Getting started with the X-CUBE-STSE01 software package

Introduction

This user manual describes how to get started with the X-CUBE-STSE01 software package.

The X-CUBE-STSE01 software package is a software component that provides several demonstration codes, which use the STSAFE-A110, STSAFE-A120, and STSAFE-L010 device features from a host microcontroller.

These demonstration codes utilize the secured element middleware (STSELib) built on the STM32Cube software technology to ease portability across different STM32 microcontrollers. In addition, it is MCU-agnostic for portability to other MCUs.

These demonstration codes illustrate the following features:

- Authentication.
- Secured data storage.
- Secured usage counter.
- Pairing.
- Key establishment.
- Local envelope wrapping.
- Key pair generation.

1 General information

The X-CUBE-STSE01 software package is a reference to integrate the STSAFE-A110, STSAFE-A120 and STSAFE-L010 secure element services into a host MCU's operating system (OS) and its application.

It contains the STSAFE-A110, STSAFE-A120 and STSAFE-L010 driver and demonstration codes to be executed on STM32 32-bit microcontrollers based on the Arm® Cortex®-M processor.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

The X-CUBE-STSE01 software package is developed in ANSI C. Nevertheless, the platform-independent architecture allows easy portability to a variety of different platforms.

The table below presents the definition of acronyms that are relevant for a better understanding of this document.

2 STSAFE secure element

The STSAFE-A110, STSAFE-A120, and STSAFE-L010 are highly secure solution that acts as a secure element providing authentication and data management services to a local or remote host. It consists of a full turnkey solution with a secure operating system running on the latest generation of secure microcontrollers.

The STSAFE-A110, STSAFE-A120, and STSAFE-L010 can be integrated in Internet of Things (IoT) devices, smart-home, smart-city and industrial applications, consumer electronics devices, consumables, and accessories. Its key features are:

- Authentication (of peripherals, IoT and USB Type-C® devices).
- Secure channel establishment with remote host including transport layer security (TLS) handshake.
- Signature verification service (secure boot and firmware upgrade).
- Usage monitoring with secure counters.
- Pairing and secure channel with host application processor.
- Wrapping and unwrapping of local or remote host envelopes.
- On-chip key pair generation.

3 STSecureElement Library (STSELib) description

This section details the STSELib middleware software package content and the way to use it.

3.1 General description

The STSELib middleware is a set of software components designed to:

- Interface the STSAFE-A110, STSAFE-A120, and STSAFE-L010 secure element device with an MCU
- Implement the most generic STSAFE-A110, STSAFE-A120, and STSAFE-L010 use cases

The STSELib middleware is fully integrated within ST software packages as a middleware component to add secure element features.

The STSELib middleware provides a complete set of high-level application programming interface functions to the embedded system developer. This middleware abstracts the build and the sequencing of the commands required to ensure device, accessories, and consumable brand protection using STMicroelectronics STSAFE-A secure element family.

