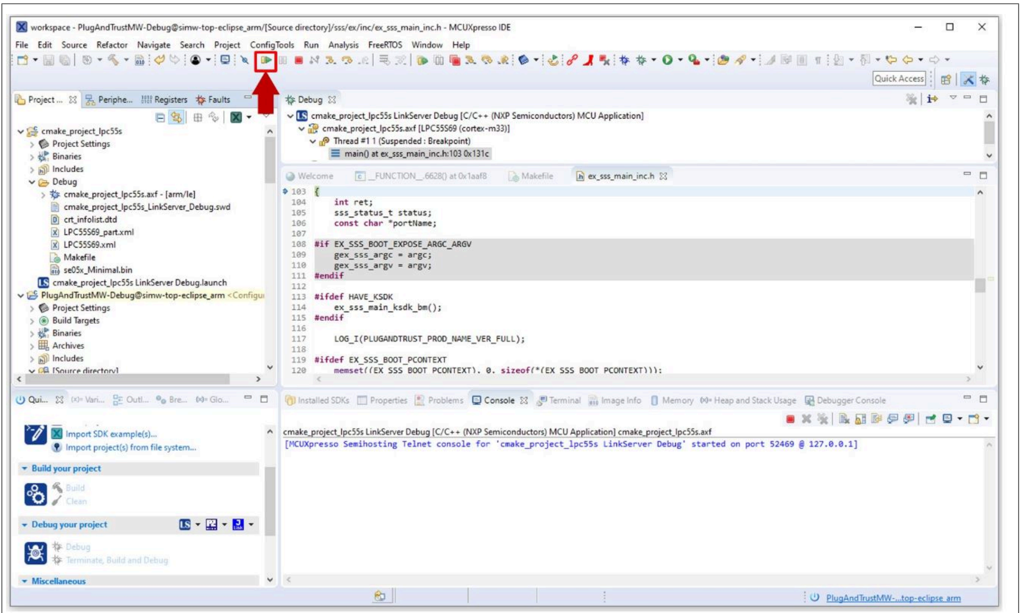
Device	Name	TAP Id	Details
✓ 0	Cortex-M33	0x6ba02477	APID:84770001
1	Cortex-M33	0x6ba02477	APID:84770001

CDT Build Console [cmake_project_lpc55s]

```
[ 67%] Built target mbedtls
[ 75%] Built target smCom
[ 78%] Built target se05x
[ 82%] Built target a7x_utils
[ 92%] Built target SSS_APIs
[ 98%] Built target ex_common
[ 98%] Linking C executable ../../../bin/se05x_Minimal.axf
se05x_Minimal.axf -> se05x_Minimal.bin
[100%] Built target se05x_Minimal

18:56:15 Build Finished. 0 errors, 0 warnings. (took 6s.601ms)
```

Figure 45. Select SWD configuration

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 46.



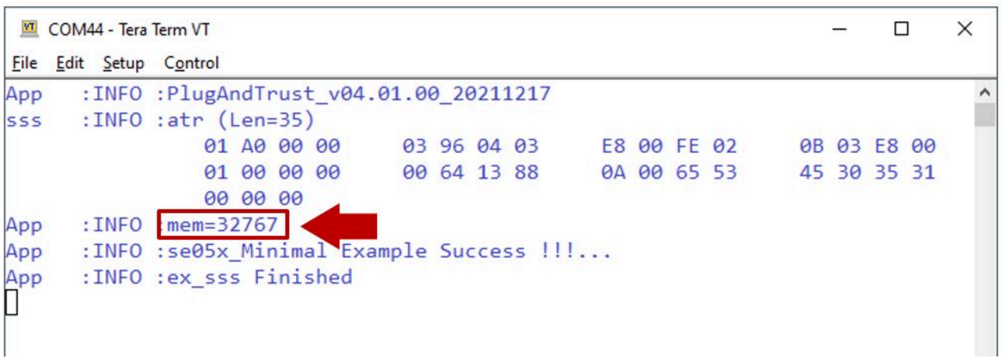
```
cmake_project_lpc55s LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application]
  cmake_project_lpc55s.axf [LPC55S69 (cortex-m33)]
    Thread #1 1 (Suspended : Breakpoint)
      main() at ex_sss_main_inc.h:103 0x131c
```

```
{
    int ret;
    sss_status_t status;
    const char *portName;

#if EX_SSS_BOOT_EXPOSE_ARGC_ARGV
    gex_sss_argc = argc;
    gex_sss_argv = argv;
#endif

#ifdef HAVE_KSDK
    ex_sss_main_ksdk_bm();
#endif

    LOG_I(PLUGANDTRUST_PROD_NAME_VER_FULL);

#ifdef EX_SSS_BOOT_PCONTEXT
    memset((EX_SSS_BOOT_PCONTEXT), 0, sizeof(*(EX_SSS_BOOT_PCONTEXT)));
```

```
cmake_project_lpc55s LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application] cmake_project_lpc55s.axf
[MCUXpresso Semihosting Telnet console for 'cmake_project_lpc55s LinkServer Debug' started on port 52469 @ 127.0.0.1]
```

