

AN14238

Getting started with EdgeLock A30 secure authenticator support package

Rev. 2.0 — 2 September 2025

Application note

Document information

Information	Content
Keywords	EdgeLock A30 secure authenticator, NX Middleware
Abstract	This document is the entry point for getting familiar with EdgeLock A30 support package contents and how to get started with them.



1 About EdgeLock A30 secure authenticator

EdgeLock A30 is a secure authentication IC for IoT platforms, electronic accessories and consumable devices such as home electronic devices, mobile accessories and medical supplies.

EdgeLock A30 supports on-chip ECC key generation to make sure that private keys are never exposed outside the IC. It performs cryptographic operations for security critical communication and control functions. EdgeLock A30 is Common Criteria EAL 6+ security certified with AVA_VAN.5 on product level and supports a generic Crypto API providing AES, ECDSA, ECDH, SHA, HMAC and HKDF cryptographic functionality.

- Asymmetric cryptography features support 256-bit ECC over the NIST P-256 and brainpool P256r1 curves.
- Symmetric cryptography features support both AES-128 and AES-256.
- PKI-based mutual authentication based on the Sigma-I protocol.
- Symmetric three-pass Mutual Authentication protocol.
- Secure messaging channel using either AES-128 or AES-256 session encryption/decryption and MAC.
- Number of supported persistent key entries:
 - Up to five persistent EC key entries for Sigma-I and crypto primitives.
 - Up to five persistent AES key entriesfor mutual authentication, Secure Dynamic Messaging and key update. These five AES key entries are not consuming any user memory.
 - Up to eight persistent AES key entriesfor crypto operations.

The Common Criteria security certification ensures that the IC security measures and protection mechanisms have been evaluated against sophisticated noninvasive and invasive attack scenarios.

- A30 supports an I²C contact interface and has two additional GPIOs.
- A30 supports a low-power design, and consumes only 5 µA at Deep-Power-Down mode when an external VDD is supplied.

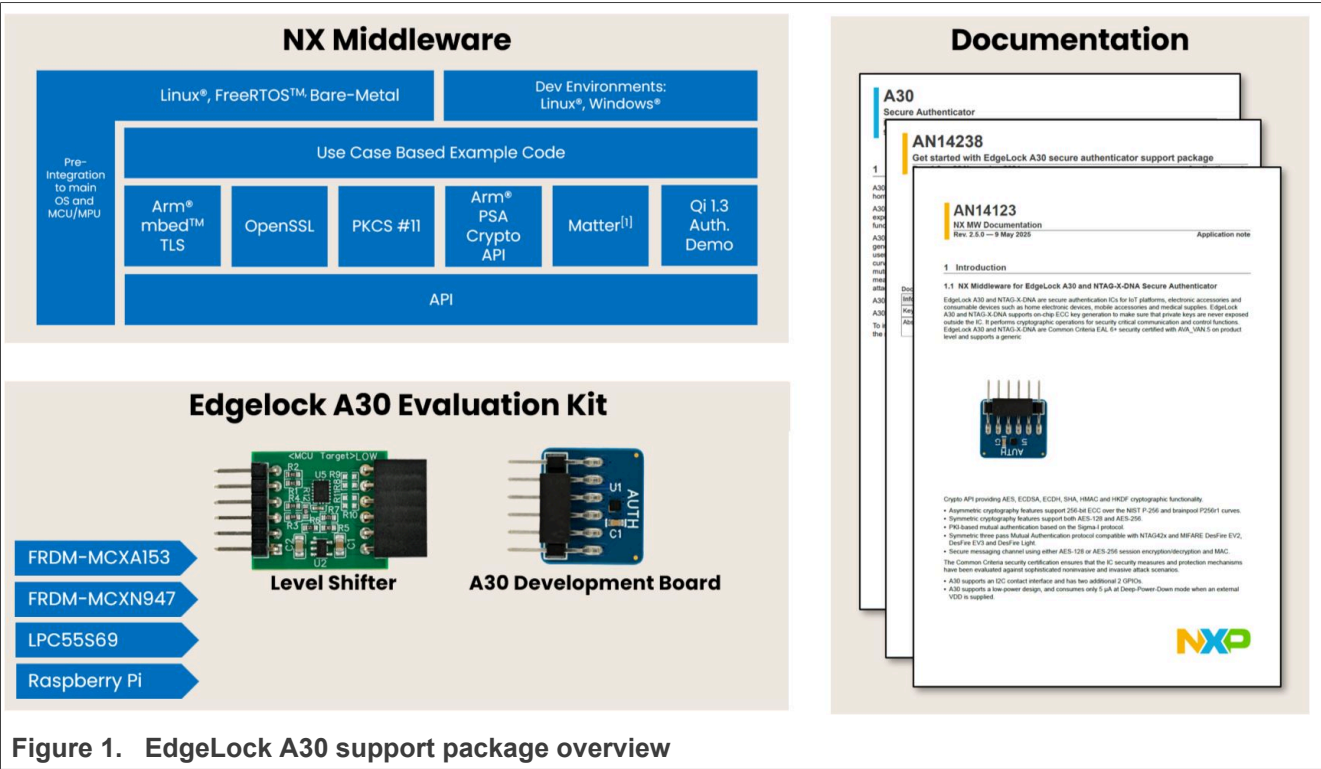


Figure 1. EdgeLock A30 support package overview

Getting started with EdgeLock A30 secure authenticator support package

Delivered as a ready-to-use solution, the EdgeLock A30 includes a complete product support package that simplifies design-in and reduces time to market. The EdgeLock A30 support package offers:

- EdgeLock A30 development kit
- NX Middleware
 - Software enablement for MCUs and MPUs.
 - Integration with the most common cryptographic libraries like OpenSSL, Mbed TLS and PKCS #11.
 - Multi-platform software enablement targeting freeRTOS and Linux as well as Windows as evaluation platform.
 - Sample code for major IoT and secure authentication use cases.
- Documentation

This document lists the existing material within EdgeLock A30 support package, organized in the following sections:

- [Section 2 "EdgeLock A30 development kit"](#)
- [Section 3 "MCU/MPU reference boards"](#)
- [Section 4 "NX Middleware"](#)
 - [Section 4.4 "Evaluation using a Windows PC"](#)
 - [Section 4.5 "Evaluation using a MCU board"](#)
 - [Section 4.6 "Evaluation using Raspberry Pi"](#)
 - [Section 4.7 "Device provisioning"](#)
 - [Section 4.8 "Device configuration"](#)
- [Section 5 "EdgeLock 2Go service for EdgeLock A30"](#)
- [Section 6 "Supported EdgeLock A30 documentation"](#)
- [Section 7 "Appendix"](#)
 - [Section 7.1 "EdgeLock A30 product identification"](#)
 - [Section 7.2 "Available free memory of EdgeLock A30 ICs"](#)
 - [Section 7.3 "EdgeLock A30 application circuit diagram"](#)
 - [Section 7.4 "EdgeLock A30 WLCSP16 pin description"](#)
 - [Section 7.5 "EdgeLock A30 HVQFN20 pin description "](#)
 - [Section 7.6 "EdgeLock A30 development board schematic "](#)
 - [Section 7.7 "Level shifter board schematic"](#)

2 EdgeLock A30 development kit

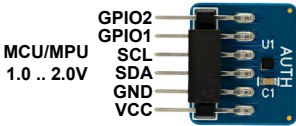

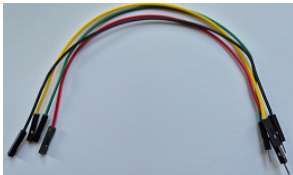
The A30-EVAL is a flexible and easy-to-use development kit for the EdgeLock A30 secure authenticator. It can be used in various MCU and MPU platforms. This kit allows the evaluation of the EdgeLock A30 secure authenticator features and simplifies the development of secure IoT and authentication applications.

The A30 development kit including:

- EdgeLock A30 development boards
- Level shifter board
- Jumper wires

Table 1 summarizes the contents of the evaluation kit.

Table 1. EdgeLock A30 development kit A30-EVAL

Part number	12NC	Number of pieces	Content	Picture
A30-EVAL	935505 094598	3	A30 evaluation board	
		1	Level shifter board	
		6	Jumper wires	

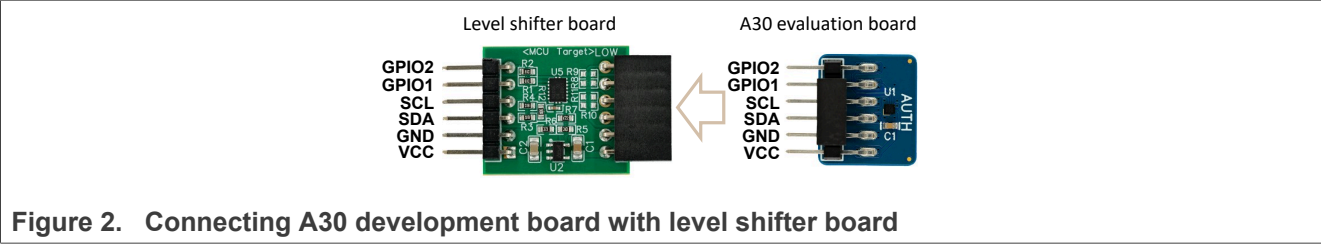
The EdgeLock A30 is optimized for battery-powered applications and systems utilizing MCUs or MPUs with a 1.8 V supply voltage. To ensure compatibility and power efficient operation in such environments, the EdgeLock A30 supports a supply voltage range from 1.0 V to 2.0 V.

While most modern MCU and MPU families support operation at 1.8 V, many evaluation kits continue to use higher voltages such as 3.3 V or even 5 V. To facilitate seamless and rapid prototyping, the EdgeLock A30 development kit integrates a level shifter that translates voltage levels as required, ensuring compatibility across different platforms.

The corresponding board schematics can be found in [Section 7.6](#) and [Section 7.7](#).

Figure 2 shows how to connect the Level Shifter and A30 board.

Getting started with EdgeLock A30 secure authenticator support package

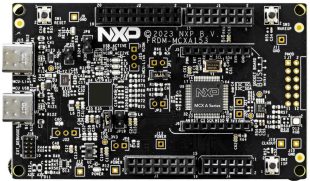
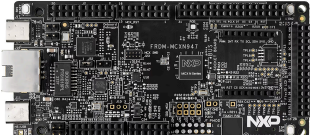
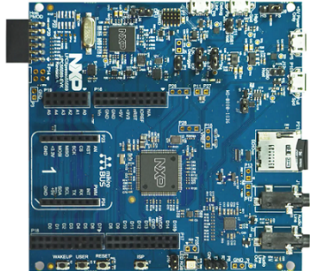


3 MCU/MPU reference boards

The EdgeLock A30 secure authenticator IC is designed to be used as a part of an IoT system. It works as an auxiliary security authenticator device attached to a host controller (MCU or MPU board). The host controller communicates with EdgeLock A30 through an I²C interface with the host controller being the controller and the EdgeLock A30 being the target.

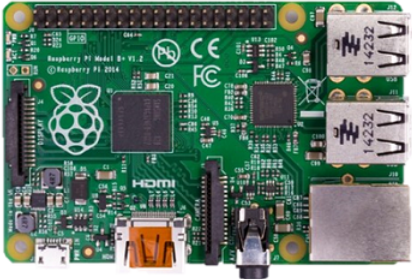
The EdgeLock A30 can be connected to any MCU/MPU on the market supporting the I²C interface. To enable easy evaluation of the EdgeLock A30, the NX Middleware supports the NXP FRDM-MCXA153, FRDM-MCXN947 and the LPCXpresso 55S69 demo board as an MCU reference platform.

Table 2. MCU reference boards

Part number	12NC	Content	Picture
FRDM-MCXA153	935455513598	FRDM development board for MCX A14x/A15x MCUs	
FRDM-MCXN947	935455837598	FRDM development board for MCX N94/N54 MCUs	
LPC55S69-EVK	935377412598	LPCXpresso55S69 development board	

The Raspberry Pi is used to demonstrate the embedded Linux enablement of A30. The NX Middleware is compatible with all Raspberry Pi board versions and has been validated on the Raspberry Pi 4 Model B.

Table 3. MPU reference boards

Part number	Content	Picture
Raspberry Pi	Raspberry Pi (any model)	

4 NX Middleware

4.1 Overview

The NX Middleware is a unified software stack that simplifies the integration of the EdgeLock A30 secure authenticator IC into MCU or MPU-based software platforms. It is a single software stack designed to facilitate the integration of EdgeLock A30 secure authenticator IC into your MCU or MPU software. The NX Middleware abstracts the commands and communication interface exposed by EdgeLock A30. It is directly accessible from stacks like mbedTLS, OpenSSL and PKCS #11. In addition, it includes code examples for quick integration of features and use cases such as TLS and AWS cloud service onboarding. It also comes with support for reference MCU/MPU platforms and can be ported to multiple host platforms and host operating systems.

Figure 3 is a simplified representation of the layers and components of NX Middleware:

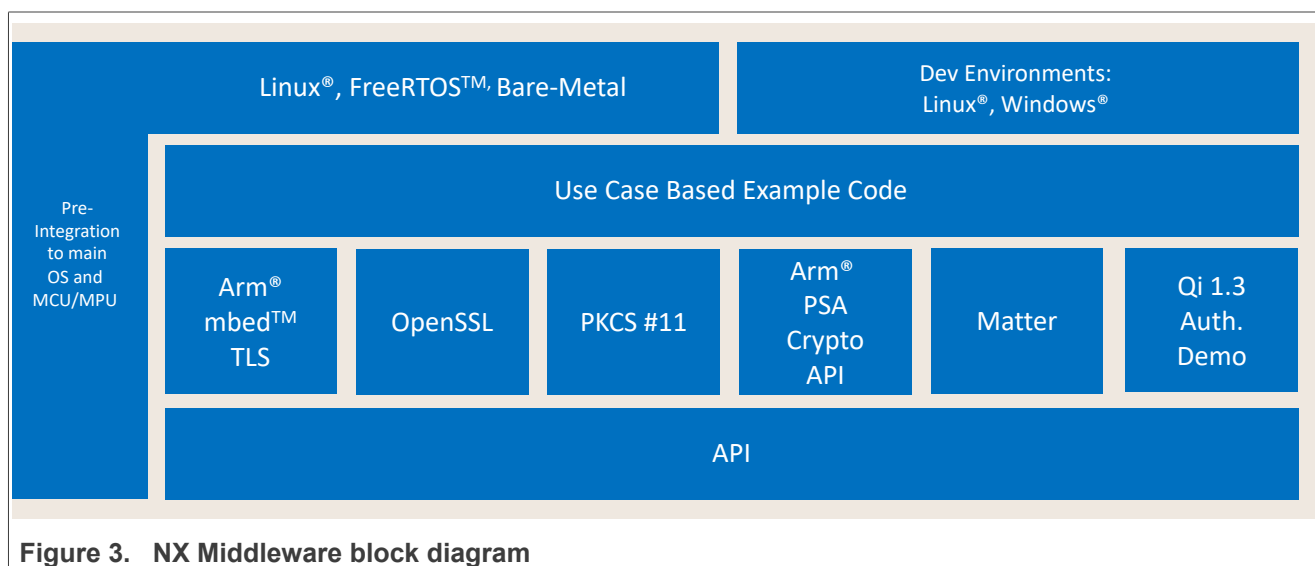


Figure 3. NX Middleware block diagram

The NX Middleware is delivered with CMake files that include the set of directives and instructions describing the project's source files and targets. The CMake files allow developers to build NX Middleware in their target platform, enable or disable features or change setting flags, among others.

The CMake based compilation option is provided as a convenient way for developers to run a project example on different target platforms:

- Windows/Linux PC for evaluation purpose
- MCU boards
- MPU boards

The NX Middleware has built-in cryptographic and abstracts the commands as well the communication interface exposed by NXP EdgeLock A30 secure authenticator IC. The NX Middleware is directly accessible from the following cryptographic libraries:

- ARM mbedTLS and PSA (Platform Security Architecture)
- OpenSSL
- PKCS #11

The NX Middleware can be downloaded via GitHub <https://github.com/NXP/nxmw>.

4.2 Architecture overview

Figure 4 gives a brief overview of the NX Middleware.

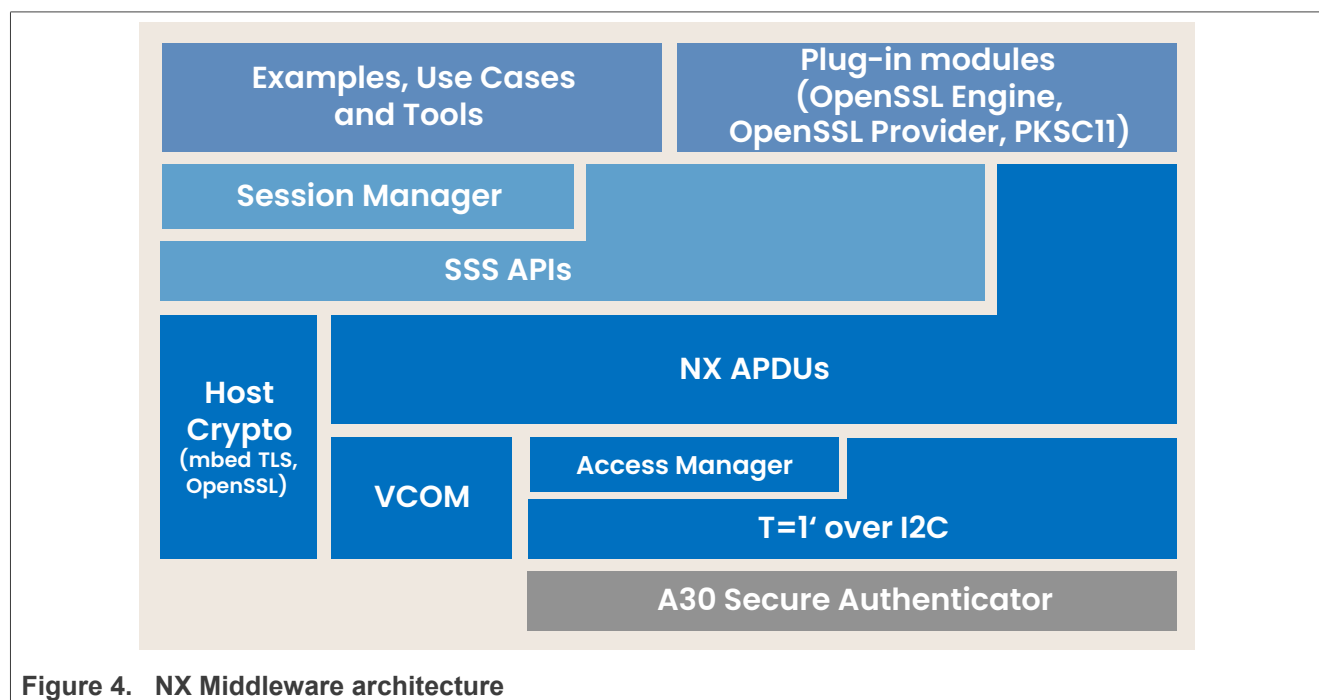


Figure 4. NX Middleware architecture

- Session Manager
 - APIs to open a session. The following sessions are supported:
 - Plain session
 - PKI based asymmetric Mutual Authentication (Sigma-I-Verifier or Sigma-I-Prover)
 - AES-based Symmetric Mutual Authentication
 - Note: Both mutual authentication methods initiate a MIFARE DESFire compatible EV2 secure messaging channel (authenticated session).
- SSS APIs
 - Provides abstraction APIs for EdgeLock A30, OpenSSL and mbedTLS host crypto.
 - SSS APIs are supporting common crypto operations.
- NX APDUs
 - Implements the EdgeLock A30 authenticator commands (APDUs)
- Access Manager
 - Manage access from multiple Linux processes to EdgeLock A30. Client processes connect over the JRCPv1 protocol to the Access Manager.

