# PSA Certified™ Functional API Step-by-step guide


| | | | |
|---|---|---|---|
| Document number: | JSADEN006 | Version: | 1.6 |
| Date of Issue: | 15/02/2019 | | |
| Author: | Arm | | |

# PSA Functional API certified step by step guide

## Getting your product PSA Functional API certified

### Background:

PSA Functional API certification is an API compliance program that ensures PSA Root of Trust (PSA-RoT) security functions can be accessed using the PSA Functional APIs. Test Suites are provided so that chip vendors, OS providers and OEMs can check the correct functioning of the PSA Functional APIs and help enable a security ecosystem based on interoperable solutions.

The following is provided as guidance for developers wanting to showcase their PSA Functional API certified solutions on the psacertified.org website. If you have further questions on PSA Functional API certification, please email psacertified@arm.com.

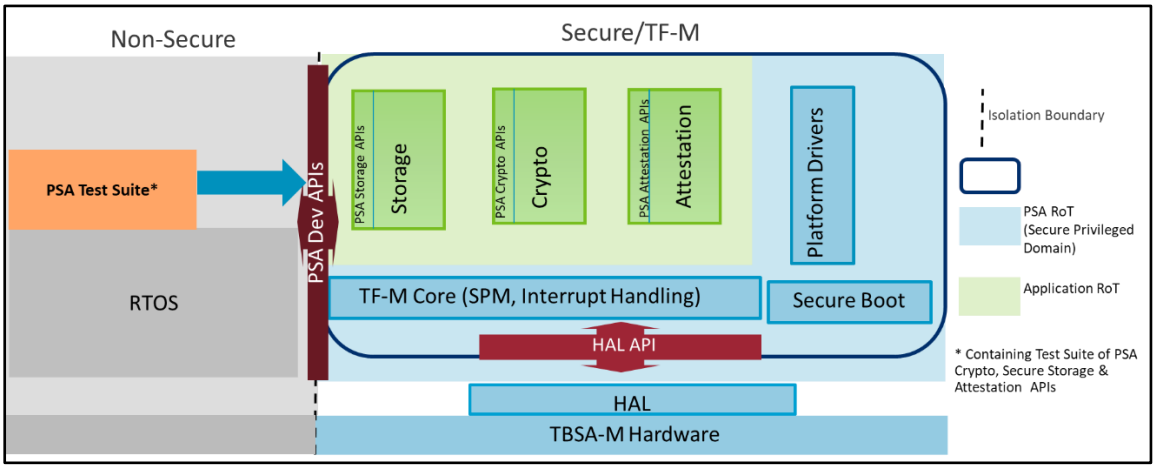


Fig 1: PSA Functional API Test Suites testing security functions of Trusted Firmware -M

### Executive Summary:

PSA Functional API certification requires developers to run the API compliance test suite on their implementation and submit the results for Crypto, Secure Storage and Attestation APIs. If the vendor wishes to use the PSA Functional API logo they can request a trademark agreement by email from Arm. There is also an opportunity for them to showcase their PSA Functional API Certified product on the psacertifed.org website (see below how to request this).

Note: TF-M refers to Trusted Firmware for Cortex-M, an open source software reference implementation for Arm v8-M based chips.

## Getting your product PSA Functional API certified

The steps to gain PSA Functional API certification are slightly different between OS vendor and chip developer:

### Steps for RTOS vendor

1. Download the API compliance test suite source code and porting guide. There are tests for Crypto, Attestation and Secure Storage APIs
2. Port the API compliance test suite application to the RTOS
3. Run and pass the compliance checker tests with the following micro steps
    - Integrate the RTOS with a Functional API certified device (e.g. Musca dev board & TF-M)
    - Interface RTOS to the TF-M PSA Dev APIs for PSA Crypto, Secure Storage and Attestation API on target device
    - RTOS can natively implement a sub-set of the APIs and leverage TF-M implementation for remaining
    - Run the PSA API tests and capture the output report including pass/fail end result
4. Submit the output log of the test suite run with the RTOS product info by emailing it to psacertified@arm.com
5. Acknowledgement and approval of the test report will be communicated by the scheme manager
6. An authorized product owner can request to have their PSA Functional API certified RTOS product showcased on the psacertified.org website
7. An authorized product owner can request a trademark license to use the appropriate PSA "Functional API certified" logo through psacertified@arm.com

