 **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

[1 General information 5](#_Toc11255749)

[2 STSAFE-A100 Secure Element 6](#_Toc11255750)

[3 Software Package description 8](#_Toc11255751)

[3.1 General description 8](#_Toc11255752)

[3.2 Architecture 9](#_Toc11255753)

[3.3 CORE Module 11](#_Toc11255754)

[3.4 SERVICE Module 14](#_Toc11255755)

[3.5 CRYPTO Module 16](#_Toc11255756)

[3.6 Templates 18](#_Toc11255757)

[3.7 Folder structure 19](#_Toc11255758)

[4 How To: Integration and Configuration 20](#_Toc11255759)

[4.1 Integration Steps 20](#_Toc11255760)

[4.2 Configuration Steps 20](#_Toc11255761)

[5 License information 22](#_Toc11255762)

[6 Revision history 23](#_Toc11255763)

List of tables

[Table 1. List of acronyms 5](#_Toc11255764)

[Table 2. CORE module exported API 11](#_Toc11255765)

[Table 3. SERVICE module exported API 15](#_Toc11255766)

[Table 4. CRYPTO module exported API 17](#_Toc11255767)

[Table 5. Templates 18](#_Toc11255768)

[Table 6. Document revision history 23](#_Toc11255769)

List of figures

[Figure 1. STSAFE-A1xx Middleware Architecture 9](#_Toc11255776)

[Figure 2. STSAFE-A1xx Middleware into STM32Cube application 10](#_Toc11255777)

[Figure 3. CORE Module Architecture 11](#_Toc11255778)

[Figure 4. CORE Module Architecture 14](#_Toc11255779)

[Figure 5. CRYPTO Module Architecture 16](#_Toc11255780)

[Figure 6. Project file structure 19](#_Toc11255781)

# General information

The STSAFE-A1xx middleware software component is developed in ANSI C and runs on STM32 32-bit microcontrollers based on the Arm®([[1]](#footnote-1)) 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 |

# 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
* 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
  + 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](http://www.st.com) for additional information on the device.

# Software Package description

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

## 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](http://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-STSAFE).

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

The following integrated development environments are supported:

• IAR Embedded Workbench® for Arm® (EWARM)

• Keil® Microcontroller Development Kit (MDK-ARM)

• System Workbench for STM32

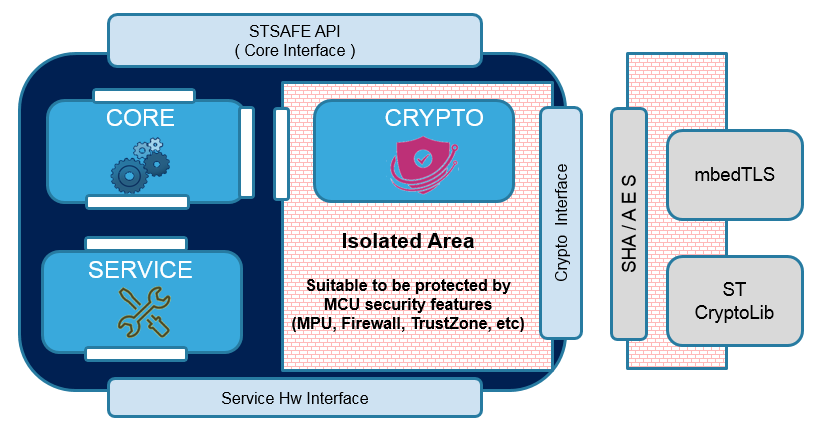
Refer to the release notes available in the package root folder for information about the IDE versions supported.