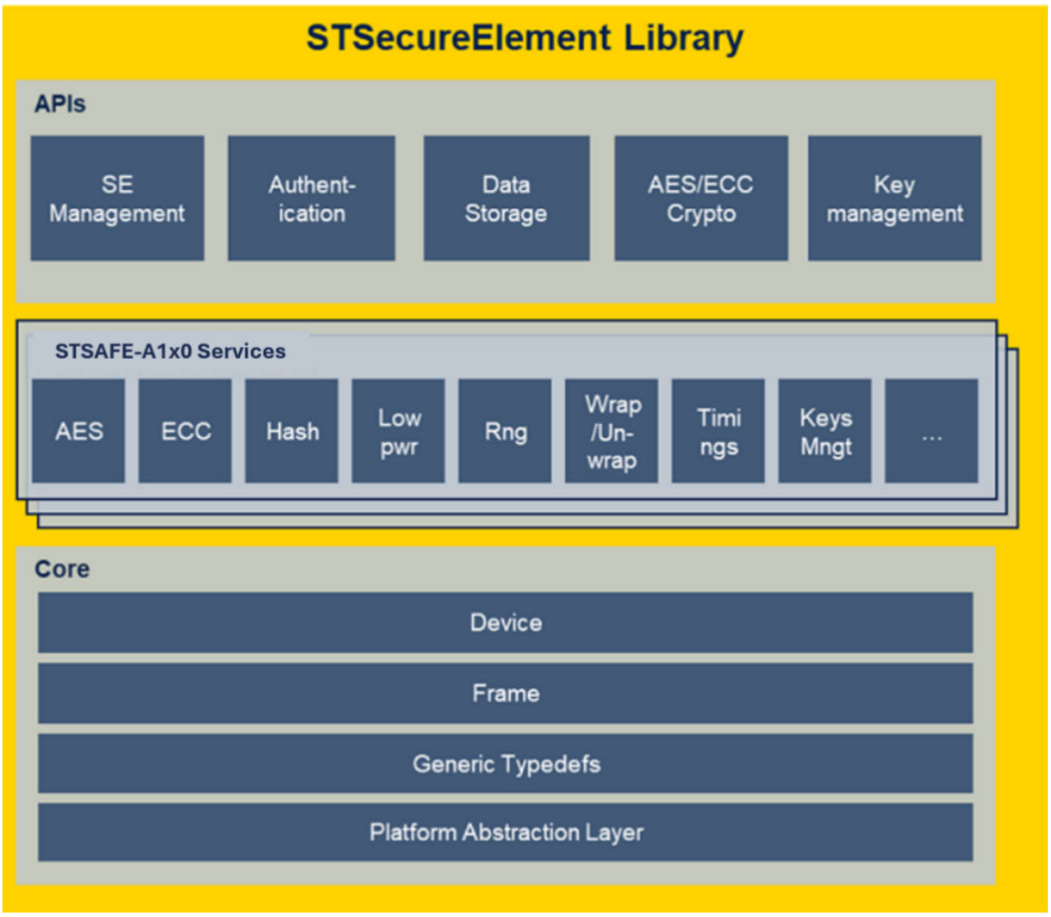
This middleware allows a seamless integration of one or multiple STSAFE-A in various host MCU/MPU ecosystems.

Refer to the release notes available in the package root folder for information about the supported IDE versions.

3.2 Architecture

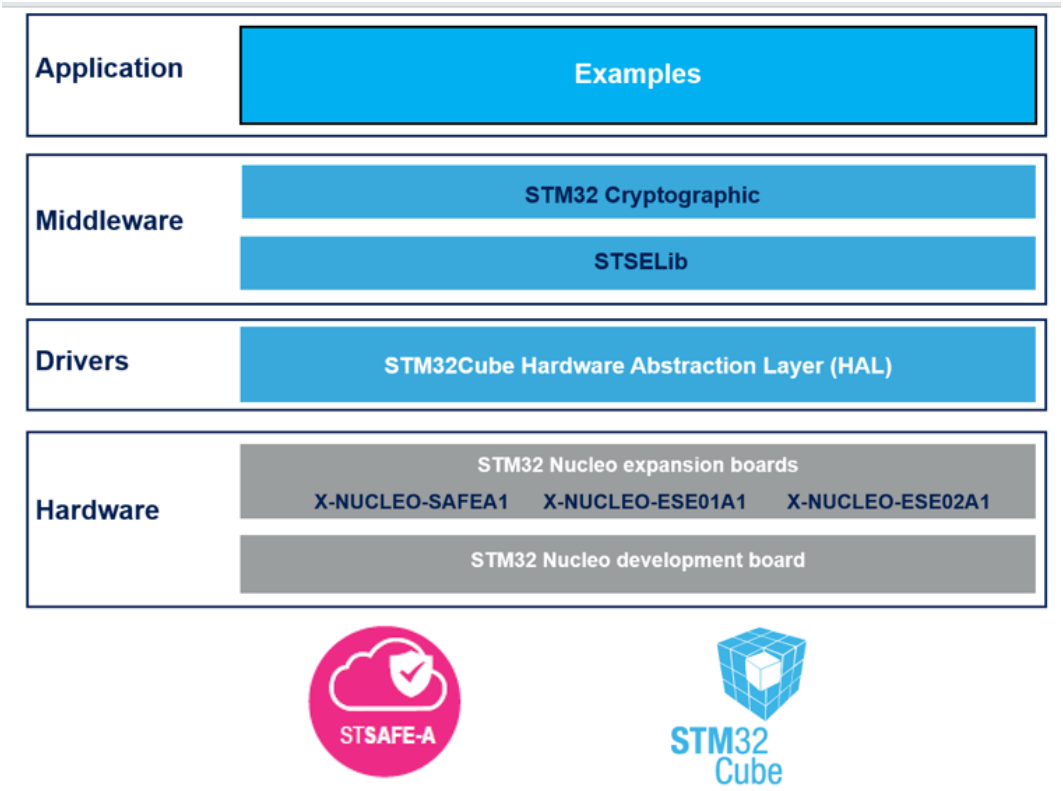
The STSELib middleware is composed of three software modules as illustrated in the figure below. Each layer provides a different level of system abstraction to the embedded system developer.