T=1' over I²C communication protocol according to Global Platform.

4.3 Code documentation

The code documentation provided as part of NX Middleware package in PDF format and as a part of the GitHub release. The primary audience are programmers, developers, system architects and system designers.

The PDF version of the NX Middleware documentation ([AN14123](#)) is located in the `nxmw\doc` folder as shown in [Figure 5](#).

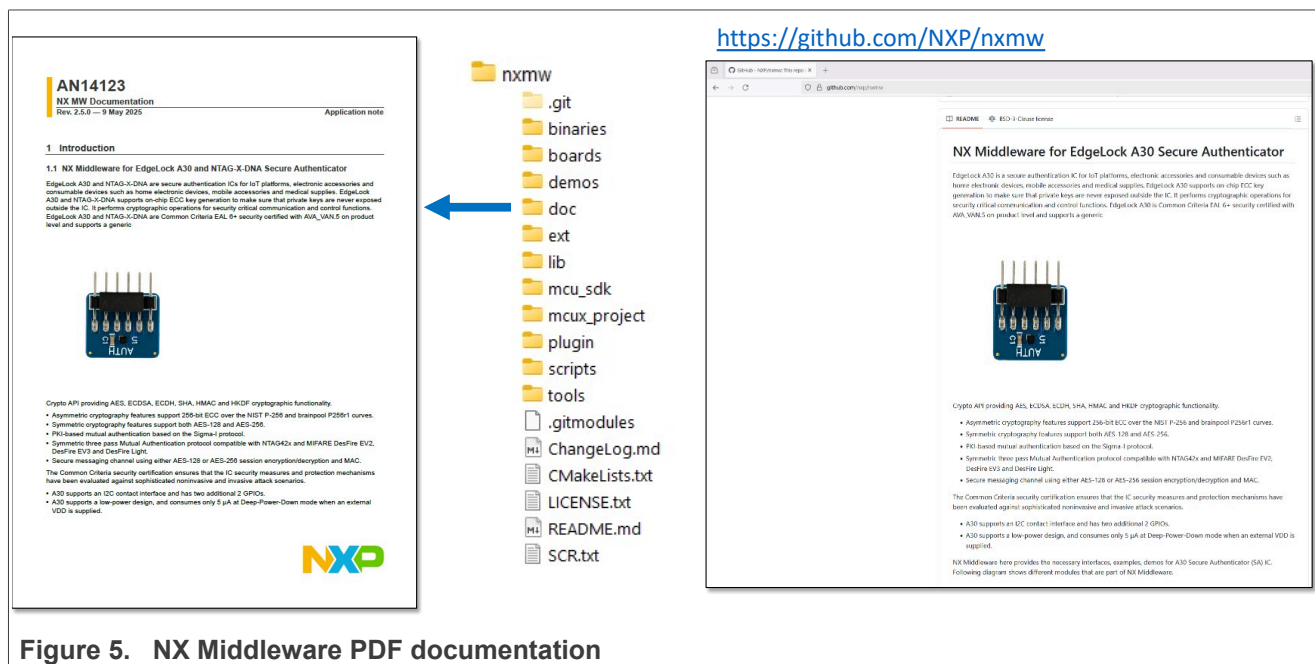


Figure 5. NX Middleware PDF documentation

4.4 Evaluation using a Windows PC

To accelerate evaluation and prototyping, the NX Middleware enables the protocol stack to be built and executed directly on a Windows PC, eliminating the need for immediate porting to the target MCU or MPU platform.

To connect the EdgeLock A30 evaluation board to a Windows PC, one of the following supported MCU development boards is required:

- [FRDM-MCXA153](#)
- [FRDM-MCXN947](#)
- [LPC55S69-EVK](#)

These development boards connect to a Windows PC via USB and must be running the T=1 over I²C firmware. This firmware serves as a communication bridge between the PC's virtual COM (VCOM) port and the EdgeLock A30.

The EdgeLock A30 and the level shifter board must be connected as illustrated in [Figure 7](#), [Figure 8](#) and [Figure 9](#).

When the NX Middleware is compiled for the Windows platform, all low-level APDU commands and responses intended for the EdgeLock A30 are transmitted over the VCOM interface, rather than directly using the T=1 over I²C protocol. T=1 over I²C protocol is handled by the MCU firmware.

Getting started with EdgeLock A30 secure authenticator support package

The required T=1 over I²C firmware is available both as an [MCU project](#) and as [a pre-compiled binary](#) on GitHub. The pre-compiled binaries can be easily programmed onto the supported MCU boards using the GUI Flash Tool integrated into the [MCUXpresso IDE](#).

For detailed instructions, refer to the MCUXpresso IDE User Guide, section 17.1.3 "Advanced GUI Flash Tool – Programming an Arbitrary Binary".

Alternatively, the firmware can be programmed without using the MCUXpresso IDE by using the standalone LinkFlash utility. The GUI LinkFlash utility, included with [LinkServer](#), is designed for programming and managing flash memory on NXP target devices.

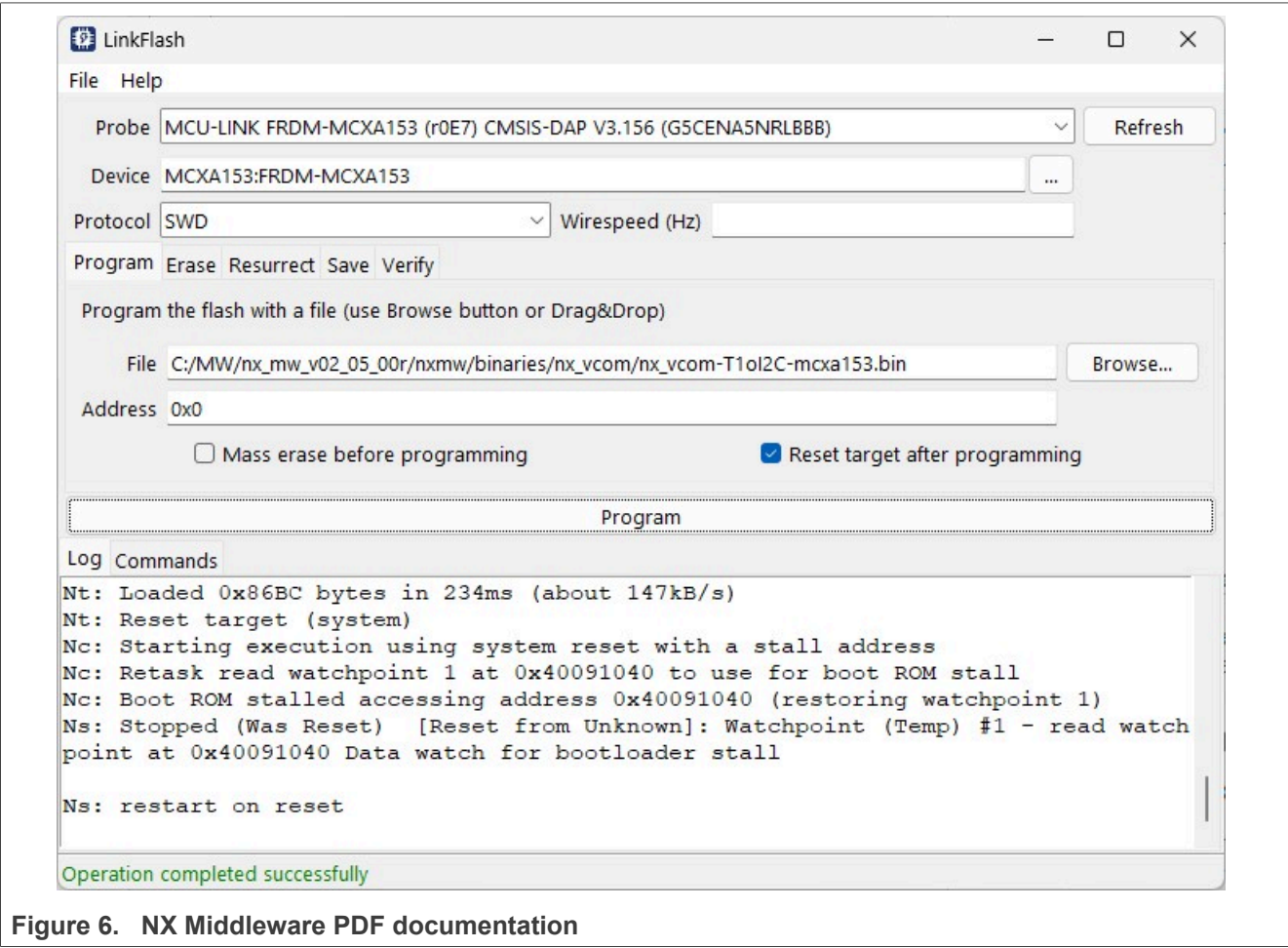
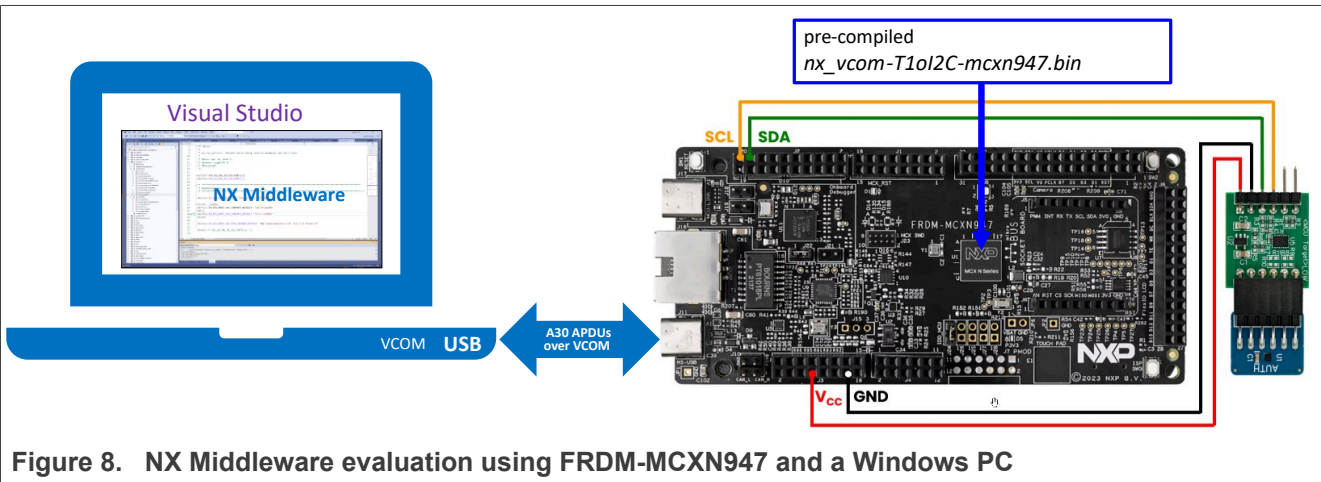
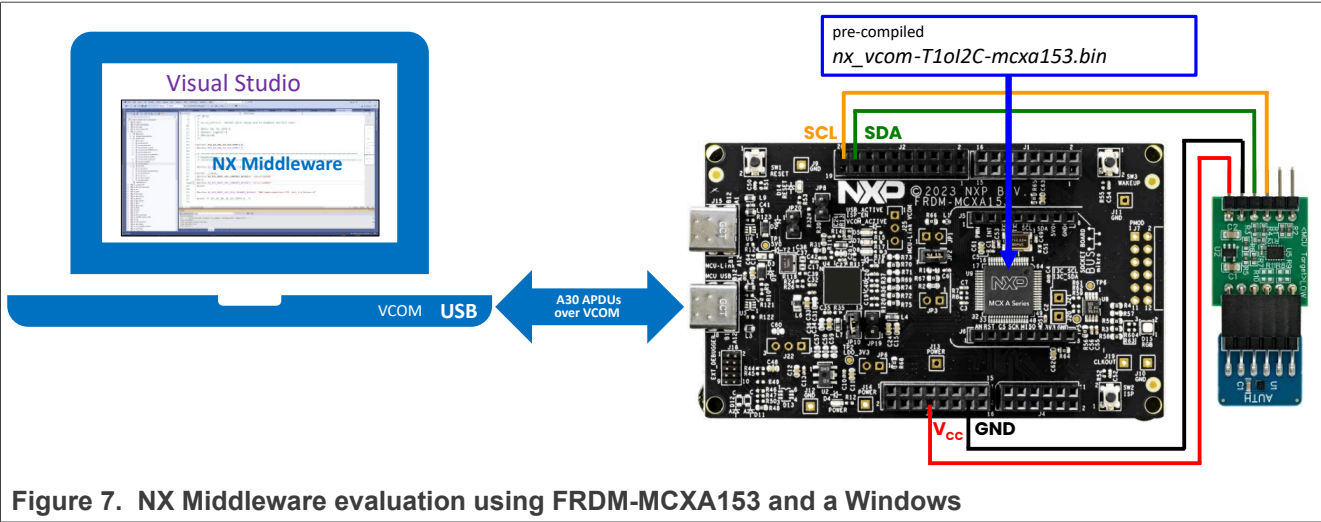
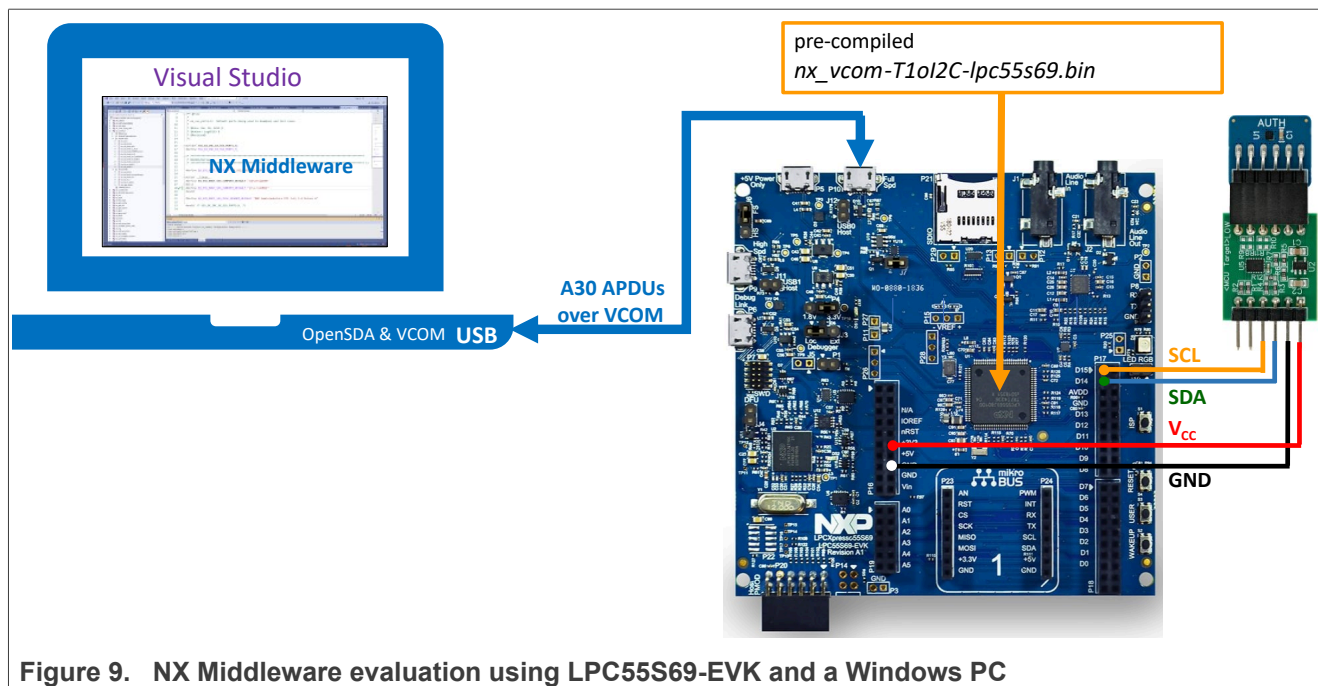


Figure 6. NX Middleware PDF documentation

Getting started with EdgeLock A30 secure authenticator support package





4.4.1 Building NX Middleware on Windows

4.4.1.1 Prerequisites

To successfully build the NX Middleware on a Windows PC, ensure the following tools are installed:

- [Visual Studio](#) (Version 2015 or later)
- [Python 3](#) (32-bit or 64-bit, Version 3.10.x to 3.11.x)
- [Git](#) (actual version)
- [West \(Zephyr's meta-tool\)](#) (Version 1.2.0 or later)
- [CMake](#) (Version 3.30.0 or later). Preferred installation path: C:/opt/cmake/bin/cmake.exe
- [OpenSSL](#) (64-bit, Version 3.4.0 or later)

4.4.1.2 Download or Clone the NX Middleware

Note: Avoid using spaces in the path for <nx_mw_root_folder>

Option A: Download the latest release of the NX Middleware via West

```
west init -m https://github.com/NXP/nxmw.git --mf mcu_sdk/west.yml workspace
cd workspace
west update
```

Resulting folder structure::

```
| - nx_mw_root_folder>
  | - workspace
    | - .west
    | - mcuxsdk
    | - nxmw
```

Option B: Clone the latest release of the NX Middleware via Git

```
git clone --recurse-submodules https://github.com/NXP/nxmw.git nxmw
cd nxmw
west init -l --mf mcu_sdk/west.yml
cd ..
west update
```

Resulting folder structure::

```
| - nx_mw_root_folder>
  | - .west
  | - mcuxsdk
  | - nxmw
```

4.4.1.3 Set Up the NX Middleware Development Environment

Note: The `env_setup.bat` script configures the necessary environment variables for the build tools. If your installations of MCUXpresso, Visual Studio, Java, Python, or CMake are in non-default directories, please update the paths in `env_setup.bat` accordingly.