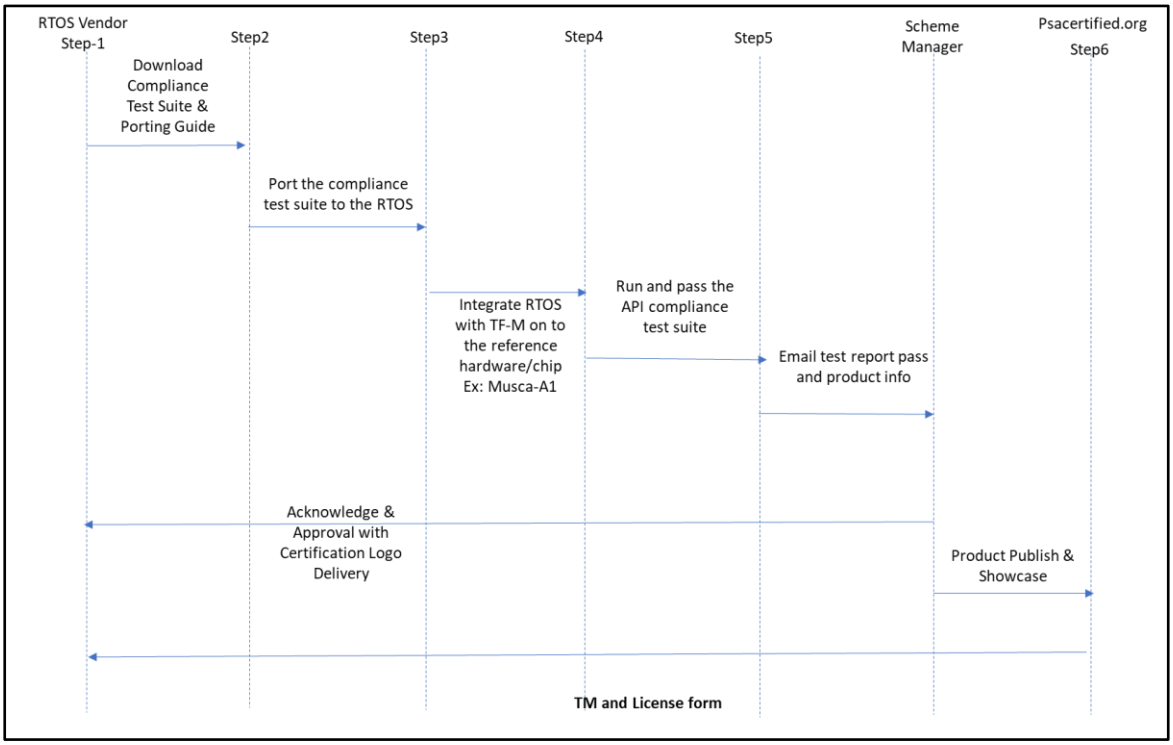


Fig 2. Process flow for PSA Functional API Certified – RTOS Vendor

## Chip/Device vendor:

1. Port TF-M to your chip or implement PSA Functional APIs according to the PSA specification. The security functions in scope are: Crypto, Attestation and Secure Storage
2. Provide the implementation as part of the reference trusted code for the target chip/device
3. Select an RTOS that has ported the compliance checker and successfully run it on an Arm reference platform with a compliant TF-M or any other PSA Functional API certified device
4. Port and integrate the RTOS with the target device and its TF-M/trusted firmware
5. Perform test and pass the compliance test suite
6. Submit the output log of the test suite run with the chip/device product info by emailing it to psacertified@arm.com
8. Acknowledgement and approval of the test report will be communicated by the scheme manager
9. An authorized product owner can request to showcase the PSA Functional API certified chip product on the psacertified.org website
10. An authorized product owner can request a trademark license to use the appropriate PSA "Functional API certified" logo through psacertified@arm.com