Figure 1. STSELib architecture



The figure below shows the STSELib middleware integrated in a standard STM32Cube application, running on an X-NUCLEO-SAFEA1, X-NUCLEO-ESE01A1 or X-NUCLEO-ESE02A1 expansion board mounted on an STM32 Nucleo board.

Figure 2. X-CUBE-STSE01 application block diagram



To provide the best hardware and platform independence, the STSELib middleware is not directly connected to the STM32Cube HAL, but through interface files implemented at application level

3.3 Application Programming Interface (API) layer

This software layer is the entry point for the system application. It provides a set of high-level functions allowing interaction with STMicroelectronics Secure Elements. The Api layer provides abstraction for different application like Secure Element Management, Authentication, Data Storage, Key Management.

3.4 Service layer

The SERVICE layer provides a set of product services that format all commands supported by the targeted secure element and reports response to higher layers API/Application. This layer can be used directly from Application (for advanced user).

3.5 Core layer

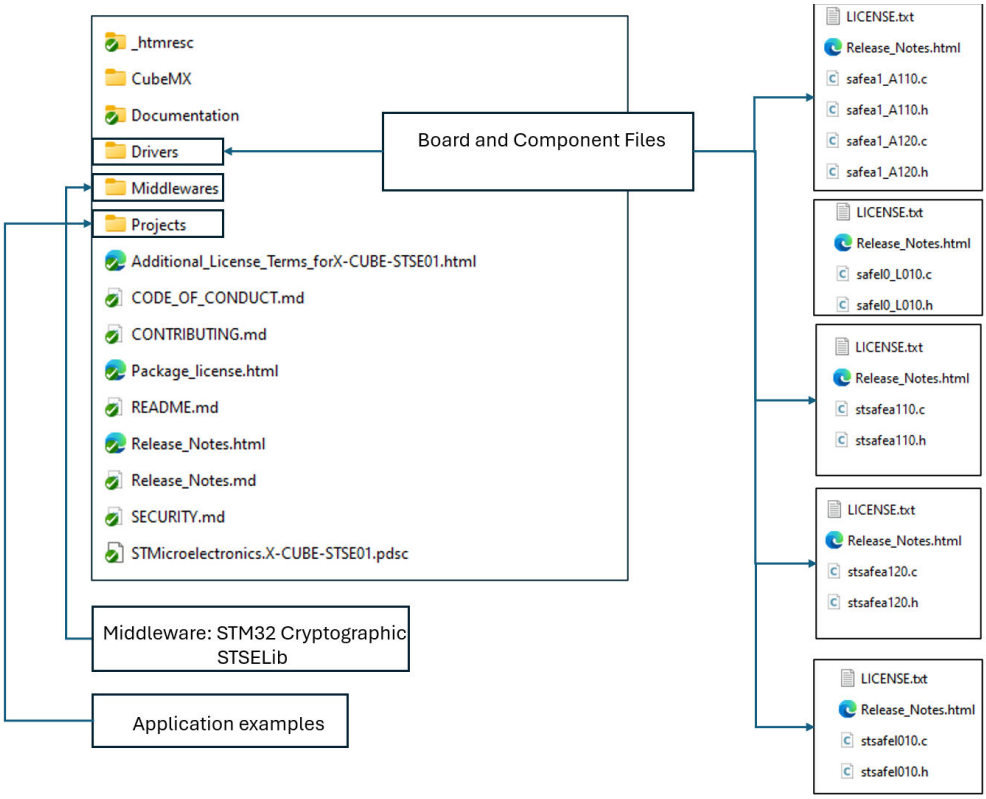
Contains generic definition for ST Secure Element and functions for communicating with target secure element.