Figure 46. Resume `se05x_minimal` project example

7. The project example should now be running into your LPC55S69. 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 47 .



```
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 47. TeraTerm logs - `se05x_minimal` project example

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

5.6.4 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_ 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 10. 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						

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

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
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 11. CMake Settings for SE051 product variants

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

Table 12. 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						

5.6.4.1 Example: SE050E CMake build settings

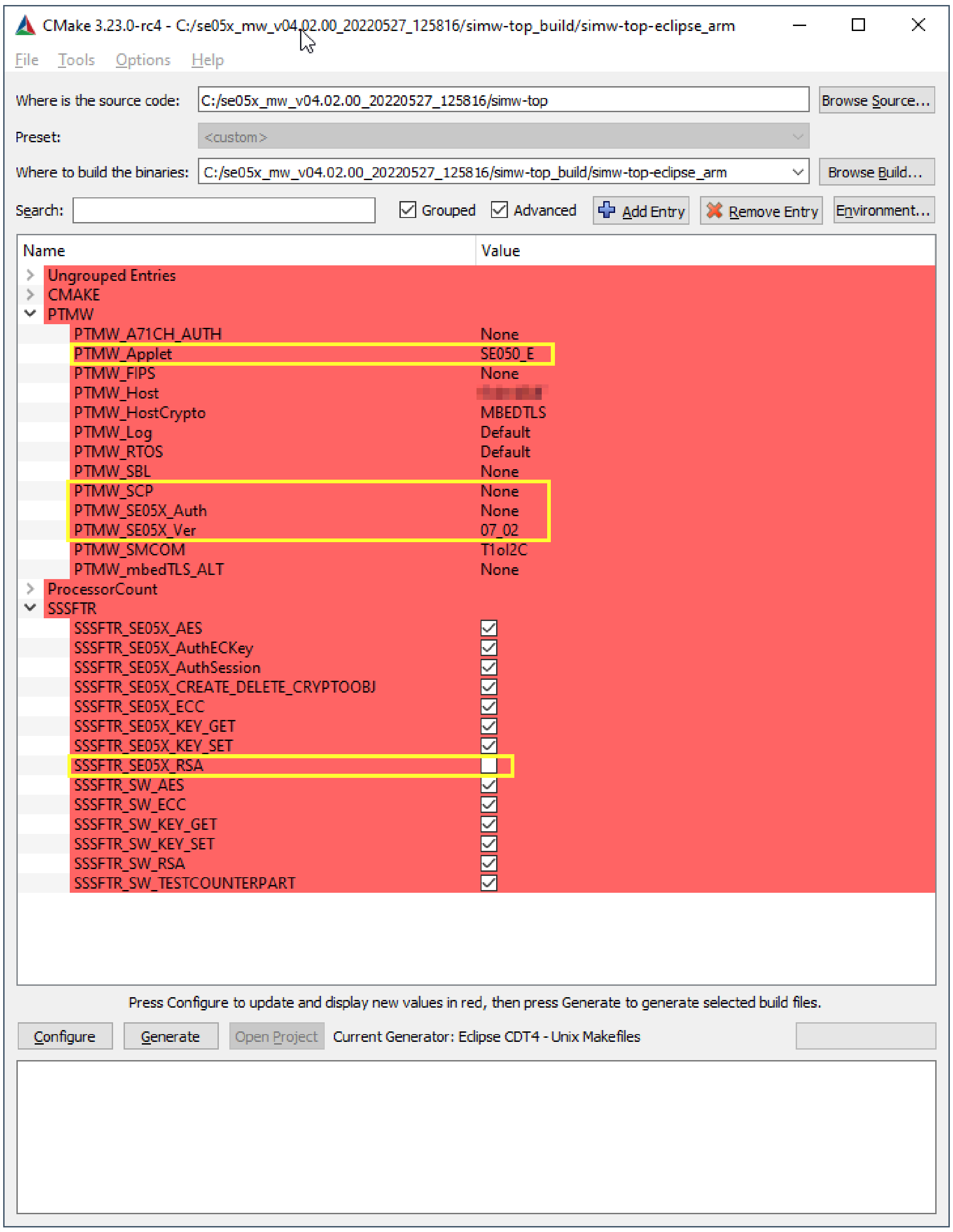
The following images show the configuration for the SE050E development board OM-SE05ARD-E according to Table 8.

- 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_SCP`.

How to enable Platform SCP is described in Section 6.



Where is the source code: C:/se05x_mw_v04.02.00_20220527_125816/simw-top

Preset: <custom>

Where to build the binaries: C:/se05x_mw_v04.02.00_20220527_125816/simw-top_build/simw-top-eclipse_arm

Search:

Name	Value
Ungrouped Entries	
CMAKE	
PTMW	
PTMW_A71CH_AUTH	None
PTMW_Applet	SE050_E
PTMW_FIPS	None
PTMW_Host	[redacted]
PTMW_HostCrypto	MBEDTLS
PTMW_Log	Default
PTMW_RTOS	Default
PTMW_SBL	None
PTMW_SCP	None
PTMW_SE05X_Auth	None
PTMW_SE05X_Ver	07_02
PTMW_SMCOM	T1oI2C
PTMW_mbedTLS_ALT	None
ProcessorCount	
SSSFTR	
SSSFTR_SE05X_AES	✓
SSSFTR_SE05X_AuthECKey	✓
SSSFTR_SE05X_AuthSession	✓
SSSFTR_SE05X_CREATE_DELETE_CRYPTOOBJ	✓
SSSFTR_SE05X_ECC	✓
SSSFTR_SE05X_KEY_GET	✓
SSSFTR_SE05X_KEY_SET	✓
SSSFTR_SE05X_RSA	☐
SSSFTR_SW_AES	✓
SSSFTR_SW_ECC	✓
SSSFTR_SW_KEY_GET	✓
SSSFTR_SW_KEY_SET	✓
SSSFTR_SW_RSA	✓
SSSFTR_SW_TESTCOUNTERPART	✓

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Eclipse CDT4 - Unix Makefiles

Figure 48. 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 LPC55S69 MCUXpresso SDK
- Section 6.3 How to enable Platform SCP in the LPC55S69 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 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 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.*

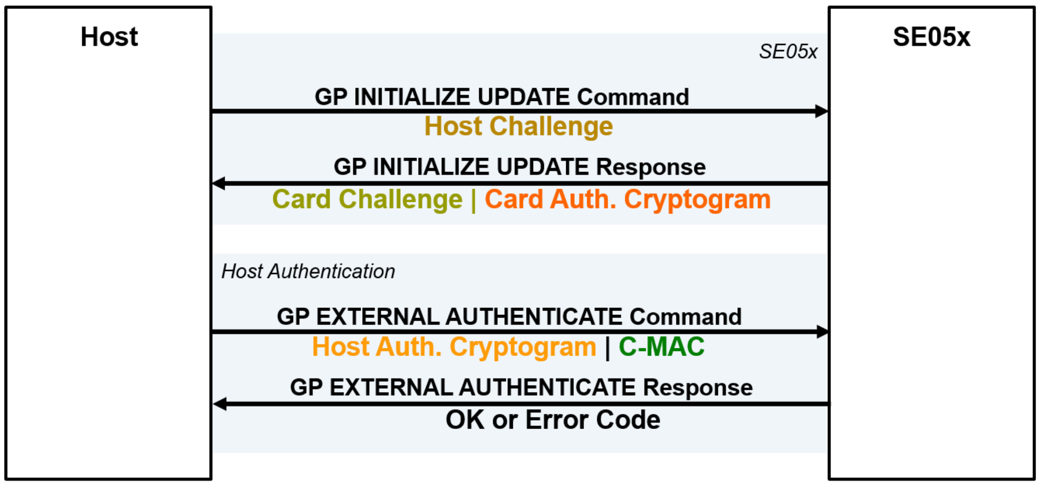

Host
SE05x
SE05x
GP INITIALIZE UPDATE Command
Host Challenge
GP INITIALIZE UPDATE Response
Card Challenge | Card Auth. Cryptogram
Host Authentication
GP EXTERNAL AUTHENTICATE Command
Host Auth. Cryptogram | C-MAC
GP EXTERNAL AUTHENTICATE Response
OK or Error Code

Figure 49. SPC03 mutual authentication – principle

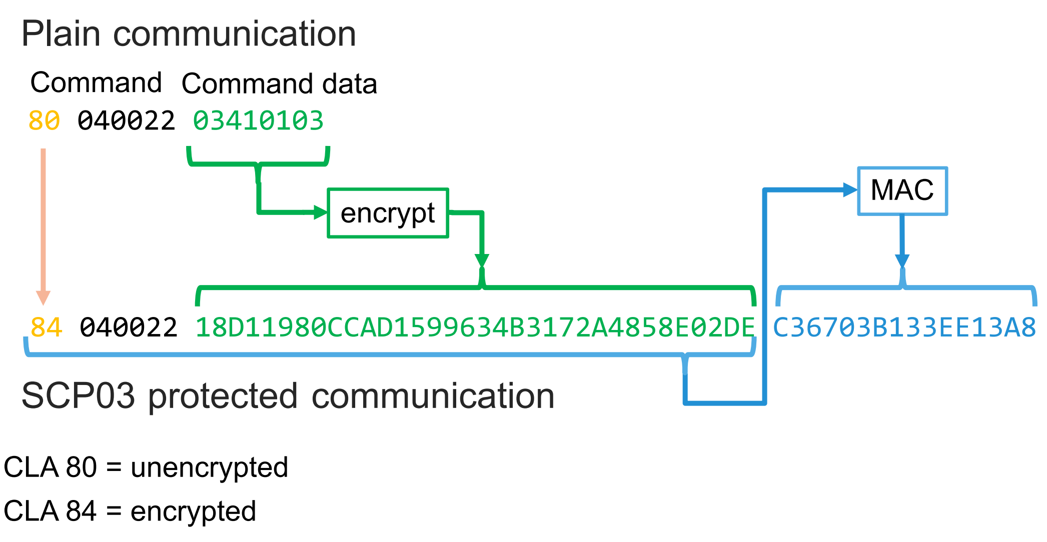


Figure 50. SPC03 Encryption and MACing principle

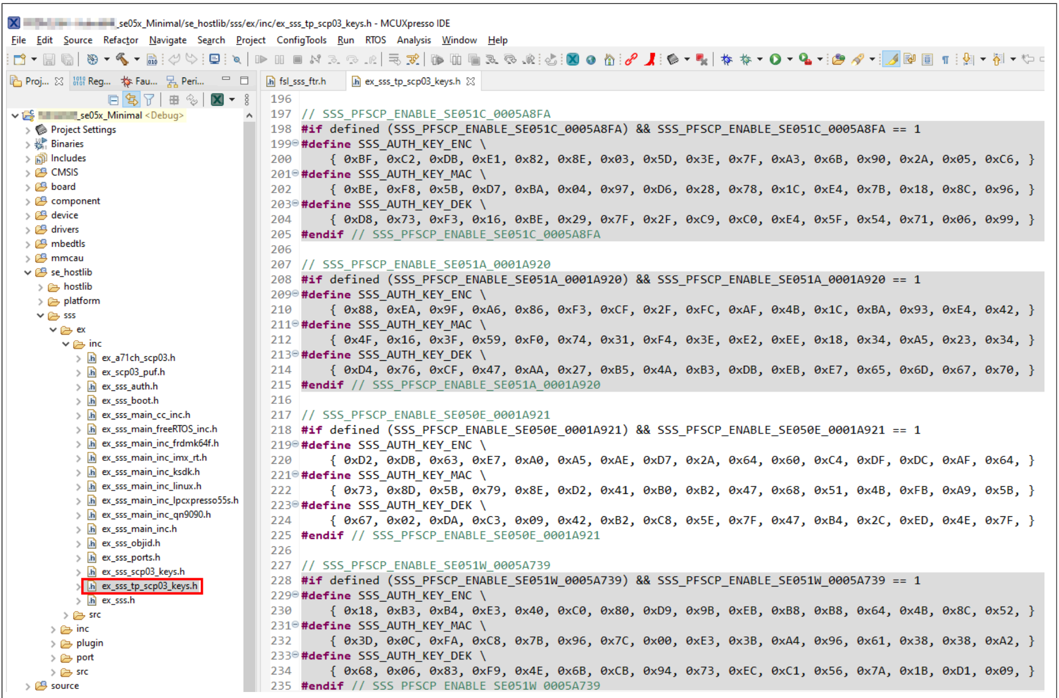
6.2 How to configure the Platform SCP keys in the LPC55S69 MCUXpresso SDK

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\
```

