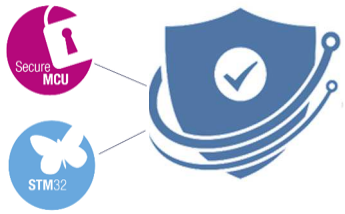
User manual

Getting started with STSAFE-A1xx MW V3 Middleware Software for STSAFE-A1xx

Introduction

This user manual describes how to get started with the STSAFE-A1xx Middleware software package.
The STSAFE-A1xx Middleware is a software component providing a complete set of APIs to access all the STSAFE-A1xx device features from a host microcontroller. The middleware is built on STM32Cube software technology to ease portability across different STM32 microcontrollers. Also, it is MCU agnostic for additional portability across different MCUs.
The STSAFE-A1xx Middleware is a complete library integrating both low level communication drivers to interface the STSAFE-A1xx hardware, and a higher level processing exporting a set of command APIs to easily access the device features from the host microcontroller.
The middleware architecture improves code robustness and security level, as well as reliability, maintainability, scalability.
Although the software architecture has been designed to support different STSAFE-A1xx devices, the current middleware version implementation has been developed and tested for the STSAFE-A100 Secure Element device only.



Contents

List of tables

List of figures

1 General information

The STSAFE-A1xx middleware software component is developed in ANSI C and runs on STM32 32-bit microcontrollers based on the Arm® Cortex®-M processor. Nevertheless, the platform independent architecture allows an easy portability on a varieties of different platforms.
Table 1 presents the definition of acronyms that are relevant for a better understanding of this document.

Table 1. List of acronyms

Term	Definition
API	Application programming interface
BSP	Board support package
CA	Certification authority
CC	Common Criteria
HAL	Hardware abstraction layer
HW	Hardware
IDE	Integrated development environment
I2C	Inter-Integrated Circuit
IoT	Internet of things
LL	Low Level drivers
MCU	Microcontroller Unit
MPU	Memory Protection Unit
MW	Middleware
SE	Secure Element
SW	Software
TLS	Transport Layer Security

2 STSAFE-A100 Secure Element

The STSAFE-A100 is a 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-A100 can be integrated in IoT (Internet of things) devices, smart-home, smart-city and industrial applications, consumer electronics devices, consumables and accessories. 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.
- Security features:
    - Latest generation of highly secure MCUs:
        - CC EAL5+ AVA_VAN5 Common Criteria certified.
        - Active shield.
        - Monitoring of environmental parameters.
        - Protection mechanism against faults.
        - Unique serial number on each die.
        - Protection against side-channel attacks.
    - Advanced asymmetric cryptography:
        - Elliptic curve cryptography (ECC) with NIST or Brainpool 256-bit and 384-bit curves.
        - Elliptic curve digital signature algorithm (ECDSA) with SHA-256 and SHA-384 for digital signature generation and verification.
        - Elliptic curve Diffie-Hellman (ECDH) for key establishment.
    - Advanced symmetric cryptography:
        - Key wrapping and unwrapping using AES-128/AES-256.
        - Secure channel protocols using AES-128.
    - Secure operating system:
        - Secure STSAFE-A100 kernel for authentication and data management.
        - Protection against logical and physical attacks.
    - Hardware features:
        - Highly secure MCU platform.
        - 6 Kbytes of configurable non-volatile memory.
        - Highly reliable CMOS EEPROM technology.
        - 30 years' data retention at 25 °C.
        - 500 000 erase / program cycles endurance at 25 °C.
        - 1.62 V to 5.5 V continuous supply voltage.
        - Operating temperature: –40 to 105 °C.
    - Protocol:
        - I²C-bus slave interface.
        - Up to 400 kbps transmission speed (Fast mode) and true open-

drain pads.
        - 7-bit addressing.
    - Packages:
        - ECOPACK®-compliant SO8N 8-lead plastic small outline and UFDFPN 8-lead ultra thin profile fine pitch dual flat packages.

Please refer to the STSAFE-A100 Datasheet available on www.st.com for additional information on the device.

3 Software Package description

This chapter details the STSAFE-A1xx middleware software package content and the way to use it.