Open a command prompt and navigate to the following directory:

```
cd nxmw\scripts
```

Run the environment setup script:

```
env_setup.bat
```

In the same directory, execute the Python script to generate the build files:

```
python create_cmake_projects.py
```

Depending on your Python installation, you may need to use:

```
py create_cmake_projects.py
```

The build files will be generated in: `<nx_mw_root_folder>\nxmw_build`

The Visual Studio solution file will be located at: `<nx_mw_root_folder>\nxmw_build\se_x86\NxMW.sln`

4.4.1.4 Open CMake GUI to explore the default CMake settings

Navigate to <nx_mw_root_folder>\nxmw_build\se_x86\ and run the follwing command to open the CMake GUI.

```
cmake-gui .
```

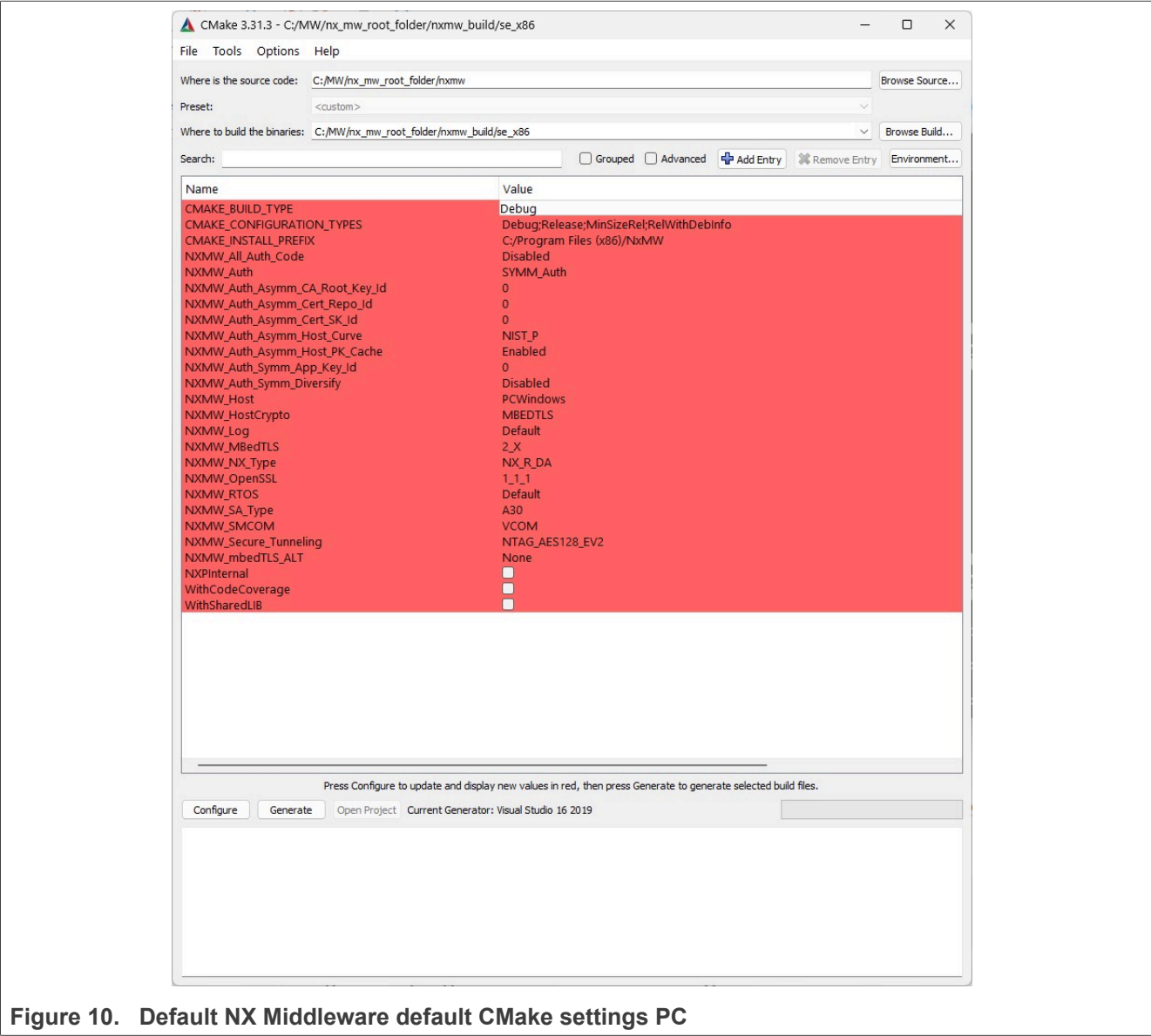


Figure 10. Default NX Middleware default CMake settings PC

The default EdgeLock A30 CMake settings are:

```
CMAKE_BUILD_TYPE:STRING=Debug
CMAKE_CONFIGURATION_TYPES:STRING=Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX:PATH=C:/Program Files (x86)/NxMW
NXMW_All_Auth_Code:STRING=Disabled
```

```
NXMW_Auth:STRING=SYMM_Auth
NXMW_Auth_Asymm_CA_Root_Key_Id:STRING=0
NXMW_Auth_Asymm_Cert_Repo_Id:STRING=0
NXMW_Auth_Asymm_Cert_SK_Id:STRING=0
NXMW_Auth_Asymm_Host_Curve:STRING=NIST_P
NXMW_Auth_Asymm_Host_PK_Cache:STRING=Enabled
NXMW_Auth_Symm_App_Key_Id:STRING=0
NXMW_Auth_Symm_Diversify:STRING=Disabled
NXMW_Host:STRING=PCWindows
NXMW_HostCrypto:STRING=MBEDTLS
NXMW_Log:STRING=Default
NXMW_NX_Type:STRING=NX_R_DA
NXMW_OpenSSL:STRING=1_1_1
NXMW_RTOS:STRING=Default
NXMW_SA_Type=A30
NXMW_SMCOM:STRING=VCOM
NXMW_Secure_Tunneling:STRING=NTAG_AES128_EV2
NXPInternal:BOOL=ON
NXTST_FTR_TEST_COUNT:BOOL=OFF
WithCodeCoverage:BOOL=OFF
WithSharedLIB:BOOL=OFF
```

There is no need to modify any CMake option for the following examples. For further details please refer to the NX Middleware documentation on [GitHub](#).

4.4.1.5 Build NX Middleware using the command line

To build the NX Middleware including all demo/examples using the command line run:

```
cmake --build .
```

To build a single example run:

```
cmake --build . --target <example_name>
```

For example, the build only the ECC example code run:

```
cmake --build . --target ex_ecc
```

4.4.2 Running NX Middleware examples from command line

Navigate to <nx_mw_root_folder>\nx-mw-top_build\se_x86\bin\Debug

Use the corresponding VCOM port number to run e.g. the example nx_ecc.exe

```
nx_ecc.exe COMxx
```

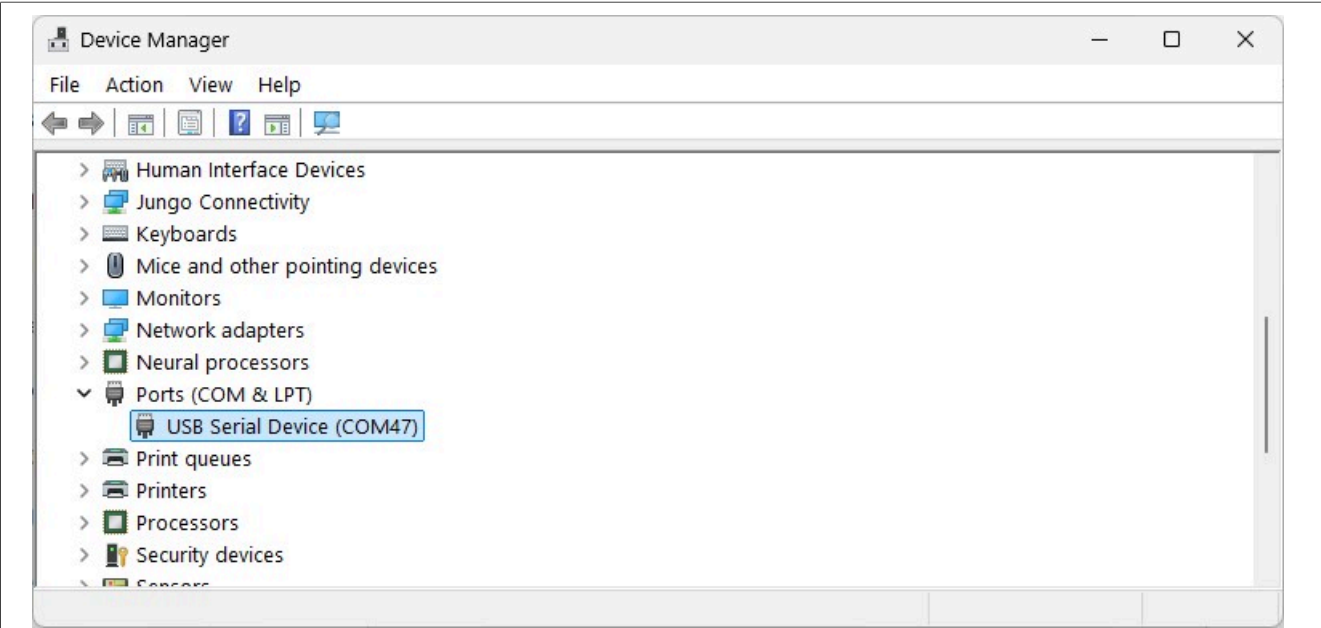



Figure 11. Windows Device Manger – Identify the correct VCOM port PC

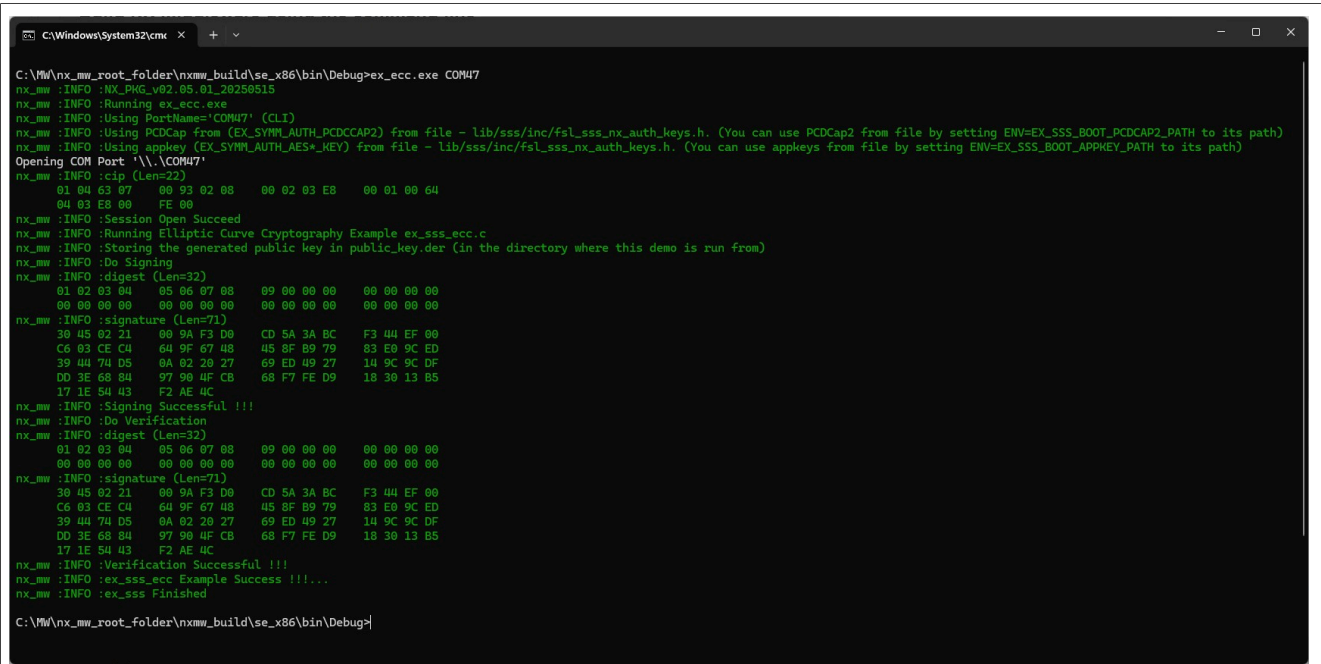


Figure 12. Executing the nx_ecc.exe example PC

4.4.3 Running NX Middleware example projects in Visual Studio

4.4.3.1 Opening the Project in Visual Studio

Navigate to the following directory: `<nx_mw_root_folder>\nxmw_build\se_x86\`. Locate and double-click the `NxMW.sln` file to open the solution in Visual Studio.

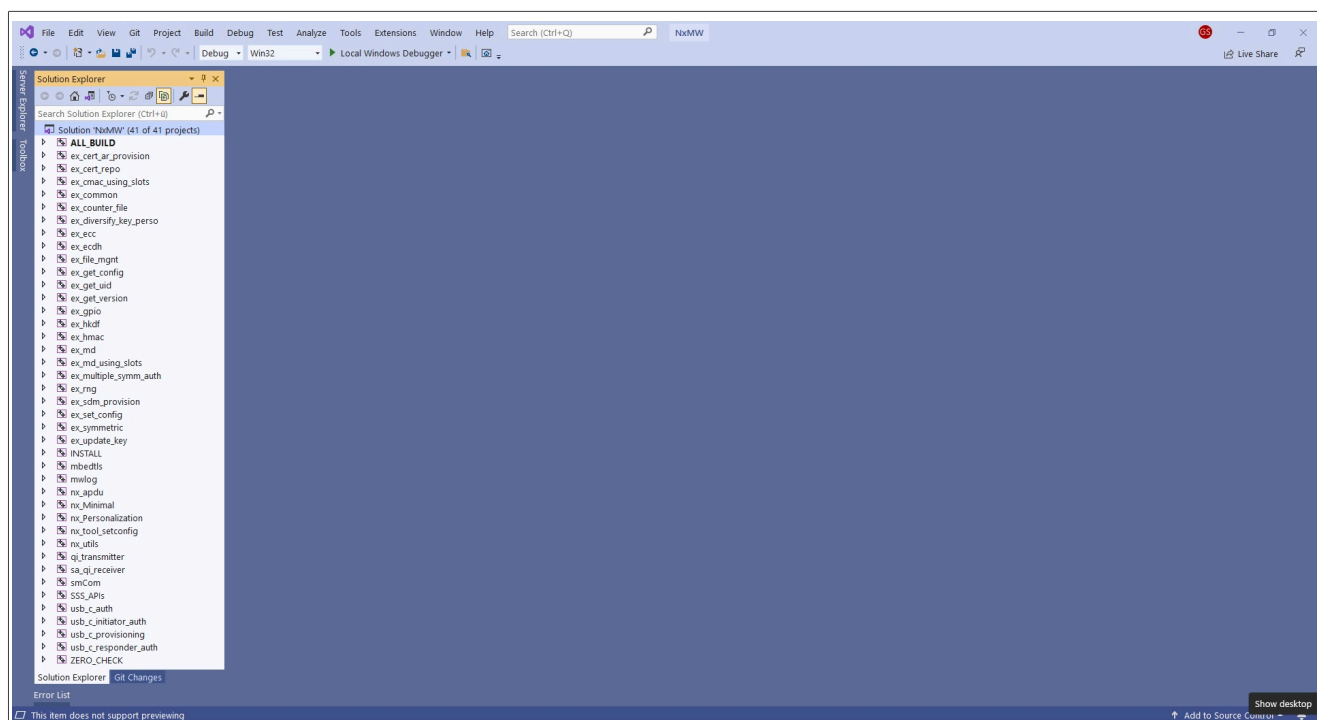


Figure 13. NxMW.sln Visual Studio project workspace

4.4.3.2 Configuring the VCOM Port

To configure the correct COM port for the NX middleware project:

- Navigate to the `ex_common` project folder.
- Open the `ex_sss_ports.h` file located in the headers directory.
- Locate the following line:

```
#define EX_SSS_BOOT_SSS_COMPORT_DEFAULT "\\.\COMx"
```

Replace `COMx` with the actual COM port number assigned to your device (e.g., `COM3`).

Getting started with EdgeLock A30 secure authenticator support package

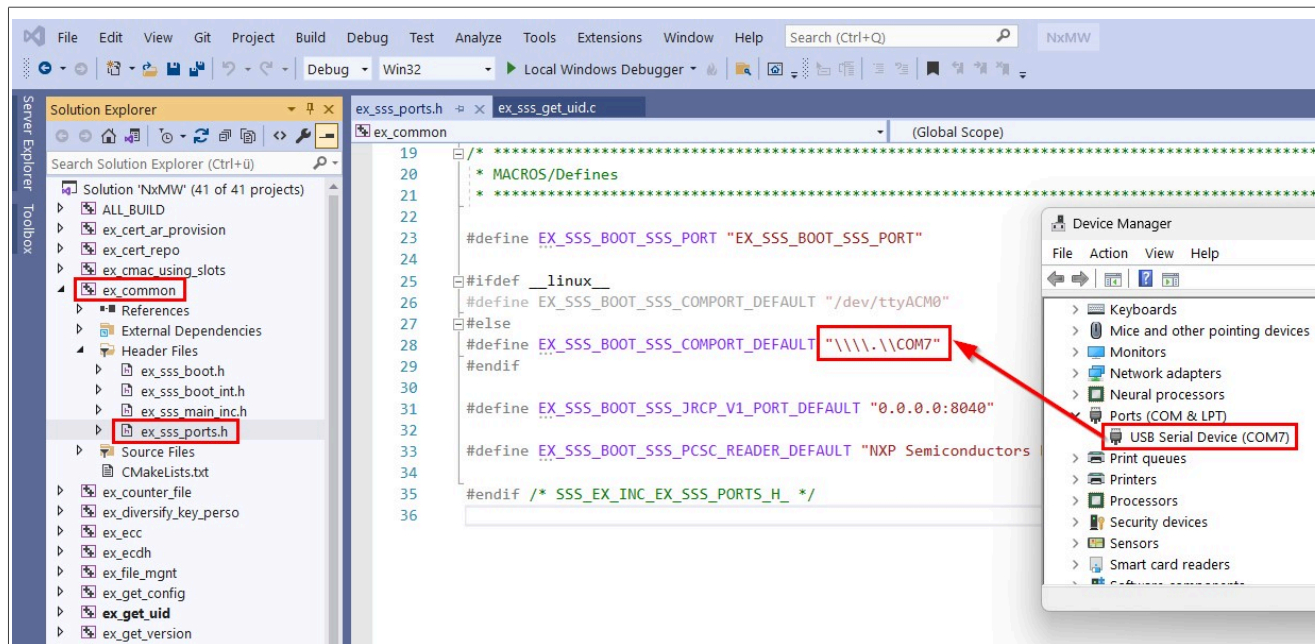


Figure 14. Change VCOM port number

4.4.3.3 Set an Example Project as Startup

To define an example project (e.g., `ex_get_uid`) as the startup project:

- In Solution Explorer, right-click on the `ex_get_uid` project.
- Select *Set as StartUp Project* from the context menu.

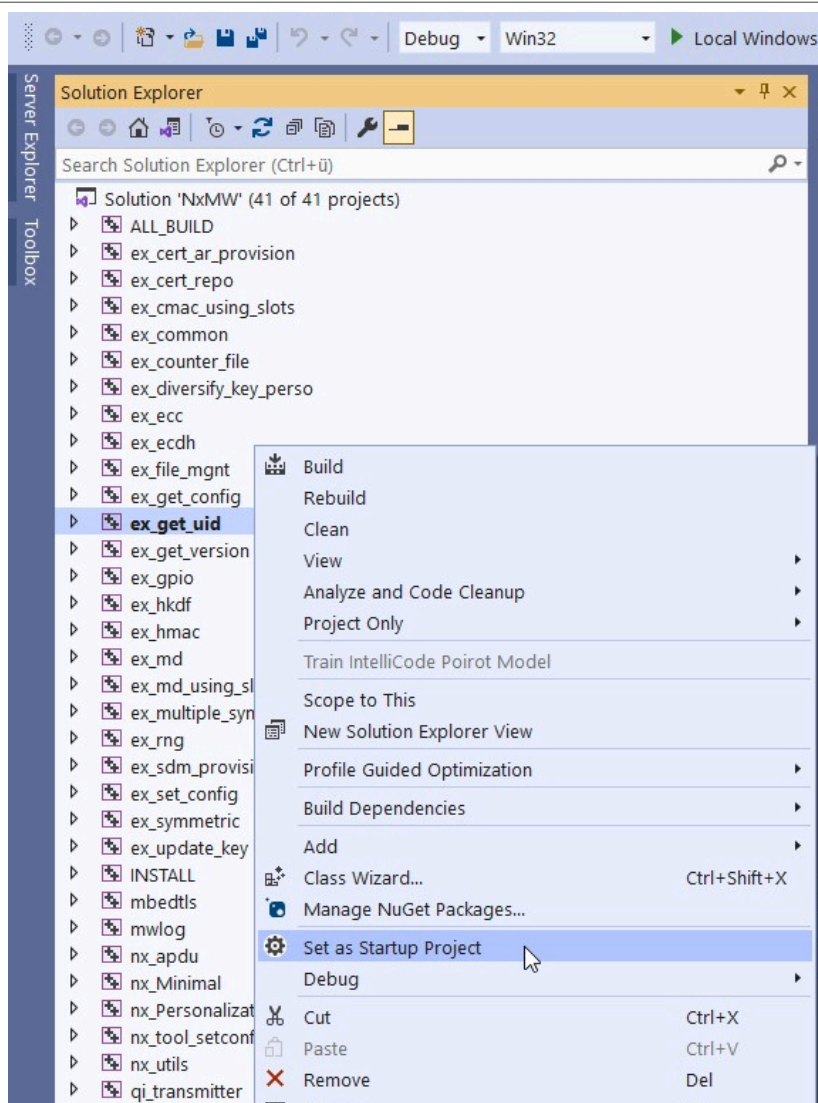


Figure 15. Set `ex_get_uid` as Visual Studio startup project

4.4.3.4 Build and Run the Project

- Click Build to compile the project.
- Once the build completes successfully, click Start to run the application.