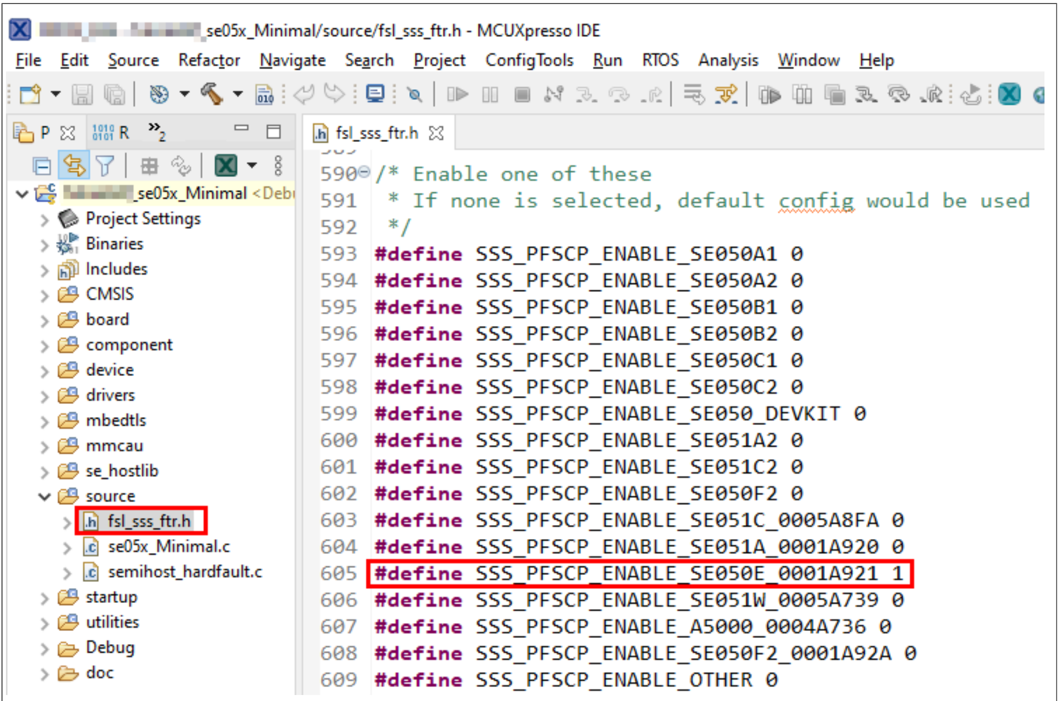


```
// SSS_PFSCP_ENABLE_SE051C_0005A8FA
#if defined (SSS_PFSCP_ENABLE_SE051C_0005A8FA) && SSS_PFSCP_ENABLE_SE051C_0005A8FA == 1
#define SSS_AUTH_KEY_ENC \
    { 0xBF, 0xC2, 0xDB, 0xE1, 0x82, 0x8E, 0x03, 0x5D, 0x3E, 0x7F, 0xA3, 0x6B, 0x90, 0x2A, 0x05, 0xC6, }
#define SSS_AUTH_KEY_MAC \
    { 0xBE, 0xF8, 0x5B, 0xD7, 0xBA, 0x04, 0x97, 0xD6, 0x28, 0x78, 0x1C, 0xE4, 0x7B, 0x18, 0x8C, 0x96, }
#define SSS_AUTH_KEY_DEK \
    { 0xD8, 0x73, 0xF3, 0x16, 0xBE, 0x29, 0x7F, 0x2F, 0xC9, 0xC0, 0xE4, 0x5F, 0x54, 0x71, 0x06, 0x99, }
#endif // SSS_PFSCP_ENABLE_SE051C_0005A8FA

// SSS_PFSCP_ENABLE_SE051A_0001A920
#if defined (SSS_PFSCP_ENABLE_SE051A_0001A920) && SSS_PFSCP_ENABLE_SE051A_0001A920 == 1
#define SSS_AUTH_KEY_ENC \
    { 0x88, 0xEA, 0x9F, 0xA6, 0x86, 0xF3, 0xCF, 0x2F, 0xFC, 0xAF, 0x4B, 0x1C, 0xBA, 0x93, 0xE4, 0x42, }
#define SSS_AUTH_KEY_MAC \
    { 0x4F, 0x16, 0x3F, 0x59, 0xF0, 0x74, 0x31, 0xF4, 0x3E, 0xE2, 0xEE, 0x18, 0x34, 0xA5, 0x23, 0x34, }
#define SSS_AUTH_KEY_DEK \
    { 0xD4, 0x76, 0xCF, 0x47, 0xAA, 0x27, 0xB5, 0x4A, 0xB3, 0xDB, 0xEB, 0xE7, 0x65, 0x6D, 0x67, 0x70, }
#endif // SSS_PFSCP_ENABLE_SE051A_0001A920

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

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

Figure 51. 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.


[redacted] se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
fsl_sss_ftr.h
[redacted] se05x_Minimal <Debu
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
mmcau
se_hostlib
source
fsl_sss_ftr.h
se05x_Minimal.c
semihost_hardfault.c
startup
utilities
Debug
doc

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

Figure 52. 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 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[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 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

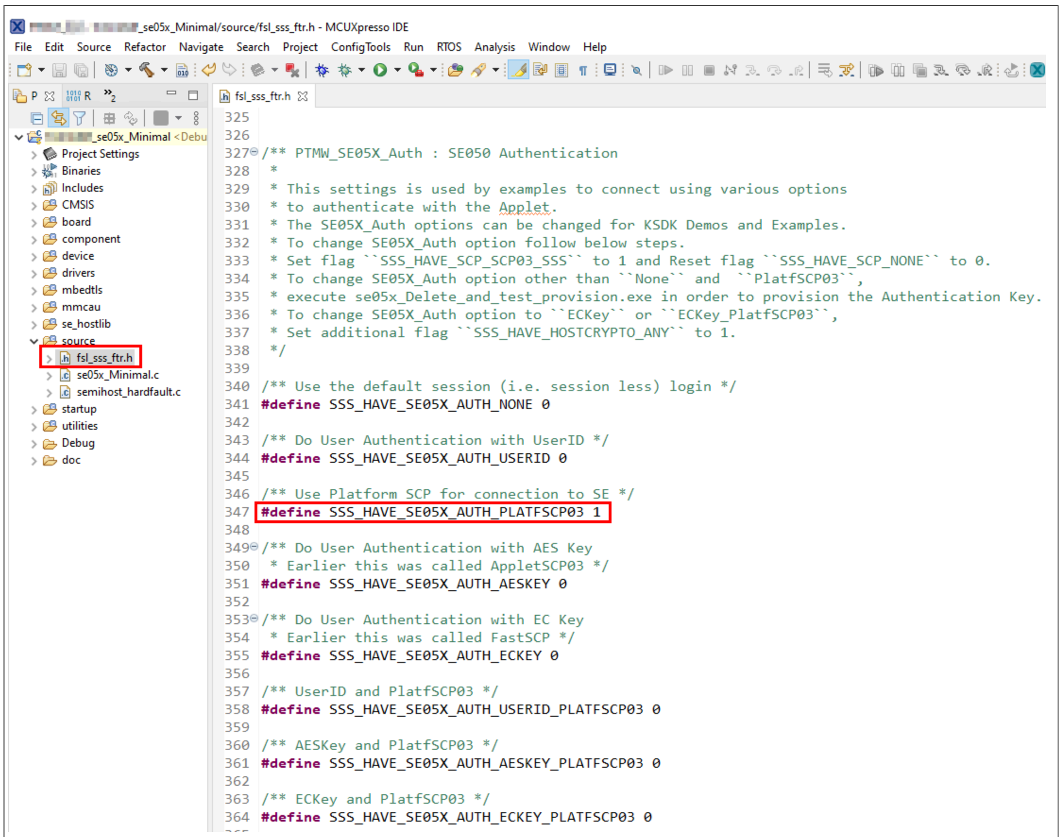
Variant	OEF ID	Platform SCP key define to be set to '1'
A5000 Dev. Board OM-A5000ARD	A736	`SSS_PFSCP_ENABLE_A5000_0004A736`
A5000	A736	`SSS_PFSCP_ENABLE_A5000_0004A736`

In the next step it is necessary to enable Platfrom SCP in the 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 LPC55S69 MCUXpresso SDK

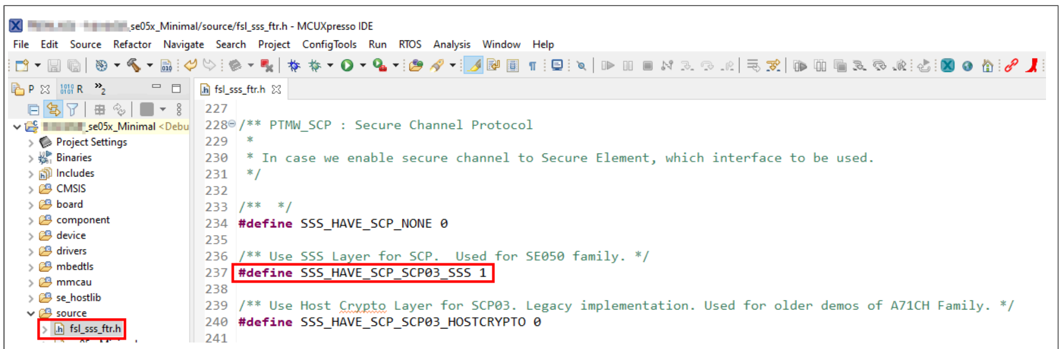
To enable Platform SCP is required to rebuild the SDK with the following options:

- 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`.



```
/** PTMW_SE05X_Auth : SE050 Authentication
 *
 * This settings is used by examples to connect using various options
 * to authenticate with the Applet.
 * The SE05X_Auth options can be changed for KSDK Demos and Examples.
 * To change SE05X_Auth option follow below steps.
 * Set flag ``SSS_HAVE_SCP_SCP03_SSS`` to 1 and Reset flag ``SSS_HAVE_SCP_NONE`` to 0.
 * To change SE05X_Auth option other than ``None`` and  ``PlatfSCP03``,
 * execute se05x_Delete_and_test_provision.exe in order to provision the Authentication Key.
 * To change SE05X_Auth option to ``ECKey`` or ``ECKey_PlatfSCP03``,
 * Set additional flag ``SSS_HAVE_HOSTCRYPTO_ANY`` to 1.
 */

/** Use the default session (i.e. session less) login */
#define SSS_HAVE_SE05X_AUTH_NONE 0

/** Do User Authentication with UserID */
#define SSS_HAVE_SE05X_AUTH_USERID 0

/** Use Platform SCP for connection to SE */
#define SSS_HAVE_SE05X_AUTH_PLATFSCP03 1

/** Do User Authentication with AES Key
 * Earlier this was called AppletSCP03 */
#define SSS_HAVE_SE05X_AUTH_AESKEY 0

/** Do User Authentication with EC Key
 * Earlier this was called FastSCP */
#define SSS_HAVE_SE05X_AUTH_ECKEY 0

/** UserID and PlatfSCP03 */
#define SSS_HAVE_SE05X_AUTH_USERID_PLATFSCP03 0

/** AESKey and PlatfSCP03 */
#define SSS_HAVE_SE05X_AUTH_AESKEY_PLATFSCP03 0

/** ECKey and PlatfSCP03 */
#define SSS_HAVE_SE05X_AUTH_ECKEY_PLATFSCP03 0
```

Figure 53. Feature file fsl_sss_ftr.h - Option PTMW_SE05X_Auth - PlatformSCP enabled


se05x_Minimal/source/fsl_sss_ftr.h - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
fsl_sss_ftr.h
se05x_Minimal
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
mbedtls
mmcau
se_hostlib
source
fsl_sss_ftr.h

```
/** PTMW_SCP : Secure Channel Protocol
 *
 * In case we enable secure channel to Secure Element, which interface to be used.
 */

/**  */
#define SSS_HAVE_SCP_NONE 0

/** Use SSS Layer for SCP.  Used for SE050 family. */
#define SSS_HAVE_SCP_SCP03_SSS 1