3.1 General description

The STSAFE-A1xx middleware is a software component designed to interface the STSAFE-A1xx secure element device.

The STSAFE-A1xx middleware doesn't currently have a specific download link on www.st.com but is fully integrated within ST software packages as middleware component to add secure element features (e.g. X-CUBE-SBSFU, X-CUBE-SAFEA1).

The software is provided as source code under SLA0088 ST license (see License information for more details).

The following integrated development environments are supported:

- IAR Embedded Workbench® for Arm®: EWARM
- Keil® Microcontroller Development Kit: MDK-ARM
- Integrated Development Environment for STM32: STM32CubeIDE
- System Workbench for STM32: SW4STM32

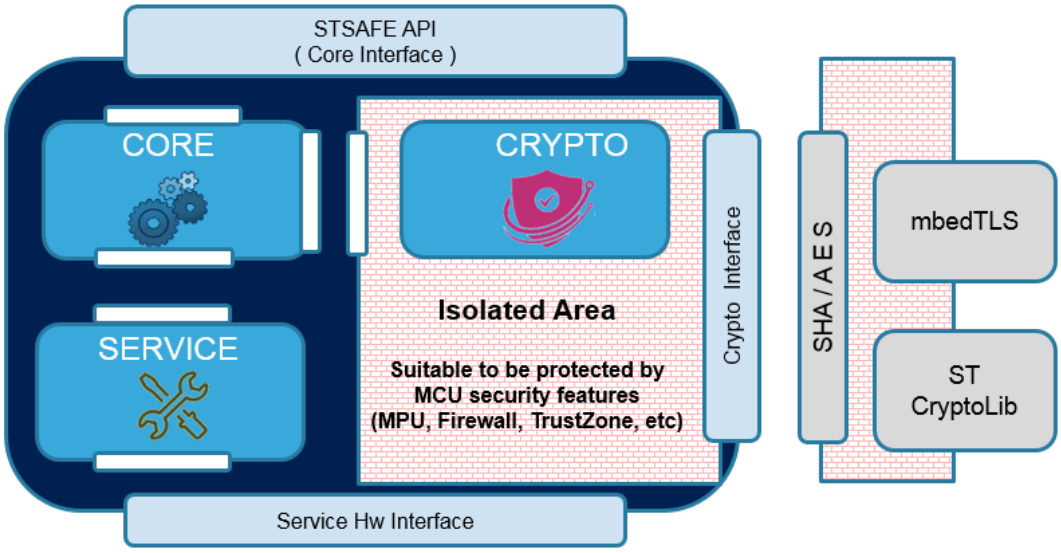
Refer to the release notes available in the package root folder for information about the IDE versions supported.

3.2 Architecture

This section describes the software components of the STSAFE-A1xx middleware software package.

Figure 1 presents a view of the STSAFE-A1xx middleware architecture and related Interfaces.

Figure 1. STSAFE-A1xx Middleware Architecture



The Middleware exposes 3 different interfaces:

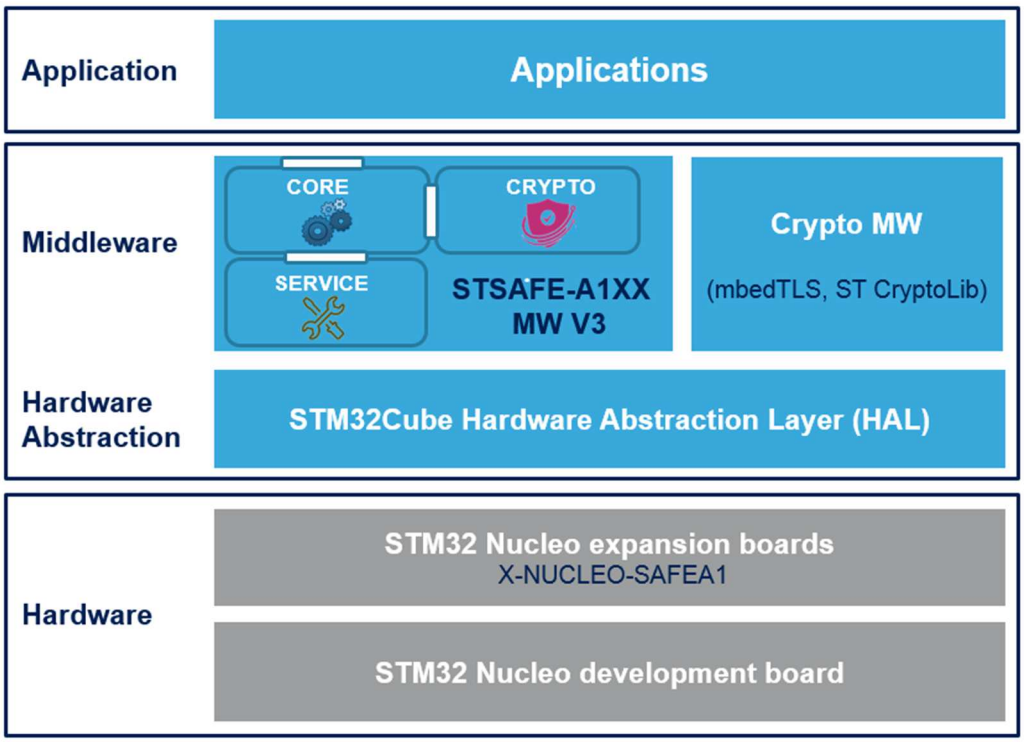
- **STSAFE API**: It's the main Application Programming Interface providing full access to all the STSAFE exported features to the upper layers (application, libraries and stacks). This interface is also referred as Core Interface because all the exported APIs are implemented into the CORE module. Upper layers that want to integrate the STSAFE-A1xx middleware shall access the STSAFE features through this interface.
- **Service Interface**: This interface is used by the STSAFE-A1xx middleware to reach the highest HW platform independence. It includes a set of generic functions to connect the specific MCU, IO Bus, Timing functions.
  Core is accessing to these functions via functions pointer defined into a structure implemented at application level following the example provided within the template *stsafea_service_interface_template.c*. This structure improves the library code reusability and guarantees an easy portability onto other devices.
- **Crypto Interface**: This interface is used by the STSAFE-A1xx middleware to access crypto keys & functions like SHA and AES needed by the middleware itself to properly operate.
  Defined as weak functions, they shall be implemented at application level following the example provided with two different offered templates:
  - *stsafea_crypto_mbedtls_interface_template.c* if the mbedTLS crypto library is used;
  - *stsafea_crypto_stlib_interface_template.c* if the ST crypto library is used;

  Alternative crypto libraries can be used by simply customizing the template source files.

Template files are provided for easy integration and customization within the upper layers.

Some examples of application integrating and using the STSAFE-A1xx middleware (e.g. X-CUBE-SBSFU, X-CUBE-SAFEA1) are based on STM32CubeHAL, the hardware abstraction layer for STM32 microcontrollers.
Figure 2 shows the STSAFE-A1xx middleware integrated into a standard STM32Cube application, running on a STM32 Nucleo board plus the X-NUCLEO-SAFEA1 expansion board.

Figure 2. STSAFE-A1xx Middleware into STM32Cube application



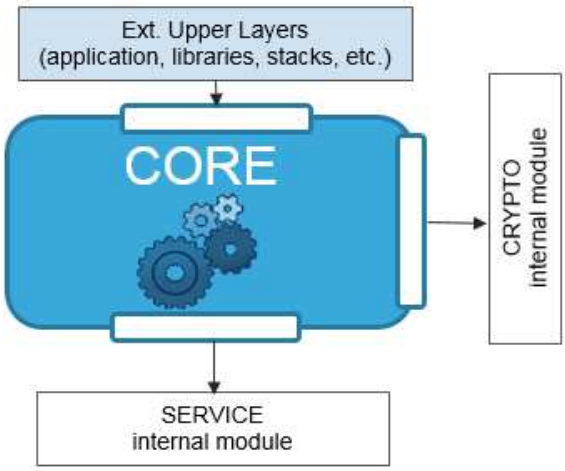
To provide the best HW and platform independencies, the STSAFE-A1xx middleware is not directly connected to the STM32Cube HAL.
This connection is done through the interfaces files implemented at application level (*stsafea_service_interface_template.c, stsafea_interface_conf.h*).