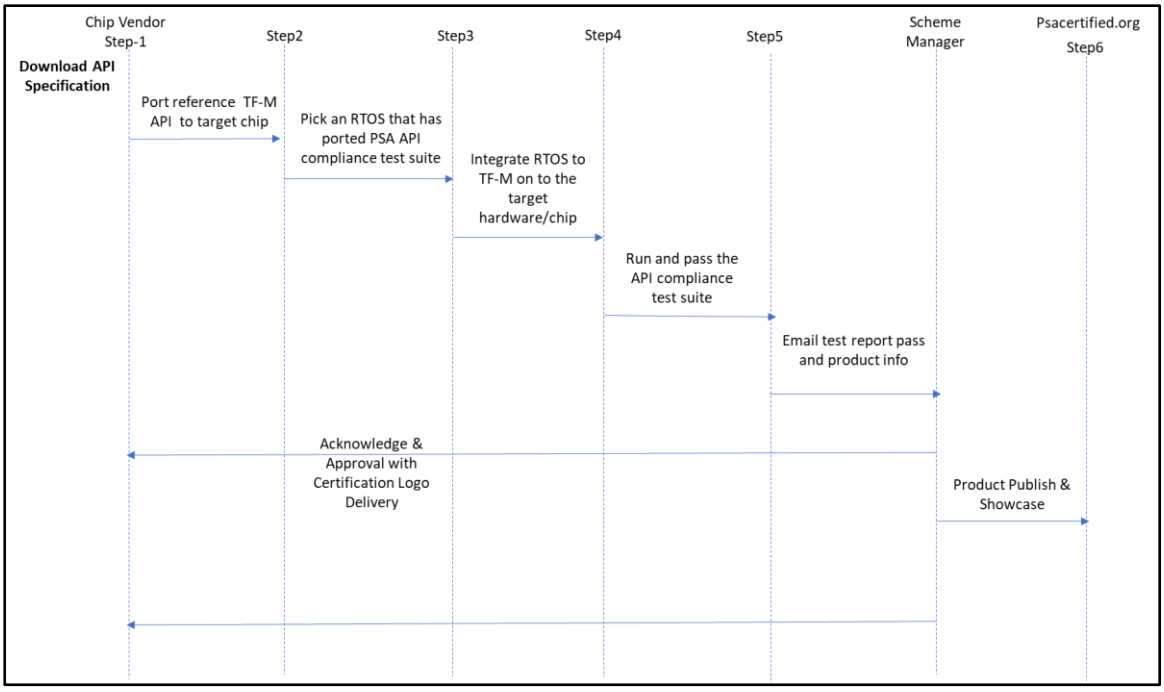


Fig 3. Process flow for PSA Functional API Certified – Chip Vendor

## Showcasing your PSA Functional API certified product

If the developer product has passed and wants to showcase the products on psacertified.org, they should send the following to psacertified@arm.com along with the test suite output report:

1. Test suite output report
2. Company Logo
3. Product name or Product Family name

4. Short product description (25 words)
5. Image or graphic to represent the product
6. Link to the developer's website for the product (if appropriate)
7. Please state whether your company would like to use the PSA Certified logo and trademarks

If the Developer wishes to use the PSA Certified logos and trademarks, a trademark agreement will be sent by return email.


Fig 4. Example- Certified products showcase on [www.psacertified.org](http://www.psacertified.org)

## Q&A:

Q. Is it only crypto, attestation and secure storage test suites I need to run? - Yes
Q. Do I need to run any other test suites e.g. IPC, TBSA ACK? – No
Q. Can I run in library mode? - Yes
Q. Is PSA Functional API same as Developers facing API? - Yes

## Appendix-A – Sample PSA Test Suite Output

```
***** PSA
Architecture
Test Suite -
Version 0.8
*****
                Running.. Attestation Suite
                ****************************************
                TEST: 801 | DESCRIPTION: Testing initial attestation APIs
                [Info] Executing tests from non-secure
                [Check 1] Test psa_initial_attestation_get_token with Challenge 32
                [Check 2] Test psa_initial_attestation_get_token with Challenge 48
                [Check 3] Test psa_initial_attestation_get_token with Challenge 64
                [Check 4] Test psa_initial_attestation_get_token with zero challenge size
                [Check 5] Test psa_initial_attestation_get_token with small challenge size
                [Check 6] Test psa_initial_attestation_get_token with invalid challenge size
                [Check 7] Test psa_initial_attestation_get_token with large challenge size
                [Check 8] Test psa_initial_attestation_get_token with zero as token size
                [Check 9] Test psa_initial_attestation_get_token with small token size
                [Check 10] Test psa_initial_attestation_get_token_size with Challenge 32
                [Check 11] Test psa_initial_attestation_get_token_size with Challenge 48
                [Check 12] Test psa_initial_attestation_get_token_size with Challenge 64
                [Check 13] Test psa_initial_attestation_get_token_size with zero challenge size
                [Check 14] Test psa_initial_attestation_get_token_size with small challenge size
                [Check 15] Test psa_initial_attestation_get_token_size with invalid challenge
                size
                [Check 16] Test psa_initial_attestation_get_token_size with large challenge size
                TEST RESULT: PASSED
                ****************************************
                ********** Attestation Suite Report **********
                TOTAL TESTS : 1
                TOTAL PASSED : 1
                TOTAL SIM ERROR : 0
                TOTAL FAILED : 0
                TOTAL SKIPPED : 0
                ****************************************
```

***** PSA Architecture Test Suite - Version 0.8 *****

Running.. Crypto Suite

**********

TEST: 201 | DESCRIPTION: Testing psa_crypto_init API: Basic
[Info] Executing tests from non-secure
[Check 1] Test calling crypto functions before psa_crypto_init
[Check 2] Test psa_crypto_init
[Check 3] Test multiple psa_crypto_init
TEST RESULT: PASSED

**********

TEST: 202 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_import_key 16 Byte AES
[Check 2] Test psa_import_key 24 Byte AES
[Check 3] Test psa_import_key 32 Byte AES
[Check 4] Test psa_import_key with DES 64 bit key
[Check 5] Test psa_import_key with Triple DES 2-Key
[Check 6] Test psa_import_key with Triple DES 3-Key
[Check 7] Test psa_import_key with key data greater than the algorithm size
[Check 8] Test psa_import_key with incorrect key data size
[Check 9] Test psa_import_key with incorrect key type
[Check 10] Test psa_import_key with already occupied key slot
[Check 11] Test psa_import_key with invalid key slot
[Check 12] Test psa_import_key with zero key slot
TEST RESULT: PASSED