/** Use Host Crypto Layer for SCP03. Legacy implementation. Used for older demos of A71CH Family. */
#define SSS_HAVE_SCP_SCP03_HOSTCRYPTO 0
```

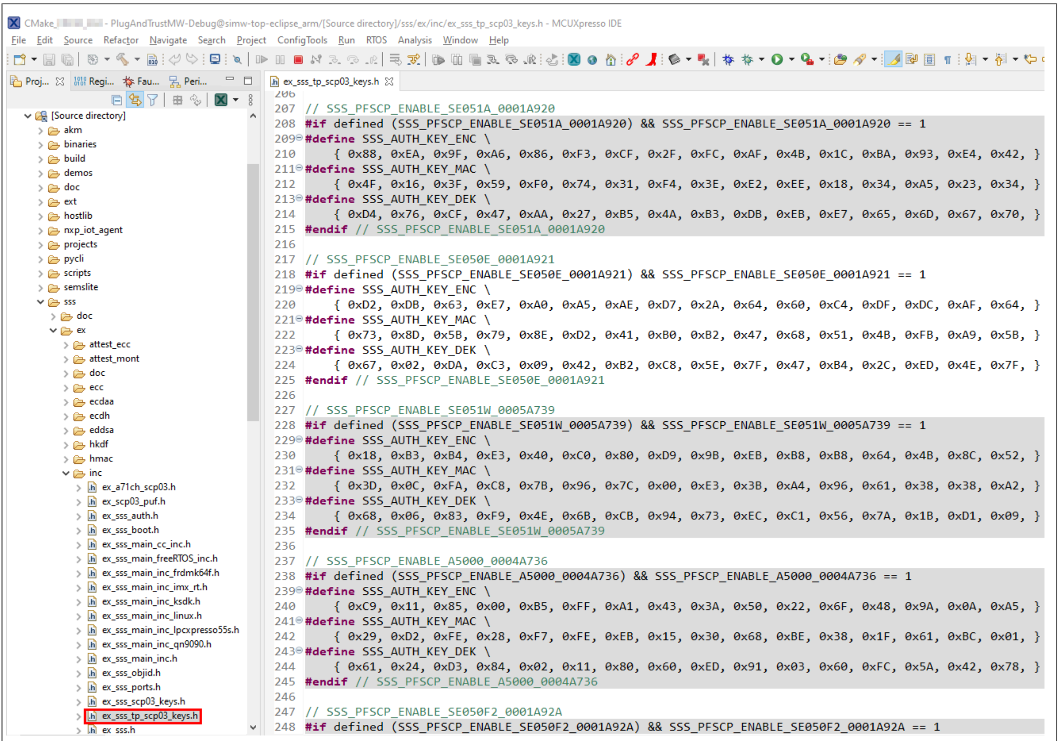
Figure 54. Feature file fsl_sss_ftr.h - Option PTMW_SCP - PlatformSCP enabled

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\`

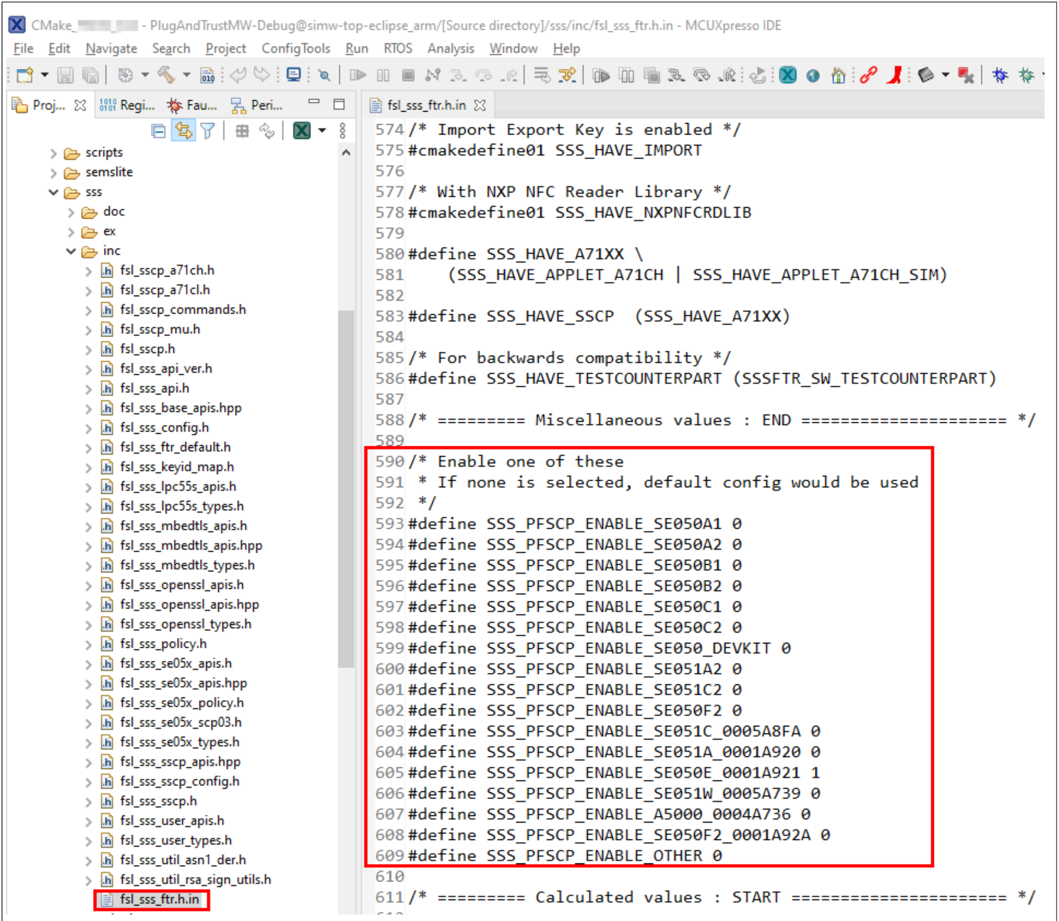


```
// SSS_PFSCP_ENABLE_SE051A_0001A920
#if defined (SSS_PFSCP_ENABLE_SE051A_0001A920) && SSS_PFSCP_ENABLE_SE051A_0001A920 == 1
#define SSS_AUTH_KEY_ENC \
    { 0x88, 0xEA, 0x9F, 0xA6, 0x86, 0xF3, 0xCF, 0x2F, 0xFC, 0xAF, 0x4B, 0x1C, 0xBA, 0x93, 0xE4, 0x42, }
#define SSS_AUTH_KEY_MAC \
    { 0x4F, 0x16, 0x3F, 0x59, 0xF0, 0x74, 0x31, 0xF4, 0x3E, 0xE2, 0xEE, 0x18, 0x34, 0xA5, 0x23, 0x34, }
#define SSS_AUTH_KEY_DEK \
    { 0xD4, 0x76, 0xCF, 0x47, 0xAA, 0x27, 0xB5, 0x4A, 0xB3, 0xDB, 0xEB, 0xE7, 0x65, 0x6D, 0x67, 0x70, }
#endif // SSS_PFSCP_ENABLE_SE051A_0001A920

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

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

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

// SSS_PFSCP_ENABLE_SE050F2_0001A92A
#if defined (SSS_PFSCP_ENABLE_SE050F2_0001A92A) && SSS_PFSCP_ENABLE_SE050F2_0001A92A == 1
```

Figure 55. 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.



```
/* Import Export Key is enabled */
#cmakedefine01 SSS_HAVE_IMPORT

/* With NXP NFC Reader Library */
#cmakedefine01 SSS_HAVE_NXPNFCRDLIB

#define SSS_HAVE_A71XX \
    (SSS_HAVE_APPLET_A71CH | SSS_HAVE_APPLET_A71CH_SIM)

#define SSS_HAVE_SSCP  (SSS_HAVE_A71XX)

/* For backwards compatibility */
#define SSS_HAVE_TESTCOUNTERPART (SSSFTR_SW_TESTCOUNTERPART)