3.3 CORE Module

The CORE module is the core of the Middleware and implements the commands called by the upper layers (application, libraries, stack, etc.) in order to properly use the STSAFE-A1xx features.

Figure 3 presents a view of the CORE module architecture.

Figure 3. CORE Module Architecture



The CORE module is a multi-interface software component connected to:

- Upper layers: external connection through the exported API described in Table2.
- Crypto layer: internal connection to the CRYPTO module.
- Service HW layer: internal connection to the SERVICE module.

A complete API documentation of the CORE module is provided within the STSAFE-A1xx Middleware software package in the root folder (see *STSAFE-A1xx_Middleware.chm* file).
Also please refer to the STSAFE-A110 Datasheet for a brief explanation of the command set which the commands API listed in the following table are related to.

Table 2. CORE module exported API

API Category	Function
Initialization Configuration	**StSafeA_Init** *To create, initialize and assign the STSAFE-A1xx device handle*
	StSafeA_GetVersion *To return the STSAFE-A1xx Middleware revision*
General Purpose Commands	**StSafeA_Echo** *To execute the echo command, expecting receive back same data passed in the command from device*
	StSafeA_Reset *To reset the volatile attributes to their initial value*
	StSafeA_GenerateRandom

	To generates a number of random bytes
	StSafeA_Hibernate *To put the device in hibernation*
	StSafeA_StartSignatureSession (only for STSAFE-A100) *To start an asymmetric cryptographic session*
Data Partition Commands	**StSafeA_DataPartitionQuery** *Query command to retrieve data partition configuration*
	StSafeA_Decrement *To decrement the one-way counter in a counter zone*
	StSafeA_Read *To read data from a data partition zone*
	StSafeA_Update *To update data through zone partition*
Private and Public Key Commands	**StSafeA_GenerateKeyPair** *To generate a key-pair in a private key slot*
	StSafeA_GenerateSignature *To return the ECDSA signature over a message digest*
	StSafeA_VerifyMessageSignature *To verify message authentication*
	StSafeA_EstablishKey *To establish shared secret between 2 hosts by using asymmetric cryptography*
	StSafeA_GetSignature (only for STSAFE-A100) *To return the ECDSA signature over the command response sequence since the start of a signature session*
	StSafeA_VerifyEntitySignature (only for STSAFE-A100) *To verify entity authentication*
Administrative Commands	**StSafeA_ProductDataQuery** *Query command to retrieve product data*
	StSafeA_I2cParameterQuery *Query command to retrieve I2C address and low power mode configuration*
	StSafeA_LifeCycleStateQuery *Query command to retrieve the life cycle state product status*
	StSafeA_PublicKeySlotQuery (only for STSAFE-A100) *Query command to retrieve public key information (presence & curve ID)*
	StSafeA_HostKeySlotQuery *Query command to retrieve the host key information (presence & host C-MAC counter)*
	StSafeA_PutAttribute *To put attributes in the STSAFE-Axxx device like keys, password, I2C parameters according to the attribute TAG*
	StSafeA_DeletePassword *To delete the password form its slot*
	StSafeA_VerifyPassword *To perform password verification and remembers the outcome of the verification for future command authorization*
	StSafeA_CommandAuthorizationConfigurationQuery (only for STSAFE-A110) *Query command to retrieve command authorization configuration*

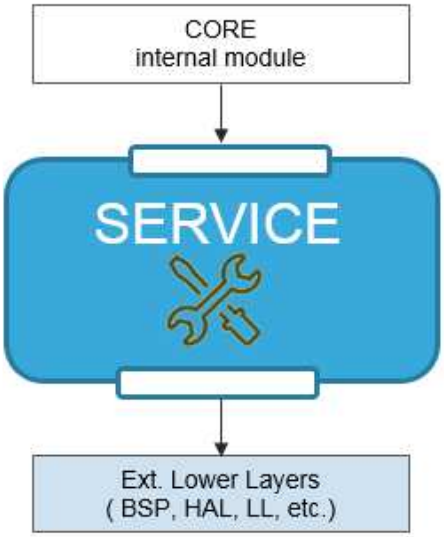
Local Envelope Commands	**StSafeA_LocalEnvelopeKeySlotQuery** *Query command to retrieve local envelope keys information (slot number, presence & key length) for the available key slots*
	StSafeA_GenerateLocalEnvelopeKey *To generate a key in a local envelope key slot*
	StSafeA_WrapLocalEnvelope *To wrap data and typically working keys that are entirely managed by the host, with a local envelope key and the [AES key wrap] algorithm*
	StSafeA_UnwrapLocalEnvelope *To unwrap a local envelope with a local envelope key*
Additional Commands	**StSafeA_RawCommand** *To execute a raw command and receive the related response*