Core layer handles the framing of the messages as well as provides the platform abstraction for the above layers.

3.6 Folder structure

The figure below presents the folder structure of the X-CUBE-STSE01.

Figure 3. Project file structure



4 Demonstration software

This section illustrates demonstration software based on the STSELib middleware.

4.1 Authentication

This demonstration illustrates the command flow where the STSAFE-A110/STSAFE-A120/STSAFE-L010 is mounted on a device that authenticates to a remote host (IoT device case), the local host being used as a pass-through to the remote server.

The scenario where the STSAFE-A110/STSAFE-A120/STSAFE-L010 is mounted on a peripheral that authenticates to a local host, for example for games, mobile accessories or consumables, is exactly the same.

Note: For demonstration purposes, the local and remote hosts are the same device here.

1. Extract, parse, and verify the STSAFE-A110/ STSAFE-A120/STSAFE-L010's public certificate stored in the data partition zone 0 of the device to get the public key:
   - Read the certificate using the STSELib middleware through the STSAFE-A110/STSAFE-A120/STSAFE-L010's zone 0.
   - Parse the certificate using the cryptographic library's parser.
   - Read the CA certificate (available through the code).
   - Parse the CA certificate using the cryptographic library's parser.
   - Verify the certificate validity using the CA certificate through the cryptographic library.
   - Get the public key from the STSAFE-A110/STSAFE-A120/STSAFE-L010 X.509 certificate.
2. Generate and verify the signature over a challenge number:
   - Generate a challenge number (random number).
   - Hash the challenge.
   - Fetch a signature over the hashed challenge using the STSAFE-A110/ STSAFE-A120/STSAFE-L010 private key slot 0 through the STSELib middleware.
   - Parse the generated signature using the cryptographic library.
   - Verify the generated signature using the STSAFE-A110/STSAFE-A120/STSAFE-L010's public key through the cryptographic library.
   - When this is valid, the host knows that the peripheral or IoT is authentic.

4.2 Pairing (host Key provisioning)

This code example establishes a pairing between a device and the MCU that it is connected to. The pairing allows the exchanges between the device and the MCU to be authenticated (that is, signed and verified).

The STSAFE-A110/STSAFE-A120 (not available on STSAFE-L010) device become usable only in combination with the MCU that it is paired with.

The pairing consists of the host MCU sending a host MAC key and a host cipher key to the STSAFE-A110/ STSAFE-A120.

Both keys are stored to the protected NVM of the STSAFE-A110/STSAFE-A120 and should be stored to the flash memory of the STM32 device.

By default, in this example, the host MCU sends well-known keys to the STSAFE-A110/STSAFE-A120 (see command flow below) that are highly recommended to use for demonstration purposes. The code also allows the generation of random keys.

Moreover, the code example generates a local envelope key when the corresponding slot is not already populated in the STSAFE-A110/STSAFE-A120. When the local envelope slot is populated, the STSAFE-A110/STSAFE-A120 device allows the host MCU to wrap/unwrap a local envelope to securely store a key on the host MCU's side.

Note: Note: The pairing code example must be executed successfully prior to executing all the following code examples.