**********

TEST: 203 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_export_key 16 Byte AES
[Check 2] Test psa_export_key 24 Byte AES
[Check 3] Test psa_export_key 32 Byte AES
[Check 4] Test psa_export_key with DES 64 bit key
[Check 5] Test psa_export_key with Triple DES 2-Key
[Check 6] Test psa_export_key with Triple DES 3-Key
[Check 7] Test psa_export_key with key policy verify
[Check 8] Test psa_export_key with invalid key slot
[Check 9] Test psa_export_key with zero key slot
[Check 10] Test psa_export_key with less buffer size
[Check 11] Test psa_export_key with empty key slot
TEST RESULT: PASSED

```
**************************************
TEST: 204 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_export_public_key 16 Byte AES
[Check 2] Test psa_export_public_key 24 Byte AES
[Check 3] Test psa_export_public_key 32 Byte AES
[Check 4] Test psa_export_public_key with DES 64 bit key
[Check 5] Test psa_export_public_key with Triple DES 2-Key
[Check 6] Test psa_export_public_key with Triple DES 3-Key
TEST RESULT: PASSED
**************************************
TEST: 205 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_destroy_key 16 Byte AES
[Check 2] Test psa_destroy_key 24 Byte AES
[Check 3] Test psa_destroy_key 32 Byte AES
[Check 4] Test psa_destroy_key with DES 64 bit key
[Check 5] Test psa_destroy_key with Triple DES 2-Key
[Check 6] Test psa_destroy_key with Triple DES 3-Key
[Check 7] Test psa_destroy_key with invalid key slot
[Check 8] Test psa_destroy_key with zero key slot
[Check 9] Test psa_destroy_key with empty key slot
TEST RESULT: PASSED
**************************************
TEST: 206 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_get_key_information 16 Byte AES
[Check 2] Test psa_get_key_information 24 Byte AES
[Check 3] Test psa_get_key_information 32 Byte AES
[Check 4] Test psa_get_key_information wit DE¦[Check 5] Test
psa_get_key_information with Triple DES 2-Key
[Check 6] Test psa_get_key_information with Triple DES 3-Key
[Check 7] Test psa_get_key_information with invalid key slot
[Check 8] Test psa_get_key_information with zero key slot
[Check 9] Test psa_get_key_information with empty key slot
TEST RESULT: PASSED
**************************************
TEST: 207 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_set_key_policy 16 Byte AES
[Check 2] Test psa_set_key_policy 24 Byte AES
[Check 3] Test psa_set_key_policy 32 Byte AES
[Check 4] Test psa_set_key_policy with DES 64 bit key
[Check 5] Test psa_set_key_policy with Triple DES 2-Key
[Check 6] Test psa_set_key_policy with Triple DES 3-Key
[Check 7] Test psa_set_key_policy with already occupied key slot
[Check 8] Test psa_set_key_policy with invalid key slot
```