3.4 SERVICE Module

The SERVICE module is the low layer of the Middleware and implements a full hardware abstraction in terms of MCU and HW platform.

Figure 4 presents a view of the SERVICE module architecture.

Figure 4. CORE Module Architecture



The SERVICE module is a dual-interface software component connected to:

- Core layer: internal connection to the CORE module through the exported API described in Table 3.
- External Lower layers: such as BSP, HAL, LL.
  Functions must be implemented at external higher layers called *service interface* layers. They are based on the *stsafea_service_interface_template.c* template file.

A complete API documentation of the SERVICE module is provided within the STSAFE-A1xx Middleware software package in the root folder (see *STSAFE-A1xx_Middleware.chm* file).

Table 3. SERVICE module exported API

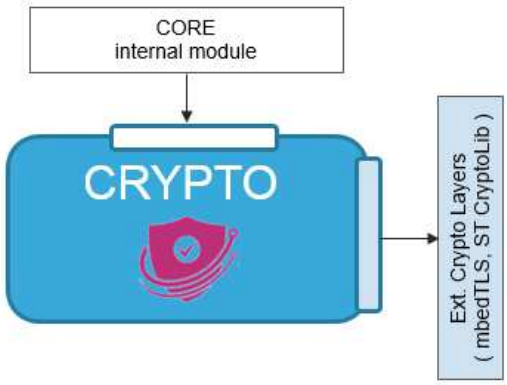
API Category	Function
Initialization Configuration	**StSafeA_HW_Init** *To initialize the communication bus and the IO pins needed to operate the STSAFE-Axxx device*
Low Level operation functions	**StSafeA_Transmit** *To prepare the command to be transmitted and call the low level bus to execute. Compute and concatenate CRC if supported*
	StSafeA_Receive *To receive data from STSAFE-Axxx by using the low level bus functions to retrieve it. Check the CRC, if supported.*
	StSafeA_Delay *To provide a delay in milliseconds*

3.5 CRYPTO Module

The CRYPTO module represents the cryptographic part of the Middleware and implements all the sensitive operations such as MAC, SHA, encryption and decryption.
The CRYPTO module if completely independent from the other middleware modules and for this reason can be easily encapsulated inside an isolated secure area suitable to be protected by MCU security features such as Memory Protection Unit (MPU), Firewall, TrustZone, etc.

Figure 3 presents a view of the CORE module architecture.

Figure 5. CRYPTO Module Architecture



The CRYPTO module is a dual-interface software component connected to:

- Core layer: internal connection to the CORE module through the exported API described in Table 4;
- External crypto library: mbedTLS and ST CryptoLib are currently supported. Weak functions must be implemented at external higher layers and are based on the:
    - *stsafea_crypto_mbedtls_interface_template.c* for mbedTLS crypto library;
    - *stsafea_crypto_stlib_interface_template.c* for the ST crypto library;

  Additional crypto libraries can be easily supported adapting the crypto interface template file.

A complete API documentation of the CRYPTO module is provided within the STSAFE-A1xx Middleware software package in the root folder (see *STSAFE-A1xx_Middleware.chm* file).