Command flow

1. Generate the local envelope key in the STSAFE-A110/STSAFE-A120 using the STSELib middleware.
   By default, this command is activated
   This operation occurs only if the STSAFE-A110/STSAFE-A120's local envelope key slot is not already populated.
2. Define two 128-bit numbers to use as the host MAC key and the host cipher key.
   By default, golden known keys are used. They have the following values:
   – 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
   – Host cipher Key
   0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF, 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF
3. Store the host MAC key and the host cipher key to their respective slot in the STSAFE-A110/STSAFE-A120.
4. Store the host MAC key and the host cipher key to the STM32's flash memory.

4.3 Key establishment (symmetric key AES-128 CMAC)

This demonstration illustrates the case where the STSAFE A devices (not available on STSAFE-L0) are mounted on a device (such as an IoT device), which communicates with a remote server, and needs to establish a secure channel to exchange data with it.

In this example, the STM32 device plays the role of both the remote server (remote host) and the local host that is connected to the STSAFE A series device.

The goal of this use case is to show how to establish a shared secret between the local host and the remote server using the elliptic curve Diffie-Hellman scheme with a static (ECDH) or ephemeral (ECDHE) key in the STSAFE A series device.

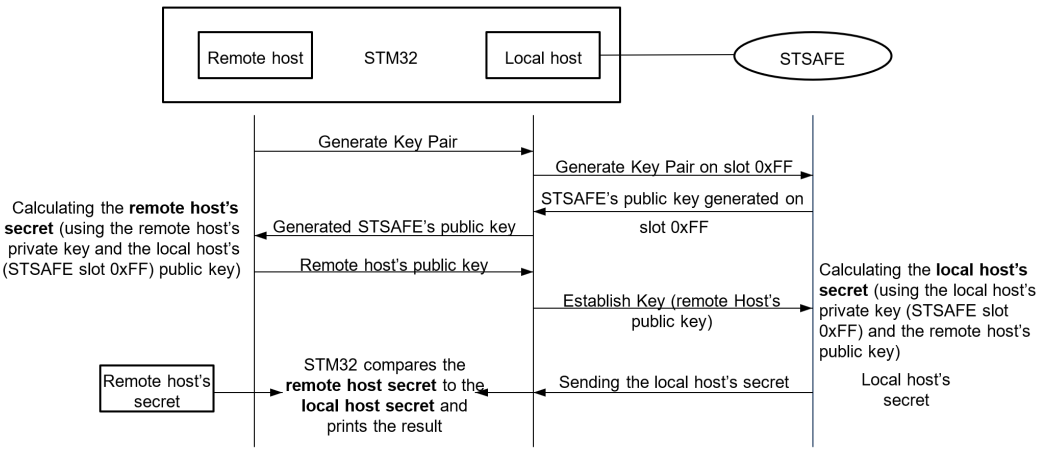
The shared secret should be further derived to one or more working keys (not illustrated here). These working keys can then be used in communication protocols such as TLS, for example for protecting the confidentiality, integrity and authenticity of the data that is exchanged between the local host and the remote server.

Command flow

The Figure 4. Key establishment command flow illustrates the command flow:

- The remote host's private and public keys are hard coded in the code example.
- The local host sends the `Generate Keypair` command to the STSAFE-A110/STSAFE-A120 to generate the key pair on its ephemeral slot (slot 0xFF).
- The STSAFE-A110/STSAFE-A120 sends back the public key (which corresponds to slot 0xFF) to the STM32 (representing the remote host).
- The STM32 computes the remote host's secret (using the STSAFE device's public key and the remote host's private key).
- The STM32 sends the remote host's public key to the STSAFE-A110/STSAFE-A120 and asks the STSAFE-A110/STSAFE-A120 to compute the local host's secret using the API.
- The STSAFE-A110/ STSAFE-A120 sends back the local host's secret to the STM32.
- The STM32 compares the two secrets and prints the result. If the secrets are the same, the secret establishment is successful.