Getting started with EdgeLock A30 secure authenticator support package

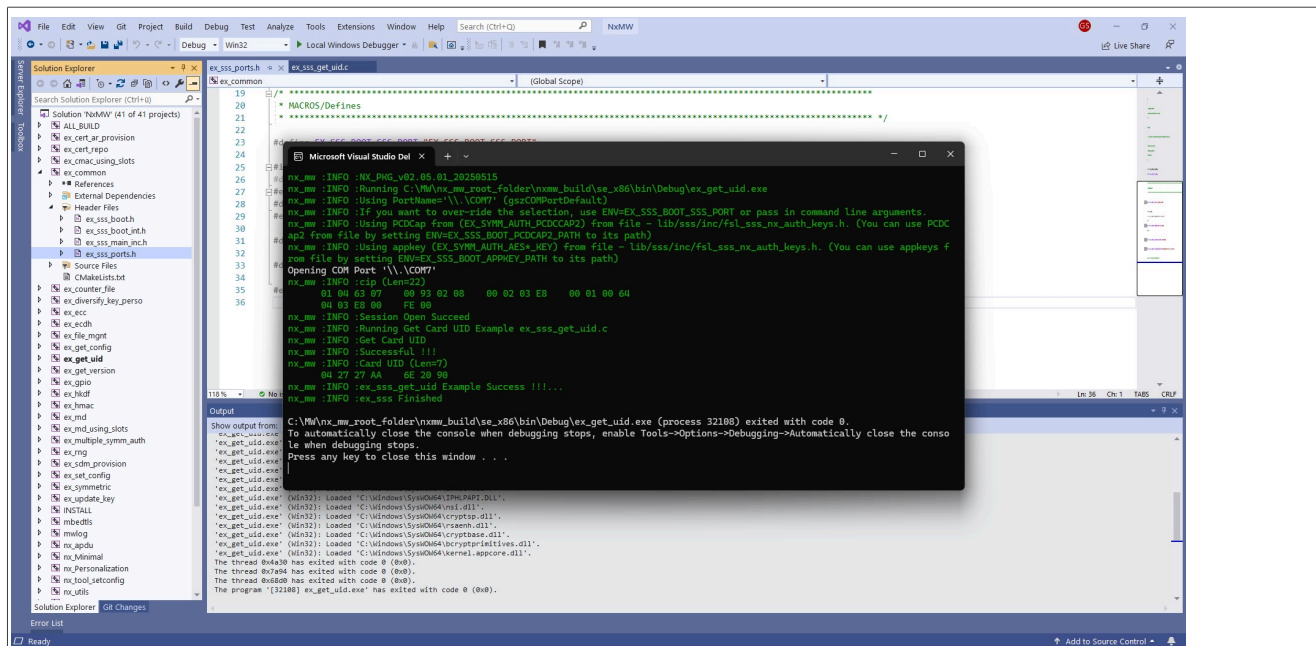


Figure 16. Run `ex get uid project`

4.5 Evaluation using a MCU board

The NX Middleware includes a set of MCU-based examples that demonstrate the use of EdgeLock A30 in the latest authenticator security use cases. The following MCU boards are used as reference platforms:

- [FRDM-MCXA153](#)
- [FRDM-MCXN947](#)
- [LPC55S69-EVK](#)

The NX Middleware stack is designed for easy portability to other MCU or MPU platforms.. For detailed implementation guidance, please refer to the NX Middleware documentation available on [GitHub](#).

MCU project examples can be imported into [MCUXpresso](#) in one of the following formats:

- [Standalone MCUXpresso projects](#)
- [CMake-based projects](#)

These examples provide a quick and practical way to evaluate EdgeLock A30 features. The source code is modular and can be adapted for customer-specific implementations.

To run the examples, connect the reference boards to a Windows PC via USB.

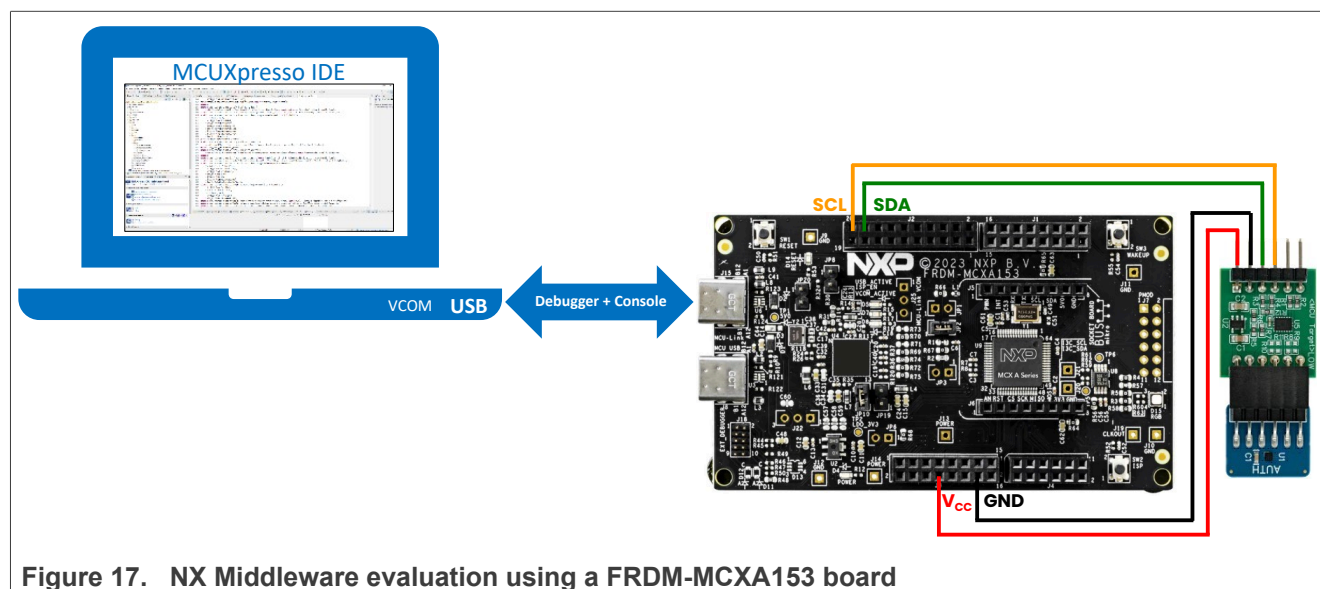


Figure 17. NX Middleware evaluation using a FRDM-MCXA153 board

Getting started with EdgeLock A30 secure authenticator support package

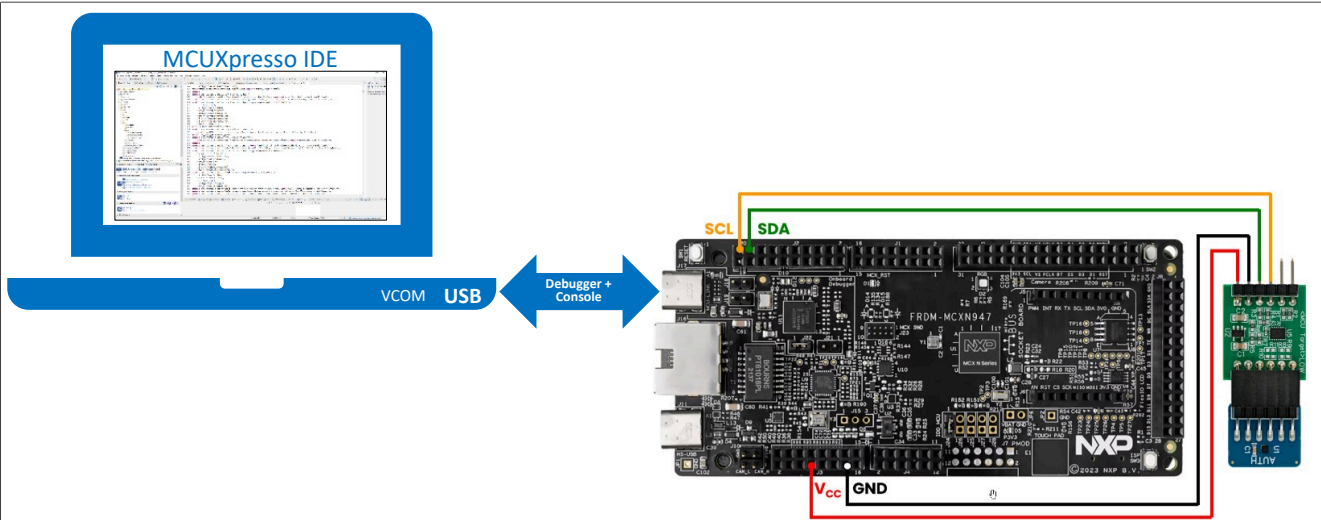


Figure 18. NX Middleware evaluation using a FRDM-MCXA947 board

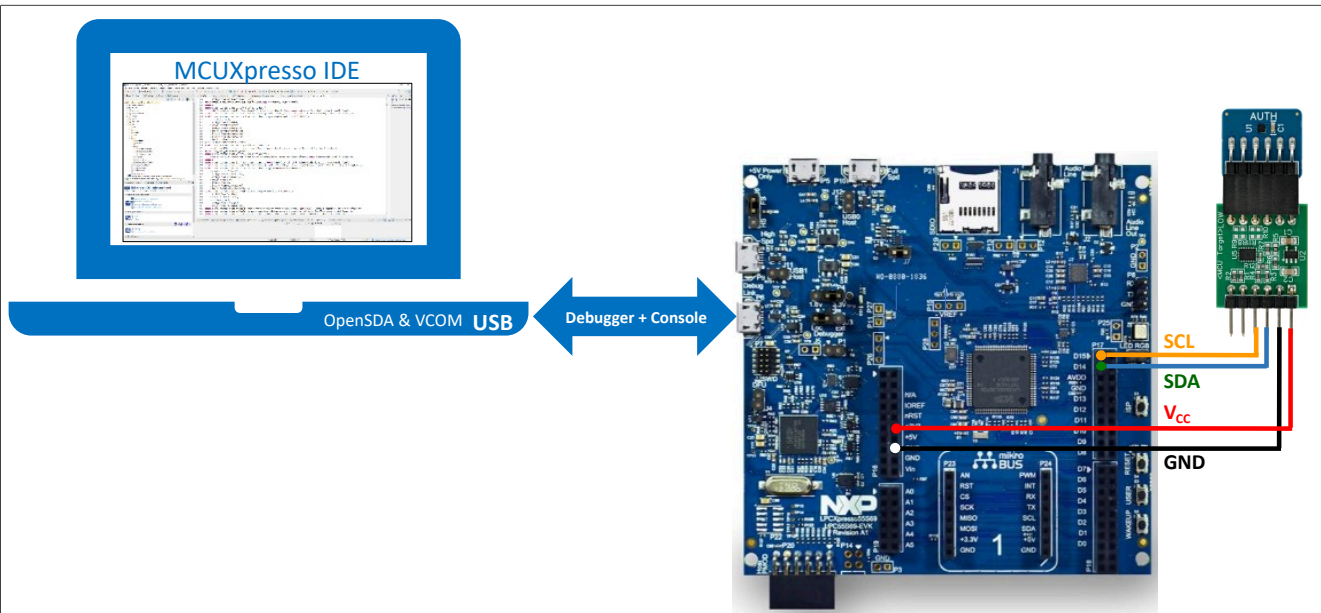


Figure 19. NX Middleware evaluation using a LPC55S69-EVK

4.5.1 Standalone MCUXpresso projects

MCUXpresso standalone demo projects for the supported MCU boards are provided in their respective `MCUX_project` folders on GitHub:

- [FRDM-MCXA153 MCUXpresso projects](#)
- [FRDM-MCXN947 MCUXpresso projects](#)
- [LPC55S69-EVK MCUXpresso projects](#)

To build and run the example projects, refer to the online guide [Getting Started on MCUs Using Standalone MCUXpresso Projects](#), which provides step-by-step instructions for setting up the development environment, importing projects, and executing them on supported hardware.

Note: All project source files are symbolic links to the shared middleware components and are not physical copies. The only exception is the `fsl_sss_ftr.h` file, which is unique to each project and contains board-specific configuration settings. This distinction is important when creating new standalone MCUXpresso projects, as the symbolic linking and configuration must be handled correctly. Instructions for creating new projects are available in the documentation on [GitHub](#).

Note: Projects are linked to specific versions of third-party software. Most demos use mbedTLS v2 by default. To switch to [mbedTLS v3](#), you must:

- Remove mbedTLS v2 from the source tree
- Add mbedTLS v3
- Update the macros in `fsl_sss_ftr.h` accordingly

Note: The FRDM-MCXA153 is a low-power MCU based on the Arm® Cortex®-M33 core, featuring up to 128 KB of flash and 32 KB of RAM. Due to its limited memory resources, adjustments to stack and heap sizes may be necessary depending on the selected authentication method - Symmetric or Asymmetric (Sigma-I). Detailed memory configuration guidelines are available in the project's documentation on [GitHub](#).

4.5.1.1 Feature Configuration File (`fsl_sss_ftr.h`)

Each MCUXpresso project includes a dedicated feature configuration file named `fsl_sss_ftr.h`. This file defines compile-time settings that tailor the middleware stack to the specific MCU board, host environment, communication interface, cryptographic library, and authentication method.

The file enables or disables features using preprocessor macros, ensuring that only the relevant components are included during build time. This modular approach simplifies project customization and supports a wide range of use cases.

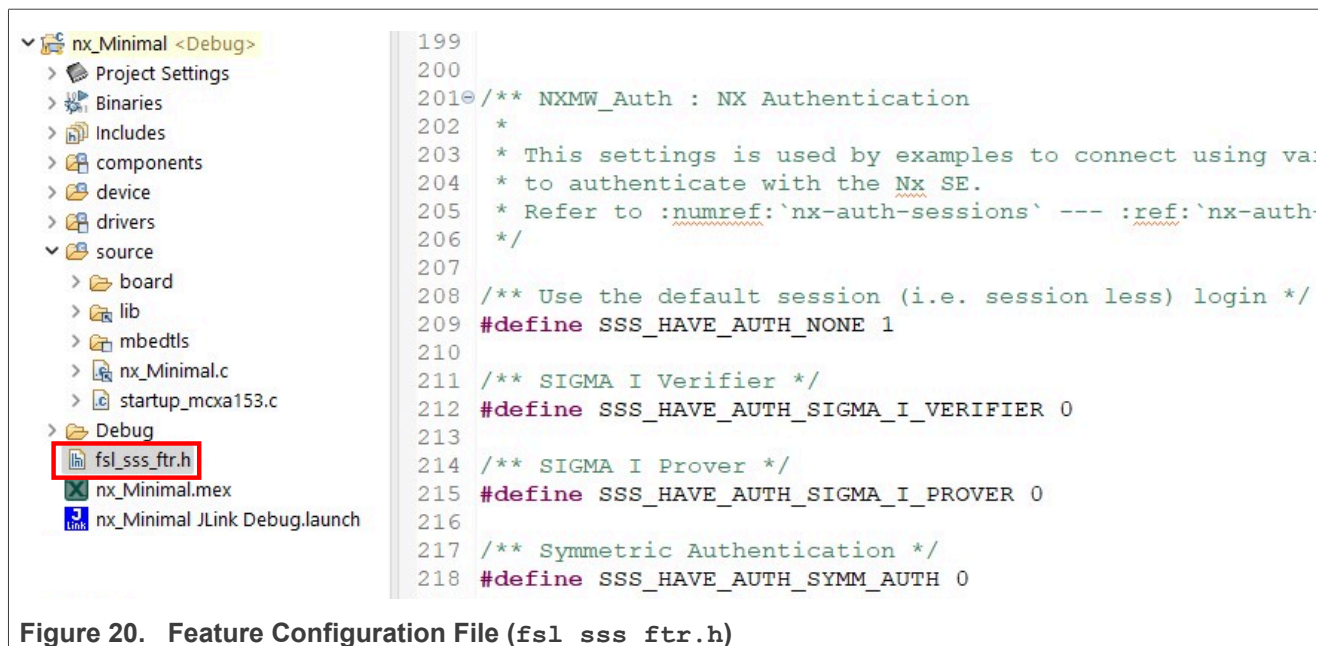


Figure 20. Feature Configuration File (fsl_sss_ftr.h)

4.6 Evaluation using Raspberry Pi

The NX Middleware offers several components and example code to implement and verify EdgeLock A30 on devices running an embedded Linux distribution:

- OpenSSL engine compatible with OpenSSL versions 1.1.1
- OpenSSL provider compatible with OpenSSL versions 3.0
- PKCS#11 Plugin
- SSS API

The Raspberry Pi was selected as a reference MPU platform running embedded Linux. For further details, refer to the following chapters in the GitHub documentation:

- [Raspberry Pi Build](#)
- [Access Manager](#)
- [Introduction on OpenSSL engine](#)
- [Introduction on OpenSSL provider](#)
- [PKCS#11 Plugin](#)
- [OpenSSL Engine: TLS Client example](#)
- [AWS Demo for Raspberry Pi](#)

Note: If several Linux processes want to access EdgeLock A30 at the same time, it is necessary to use the Access Manager. The Access Manager manages the simultaneous access of several Linux processes to EdgeLock A30. Linux client processes are connected to the Access Manager via the JRCPv1 protocol.

Figure 21 shows the principal hardware setup.