/* ========= Miscellaneous values : END ===================== */

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

/* ========= Calculated values : START ===================== */
```

Figure 56. 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 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[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 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 Plug & Trust middleware. Section 6.5 describes how to enable Platform SCP in the CMake-based build system.

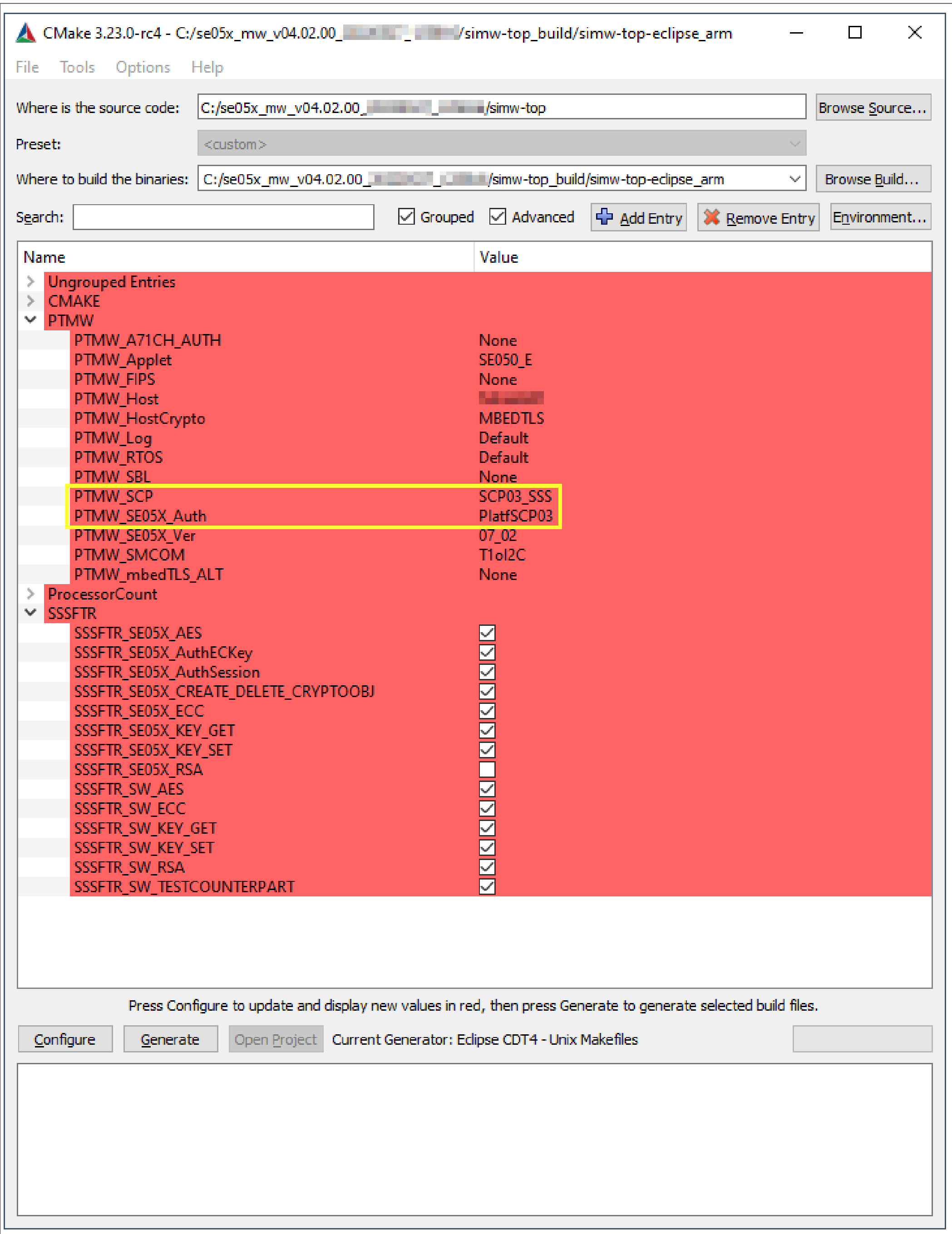
6.5 How to enable Platform SCP in the CMake-based build system

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.



CMake 3.23.0-rc4 - C:/se05x_mw_v04.02.00_[redacted]_[redacted]/simw-top_build/simw-top-eclipse_arm

Where is the source code: C:/se05x_mw_v04.02.00_[redacted]_[redacted]/simw-top

Preset: <custom>

Where to build the binaries: C:/se05x_mw_v04.02.00_[redacted]_[redacted]/simw-top_build/simw-top-eclipse_arm

Name	Value
Ungrouped Entries	
CMAKE	
PTMW	
PTMW_A71CH_AUTH	None
PTMW_Applet	SE050_E
PTMW_FIPS	None
PTMW_Host	[redacted]
PTMW_HostCrypto	MBEDTLS
PTMW_Log	Default
PTMW_RTOS	Default
PTMW_SBL	None
PTMW_SCP	SCP03_SSS
PTMW_SE05X_Auth	PlatfSCP03
PTMW_SE05X_Ver	07_02
PTMW_SMCOM	T1oI2C
PTMW_mbedTLS_ALT	None
ProcessorCount	
SSSFTR	
SSSFTR_SE05X_AES	✓
SSSFTR_SE05X_AuthECKey	✓
SSSFTR_SE05X_AuthSession	✓
SSSFTR_SE05X_CREATE_DELETE_CRYPTOOBJ	✓
SSSFTR_SE05X_ECC	✓
SSSFTR_SE05X_KEY_GET	✓
SSSFTR_SE05X_KEY_SET	✓
SSSFTR_SE05X_RSA	☐
SSSFTR_SW_AES	✓
SSSFTR_SW_ECC	✓
SSSFTR_SW_KEY_GET	✓
SSSFTR_SW_KEY_SET	✓
SSSFTR_SW_RSA	✓
SSSFTR_SW_TESTCOUNTERPART	✓

Press Configure to update and display new values in red, then press Generate to generate selected build files.

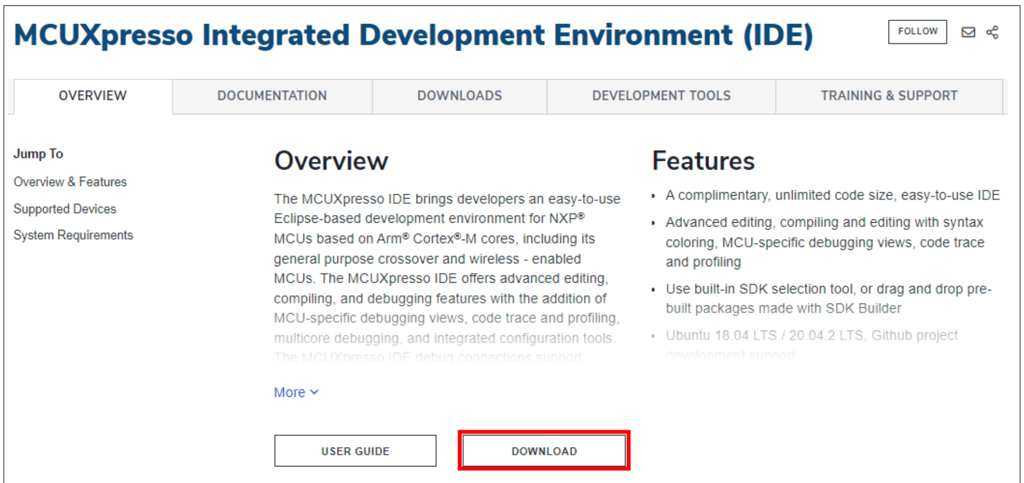
Current Generator: Eclipse CDT4 - Unix Makefiles

Figure 57. 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 58:



MCUXpresso Integrated Development Environment (IDE)

Overview

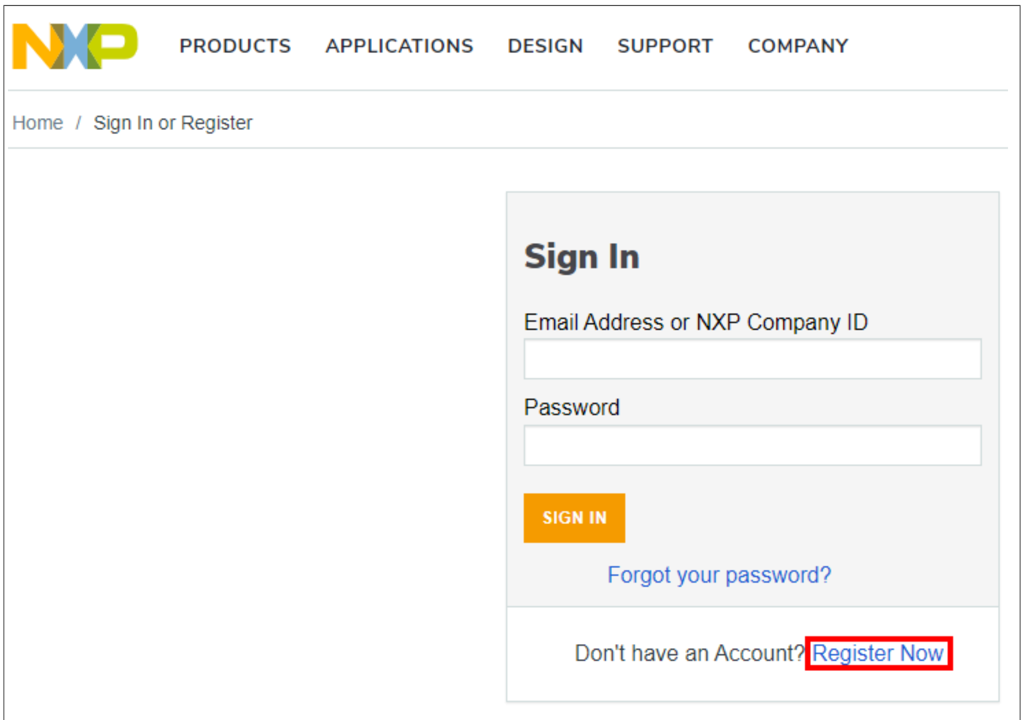
The MCUXpresso IDE brings developers an easy-to-use Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores, including its general purpose crossover and wireless - enabled MCUs. The MCUXpresso IDE offers advanced editing, compiling, and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools. The MCUXpresso IDE debug connections support

Features

- A complimentary, unlimited code size, easy-to-use IDE
- Advanced editing, compiling and editing with syntax coloring, MCU-specific debugging views, code trace and profiling
- Use built-in SDK selection tool, or drag and drop pre-built packages made with SDK Builder