## **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



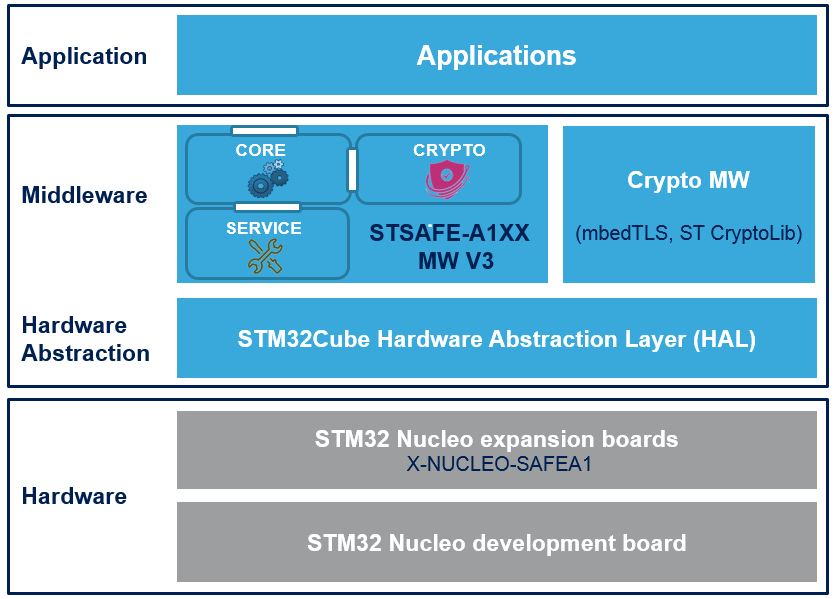
The Middleware exposes 3 different interfaces:

* **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 Hw 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. This structure improves the library code reusability and guarantees an easy portability onto other devices.  
  Defined as weak functions, they shall be implemented at application level following the example provided within the template *stsafea\_service\_interface\_template.c*. The template is providing for easy integration and customization within the upper layers.
* **Crypto Interface**: This interface is used by the STSAFE-A1xx middleware to access crypto 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. The 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-STSAFE) 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-STSAFEA1x expansion board.

Figure 2. STSAFE-A1xx Middleware into STM32Cube application



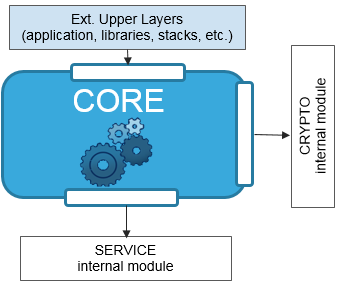
To provide the best HW and platform independencies, the STSAFE-A1xx middleware is not directly connected to the STM32Cube HAL.   
This connection is done through the interfaces files implemented at application level (*stsafea\_service\_interface\_template.c, stsafea\_interface\_conf.h*).

## **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](#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-A100 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

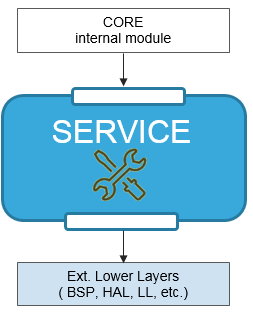
|  |  |
| --- | --- |
| **API Category** | **Function** |
| Initialization Configuration | **StSafeA\_Init** |
| *To create, initialize and assign the STSAFE-A1xx device handle* |
| **StSafeA\_GetVersion** |
| *To return the STSAFE-A1xx Middleware revision* |
| General Purpose Commands | **StSafeA\_Echo** *To executes the echo command, expecting to receive back from STSAFEA the same data passed in the command* |
| **StSafeA\_Reset** *To reset the volatile attributes to their initial value* |
| **StSafeA\_GenerateRandom** *To generates a number of random bytes* |
| **StSafeA\_StartSignatureSession** *To start an asymmetric cryptographic session* |
| **StSafeA\_Hibernate** *To put the STSAFE-Axxx device in hibernation* |
| Data Partition Commands | **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\_GetSignature** *To return the ECDSA signature over the command response sequence since the start of a signature session* |
| **StSafeA\_GenerateSignature** *To return the ECDSA signature over a message digest* |
| **StSafeA\_VerifyMessageSignature** *To verify message authentication* |
| **StSafeA\_VerifyEntitySignature** *To verify entity authentication* |
| **StSafeA\_EstablishKey** *To establish shared secret between 2 hosts by using asymmetric cryptography* |
| Administrative Commands  Administrative Commands | **StSafeA\_ProductDataQuery** *Query command to retrieve product data* |
| **StSafeA\_DataPartitionQuery** *Query command to retrieve data partition configuration* |
| **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** *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\_LocalEnvelopeKeySlotQuery** *Query command to retrieve local envelope keys information (slot number, presence & key length) for the available key slots* |
| **StSafeA\_CommandAuthorizationConfigurationQuery** *Query command to retrieve command authorization configuration* |
| **StSafeA\_PutAttribute** *To put attributes in the STSAFE-Axxx device like keys, password, I2C parameters according to the attribute TAG* |
| **StSafeA\_PutPublicKeySlot** *To fill the public key slot* |
| **StSafeA\_GenerateKeyPair** *To generate a key-pair in a private key slot* |
| **StSafeA\_GenerateLocalEnvelopeKey** *To generate a key in a local envelope key slot* |
| **StSafeA\_DeletePassword** *To delete the password form its 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* |
| **StSafeA\_VerifyPassword** *To perform password verification and remembers the outcome of the verification for future command authorization* |
| Additional Commands | **StSafeA\_RawCommand** *To execute a raw command and receive the related response* |
|