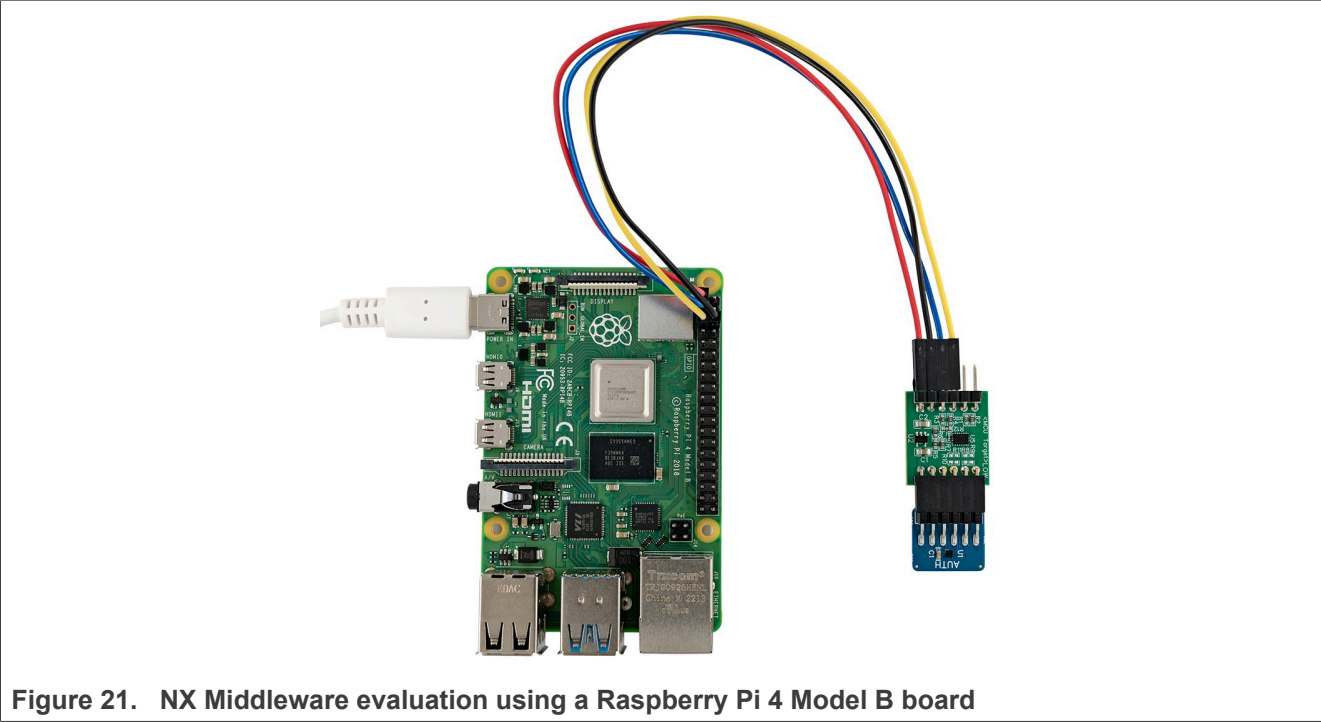


Figure 22 and Table 4 are showing the detailed connection of the EdgeLock A30 to the Raspberry Pi:

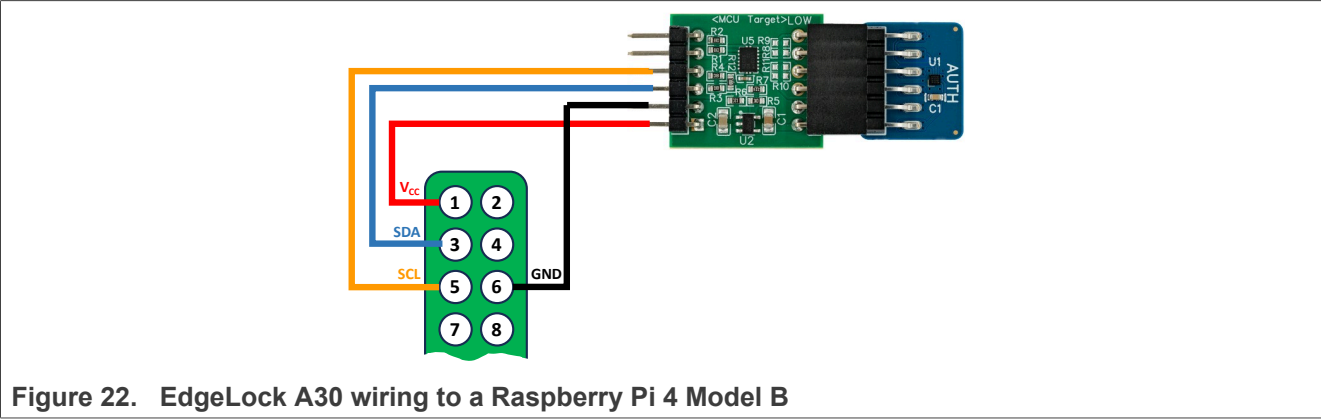


Table 4. EdgeLock A30 wiring to a Raspberry Pi 4 Model B board

Raspberry Pi 4 Model B (# jumper - # pin)	Level Shifter Board (# jumper - # pin)
J8-P5 (SCL)	SCL (HIGH<MCU)
J8-P3 (SDA)	SDA (HIGH<MCU)
J8-P6 (GND)	GND (HIGH<MCU)
J8-P1 (3V3)	VCC (HIGH<MCU)

4.7 Device provisioning

The NX Middleware provides the following tools and examples for provisioning:

- NX Personalization Example
- NX-CLI Tool

4.7.1 NX personalization example

This example project demonstrates how to provision keys and certificates for the EdgeLock A30 Secure Authenticator, which are essential for enabling Sigma-I authentication.

Note: *This demo does not modify the default symmetric keys. For production deployments, it is recommended to either provision symmetric keys or disable symmetric authentication entirely.*

Symmetric key provisioning can be performed as shown in the [ex_update_key](#) and [diversify-key-perso](#) examples. For more information, refer to the online documentation.

4.7.2 NX-CLI Tool

The NX-CLI Tool is a command-line utility for evaluating the EdgeLock A30 Secure Authenticator in Windows and Linux environments. It supports the creation of scripts for provisioning and configuring the EdgeLock A30 device.

Key Features:

- Generate an EC key pair inside the EdgeLock A30 Secure Authenticator
 - The private EC key is stored securely inside the EdgeLock A30 Secure Authenticator
 - The public EC key is exported into a DER formatted file
- Generate `reference keys` and export them in PEM format
- Import private EC keys from PEM files into the EdgeLock A30 Secure Authenticator
- Manage the Sigma-I Certificate Repository (create, activate, reset)
- Insert X.509 certificates in PEM format into the EdgeLock A30 Secure Authenticator
- Configure Sigma-I certificate repository access conditions
- Import Sigma-I host CA root keys from PEM files
- Generate random numbers within the EdgeLock A30 Secure Authenticator
- Create, read and write standard data files
- Retrieve the EdgeLock A30 Secure Authenticator device UID
- Configure the EdgeLock A30 I2C address

Note: *During EC key pair generation, the private EC key is securely stored inside A30, ensuring it remains protected and inaccessible. The corresponding public EC key is returned to the host system. To optimize user NV memory usage, the A30 does not store the public EC key. Instead, the public key is typically embedded in an X.509 certificate. A30 supports storing X.509 certificates inside standard data files.*

Note: *Standard OpenSSL and MbedTLS 3.0 APIs require a private key to perform cryptographic operations. Instead of using an actual private key, a `reference key` structure is provided that points to the private key securely stored inside the Secure Authenticator. This `reference key` is compatible with OpenSSL and ARM mbedTLS and behaves like a real key from the API's perspective, but it does not expose any secret material. This approach ensures secure key handling while maintaining compatibility with standard cryptographic libraries.*

The NX-CLI Tool supports the following commands:

Table 5. NX-CLI Tool commands

Command	Description
connect	Connect to Secure Authenticator
disconnect	Disconnect from Secure Authenticator
get-uid	Retrieve UID from Secure Authenticator
genkey	Generate ECC key in Secure Authenticator and export public key
get-ref-key	Generate a reference key
rand	Generate random numbers from Secure Authenticator
setkey	Import a private key into Secure Authenticator
certrepo-*	Certificate repository management commands
create-bin	Create a standard data file in Secure Authenticator.
setbin	Write data to a standard data file in Secure Authenticator
getbin	Read data from a standard data file in Secure Authenticator
list-fileid	List file IDs in Secure Authenticator
list-eckey	List EC keys and their properties in Secure Authenticator
set-i2c_mgnt	Configure I2C settings
set-cert_mgnt	Configure certificate settings

For more details, refer to the NX-CLI Tool [GitHub](#) documentation.

The following example demonstrates how to retrieve the Secure Authenticator's device UID using the NX-CLI Tool in a Windows environment.

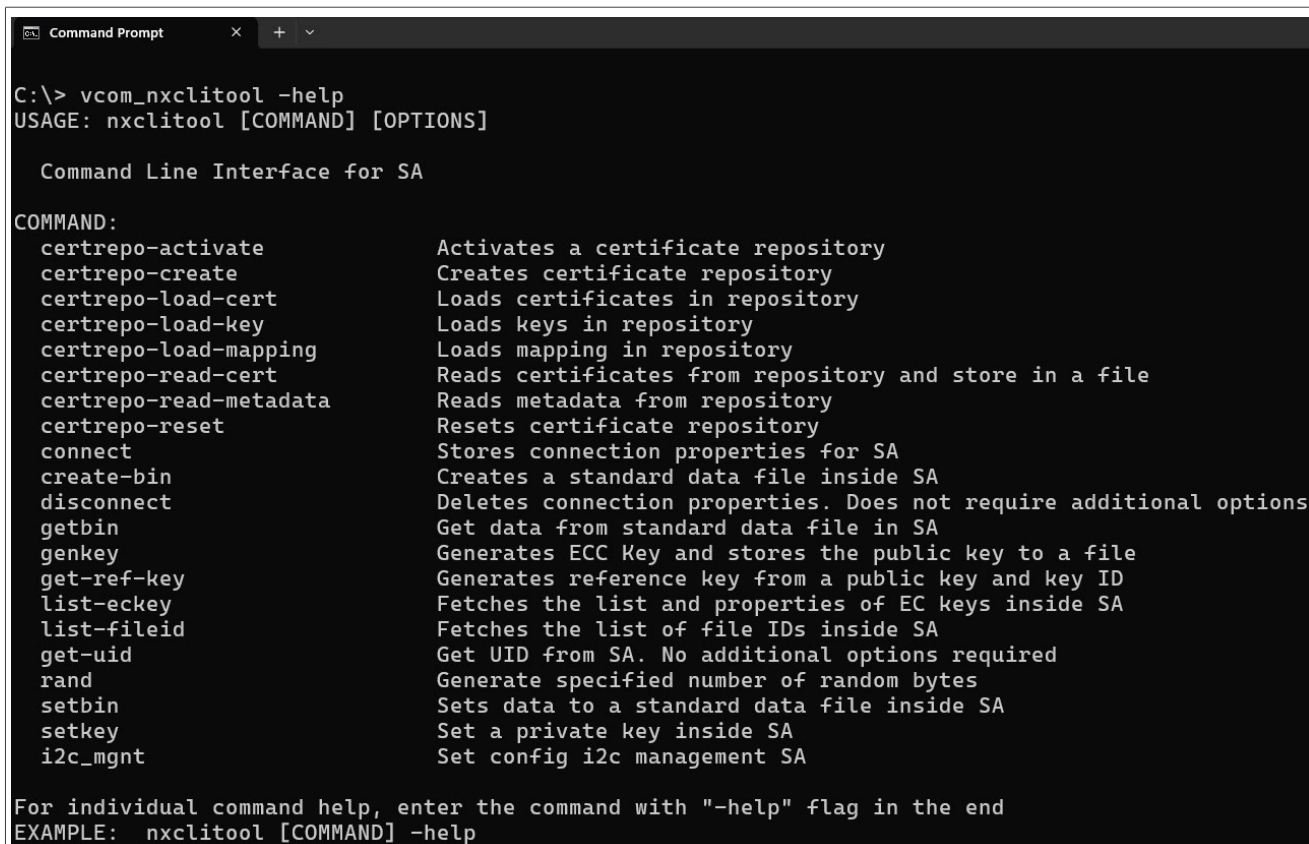
Note: This example assumes that the VCOM to T=1 over I²C firmware is running on a supported NXP MCU development board. For more information on supported boards and setup, refer to [Section 4.4](#). Pre-compiled binaries for both the VCOM to T=1 over I²C firmware and the Windows version of the NX-CLI Tool are available on [GitHub](#):

- Link to the pre-compiled VCOM to T=1 over I²C firmware binary:
 - [nx_vcom-T1oI2C-mcxa153.bin](#)
 - [nx_vcom-T1oI2C-mcxn947.bin](#)
 - [nx_vcom-T1oI2C-lpc55s69.bin](#)
- Link to the pre-compiled Windows NX-CLI-Tool binary: [vcom_nxclitool.exe](#)

Note: It is recommended to add the location of the `vcom_nxclitool.exe` binary to the Windows environment variable `Path`.

Use the following command to display the `vcom_nxclitool` built in help:

```
vcom_nxclitool help
```



```
C:\> vcom_nxclitool -help
USAGE: nxclitool [COMMAND] [OPTIONS]

Command Line Interface for SA

COMMAND:
certrepo-activate      Activates a certificate repository
certrepo-create        Creates certificate repository
certrepo-load-cert     Loads certificates in repository
certrepo-load-key      Loads keys in repository
certrepo-load-mapping  Loads mapping in repository
certrepo-read-cert     Reads certificates from repository and store in a file
certrepo-read-metadata Reads metadata from repository
certrepo-reset         Resets certificate repository
connect               Stores connection properties for SA
create-bin            Creates a standard data file inside SA
disconnect            Deletes connection properties. Does not require additional options
getbin               Get data from standard data file in SA
genkey               Generates ECC Key and stores the public key to a file
get-ref-key          Generates reference key from a public key and key ID
list-eckey           Fetches the list and properties of EC keys inside SA
list-fileid          Fetches the list of file IDs inside SA
get-uid              Get UID from SA. No additional options required
rand                 Generate specified number of random bytes
setbin               Sets data to a standard data file inside SA
setkey               Set a private key inside SA
i2c_mngt             Set config i2c management SA

For individual command help, enter the command with "-help" flag in the end
EXAMPLE: nxclitool [COMMAND] -help
```

Figure 23. NX-CLI-Tool - help

To establish a connection, use the following command:

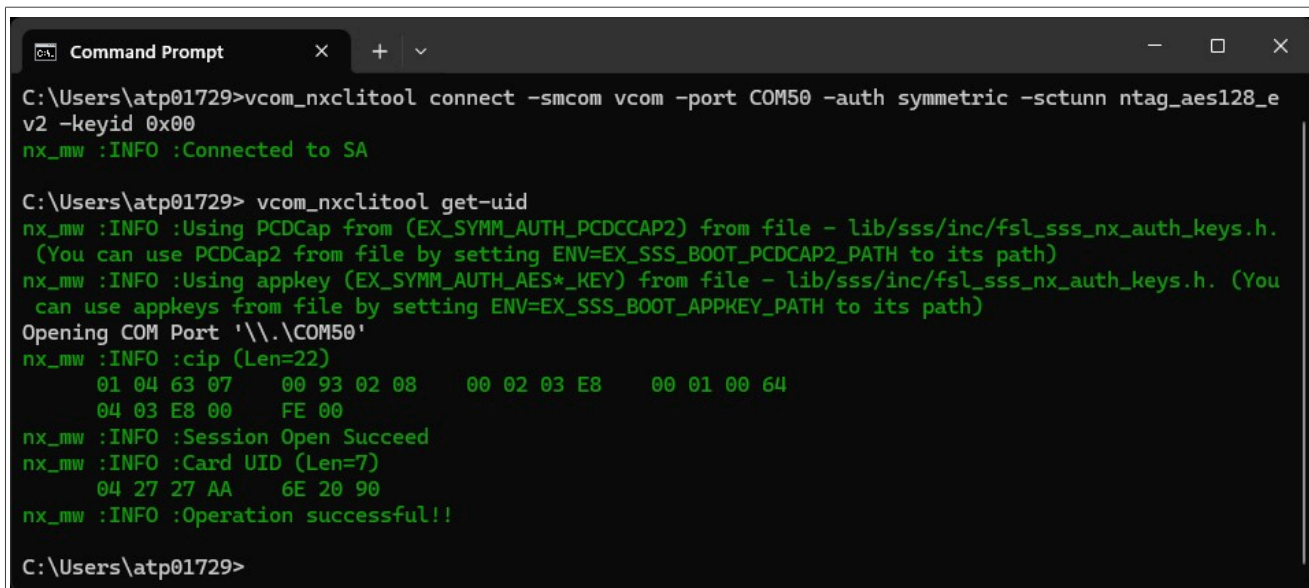
```
vcom_nxclitool connect -smcom vcom -port COMxx -auth symmetric ^
                    -sctunn ntag_aes128_ev2 -keyid 0x00
```

This command specifies:

- `-smcom vcom`: Host communication via VCOM
- `-port COMxx`: VCOM port number (replace COMxx with the actual port)
- `-auth symmetric`: Symmetric authentication method
- `-sctunn ntag_aes128_ev2`: Secure channel/tunnel type
- `-keyid 0x00`: Authentication key number/slot

Once connected, retrieve the UID with:

```
vcom_nxclitool get-uid
```



```

C:\Users\atp01729>vcom_nxclitool connect -smcom vcom -port COM50 -auth symmetric -sctunn ntag_aes128_e
v2 -keyid 0x00
nx_mw :INFO :Connected to SA

C:\Users\atp01729> vcom_nxclitool get-uid
nx_mw :INFO :Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h.
(You can use PCDCap2 from file by setting ENV=EX_SSS_BOOT_PCDCCAP2_PATH to its path)
nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h. (You
can use appkeys from file by setting ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
Opening COM Port '\\.\COM50'
nx_mw :INFO :cip (Len=22)
01 04 63 07 00 93 02 08 00 02 03 E8 00 01 00 64
04 03 E8 00 FE 00
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :Card UID (Len=7)
04 27 27 AA 6E 20 90
nx_mw :INFO :Operation successful!!

C:\Users\atp01729>

```

Figure 24. NX-CLI-Tool - get-uid command

4.8 Device configuration

The NX Middleware provides command-line examples to set and retrieve the chip configuration. The configuration options are described in the [DS9767xx](#) EdgeLock A30 Secure Authenticator data sheet chapter 6.5.2 Card Configuration.

4.8.1 Get configuration

The NX Middleware example get configuration retrieves the chip configuration and print the information to the console. This example can be executed on all supported environments (Windows, Linux and MCUs).

The following example demonstrates how to get the configuration in a Windows environment. Instructions on how to build the example are provided in [Section 4.4.1](#).

Note: This example assumes that the VCOM to T=1 over I²C firmware is running on a supported NXP MCU development board. For more information on supported boards and setup, refer to [Section 4.4](#). Pre-compiled binaries for both the VCOM to T=1 over I²C firmware is available on [GitHub](#).

To invoke the get configuration example (see also [Section 4.4.2](#)), use the following command:

```
ex_get_config.exe COMxx
```

This command specifies:

- `-port COMxx`: VCOM port number (replace COMxx with the actual port)

An EdgeLock A30 chip fresh from the factory returns the following configuration.