[Check 9] Test psa_set_key_policy with zero key slot
[Check 10] Test psa_set_key_policy with invalid usage
TEST RESULT: PASSED
**********
TEST: 208 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_get_key_policy 16 Byte AES
[Check 2] Test psa_get_key_policy 24 Byte AES
[Check 3] Test psa_get_key_policy 32 Byte AES
[Check 4] Test psa_get_key_policy with DES 64 bit key
[Check 5] Test psa_get_key_policy with Triple DES 2-Key
[Check 6] Test psa_get_key_policy with Triple DES 3-Key
[Check 7] Test psa_get_key_policy with invalid key slot
[Check 8] Test psa_get_key_policy with zero key slot
TEST RESULT: PASSED
**********
TEST: 209 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_set_key_lifetime 16 Byte AES
[Check 2] Test psa_set_key_lifetime with Triple DES 2-Key
[Check 3] Test psa_set_key_lifetime with invalid key slot
[Check 4] Test psa_set_key_lifetime with zero key slot
[Check 5] Test psa_set_key_lifetime with invalid lifetime
TEST RESULT: PASSED
**********
TEST: 210 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_get_key_lifetime 16 Byte AES
[Check 2] Test psa_get_key_lifetime with Triple DES 2-Key
[Check 3] Test psa_get_key_lifetime with invalid key slot
[Check 4] Test psa_get_key_lifetime with zero key slot
TEST RESULT: PASSED
**********
TEST: 211 | DESCRIPTION: Testing crypto hash functions APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_hash_setup with MD2 algorithm
[Check 2] Test psa_hash_setup with MD4 algorithm
[Check 3] Test psa_hash_setup with MD5 algorithm
[Check 4] Test psa_hash_setup with RIPEMD160 algorithm
[Check 5] Test psa_hash_setup with SHA1 algorithm
[Check 6] Test psa_hash_setup with SHA224 algorithm
[Check 7] Test psa_hash_setup with SHA256 algorithm
[Check 8] Test psa_hash_setup with SHA384 algorithm
[Check 9] Test psa_hash_setup with SHA512 algorithm
[Check 10] Test psa_hash_setup with Invalid algorithm
TEST RESULT: PASSED
**********

```
TEST: 212 | DESCRIPTION: Testing crypto hash functions APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_hash_update with MD2 algorithm
[Check 2] Test psa_hash_update with MD4 algorithm
[Check 3] Test psa_hash_update with MD5 algorithm
[Check 4] Test psa_hash_update with RIPEMD160 algorithm
[Check 5] Test psa_hash_update with SHA1 algorithm
[Check 6] Test psa_hash_update with SHA224 algorithm
[Check 7] Test psa_hash_update with SHA256 algorithm
[Check 8] Test psa_hash_update with SHA384 algorithm
[Check 9] Test psa_hash_update with SHA512 algorithm
[Check 10] Test psa_hash_update without hash setup
[Check 11] Test psa_hash_update with completed operation handle
TEST RESULT: PASSED
****************************************
TEST: 213 | DESCRIPTION: Testing crypto hash functions APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_hash_verify with MD2 algorithm
[Check 2] Test psa_hash_verify with MD4 algorithm
[Check 3] Test psa_hash_verify with MD5 algorithm
[Check 4] Test psa_hash_verify with RIPEMD160 algorithm
[Check 5] Test psa_hash_verify with SHA1 algorithm
[Check 6] Test psa_hash_verify with SHA224 algorithm
[Check 7] Test psa_hash_verify with SHA256 algorithm
[Check 8] Test psa_hash_verify with SHA384 algorithm
[Check 9] Test psa_hash_verify with SHA512 algorithm
[Check 10] Test psa_hash_verify with incorrect expected hash
[Check 11] Test psa_hash_verify with incorrect hash length
[Check 12] test psa_hash_verify with inactive & invalid operation handle
TEST RESULT: PASSED
****************************************
TEST: 214 | DESCRIPTION: Testing crypto hash functions APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_hash_finish with MD2 algorithm
[Check 2] Test psa_hash_finish with MD4 algorithm
[Check 3] Test psa_hash_finish with MD5 algorithm
[Check 4] Test psa_hash_finish with RIPEMD160 algorithm
[Check 5] Test psa_hash_finish with SHA1 algorithm
[Check 6] Test psa_hash_finish with SHA224 algorithm
[Check 7] Test psa_hash_finish with SHA256 algorithm
[Check 8] Test psa_hash_finish with SHA384 algorithm
[Check 9] Test psa_hash_finish with SHA512 algorithm
[Check 10] test psa_hash_finish with inactive operation handle
[Check 11] test psa_hash_finish with invalid hash buffer size
TEST RESULT: PASSED
****************************************
TEST: 215 | DESCRIPTION: Testing crypto hash functions APIs
```

[Info] Executing tests from non-secure
[Check 1] Test psa_hash_abort with MD2 algorithm
[Check 2] Test psa_hash_abort with MD4 algorithm
[Check 3] Test psa_hash_abort with MD5 algorithm
[Check 4] Test psa_hash_abort with RIPEMD160 algorithm
[Check 5] Test psa_hash_abort with SHA1 algorithm
[Check 6] Test psa_hash_abort with SHA224 algorithm
[Check 7] Test psa_hash_abort with SHA256 algorithm
[Check 8] Test psa_hash_abort with SHA384 algorithm
[Check 9] Test psa_hash_abort with SHA512 algorithm
[Check 10] Test psa_hash_finish after calling psa_hash_abort
TEST RESULT: PASSED
****************************************

TEST: 223 | DESCRIPTION: Testing crypto key management APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_key_policy_get_usage with usage as encrypt
[Check 2] Test psa_key_policy_get_usage with usage as decrypt
[Check 3] Test psa_key_policy_get_usage with usage as derive
[Check 4] Test psa_key_policy_get_usage with usage as export
[Check 5] Test psa_key_policy_get_usage with usage as sign
[Check 6] Test psa_key_policy_get_usage with usage as verify
TEST RESULT: PASSED
****************************************

TEST: 224 | DESCRIPTION: Testing crypto AEAD APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_aead_encrypt - CCM - 16B AES - 13B nounce & 8B addi data
[Check 2] Test psa_aead_encrypt - AES-CCM
[Check 3] Test psa_aead_encrypt - GCM - 16B AES - 12B Nounce & 12B addi data
[Check 4] Test psa_aead_encrypt - DES Key
[Check 5] Test psa_aead_encrypt - Empty key slot
[Check 6] Test psa_aead_encrypt - Zero key slot
[Check 7] Test psa_aead_encrypt - Invalid key slot
[Check 8] Test psa_aead_encrypt - Unsupported Algorithm
[Check 9] Test psa_aead_encrypt - Invalid key usage
[Check 10] Test psa_aead_encrypt - Small output buffer size
TEST RESULT: PASSED
****************************************