Figure 4. Key establishment command flow



4.4 Wrap/unwrap local envelopes

This demonstration illustrates the case where the STSAFE-A110/STSAFE-A120 (not available on STSAFE-L010) wraps/unwraps the local envelope in order to securely store a secret to any non-volatile memory (NVM).

Encryption/decryption keys can be securely stored in that manner to additional memory or within the STSAFE-A110/STSAFE-A120's user data memory.

The wrapping mechanism is used to protect a secret or plain text. The output of wrapping is an envelope encrypted with an AES key wrap algorithm, and that contains the key or plain text to be protected.

Command flow

Note: The local and remote hosts are the same device here.

1. Generate random data assimilated to a local envelope.
2. Wrap the local envelope using the STSELib middleware API.
3. Store the wrapped envelope.
4. Unwrap the wrapped envelope using the STSELIB middleware.
5. Compare the unwrapped envelope to the initial local envelope. They should be equal.

4.5 Key pair generation

This demonstration illustrates the command flow where the STSAFE-A (not available on STSAFE-L0) devices are mounted on a local host. A remote host asks this local host to generate a key pair (a private key and a public key) on slot 1 and then to sign a challenge (random number) with the generated private key.

The remote host is then able to verify the signature with the generated public key.

This demonstration is similar to the authentication demonstration with two differences:

- The key pair in the authentication demonstration is already generated (on slot 0), whereas, in this example, we generate the key pair on slot 1. The STSAFE-A110/STSAFE-A120 device can also generate the key pair on slot 0xFF, but only for key establishment purposes.
- The public key in the authentication demonstration is extracted from the certificate in zone 0. In this example, the public key is sent back with the STSAFE-A110/STSAFE-A120 response to the generated keypair command.

Command flow

Note: For demonstration purposes, the local and remote hosts are the same device here.

1. The host sends the `Generate Keypair` command to the STSAFE-A110/STSAFE-A120 which sends back the public key to the host MCU.
2. The host generates a challenge (48-byte random number) using the `Generate Random` API. The STSAFE-A110/STSAFE-A120 sends back the generated random number.
3. The host computes the hash of the generated number using the cryptographic library.
4. The host asks the STSAFE-A110/STSAFE-A120 to generate a signature of the computed hash using the `Generate Signature` API. The STSAFE-A110/ STSAFE-A120 sends back the generated signature.
5. The host verifies the generated signature with the public key sent by the STSAFE-A110/ STSAFE-A120 in step 1.
6. The signature verification result is printed.

5 Glossary

Table 1. **Glossary**

Abbreviation	Meaning
AES	Advanced encryption standard
ANSI	American National Standards Institute
API	Application programming interface
BSP	Board support package
CA	Certification Authority
CC	Common Criteria
C-MAC	Command message authentication code
ECC	Elliptic curve cryptography
ECDH	Elliptic curve Diffie–Hellman
ECDHE	Elliptic curve Diffie–Hellman - ephemeral
EWARM	IAR Embedded Workbench® for Arm®
HAL	Hardware abstraction layer
I/O	Input/output
IAR Systems®	World leader in software tools and services for embedded systems development.
IDE	Integrated development environment. A software application that provides comprehensive facilities to computer programmers for software development.
IoT	Internet of things
I²C	Inter-integrated circuit (IIC)
LL	Low-level drivers
MAC	Message authentication code
MCU	Microcontroller unit
MDK-ARM	Keil® microcontroller development kit for Arm®
MPU	Memory protection unit
NVM	Nonvolatile memory
OS	Operating system
SE	Secure element
SHA	Secure Hash algorithm
SLA	Software license agreement
ST	STMicroelectronics
TLS	Transport layer security
USB	Universal serial bus

Revision history

Table 2. **Document revision history**

Date	Revision	Changes
23-Jun-2025	1	Initial release.
27-Nov-2025	2	Updated Introduction, Section 1: General information, Section 2: STSAFE secure element, Section 3.1: General description, Section 3.2: Architecture, Section 3.6: Folder structure, Section 4.1: Authentication, Section 4.3: Key establishment (symmetric key AES-128 CMAC) and Section 4.5: Key pair generation.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.