Getting started with EdgeLock A30 secure authenticator support package

```

C:\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug>ex_get_config.exe COM50
nx_mw :INFO :NX_PKG_v02.05.01_20250515
nx_mw :INFO :Running ex_get_config.exe
nx_mw :INFO :Using PortName='COM50' (CLI)
nx_mw :INFO :Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/
sss/inc/fsl_sss_nx_auth_keys.h. (You can use PCDCap2 from file by setting
ENV=EX_SSS_BOOT_PCDCAP2_PATH to its path)
nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES* KEY) from file - lib/sss/
inc/fsl_sss_nx_auth_keys.h. (You can use appkeys from file by setting
ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
Opening COM Port '\\.\COM50'
nx_mw :INFO :cip (Len=22)
01 04 63 07 00 93 02 08 00 02 03 E8 00 01 00 64
04 03 E8 00 FE 00
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :Running Get Configuration Example ex_sss_set_config.c
nx_mw :INFO :=====
nx_mw :INFO :Get Manufacture Features successful !!!
nx_mw :INFO : Support ECC-based Unilateral Authentication: Enabled
nx_mw :INFO : Support Import of ECC Private Key: Enabled
nx_mw :INFO : Enable Enforce User Data Files 1k Limit: Disabled
nx_mw :INFO : Counter Support: Enabled
nx_mw :INFO : EC DSA 4.0: Disabled
nx_mw :INFO : CCM AES-256 Secure Channel: Disabled
nx_mw :INFO : NTAG AES-256 Secure Channel: Enabled
nx_mw :INFO : NTAG AES-128 Secure Channel: Enabled
nx_mw :INFO : External Crypto API ECC Support: Enabled
nx_mw :INFO : External Crypto API AES Support: Enabled
nx_mw :INFO : GPIO Support: Enabled
nx_mw :INFO : I2C IO Support: Enabled
nx_mw :INFO : NFC IO Support: Enabled
nx_mw :INFO :=====
nx_mw :INFO :Get PICC Configurations successful !!!
nx_mw :INFO : User RID (Random ISO ID): Disabled
nx_mw :INFO :=====
nx_mw :INFO :Get ATS Update Configurations successful !!!
nx_mw :INFO : User defined ATS: (Len=6)
06 78 77 71 02 80
nx_mw :INFO :=====
nx_mw :INFO :Get SAK Update Configurations successful !!!
nx_mw :INFO : User defined User defined SAK1: 0x4
nx_mw :INFO : User defined User defined SAK2: 0x20
nx_mw :INFO :=====
nx_mw :INFO :Get Secure messaging configuration successful !!!
nx_mw :INFO : Chained writing: Enabled
nx_mw :INFO :=====
nx_mw :INFO :Get Capability Data successful !!!
nx_mw :INFO : User configured PDCap2.5: 0x0
nx_mw :INFO : User configured PDCap2.6: 0x0
nx_mw :INFO :=====
nx_mw :INFO :Get ATQA Update successful !!!
nx_mw :INFO : User defined ATQA: (Len=2)
44 03
nx_mw :INFO :=====
nx_mw :INFO :Get Silent Mode Configuration successful !!!
nx_mw :INFO : Customized REQS/WUPS: Disabled
nx_mw :INFO : Silent mode: Disabled
nx_mw :INFO :=====
nx_mw :INFO :Get Enhanced Privacy Configuration successful !!!

```

Getting started with EdgeLock A30 secure authenticator support package

```
nx_mw :INFO : Originality Check: Enabled
nx_mw :INFO : Manufacturer data masking: Disabled
nx_mw :INFO : KeyID.AppPrivacyKey: Disabled
nx_mw :INFO : KeyID.AppPrivacyKey definition: 0x0
nx_mw :INFO :=====
nx_mw :INFO :Get NFC Management Configuration successful !!!
nx_mw :INFO : NFC I/O: Disabled
nx_mw :INFO : SIGMA-I Verifier: Enabled
nx_mw :INFO : SIGMA-I Prover: Enabled
nx_mw :INFO : NTAG EV2 secure messaging: Enabled
nx_mw :INFO : Secure Tunnel strength: AES-256 supported
nx_mw :INFO : Secure Tunnel strength: AES-128 supported
nx_mw :INFO : ECC-based Card-Unilateral Authentication: Enabled
nx_mw :INFO : AES-based Symmetric Authentication: Enabled
nx_mw :INFO :=====
nx_mw :INFO :Get I2C Management Configuration successful !!!
nx_mw :INFO : I2C I/O: Enabled
nx_mw :INFO : SIGMA-I Verifier: Enabled
nx_mw :INFO : SIGMA-I Prover: Enabled
nx_mw :INFO : NTAG EV2 secure messaging: Enabled
nx_mw :INFO : Secure Tunnel strength: AES-256 supported
nx_mw :INFO : Secure Tunnel strength: AES-128 supported
nx_mw :INFO : ECC-based Card-Unilateral Authentication: Enabled
nx_mw :INFO : AES-based Symmetric Authentication: Enabled
nx_mw :INFO :=====
nx_mw :INFO :Get GPIO Management Configuration successful !!!
nx_mw :INFO : GPIO 1 Mode: Disabled
nx_mw :INFO : GPIO 1 supply selection: 1V1 and 1V2 signaling in I2c mode
nx_mw :INFO : GPIO 2 Mode: Disabled
nx_mw :INFO : GPIO 2 supply selection: 1V1 and 1V2 signaling in I2c mode
nx_mw :INFO : Cmd.ManageGPIO communication modes: No protection.
nx_mw :INFO : Cmd.ManageGPIO access condition 0xf
nx_mw :INFO : Cmd.ReadGPIO communication modes: No protection.
nx_mw :INFO : Cmd.ReadGPIO access condition 0xf
nx_mw :INFO :=====
nx_mw :INFO :Get ECC Key Management Configuration successful !!!
nx_mw :INFO : Cmd.ManageKeyPair communication modes: Full protection.
nx_mw :INFO : Cmd.ManageKeyPair access condition 0x0
nx_mw :INFO : Cmd.ManageCARootKey communication modes: Full protection.
nx_mw :INFO : Cmd.ManageCARootKey access condition 0x30
nx_mw :INFO :=====
nx_mw :INFO :Get Certificate Management Configuration successful !!!
nx_mw :INFO : End Leaf certificate cache size: 0x5
nx_mw :INFO : Intermediate certificate cache size: 0x4
nx_mw :INFO : SIGMA-I Cache: Disabled
nx_mw :INFO : Cmd.ManageCertRepo communication modes: MAC protection.
nx_mw :INFO : Cmd.ManageCertRepo access condition 0x0
nx_mw :INFO :=====
nx_mw :INFO :Get Watchdog Timer Management Configuration successful !!!
nx_mw :INFO : Halt Watchdog Timer (HWDT) Value: 0x0
nx_mw :INFO : Authorization Watch-Dog Timer (AWDT1) Value: 0x0
nx_mw :INFO : Authorization Watch-Dog Timer (AWDT2) Value: 0x0
nx_mw :INFO :=====
nx_mw :INFO :Get Crypto API Management Configuration successful !!!
nx_mw :INFO : Asymmetric Crypto API: Enabled
nx_mw :INFO : Symmetric Crypto API: Enabled
nx_mw :INFO : Cmd.CryptoRequest communication modes: MAC protection.
nx_mw :INFO : Cmd.CryptoRequest access condition 0x0
nx_mw :INFO : Cmd.CryptoRequest access condition for ChangeKey command targeting
CryptoRequest Keys 0x0
```



```

nx_mw :INFO :Crypto API TB Policy: (Len=24)
      00 00 00 01      00 00 02 00      00 03 00 00      04 00 00 05
      00 00 06 00      00 07 00 00
nx_mw :INFO :Crypto API SB Policy: (Len=42)
      00 00 00 01      00 00 02 00      00 03 00 00      04 00 00 05
      00 00 06 00      00 07 00 00      08 00 00 09      00 00 0A 00
      00 0B 00 00      0C 00 00 0D      00 00
nx_mw :INFO :=====
nx_mw :INFO :Get Lock Configuration successful !!!
nx_mw :INFO : Lock bitmap: 0
nx_mw :INFO :ex_sss_get_config Example Success !!!...
nx_mw :INFO :ex_sss Finished

```

Note: Information related to an ISO/IEC14443 contactless interface is not relevant for the EdgeLock A30 and can be ignored:

- ATQA
- ATS
- Capability Data
- NFC IO Support
- NFC Management Configuration
- SAK
- Silent Mode Configuration

4.8.2 Set configuration

The `nx_tool_setconfig` can be used to configure the EdgeLock A30 Secure Authenticator in a Windows or Linux environment.

Supported Configuration Options:

- `gpio1mode`: Set GPIO1 to disabled, input, output, input tag tamper or down-stream power out
- `gpio2mode`: Set GPIO2 to disabled, input or output
- `gpio1Notif`: Set GPIO1 notification on authentication. It can be disabled, enabled for authentication or enabled for presence of NFC field
- `gpio2Notif`: Set GPIO2 notification on authentication. It can be disabled, enabled for authentication or enabled for presence of NFC field
- `gpioMgmtCM`: Set ManageGPIO communication mode
- `gpioReadCM`: Set ReadGPIO communication mode
- `gpioMgmtAC`: Set ManageGPIO access condition
- `gpioReadAC`: Set ReadGPIO access condition
- `cryptoCM`: Set Crypto API communication mode
- `cryptoAC`: Set Crypto API access condition
- `keypairCM`: Set ManageKeyPair communication mode
- `keypairAC`: Set ManageKeyPair access condition
- `caRootKeyCM`: Set ManageCARootKey communication mode
- `caRootKeyAC`: Set ManageCARootKey access condition

Please refer to the documentation on [GitHub](#) for detailed descriptions and usage guidelines.

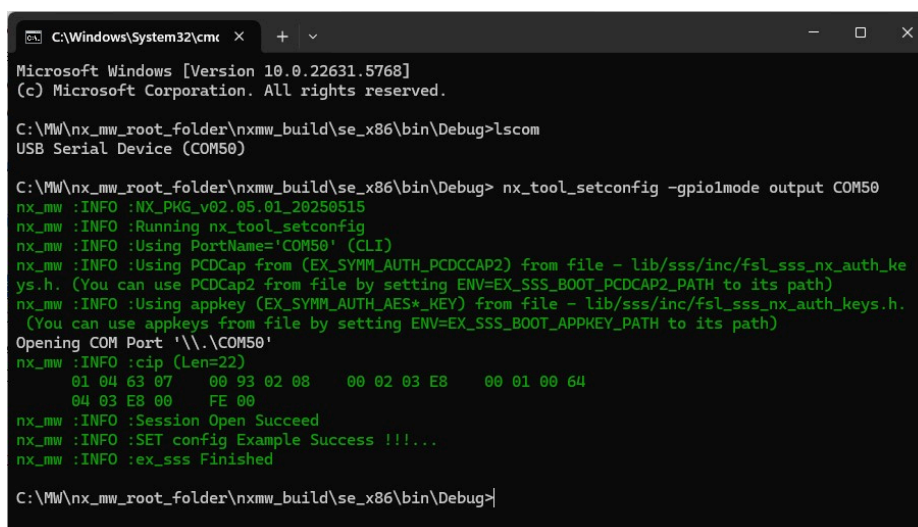
The following example illustrates how to configure GPIO1 to operate in output mode using a Windows environment.

Getting started with EdgeLock A30 secure authenticator support package

```
nx_tool_setconfig -gpiolmode output COMxx
```

This command specifies:

- `-port COMxx`: VCOM port number (replace COMxx with the actual port)



```
C:\Windows\System32\cmd
Microsoft Windows [Version 10.0.22631.5768]
(c) Microsoft Corporation. All rights reserved.

C:\MW\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug>lscom
USB Serial Device (COM50)

C:\MW\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug> nx_tool_setconfig -gpiolmode output COM50
nx_mw :INFO :NX_PKG_v02.05.01_20250515
nx_mw :INFO :Running nx_tool_setconfig
nx_mw :INFO :Using PortName='COM50' (CLI)
nx_mw :INFO :Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h. (You can use PCDCap2 from file by setting ENV=EX_SSS_BOOT_PCDCCAP2_PATH to its path)
nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES*.KEY) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h. (You can use appkeys from file by setting ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
Opening COM Port '\\.\COM50'
nx_mw :INFO :cip (Len=22)
          01 04 63 07 00 93 02 08 00 02 03 E8 00 01 00 64
          04 03 E8 00 FE 00
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :SET config Example Success !!!...
nx_mw :INFO :ex_sss Finished

C:\MW\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug>
```

Figure 25. Example `nx_tool_setconfig`

Note: The [Set Configuration Example](#) is an alternative method that demonstrates how to perform configuration operations using the NX APIs. This example can also be executed on MCUs.

5 EdgeLock 2Go service for EdgeLock A30

5.1 Overview

EdgeLock A30 products are delivered with an NXP trust-provisioned device unique application private EC key and application X.509 certificate containing the corresponding public EC key. The EdgeLock A30 application credentials can be used for Sigma-I Authentication or ECDSA signature generation.

To simplify the OEMs products process and reduce the production cost the EdgeLock A30 device UID and the application X.509 certificate can be downloaded via the EdgeLock 2Go service. This eliminates the need for OEMs to read the credential from each individual EdgeLock A30 device.

EdgeLock A30 reels are shipped with a label containing an access code (reference code and authorization key).



Figure 26. EdgeLock A30 reel

To download the EdgeLock A30 UIDs and application certificates can be simple done by the following steps:

- Go to <https://www.edgelock2go.com/downloads> and enter the reference code and authorization key.
- After successful registration, Device UIDs and application X.509 certificates become available for download. If wafer delivery is selected, the corresponding Wafer Map can also be retrieved.

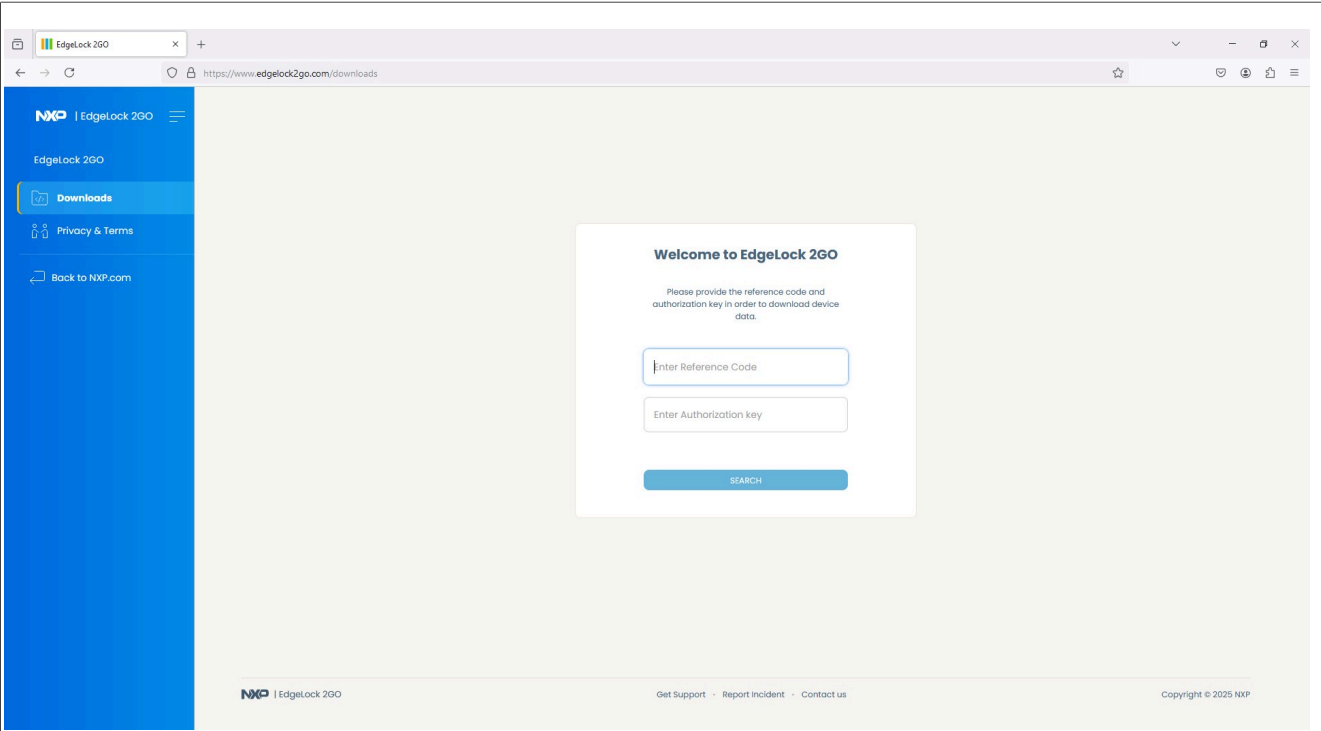


Figure 27. EdgeLock2Go web portal

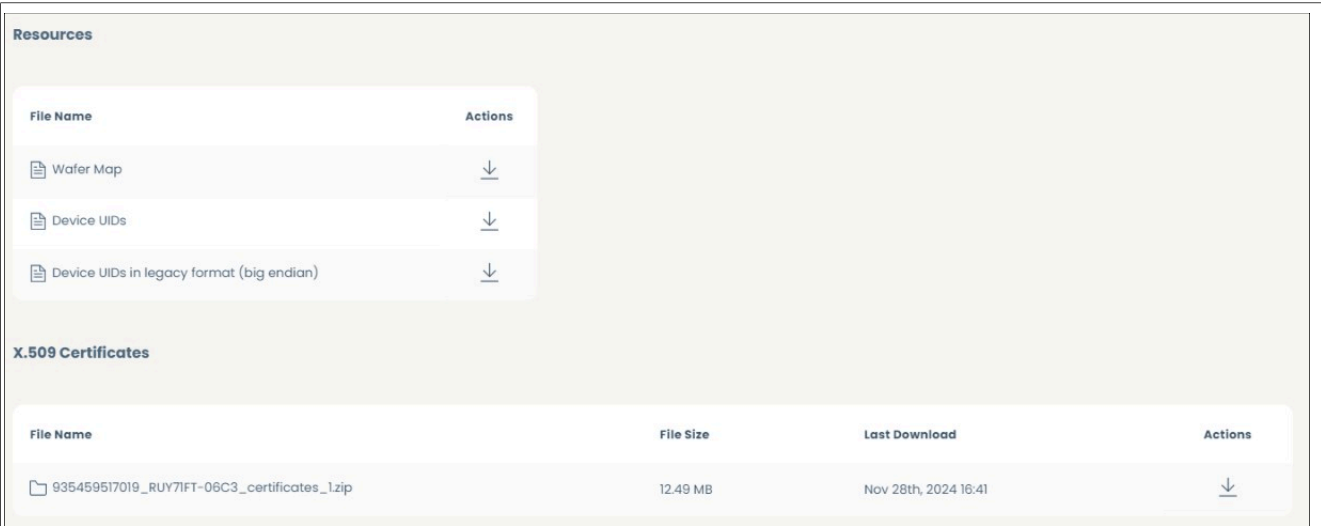


Figure 28. UID and certificate download via EdgeLock2Go

6 Supported EdgeLock A30 documentation

[Table 6](#) summarizes the EdgeLock A30 dedicated documents.

Note: Click on the hyperlink to download the document.

Table 6. Dedicated EdgeLock A30 documentation

Documentation number	Title
DS9767xx	EdgeLock A30 Secure Authenticator data sheet
AD9772xx	Edgelock A30 Delivery specification
AN14559	Migration Guide from EdgeLock A5000 to EdgeLock A30
AN14123	NX Middleware Documentation
Global Platform	GlobalPlatform Technology- APDU Transport over SPI / I ² C Version 1.0, January 2020
UM10204	I ² C-bus specification and user manual

7 Appendix

7.1 EdgeLock A30 product identification

The EdgeLock A30 product identification can be retrieved by sending the APDU GetVersion command. The GetVersion response contains the major/minor version identifiers of the IC hardware and software required for EdgeLock A30 product identification. A detailed description of the GetVersion response can be found in the [DS9767xx](#) EdgeLock A30 Secure Authenticator data sheet. The exact EdgeLock A30 product identification is covered by the following two parameters:

- Hardware Major/Minor Version value: 0xA0 0x00
- Software Major/Minor Version value: 0x00 0x01

The NX Middleware includes an example for running GetVersion on all supported platforms (Windows, Linux and MCUs). Please refer to [Section 4.4.1 Building NX Middleware on Windows](#) and [Section 4.4.2 Running NX Middleware examples from command line](#) for a detailed description how compile and run an example on a Windows environment.