TEST: 225 | DESCRIPTION: Testing crypto AEAD APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_aead_decrypt - CCM - 16B AES - 13B nounce & 8B addi data
[Check 2] Test psa_aead_decrypt - AES-CCM
[Check 3] Test psa_aead_decrypt - GCM - 16B AES - 12B Nounce & 12B addi data
[Check 4] Test psa_aead_decrypt - DES Key

[Check 5] Test psa_aead_decrypt - Empty key slot
[Check 6] Test psa_aead_decrypt - Zero key slot
[Check 7] Test psa_aead_decrypt - Invalid key slot
[Check 8] Test psa_aead_decrypt - Unsupported Algorithm
[Check 9] Test psa_aead_decrypt - Invalid key usage
[Check 10] Test psa_aead_decrypt - Small output buffer size
[Check 11] Test psa_aead_decrypt - Invalid cipher text
[Check 12] Test psa_aead_decrypt - Invalid cipher text size
TEST RESULT: PASSED
**********

TEST: 232 | DESCRIPTION: Testing crypto symmetric cipher APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_cipher_encrypt_setup 16 Byte AES
[Check 2] Test psa_cipher_encrypt_setup 24 Byte AES
[Check 3] Test psa_cipher_encrypt_setup 32 Byte AES
[Check 4] Test psa_cipher_encrypt_setup DES 64 bit key
[Check 5] Test psa_cipher_encrypt_setup Triple DES 2-Key
[Check 6] Test psa_cipher_encrypt_setup Triple DES 3-Key
[Check 7] Test psa_cipher_encrypt_setup 16 Byte raw data
[Check 8] Test psa_cipher_encrypt_setup - not a cipher algorithm
[Check 9] Test psa_cipher_encrypt_setup - unknown cipher algorithm
[Check 10] Test psa_cipher_encrypt_setup - incompatible key ARC4
[Check 11] Test psa_cipher_encrypt_setup - invalid key slot
[Check 12] Test psa_cipher_encrypt_setup - empty key slot
[Check 13] Test psa_cipher_encrypt_setup - zero as key slot
[Check 14] Test psa_cipher_encrypt_setup - incorrect usage
TEST RESULT: PASSED
**********

TEST: 233 | DESCRIPTION: Testing crypto symmetric cipher APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_cipher_decrypt_setup 16 Byte AES
[Check 2] Test psa_cipher_decrypt_setup 24 Byte AES
[Check 3] Test psa_cipher_decrypt_setup 32 Byte AES
[Check 4] Test psa_cipher_decrypt_setup DES 64 bit key
[Check 5] Test psa_cipher_decrypt_setup Triple DES 2-Key
[Check 6] Test psa_cipher_decrypt_setup Triple DES 3-Key
[Check 7] Test psa_cipher_decrypt_setup 16 Byte raw data
[Check 8] Test psa_cipher_decrypt_setup - not a cipher algorithm
[Check 9] Test psa_cipher_decrypt_setup - unknown cipher algorithm
[Check 10] Test psa_cipher_decrypt_setup - incompatible key ARC4
[Check 11] Test psa_cipher_decrypt_setup - invalid key slot
[Check 12] Test psa_cipher_decrypt_setup - empty key slot
[Check 13] Test psa_cipher_decrypt_setup - zero as key slot
[Check 14] Test psa_cipher_decrypt_setup - incorrect usage
TEST RESULT: PASSED
**********

TEST: 235 | DESCRIPTION: Testing crypto symmetric cipher APIs

[Info] Executing tests from non-secure
[Check 1] Test psa_cipher_set_iv 16 Byte AES
[Check 2] Test psa_cipher_set_iv 24 Byte AES
[Check 3] Test psa_cipher_set_iv 32 Byte AES
[Check 4] Test psa_cipher_set_iv DES 64 bit key
[Check 5] Test psa_cipher_set_iv Triple DES 2-Key
[Check 6] Test psa_cipher_set_iv Triple DES 3-Key
[Check 7] Test psa_cipher_set_iv AES - small iv buffer
[Check 8] Test psa_cipher_set_iv DES - small iv buffer
[Check 9] Test psa_cipher_set_iv AES - large iv buffer
[Check 10] Test psa_cipher_set_iv DES - large iv buffer
TEST RESULT: PASSED
**********