- Ubuntu 18.04 LTS / 20.04.2 LTS, Github project development support

Figure 58. 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 59:



PRODUCTS
APPLICATIONS
DESIGN
SUPPORT
COMPANY
Home / Sign In or Register
Sign In
Email Address or NXP Company ID
Password
SIGN IN
Forgot your password?
Don't have an Account? Register Now

Figure 59. 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 60:

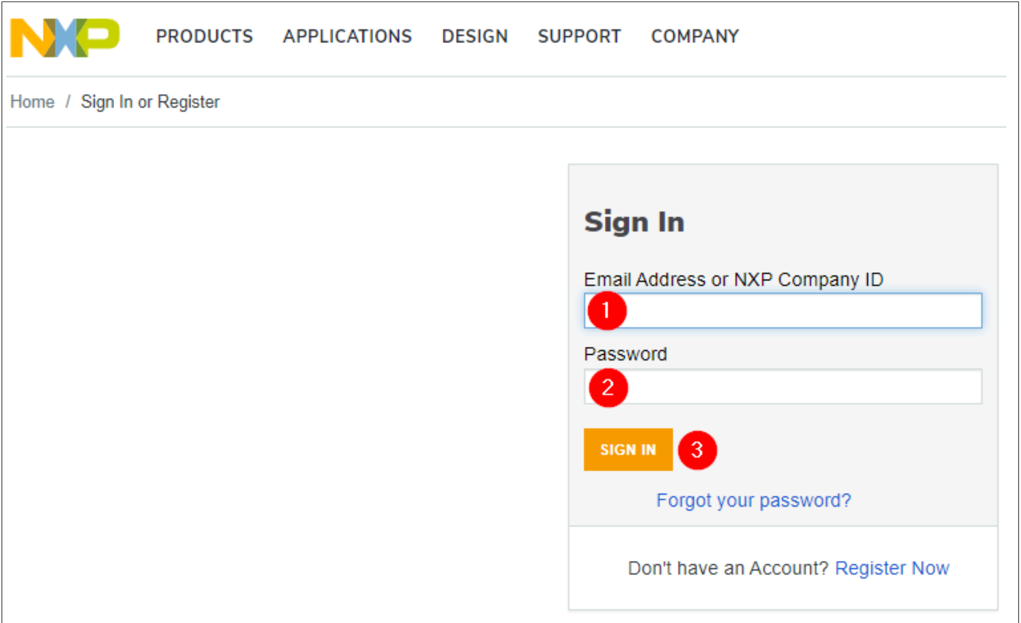
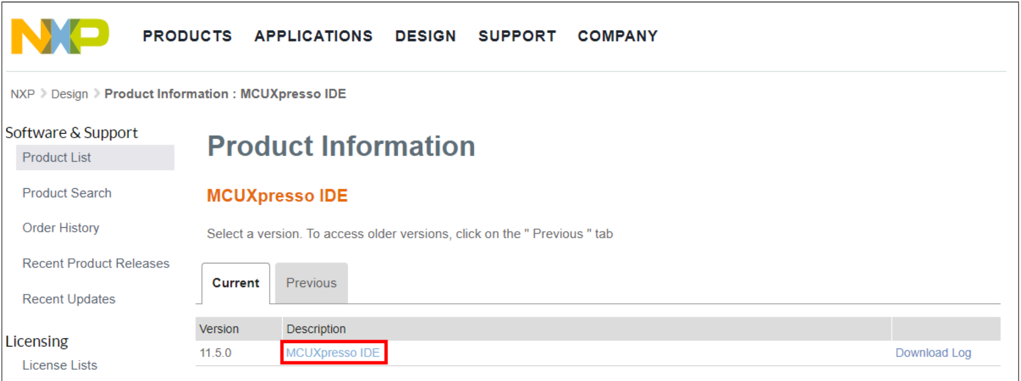

PRODUCTS
APPLICATIONS
DESIGN
SUPPORT
COMPANY
Home / Sign In or Register
Sign In
Email Address or NXP Company ID
1
Password
2
SIGN IN
3
Forgot your password?
Don't have an Account? Register Now

Figure 60. Sign-in in NXP website

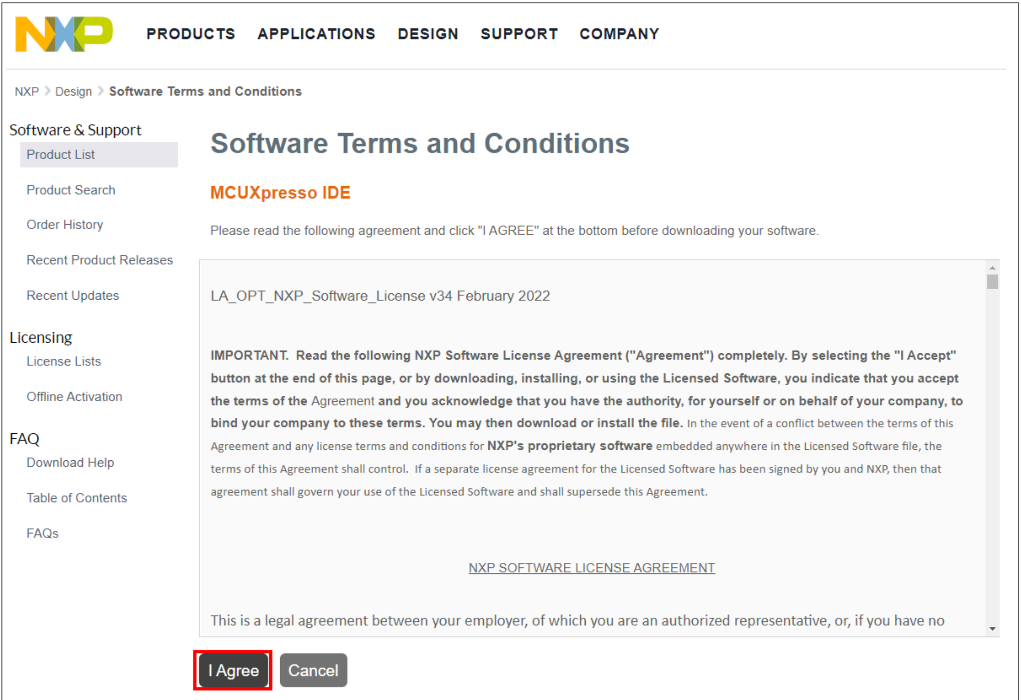
4. Click on MCUXpresso IDE as shown in Figure 61:



Version	Description	
11.5.0	MCUXpresso IDE	Download Log

Figure 61. Select MCUXpresso

5. Accept software terms and conditions as shown in Figure 62:



Software Terms and Conditions

MCUXpresso IDE

Please read the following agreement and click "I AGREE" at the bottom before downloading your software.

LA_OPT_NXP_Software_License v34 February 2022

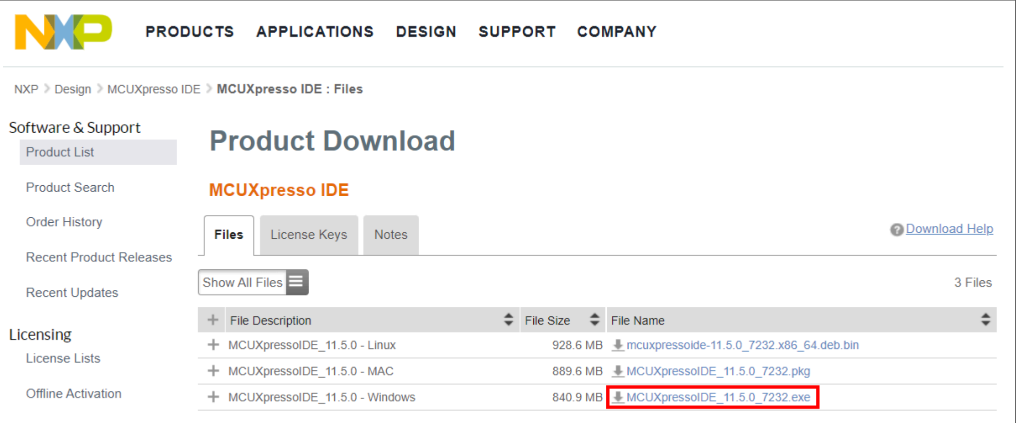
IMPORTANT. Read the following NXP Software License Agreement ("Agreement") completely. By selecting the "I Accept" button at the end of this page, or by downloading, installing, or using the Licensed Software, you indicate that you accept the terms of the Agreement and you acknowledge that you have the authority, for yourself or on behalf of your company, to bind your company to these terms. You may then download or install the file. In the event of a conflict between the terms of this Agreement and any license terms and conditions for NXP's proprietary software embedded anywhere in the Licensed Software file, the terms of this Agreement shall control. If a separate license agreement for the Licensed Software has been signed by you and NXP, then that agreement shall govern your use of the Licensed Software and shall supersede this Agreement.

NXP SOFTWARE LICENSE AGREEMENT

This is a legal agreement between your employer, of which you are an authorized representative, or, if you have no

Figure 62. 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 63:



Product Download

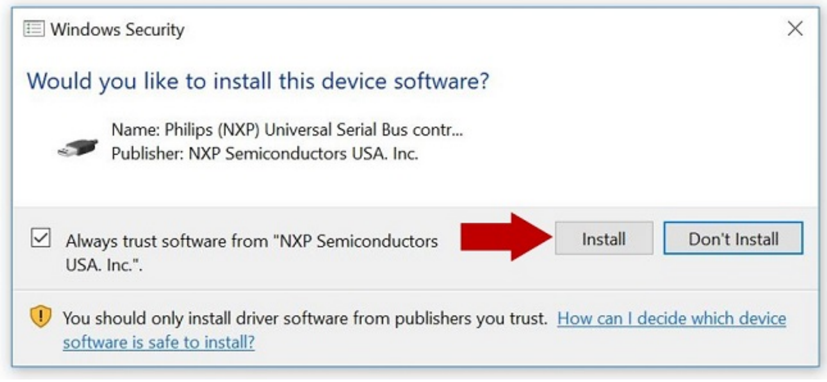
MCUXpresso IDE

File Description	File Size	File Name
MCUXpressoIDE_11.5.0 - Linux	928.6 MB	mcuxpressoide-11.5.0_7232.x86_64.deb.bin
MCUXpressoIDE_11.5.0 - MAC	889.6 MB	MCUXpressoIDE_11.5.0_7232.pkg
MCUXpressoIDE_11.5.0 - Windows	840.9 MB	MCUXpressoIDE_11.5.0_7232.exe