To invoke the GetVersion example, use the following command:

```
ex_get_version COMxx
```

This command specifies:

- -port COMxx: VCOM port number (replace COMxx with the actual port)

```
C:\MW\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug>ex_get_version COM54
nx_mw :INFO :NX_PKG_v02.05.01_20250515
nx_mw :INFO :Running ex_get_version
nx_mw :INFO :Using PortName='COM54' (CLI)
nx_mw :INFO :Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h. (You can use PCDCap2 from
file by setting ENV=EX_SSS_BOOT_PCDCCAP2_PATH to its path)
nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h. (You can use appkeys from file
by setting ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
Opening COM Port '\\.\COM54'
nx_mw :INFO :cip (Len=22)
01 04 63 07 00 93 02 08 00 02 03 E8 00 01 00 64
04 03 E8 00 FE 00
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :Running Get Card Version Example ex_sss_get_version.c
nx_mw :INFO :Get Card Version
nx_mw :INFO :Successful !!!
nx_mw :INFO :HW Vendor ID: 0x04 (NXP Semiconductors)
nx_mw :INFO :HW type: 0x0A (IoT)
nx_mw :INFO :HW subtype: 0x41 (17 pF, Tag Tamper)
nx_mw :INFO :HW major version: 0xA0
nx_mw :INFO :HW minor version: 0x00
nx_mw :INFO :HW storage size: 0x1C (16 kB)
nx_mw :INFO :HW protocol type: 0x35 (I2C and ISO/IEC 14443-4 support with Silent Mode support)
nx_mw :INFO :SW Vendor ID: 0x04 (NXP Semiconductors)
nx_mw :INFO :SW type: 0x0A (A30)
nx_mw :INFO :SW subtype: 0x01 (Standalone)
nx_mw :INFO :SW major version: 0x00
nx_mw :INFO :SW minor version: 0x01
nx_mw :INFO :SW storage size: 0x1C (16 kB)
nx_mw :INFO :SW protocol type: 0x35 (I2C and ISO/IEC 14443-4 support with Silent Mode support)
nx_mw :INFO :Card UID (Len=7)
04 27 41 AA 6E 20 90
nx_mw :INFO :BatchNo: 0x672321
nx_mw :INFO :FabKey identifier: 0x5234
nx_mw :INFO :Calendar week of card production in BCD coding: 0x09
nx_mw :INFO :The year of production in BCD coding: 0x25
nx_mw :INFO :Fab Identifier: 0x32
nx_mw :INFO :ex_sss_get_version Example Success !!!...
nx_mw :INFO :ex_sss Finished
```

Figure 29. EdgeLock A30 product identification using GetVersion

7.2 Available free memory of EdgeLock A30 ICs

EdgeLock A30 ICs are trust-provisioned during manufacturing.

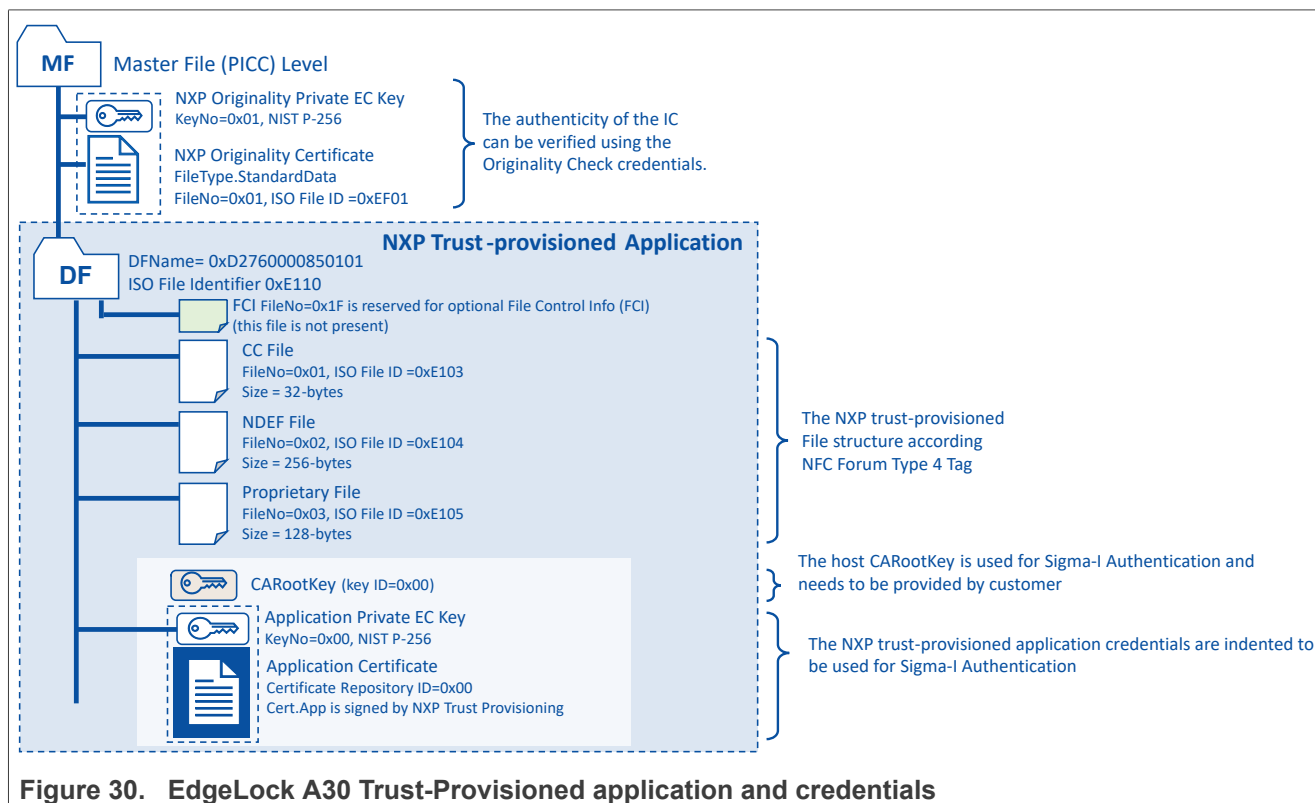


Figure 30. EdgeLock A30 Trust-Provisioned application and credentials

The default trust-provisioning includes the following components:

- ECC-Based Key Pair and Certificate at the Master File Level
 - Each device is provisioned with an ECC key pair and a corresponding certificate, enabling verification of the IC's authenticity. This is achieved through a card-unilateral authentication mechanism using a challenge-response protocol. To mitigate privacy concerns—since the protocol may leave a non-repudiable trace—the same key pair is shared across all ICs within a production batch. Additionally, this feature can be disabled if required. The NX Middleware provides the [Originality Check](#) example to verify this feature.
 - The NXP Originality Certificate is signed by NXP Trust Provisioning using a dedicated CA key pair for this product. The corresponding Root Certificate can be retrieved from <https://www.gp-ca.nxp.com/CA/getCA?caid=63709320110003>
- Pre-Configured Application with NFC Forum Type 4 Tag File Structure
 - The device includes a pre-installed application structured according to the NFC Forum Type 4 Tag specification.
 - An Application Key Pair and its associated certificate, which support EC-based Mutual Authentication using the Sigma-I protocol.
 - The option to modify the default KeyPolicy, allowing the Application Key Pair to be used for other supported elliptic curve operations.
 - The ability to replace the provisioned Application Certificate, if needed.
 - The NXP Application Certificate is signed by NXP Trust Provisioning using a dedicated CA key pair. The corresponding Root Certificate can be retrieved from <https://www.gp-ca.nxp.com/CA/getCA?caid=63709320101003>

Note: NXP provides commercial customization options for trust-provisioning. This allows for a customer-dedicated delivery configuration. This may include the provisioning of a customer-specific CARootKey. This allows to do the initial personalization with the ECC-based SIGMA-I authentication. By this, the need for a secure environment can be removed, compared to when doing the initial personalization based on the default AES keys. Additionally, also customer-specific AES keys, certificates and/or a customized file system and configuration can be provisioned. Reach out to your local sales representative for more information.

Note: For more details, please refer to EdgeLock A30 Secure Authenticator data sheet [DS9767xx](#).

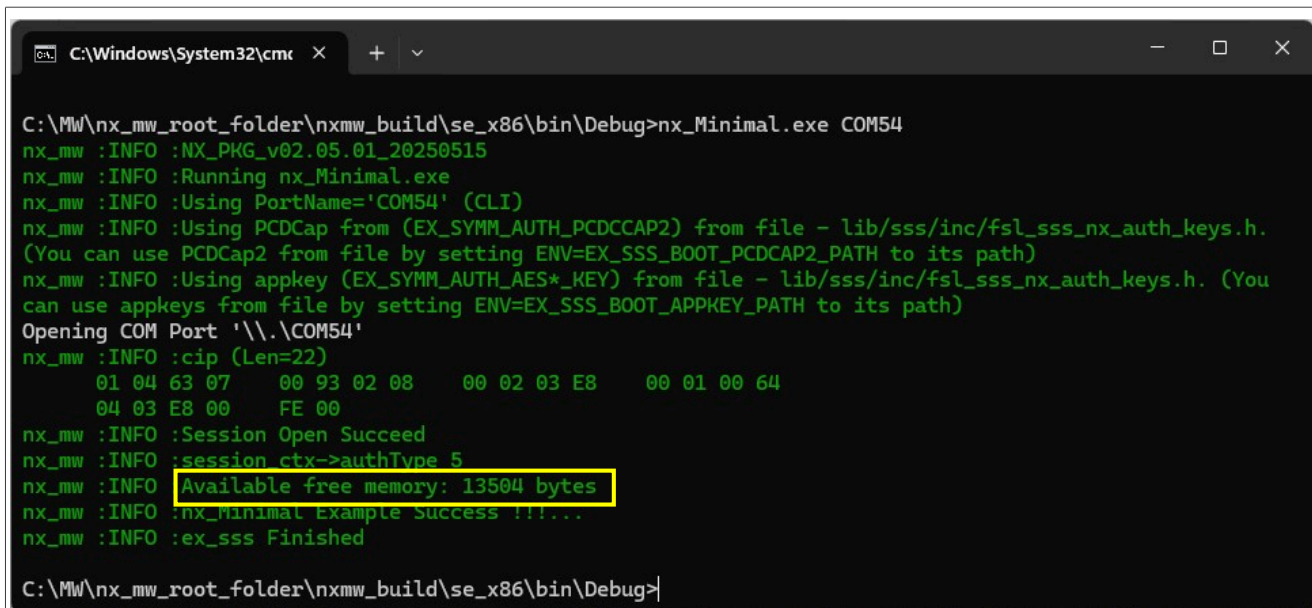
The trust-provisioned application and credentials consumes **2,880 bytes** of the total **16,384 bytes** of user memory available on the IC.

Thus, the **available free memory** for user-defined data and applications is: **16,384 bytes – 2,880 bytes = 13,504 bytes**

Table 7. Trust-Provisioned device - memory consumption

Component	Memory size	Remark
File info headers for 3 NDEF files	64 B	NDEF File System
Additional file info header for Secure Dynamic Messaging (SMD)	32 B	NDEF File System
CC File (FID 0xE103)	32 B	NDEF File System
NDEF File (FID 0xE104)	256 B	NDEF File System
Proprietary File (FID 0xE105)	128 B	NDEF File System
Host CA Root Key (Reserved Memory)	128 B	Required for Sigma-I Authentication
Application Private EC Key	96 B	Required for Sigma-I Authentication
Application Certificate (Certificate Repository Size)	2144 B	Required for Sigma-I Authentication

The available free memory of a device can be verified using the example [NX Minimal](#). Please refer to [Section 4.4.1 Building NX Middleware on Windows](#) and [Section 4.4.2 Running NX Middleware examples from command line](#) for a detailed description how compile and run an example on a Windows environment.



```
C:\Windows\System32\cmd.exe
C:\MW\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug>nx_Minimal.exe COM54
nx_mw :INFO :NX_PKG_v02.05.01_20250515
nx_mw :INFO :Running nx_Minimal.exe
nx_mw :INFO :Using PortName='COM54' (CLI)
nx_mw :INFO :Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h.
(You can use PCDCap2 from file by setting ENV=EX_SSS_BOOT_PCDCCAP2_PATH to its path)
nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/sss/inc/fsl_sss_nx_auth_keys.h. (You
can use appkeys from file by setting ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
Opening COM Port '\\.\COM54'
nx_mw :INFO :cip (Len=22)
01 04 63 07 00 93 02 08 00 02 03 E8 00 01 00 64
04 03 E8 00 FE 00
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :session ctx->authType 5
nx_mw :INFO :Available free memory: 13504 bytes
nx_mw :INFO :nx_Minimal Example Success !!!!!
nx_mw :INFO :ex_sss Finished

C:\MW\nx_mw_root_folder\nxmw_build\se_x86\bin\Debug>
```

Figure 31. Checking the available free memory using the nx_Minimal Example

7.3 EdgeLock A30 application circuit diagram

Figure 32 shows an application circuit diagram with the following design considerations:

- Power-On-Reset
 - It is recommended that the hostcontroller can perform a Power-On-Reset by controlling V_{CC} .
 - It is possible to supply A30 via the MCU/MPU GPIO. The GPIO shall be able to deliver current up to 15 mA.
- A30 triggers a Reset via $T=1'$ over I^2C protocol
 - Using a proprietary NXP S-Blocks chip reset request/response.
- A30 enables Deep Power Down via $T=1'$ over I^2C protocol
 - Using a proprietary NXP S-Block Deep Power Down request/response.

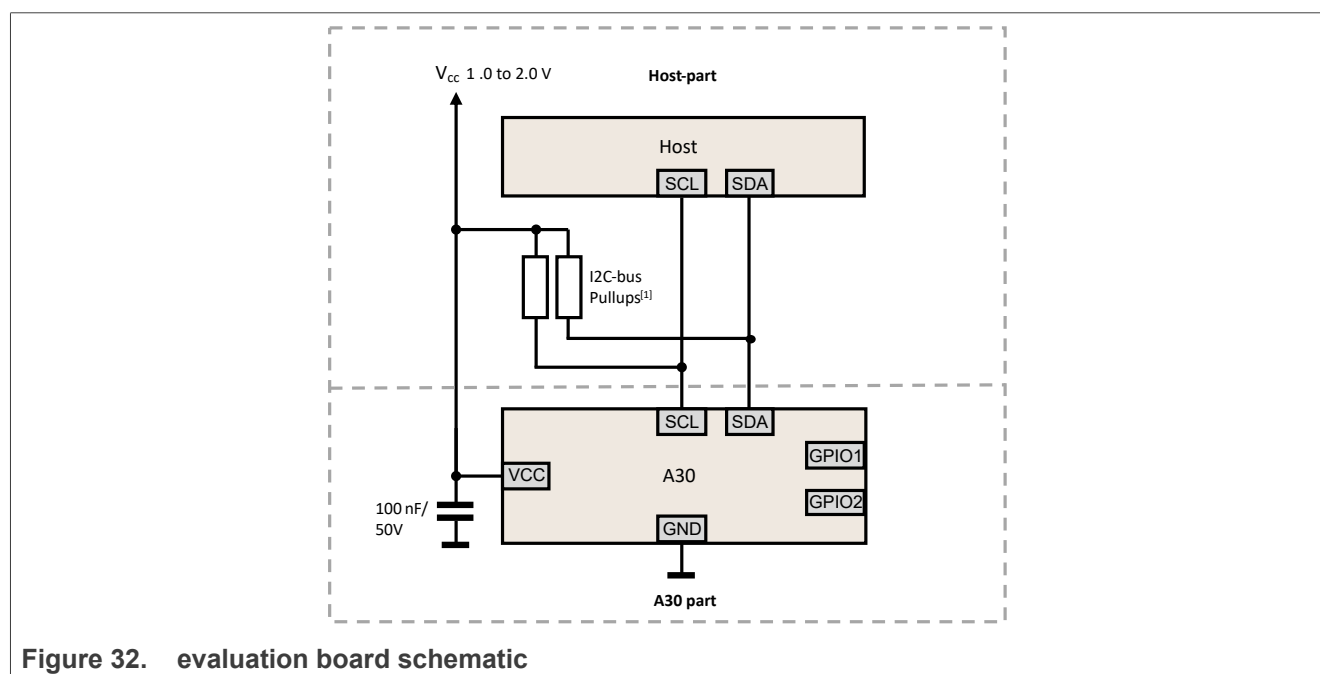


Figure 32. evaluation board schematic

Note: [1] .. According [UM10204](#) I^2C -bus specification and user manual Rev 7, chapter 7.1 Pull-up resistor sizing

7.4 EdgeLock A30 WLCSP16 pin description

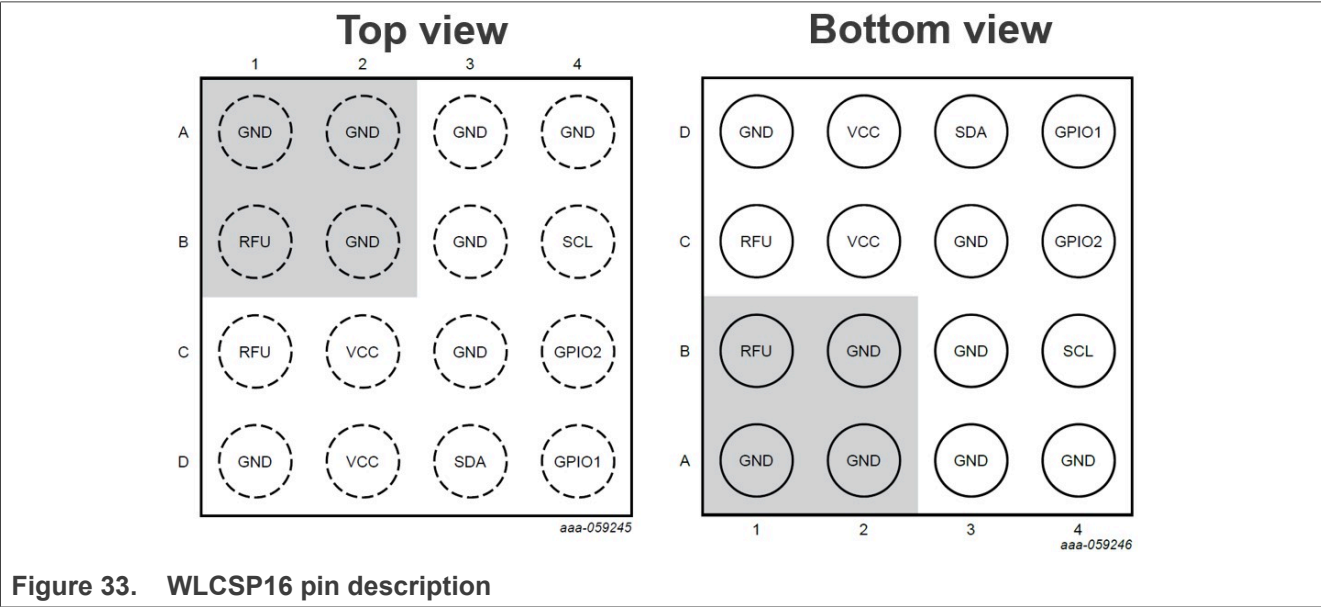


Table 8. WLCSP16 pin description

Pin	Symbol	Description
A1	GND	Ground
A2	GND	Ground
A3	GND	Ground
A4	GND	Ground
B1	r.f.u.	Connect to Ground
B2	GND	Ground
B3	GND	Ground
B4	SCL	SCL - I2C Target Clock Input
C1	r.f.u.	Connect to Ground
C2	VCC	Logic and I2C/GPIO power supply voltage input
C3	GND	Ground
C4	GPIO2	General Purpose IO
D1	GND	Ground
D2	VCC	Logic and I2C/GPIO power supply voltage input
D3	SDA	SDA - I2C Target Data
D4	GPIO1	General Purpose IO

Note: A 100nF decoupling capacitor must be located as close as possible between pin D1 and D2.