TEST: 236 | DESCRIPTION: Testing crypto symmetric cipher APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_cipher_update - Encrypt - AES CBC_NO_PADDING
[Check 2] Test psa_cipher_update - Encrypt - AES CBC_NO_PADDING (Short input)
[Check 3] Test psa_cipher_update - Encrypt - AES CBC_PKCS7
[Check 4] Test psa_cipher_update - Encrypt - AES CBC_PKCS7 (Short input)
[Check 5] Test psa_cipher_update - Encrypt - AES CTR
[Check 6] Test psa_cipher_update - Encrypt - DES CBC (nopad)
[Check 7] Test psa_cipher_update - Encrypt - 2-key 3DE -CBC (nopad)
[Check 8] Test psa_cipher_update - Encrypt - 3-key 3DE -CBC (nopad)
[Check 9] Test psa_cipher_update - small output buffer size
[Check 10] Test psa_cipher_update - Decrypt - AES CBC_NO_PADDING
[Check 11] Test psa_cipher_update - Decrypt - AES CBC_NO_PADDING (Short input)
[Check 12] Test psa_cipher_update - Decrypt - AES CBC_PKCS7
[Check 13] Test psa_cipher_update - Decrypt - AES CBC_PKCS7 (Short input)
[Check 14] Test psa_cipher_update - Decrypt - AES CTR
[Check 15] Test psa_cipher_update - Decrypt - DES CBC (nopad)
[Check 16] Test psa_cipher_update - Decrypt - 2-key 3DE -CBC (nopad)
[Check 17] Test psa_cipher_update - Decrypt - 3-key 3DE -CBC (nopad)
TEST RESULT: PASSED
**********

TEST: 237 | DESCRIPTION: Testing crypto symmetric cipher APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_cipher_finish - Encrypt - AES CBC_NO_PADDING
[Check 2] Test psa_cipher_finish - Encrypt - AES CBC_NO_PADDING (Short input)
[Check 3] Test psa_cipher_finish - Encrypt - AES CBC_PKCS7
[Check 4] Test psa_cipher_finish - Encrypt - AES CBC_PKCS7 (Short input)
[Check 5] Test psa_cipher_finish - Encrypt - AES CTR
[Check 6] Test psa_cipher_finish - Encrypt - AES CTR (short input)

[Check 7] Test psa_cipher_finish - Encrypt - DES CBC (nopad)
[Check 8] Test psa_cipher_finish - Encrypt - 2-key 3DE -CBC (nopad)
[Check 9] Test psa_cipher_finish - Encrypt - 3-key 3DE -CBC (nopad)
[Check 10] Test psa_cipher_finish - small output buffer size
[Check 11] Test psa_cipher_finish - Decrypt - AES CBC_NO_PADDING
[Check 12] Test psa_cipher_finish - Decrypt - AES CBC_NO_PADDING (Short input)
[Check 13] Test psa_cipher_finish - Decrypt - AES CBC_PKCS7
[Check 14] Test psa_cipher_finish - Decrypt - AES CBC_PKCS7 (Short input)
[Check 15] Test psa_cipher_finish - Decrypt - AES CTR
[Check 16] Test psa_cipher_finish - Decrypt - AES CTR (short input)
[Check 17] Test psa_cipher_finish - Decrypt - DES CBC (nopad)
[Check 18] Test psa_cipher_finish - Decrypt - 2-key 3DE -CBC (nopad)
[Check 19] Test psa_cipher_finish - 3-key 3DE -CBC (nopad)
TEST RESULT: PASSED
**********
TEST: 238 | DESCRIPTION: Testing crypto symmetric cipher APIs
[Info] Executing tests from non-secure
[Check 1] Test psa_cipher_abort - Encrypt - AES CBC_NO_PADDING
[Check 2] Test psa_cipher_abort - Encrypt - AES CBC_PKCS7
[Check 3] Test psa_cipher_abort - Encrypt - AES CTR
[Check 4] Test psa_cipher_abort - Encrypt - DES CBC (nopad)
[Check 5] Test psa_cipher_abort - Encrypt - 2-key 3DE -CBC (nopad)
[Check 6] Test psa_cipher_abort - Encrypt - 3-key 3DE -CBC (nopad)
[Check 7] Test psa_cipher_abort - Decrypt - AES CBC_NO_PADDING
[Check 8] Test psa_cipher_abort - Decrypt - AES CBC_PKCS7
[Check 9] Test psa_cipher_abort - Decrypt - AES CTR
[Check 10] Test psa_cipher_abort - Decrypt - DES CBC (nopad)
[Check 11] Test psa_cipher_abort - Decrypt - 2-key 3DE -CBC (nopad)
[Check 12] Test psa_cipher_abort - Decrypt - 3-key 3DE -CBC (nopad)
[Check 1] Test psa_cipher_update after psa_cipher_abort should fail
TEST RESULT: PASSED
**********
********** Crypto Suite Report **********
TOTAL TESTS : 24
TOTAL PASSED : 24
TOTAL SIM ERROR : 0
TOTAL FAILED : 0
TOTAL SKIPPED : 0
**********