## **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:

* External Lower layers: such as BSP, HAL, LL. Weak functions must be implemented at external higher layers and are based on the stsafea*\_service\_interface\_template.c* template file;
* Core layer: internal connection to the CORE module through the exported API described in [Table3](#Table3);

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\_BSP\_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* |

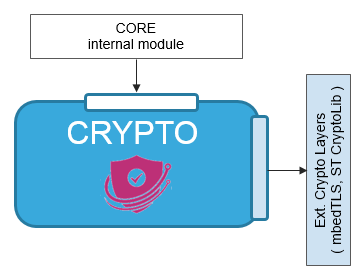
## **CRYPTO Module**

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:

* 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.

* Core layer: internal connection to the CORE module through the exported API described in [Table4](#Table4);

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* |

## **Templates**

This section describes in more details the templates available within the STSAFE-A1xx middleware software package.

All the templates listed into the [Table4](#Table4) 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.:

* 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 and to the hardware user choices.
* 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.

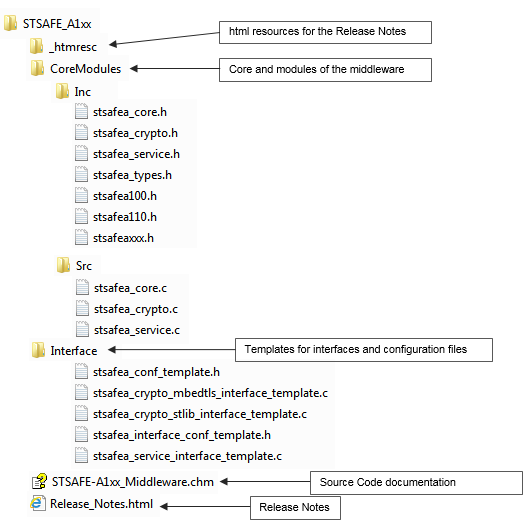
Table 5. Templates

|  |  |
| --- | --- |
| **Template Category** | **Template File** |
| 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.* |
| **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.* |

## Folder structure

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

Figure 6. Project file structure



# How To: Integration and Configuration

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

## 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.

## 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
* *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
  + STSAFEA\_USE\_OPTIMIZATION\_CRC\_TABLE
  + 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.

# 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](https://www.st.com/content/st_com/en/products/embedded-software/secure-mcu-software/stsw-stsa100.html#resource)

# Revision history

Table 6. Document revision history

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Changes** |
| 01-Jun-2019 | 1 | Initial release. |

IMPORTANT NOTICE – PLEASE 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. 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 acknowledgement.

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 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

ST and the ST logo are trademarks of ST. 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.

© 2017 STMicroelectronics – All rights reserved

1. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere [↑](#footnote-ref-1)