7.5 EdgeLock A30 HVQFN20 pin description

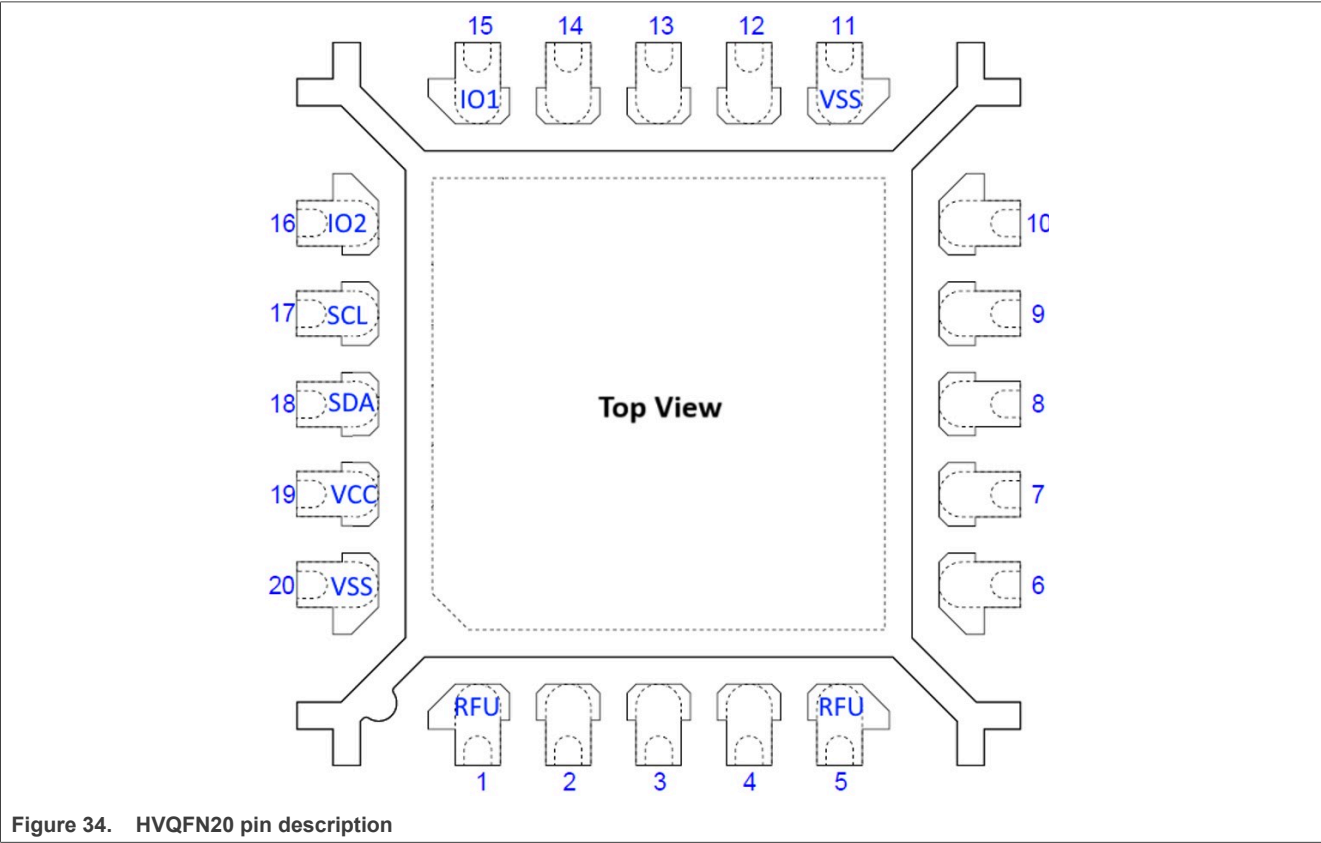


Figure 34. HVQFN20 pin description

Table 9. HVQFN20 pin description

Pin	Symbol	Description
1	r.f.u.	Connect to V _{SS}
5	r.f.u.	Connect to V _{SS}
11	VSS	Ground
15	GPIO1	General Purpose IO
16	GPIO2	General Purpose IO
17	SCL	SCL - I2C Target Clock Input
18	SDA	SDA - I2C Target Data
19	VCC	Logic and I2C/GPIO power supply voltage input
20	VSS	Ground

Note: All other pins are not connected. Not connected pins can either be left open or grounded in the application.

Note: The HVQFN20 package exposed center pad can be connected to GND.

Note: A 100nF decoupling capacitor must be located as close as possible between pin 19 and 20.

7.6 EdgeLock A30 development board schematic

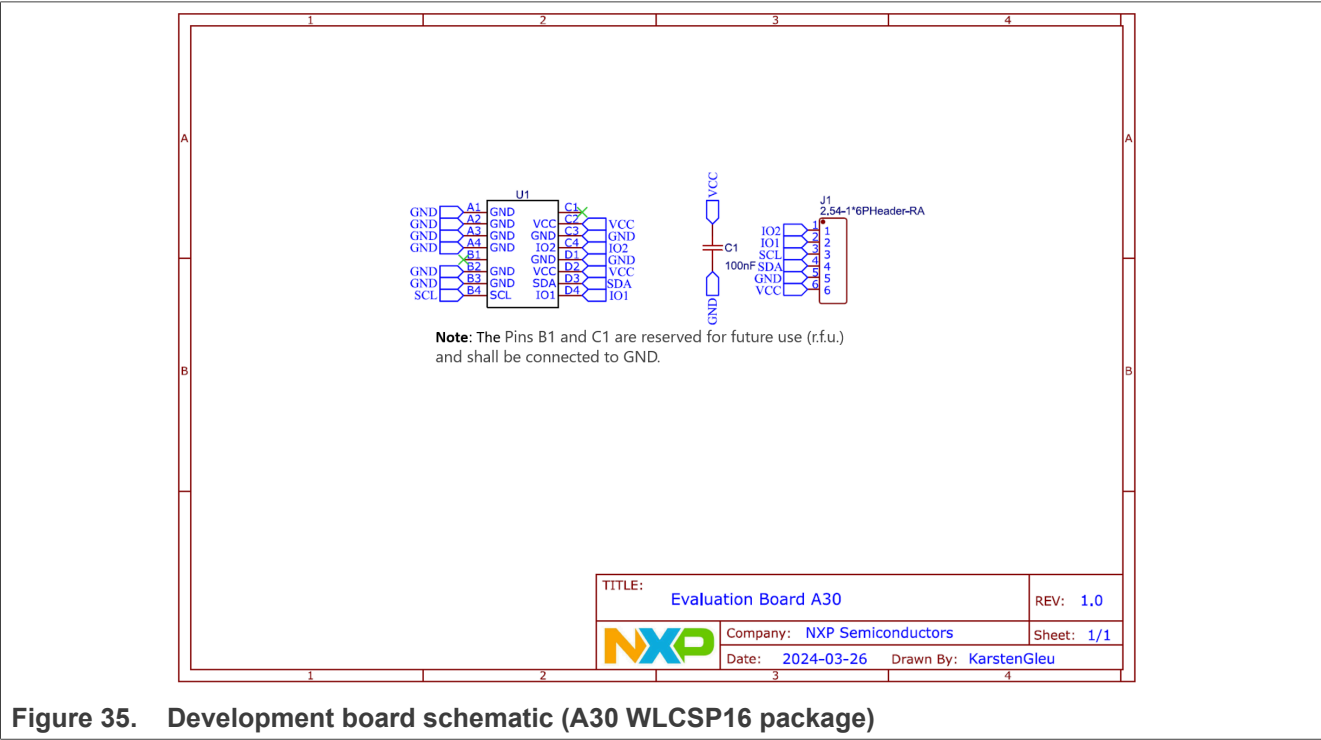


Figure 35. Development board schematic (A30 WLCSP16 package)

7.7 Level shifter board schematic

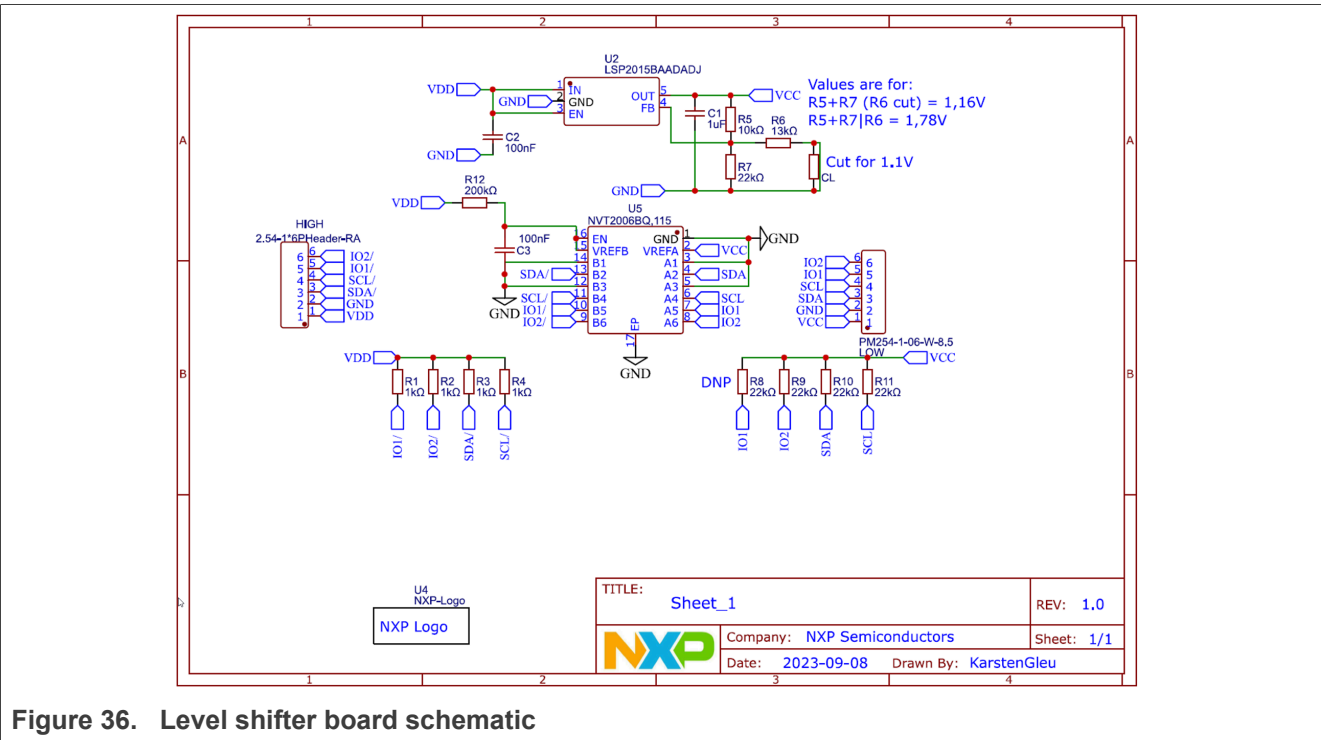


Figure 36. Level shifter board schematic

8 Abbreviations

Table 10. Abbreviations

Acronym	Description
AID	Application IDentifier
APDU	Application Protocol Data Unit
CA	Certificate Authority
C-APDU	Command APDU
CMAC	MAC according to NIST Special Publication 800-38B
CRC	Cyclic Redundancy Check
DF-Name	ISO7816 Dedicated filename
ECC	Elipctic Curve Cryptography
GUI	Graphical User Interface
IC	Integrated Circuit
IDE	Integrated Development Environment
MCU	Microcontroller Unit
MPU	Microprocessor Unit
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
r.f.u.	Reserved for Future Use
R-APDU	Response APDU
SCL	Serial Clock
SDA	Serial Data
T=1' over I ² C	GlobalPlatform Technology- APDU Transport over SPI / I²C Version 1.0, January 2020
SCL	Serial Data
UID	Unique Identifier
VCC	Voltage Common Collector
VCOM	VCOM
VSS	Voltage Source Supply (Ground)

9 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10 Revision history

Table 11. Revision history

Document ID	Release date	Description
AN14238 v.2.0	2 September 2025	<p>Editorial changes (typos, etc.). Document title changed to "Getting started with EdgeLock A30 secure authenticator support package".</p> <ul style="list-style-type: none">• Section 1 "About EdgeLock A30 secure authenticator": updated.• Section 2 "EdgeLock A30 development kit": updated.• Section 3 "MCU/MPU reference boards": updated.• Section 4 "NX Middleware": updated.• Section 4.2 "Architecture overview": updated.• Section 4.3 "Code documentation": updated.• Section 4.4 "Evaluation using a Windows PC": updated.• Section 4.4.1 "Building NX Middleware on Windows": added.• Section 4.4.2 "Running NX Middleware examples from command line": added.• Section 4.4.3 "Running NX Middleware example projects in Visual Studio": added.• Section 4.5 "Evaluation using a MCU board": updated.• Section 4.5.1 "Standalone MCUXpresso projects": added.• Section 4.6 "Evaluation using Raspberry Pi": updated.• Section 4.7 "Device provisioning": added.• Section 4.8 "Device configuration": added.• Section 5 "EdgeLock 2Go service for EdgeLock A30": updated.• Section 6 "Supported EdgeLock A30 documentation": updated.• Section 7.1 "EdgeLock A30 product identification": added.• Section 7.2 "Available free memory of EdgeLock A30 ICs": added.• Section 7.4 "EdgeLock A30 WLCSP16 pin description": added.• Section 7.5 "EdgeLock A30 HVQFN20 pin description": added.• Section 8 "Abbreviations": added.• Section 9 "Note about the source code in the document": added.
AN14238 v.1.0	15 January 2025	Initial version.

Legal information

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.

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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <https://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that 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 vulnerabilities or may support established security standards or specifications with known limitations. 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.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Tab. 1.	EdgeLock A30 development kit A30-EVAL	4	Tab. 7.	Trust-Provisioned device - memory consumption	40
Tab. 2.	MCU reference boards	6	Tab. 8.	WLCSP16 pin description	43
Tab. 3.	MPU reference boards	7	Tab. 9.	HVQFN20 pin description	44
Tab. 4.	EdgeLock A30 wiring to a Raspberry Pi 4 Model B board	26	Tab. 10.	Abbreviations	46
Tab. 5.	NX-CLI Tool commands	28	Tab. 11.	Revision history	48
Tab. 6.	Dedicated EdgeLock A30 documentation	37			

Figures

Fig. 1.	EdgeLock A30 support package overview	2	Fig. 19.	NX Middleware evaluation using a LPC55S69-EVK	23
Fig. 2.	Connecting A30 development board with level shifter board	5	Fig. 20.	Feature Configuration File (fsl_sss_ftr.h)	25
Fig. 3.	NX Middleware block diagram	8	Fig. 21.	NX Middleware evaluation using a Raspberry Pi 4 Model B board	26
Fig. 4.	NX Middleware architecture	9	Fig. 22.	EdgeLock A30 wiring to a Raspberry Pi 4 Model B	26
Fig. 5.	NX Middleware PDF documentation	10	Fig. 23.	NX-CLI-Tool - help	29
Fig. 6.	NX Middleware PDF documentation	11	Fig. 24.	NX-CLI-Tool - get-uid command	30
Fig. 7.	NX Middleware evaluation using FRDM- MCXA153 and a Windows	12	Fig. 25.	Example nx_tool_setconfig	34
Fig. 8.	NX Middleware evaluation using FRDM- MCXN947 and a Windows PC	12	Fig. 26.	EdgeLock A30 reel	35
Fig. 9.	NX Middleware evaluation using LPC55S69-EVK and a Windows PC	13	Fig. 27.	EdgeLock2Go web portal	36
Fig. 10.	Default NX Middleware default CMake settings PC	15	Fig. 28.	UID and certificate download via EdgeLock2Go	36
Fig. 11.	Windows Device Manger – Identify the correct VCOM port PC	17	Fig. 29.	EdgeLock A30 product identification using GetVersion	38
Fig. 12.	Executing the nx_ecc.exe example PC	17	Fig. 30.	EdgeLock A30 Trust-Provisioned application and credentials	39
Fig. 13.	NxMW.sln Visual Studio project workspace	18	Fig. 31.	Checking the available free memory using the nx_Minimal Example	41
Fig. 14.	Change VCOM port number	19	Fig. 32.	evaluation board schematic	42
Fig. 15.	Set ex_get_uid as Visual Studio startup project	20	Fig. 33.	WLCSP16 pin description	43
Fig. 16.	Run ex_get_uid project	21	Fig. 34.	HVQFN20 pin description	44
Fig. 17.	NX Middleware evaluation using a FRDM- MCXA153 board	22	Fig. 35.	Development board schematic (A30 WLCSP16 package)	45
Fig. 18.	NX Middleware evaluation using a FRDM- MCXA947 board	23	Fig. 36.	Level shifter board schematic	45

Contents

1	About EdgeLock A30 secure authenticator	2	9	Note about the source code in the document	47
2	EdgeLock A30 development kit	4	10	Revision history	48
3	MCU/MPU reference boards	6		Legal information	49
4	NX Middleware	8			
4.1	Overview	8			
4.2	Architecture overview	9			
4.3	Code documentation	10			
4.4	Evaluation using a Windows PC	10			
4.4.1	Building NX Middleware on Windows	13			
4.4.1.1	Prerequisites	13			
4.4.1.2	Download or Clone the NX Middleware	13			
4.4.1.3	Set Up the NX Middleware Development Environment	14			
4.4.1.4	Open CMake GUI to explore the default CMake settings	15			
4.4.1.5	Build NX Middleware using the command line	16			
4.4.2	Running NX Middleware examples from command line	16			
4.4.3	Running NX Middleware example projects in Visual Studio	18			
4.4.3.1	Opening the Project in Visual Studio	18			
4.4.3.2	Configuring the VCOM Port	18			
4.4.3.3	Set an Example Project as Startup	19			
4.4.3.4	Build and Run the Project	20			
4.5	Evaluation using a MCU board	22			
4.5.1	Standalone MCUXpresso projects	24			
4.5.1.1	Feature Configuration File (fsl_sss_ftr.h)	24			
4.6	Evaluation using Raspberry Pi	25			
4.7	Device provisioning	27			
4.7.1	NX personalization example	27			
4.7.2	NX-CLI Tool	27			
4.8	Device configuration	30			
4.8.1	Get configuration	30			
4.8.2	Set configuration	33			
5	EdgeLock 2Go service for EdgeLock A30	35			
5.1	Overview	35			
6	Supported EdgeLock A30 documentation	37			
7	Appendix	38			
7.1	EdgeLock A30 product identification	38			
7.2	Available free memory of EdgeLock A30 ICs	39			
7.3	EdgeLock A30 application circuit diagram	42			
7.4	EdgeLock A30 WLCSP16 pin description	43			
7.5	EdgeLock A30 HVQFN20 pin description	44			
7.6	EdgeLock A30 development board schematic	45			
7.7	Level shifter board schematic	45			
8	Abbreviations	46			

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