***** PSA Architecture Test Suite - Version 0.8 *****

Running.. Protected Storage Suite
**********
TEST: 401 | DESCRIPTION: UID not found check
[Info] Executing tests from non-secure
[Check 1] Call get API for UID 6 which is not set
[Check 2] Call get_info API for UID 6 which is not set
[Check 3] Call remove API for UID 6 which is not set
[Check 4] Call get API for UID 6 which is removed
[Check 5] Call get_info API for UID 6 which is removed
[Check 6] Call remove API for UID 6 which is removed
Set storage for UID 6
[Check 7] Call get API for different UID 6
[Check 8] Call get_info API for different UID 6
[Check 9] Call remove API for different UID 6
TEST RESULT: PASSED
**********
TEST: 402 | DESCRIPTION: Write once error check
[Info] Executing tests from non-secure
[Check 1] Update the flag of UID 1 with WRITE_ONCE flag
[Check 2] Try to remove the UID 1 having WRITE_ONCE flag
[Check 3] Create a new UID 2 with WRITE_ONCE flag
[Check 4] Try to remove the UID 2 having WRITE_ONCE flag
[Check 5] Try to change the length of write_once UID 2
[Check 6] Check UID removal still fails
[Check 7] Try to change the WRITE_ONCE flag to None for UID 2
[Check 8] Check UID removal still fails
TEST RESULT: PASSED
**********
TEST: 403 | DESCRIPTION: Insufficient space check
[Info] Executing tests from non-secure
[Check 1] Overload storage space
Remove all registered UIDs
[Check 2] Overload storage again to verify all previous UID removed
Remove all registered UIDs
TEST RESULT: PASSED
**********
TEST: 404 | DESCRIPTION: Data Consistency check

```
[Info] Executing tests from non-secure
[Check 1] Call get API with incorrect length
[Check 2] Old buffer invalid after length change
TEST RESULT: PASSED
*****************************************
TEST: 405 | DESCRIPTION: Success scenarios check
[Info] Executing tests from non-secure
[Check 1] Set UID with data length zero and call storage
APIs
[Check 2] Resetting the length check
TEST RESULT: PASSED
*****************************************
TEST: 406 | DESCRIPTION: Flags not supported check
[Info] Executing tests from non-secure
[Check 1] Call set API with valid flag values
TEST RESULT: PASSED
*****************************************
TEST: 407 | DESCRIPTION: Incorrect Size check
[Info] Executing tests from non-secure
Create a valid Storage
Increase the length of storage
[Check 1] Call get API with old length
Decrease the length of storage
[Check 2] Call get API with old length
[Check 3] Call get API with valid length
TEST RESULT: PASSED
*****************************************
TEST: 408 | DESCRIPTION: Invalid offset check
[Info] Executing tests from non-secure
[Check 1] Try to access data with varying valid offset
[Check 2] Try to access data with varying invalid offset
TEST RESULT: PASSED
*****************************************
TEST: 409 | DESCRIPTION: Invalid Arguments check
[Info] Executing tests from non-secure
[Check 1] Call set API with NULL pointer and data length 0
[Check 2] Create UID with zero data length
[Check 3] Try to set NULL buffer for existing UID
[Check 4] Call get API with NULL read buffer and data
length 0
[Check 5] Increase the length
TEST RESULT: PASSED
*****************************************
TEST: 410 | DESCRIPTION: UID value zero check
[Info] Executing tests from non-secure
[Check 1] Creating storage with UID 0 should fail
```

```
TEST RESULT: PASSED
******************************************
TEST: 411 | DESCRIPTION: Optional APIs: UID not found
check
[Info] Executing tests from non-secure
Test Case skipped as Optional PS APIs are not supported.
TEST RESULT: SKIPPED (Skip Code=0x2B)
******************************************
TEST: 412 | DESCRIPTION: Optional APIs: Invalid
arguments and offset invalid
[Info] Executing tests from non-secure
Test Case skipped as Optional PS APIs are not supported.
TEST RESULT: SKIPPED (Skip Code=0x2B)
******************************************
TEST: 413 | DESCRIPTION: Set_Extended and Create api :
Success
[Info] Executing tests from non-secure
Test Case skipped as Optional PS APIs are not supported.
TEST RESULT: SKIPPED (Skip Code=0x2B)
******************************************
TEST: 414 | DESCRIPTION: Optional APIs not supported
check
[Info] Executing tests from non-secure
Optional PS APIs are not supported.
[Check 1] Call to create API should fail as API not
supported
[Check 2] Create valid storage with set API
[Check 3] Call to set_extended API call should fail
[Check 4] Verify data is unchanged
TEST RESULT: PASSED
******************************************
TEST: 415 | DESCRIPTION: Create API write_once flag value
check
[Info] Executing tests from non-secure
Test Case skipped as Optional PS APIs are not supported.
TEST RESULT: SKIPPED (Skip Code=0x2B)
******************************************
********** Protected Storage Suite Report **********
TOTAL TESTS : 15
TOTAL PASSED : 11
TOTAL SIM ERROR : 0
TOTAL FAILED : 0
TOTAL SKIPPED : 4
******************************************
```