Figure 63. 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 64, Figure 65, Figure 66 and Figure 67:


Windows Security

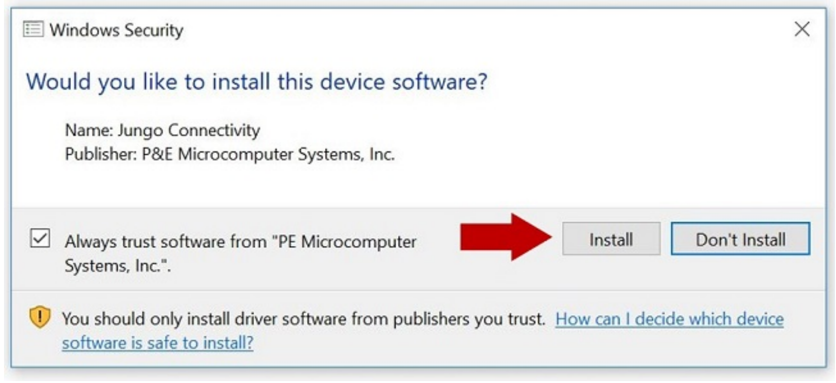
Would you like to install this device software?

Name: Philips (NXP) Universal Serial Bus contr...
Publisher: NXP Semiconductors USA. Inc.

☑ Always trust software from "NXP Semiconductors USA. Inc.".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 64. Install MCUXpresso required drivers I


Windows Security

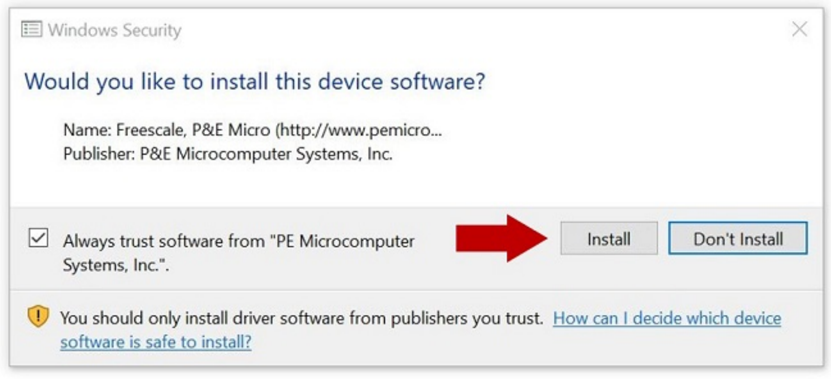
Would you like to install this device software?

Name: Jungo Connectivity
Publisher: P&E Microcomputer Systems, Inc.

☑ Always trust software from "PE Microcomputer Systems, Inc.".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 65. Install MCUXpresso required drivers II


Windows Security

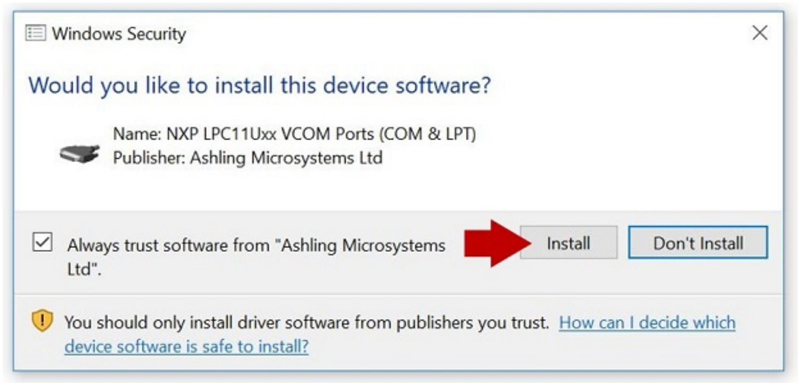
Would you like to install this device software?

Name: Freescale, P&E Micro (http://www.pemicro...
Publisher: P&E Microcomputer Systems, Inc.

☑ Always trust software from "PE Microcomputer Systems, Inc.".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 66. Install MCUXpresso required drivers III


Windows Security

Would you like to install this device software?

Name: NXP LPC11Uxx VCOM Ports (COM & LPT)
Publisher: Ashling Microsystems Ltd

☑ Always trust software from "Ashling Microsystems Ltd".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 67. Install MCUXpresso required drivers IV

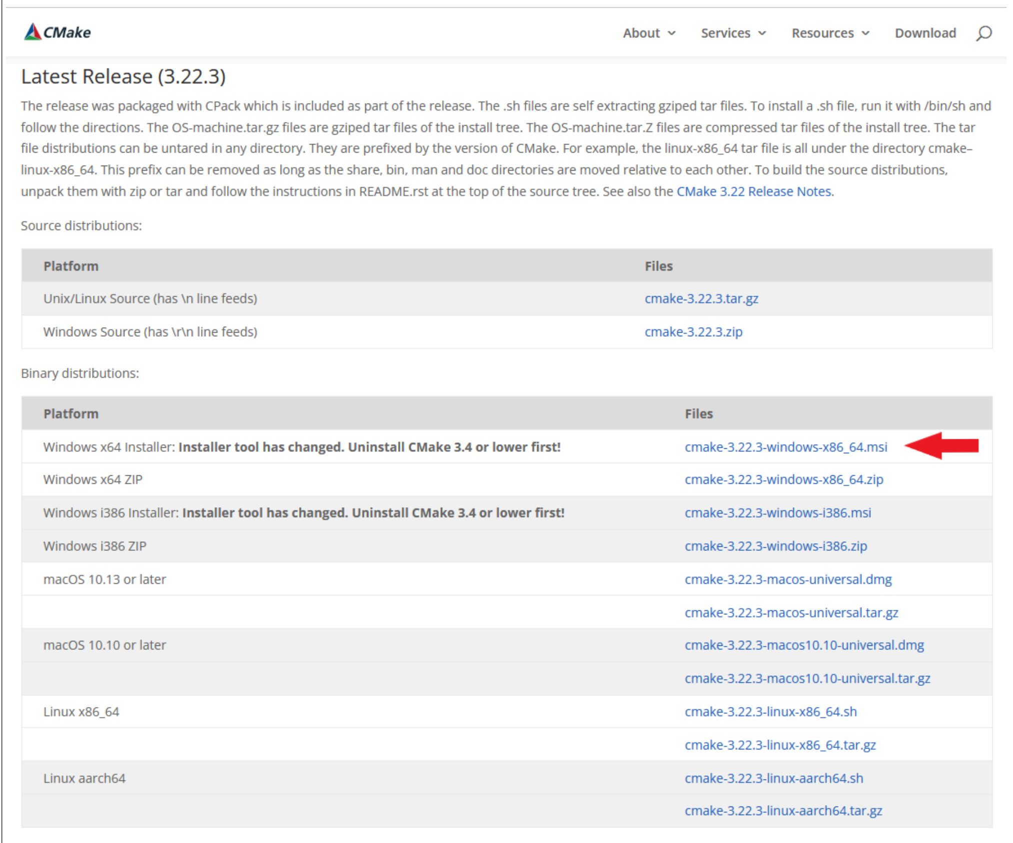
8 Appendix B: Install CMake

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 68:



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:

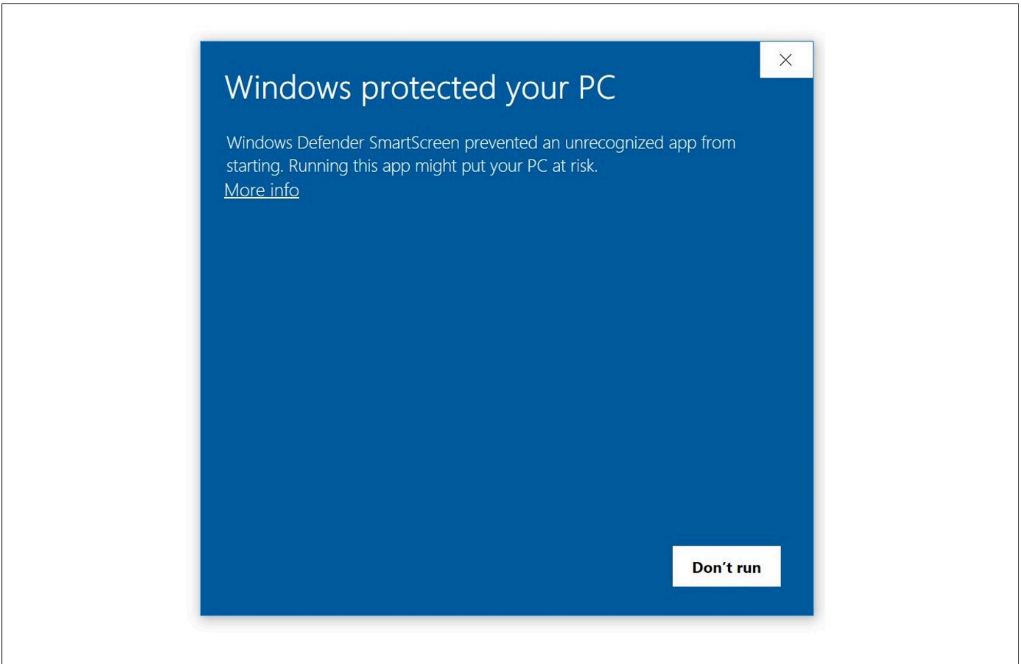
Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.22.3.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.22.3.zip

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 68. Download CMake

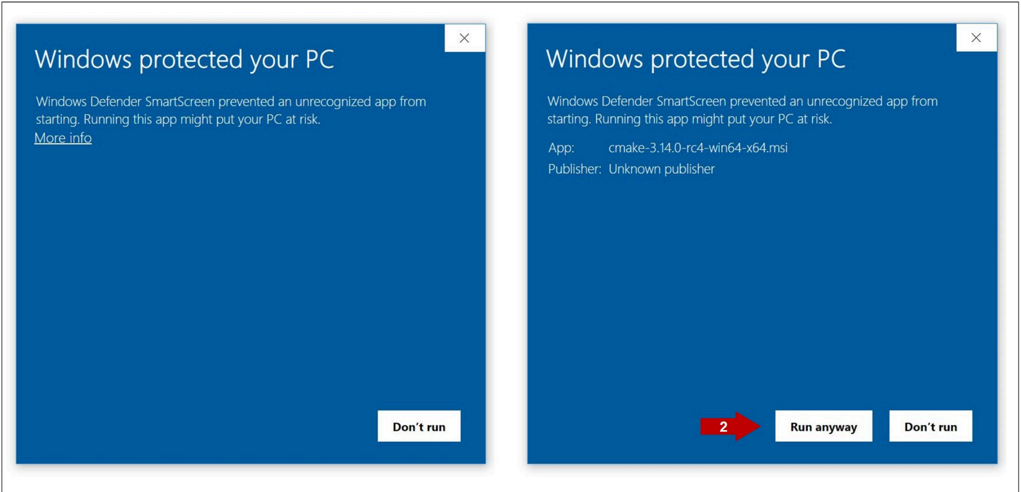
3. Double click on the downloaded installer file. Windows Defender SmartScreen might pop-up the wizard shown in Figure 69:


Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.
More info

Figure 69. 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 70:


Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.
More info

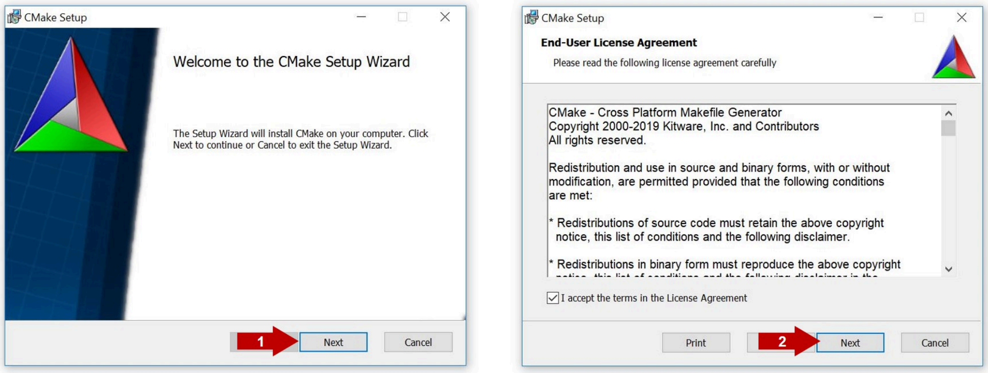
Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: cmake-3.14.0-rc4-win64-x64.msi
Publisher: Unknown publisher

Figure 70. 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 71:


CMake Setup
Welcome to the CMake Setup Wizard
1
Next
Cancel
CMake Setup
End-User License Agreement
Please read the following license agreement carefully
Print
2
Next
Cancel