Table 4. CRYPTO module exported API

API Category	Function
Crypto APIs	**StSafeA_InitHASH** *SHA Initialization. Used for STSAFE-A1xx signature session*
	StSafeA_ComputeHASH *To compute the HASH value. Used for STSAFE-A1xx signature session*
	StSafeA_ComputeCMAC *To compute the CMAC value. Used on prepared command*
	StSafeA_ComputeRMAC *Compute the RMAC value. Used on received response*
	StSafeA_DataEncryption *Executes data encryption (AES CBC) on the STSAFE-Axxx data buffer.*
	StSafeA_DataDecryption *Executes data decryption (AES CBC) on the STSAFE-Axxx data buffer.*
	StSafeA_MAC_SHA_PrePostProcess *Pre or Post Process the MAC and/or SHA before transmitting or after receiving from STSAFE_Axxx device*

3.6 Templates

This section describes in more details the templates available within the STSAFE-A1xx middleware software package.
All the templates listed into the Table 5 are provided inside the “Interface” folder available at the root level of the middleware SW package.
Template files are provided as examples to be copied and customized into the upper layers, in order to easily integrate and configure the STSAFE-A1xx Middleware.

Table 5. Templates

Template Category	Template File
Service Interface templates	**stsafea_service_interface_template.c** *Example template to show how to support the hardware services required by the STSAFE-A Middleware and offered by the specific HW, Low Level library, or BSP selected at user space.*
Crypto Interface templates	**stsafea_crypto_mbedtls_interface_template.c** *Example template to show how to support the crypto services required by the STSAFE-A Middleware and offered by the mbedTLS crypto library (Key Management, SHA, AES, etc.).*
	stsafea_crypto_stlib_interface_template.c *Example template to show how to support the crypto services required by the STSAFE-A Middleware and offered by the STM32 crypto library (Key Management, SHA, AES, etc.).*
Configuration templates	**stsafea_conf_template.h** *Example template to show how to configure the STSAFE-A Middleware such as optimizations etc.*
	stsafea_interface_conf_template.h *Example template to show how to configure and customize the interface files listed above.*

3.6.1 Service interface template

Service interface template files provide example implementation of function in charge to initialize functions pointer to specific MCU, IO Bus, Timing & CRC API offered as empty or partially empty functions inside the middleware.
These must be properly implemented at user space or anyway in the upper layers according to the hardware user choices.

3.6.2 Crypto interface template

Crypto interface template files provide example implementation of the *__weak* functions, offered as empty or partially empty functions inside the middleware. They must be properly implemented at user space or anyway in the upper layers according to the crypto user choices.