The Setup Wizard will install CMake on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

CMake - Cross Platform Makefile Generator
Copyright 2000-2019 Kitware, Inc. and Contributors
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

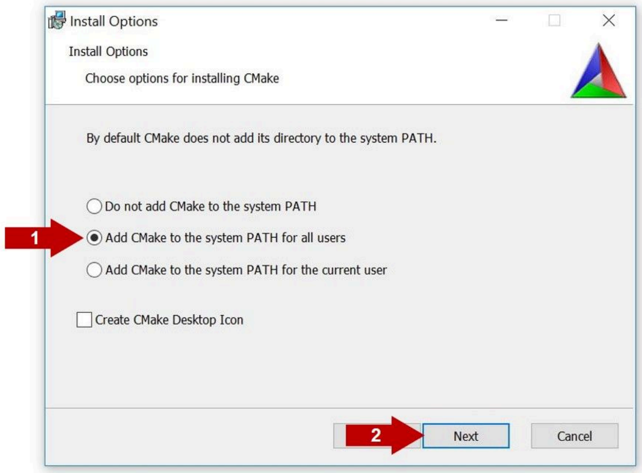
* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright

[✓] I accept the terms in the License Agreement

Figure 71. 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 72:



Install Options

Choose options for installing CMake

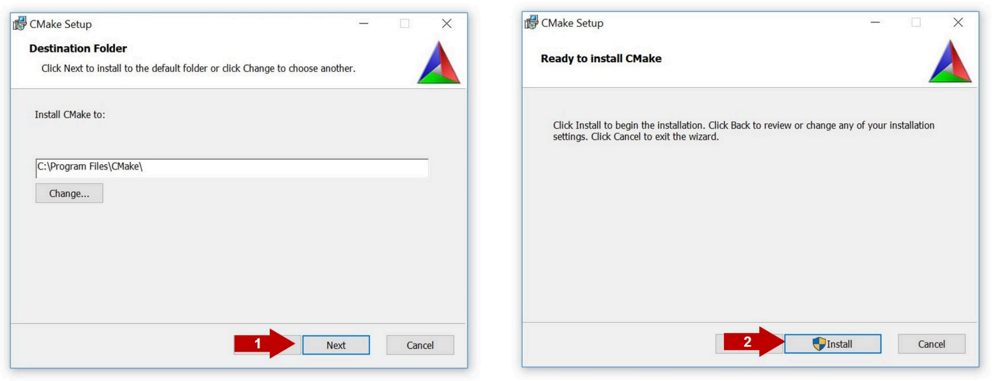
By default CMake does not add its directory to the system PATH.

○ Do not add CMake to the system PATH
● Add CMake to the system PATH for all users
○ Add CMake to the system PATH for the current user

☐ Create CMake Desktop Icon

Figure 72. Add CMake path

7. Select a destination folder, (1) click ***Next*** and then (2) click ***Install*** as shown in Figure 73:



CMake Setup

Destination Folder

Click Next to install to the default folder or click Change to choose another.

Install CMake to:

C:\Program Files\CMake\

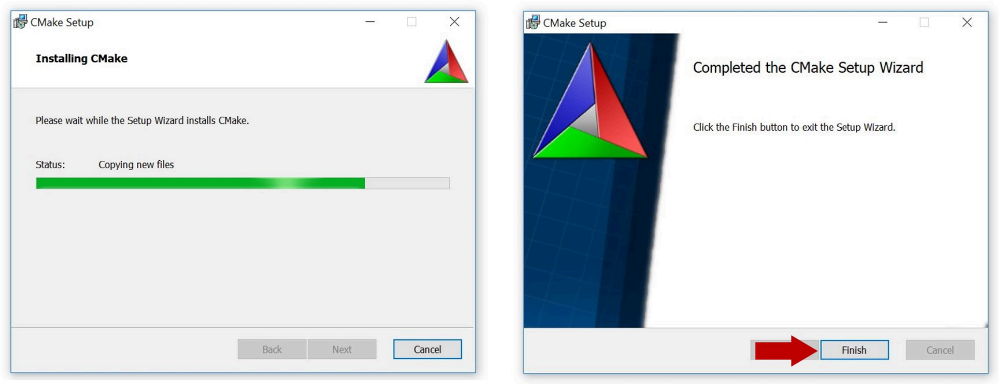
CMake Setup

Ready to install CMake

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Figure 73. Install CMake

8. Wait a few seconds until the installation is completed and click ***Finish*** as shown in Figure 74:



CMake Setup

Installing CMake

Please wait while the Setup Wizard installs CMake.

Status: Copying new files

CMake Setup

Completed the CMake Setup Wizard

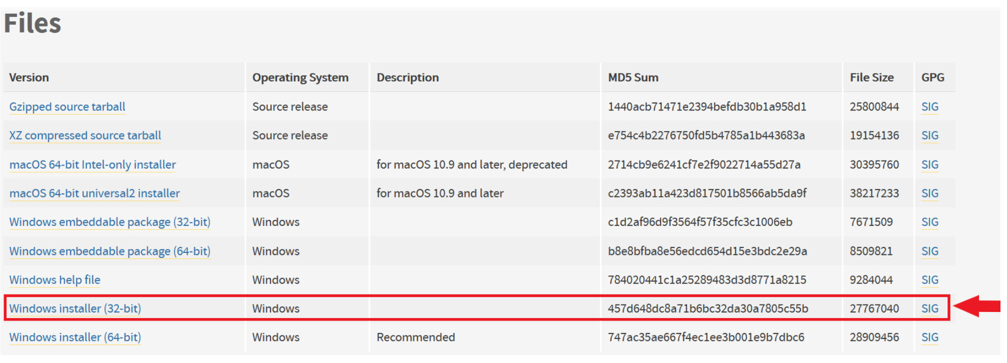
Click the Finish button to exit the Setup Wizard.

Figure 74. 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.

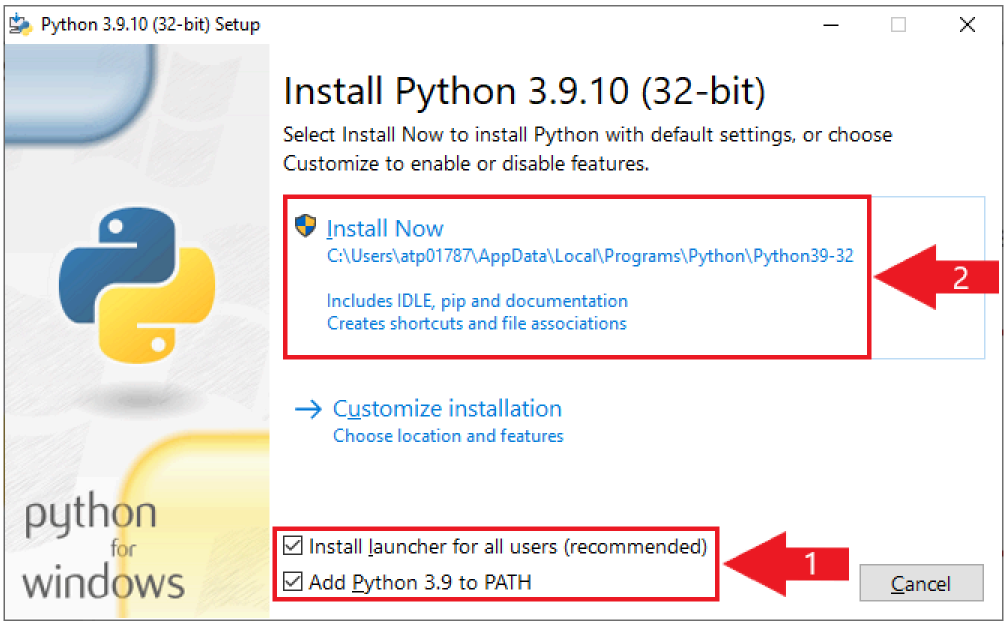


Files

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 75. 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 76:



Python 3.9.10 (32-bit) Setup

Install Python 3.9.10 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\atp01787\AppData\Local\Programs\Python\Python39-32

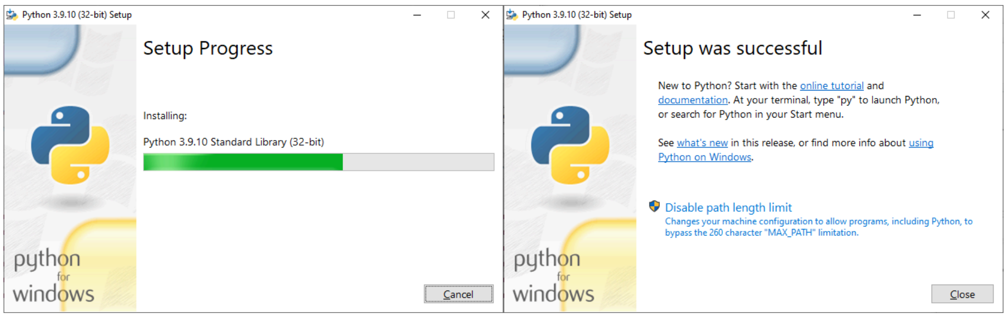
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.9 to PATH

Figure 76. Install Python 3.9.x 32 bit for Windows

3. Wait a few seconds until the installation is completed as indicated in Figure 77



Python 3.9.10 (32-bit) Setup

Setup Progress

Installing:

Python 3.9.10 Standard Library (32-bit)

Cancel

Python 3.9.10 (32-bit) Setup

Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

Disable path length limit

Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Close

Figure 77. 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.

Contents

Date of release: 4 August 2022
Document identifier: AN12542
Document number: 560833