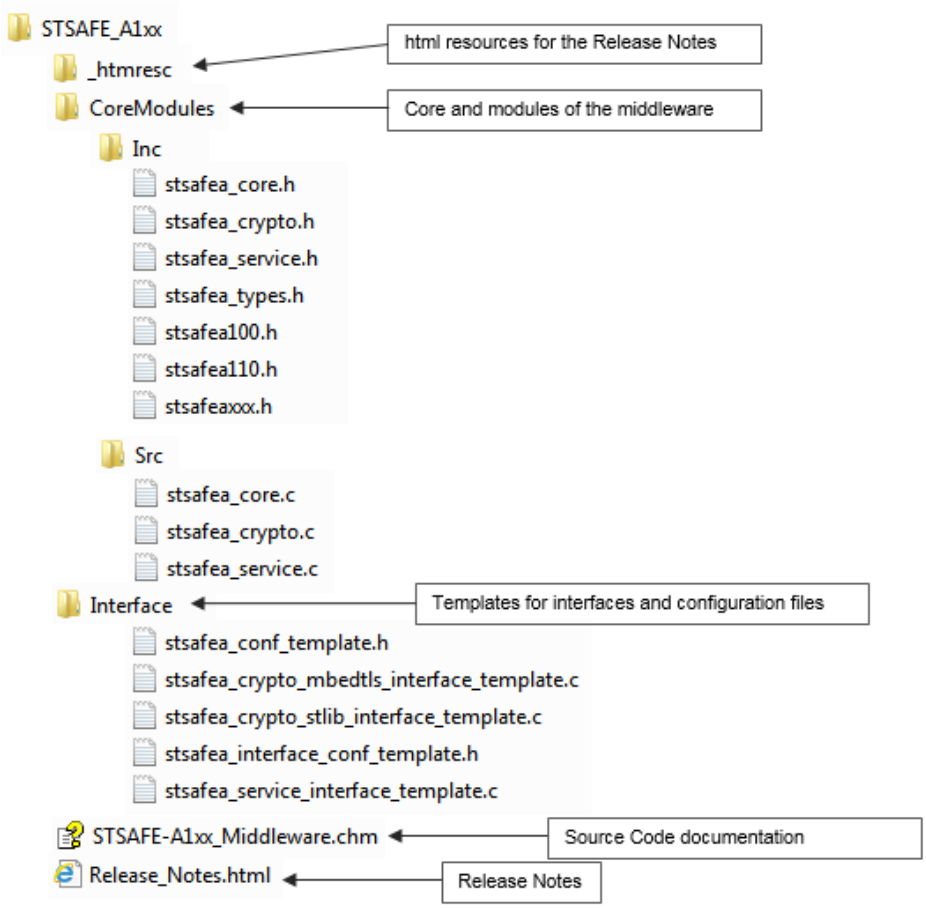
3.6.3 Configuration template

Configuration template files provide an easy way to configure the STSAFE-A1xx middleware and features that can be used in the user application, such as optimizations, specific hardware etc.

3.7 Folder structure

Figure 4 presents the folder structure of the STSAFE-A1xx middleware software package.

Figure 6. Project file structure



4 How To: Integration and Configuration

This section describes how to integrate and configure the STSAFE-A1xx Middleware in your application.

4.1 Integration Steps

Follow these steps in order to integrate the STSAFE-A1xx Middleware into your own application:

- STEP 1: At user space copy (and optionally rename) the *stsafea_service_interface_template.c* and one between *stsafea_crypto_mbedtls_interface_template.c* or *stsafea_crypto_stlib_interface_template.c* according to the crypto library that the user has already added to the application (the user can even use a different one and create/implement his own crypto interface file from scratch).
- STEP 2: At user space copy (and optionally rename) the *stsafea_conf_template.h* and the *stsafea_interface_conf_template.h*.
- STEP 3: make sure to add the right includes in your main or other user space source file that needs to interface the STSAFE-A1xx Middleware:
  #include "stsafea_core.h"
  #include "stsafea_interface_conf.h"
- STEP 4: Customize the files used in the 3 steps here above according to the user preferences.

4.2 Configuration Steps

In order to properly configure the STSAFE-A1xx Middleware into the user application, two different configuration template files are provided, to be copied and customized at user space according to the user choices:

- *stsafea_interface_conf_template.h*: This example template allows and shows how to configure the crypto and service middleware interfaces at user spaces through the following defines:
  - USE_PRE_LOADED_HOST_KEYS
  - USE_SIGNATURE_SESSION
  - MCU_PLATFORM_INCLUDE
  - MCU_PLATFORM_BUS_INCLUDE
  - MCU_PLATFORM_CRC_INCLUDE
- *stsafea_conf_template.h*: This example template allows and shows how to configure the STSAFE-A Middleware through the following defines:
  - STSAFEA_USE_OPTIMIZATION_SHARED_RAM
  - STSAFEA_USE_OPTIMIZATION_NO_HOST_MAC_ENCRYPT
  - STSAFEA_USE_FULL_ASSERT

Follow these steps in order to integrate the STSAFE-A1xx Middleware into your own application:

- STEP 1: At user space, copy (and optionally rename) the *stsafea_interface_conf_template.h* and the *stsafea_conf_template.h*.
- STEP 2: Confirm or modify the *#define* of the two header files here above according to the user platform and crypto choices.

5 License information

The STSAFE-A1xx middleware software component is licensed by ST under STSAFE DRIVER SOFTWARE LICENSE AGREEMENT (SLA0088), the “License”. You may not use this component except in compliance with the License.
You may obtain a copy of the License at the following link: SLA0088

6 Revision history

Table 6. Document revision history

Date	Revision	Changes
01-Jun-2019	1	• Initial release.
09-Apr-2020	2	• Update reference to STSAFE-A110. • Update SW package reference to X-CUBE-SAFEA1. • Update HW board reference to X-NUCLEO-SAFEA1. • Update CORE module exported API table. • Update templates description.