AUTOSAR CSM User Manual

Date (YYYY-MM-DD)	Ver.	Editor	Content (before revision-> after revision)
2021-01-15	1.0.0.0	JaeHyun Lim	Initial Version
2021-03-10	1.0.1.0	TamTV6	Update change log
2021-11-12	1.0.2.0	TamTV6	Appyling change of company name Update change log
2022-07-01	1.0.2.1	DienTC1	Clarify Copyright and update wrong company name
2022-07-20	1.0.2.2	DienTC1	Fix TM 100% coverage
2022-08-23	1.0.3.0	DienTC1	Fix UNECE security coding violations
2022-10-18	1.0.4.0	DienTC1	Allow zero length message for Hash. Support HANDLE-TERMINATION-AND-RESTART configured in Swcd_Bsw_Csm.arxml. Fix Swcdt_Bsw_Csm.template's mistake which causes Async Hash Rte generation fail.
2023-03-02	1.0.5.0	PhuocLH9	Add definition in Csm header.
2023-04-24	1.0.6.0	DienTC1	Update Change log.
2023-07-28	1.0.7.0	DienTC1	Update Change log.
2023-09-05	1.0.8.0	PhuocLH9	Support CUSTOM primitive for algorithm family, algorithm mode and algorithm secondary family following Autosar R22-11. Add user header include file in PDF. Correct wrong spelling
2023-12-29	1.0.9.0	YoungHyun Eum	Update Change log
2024-02-08	1.0.10.0	PhuocLH9	Update Change log
2024-02-28	1.0.11.0	PhuocLH9	Update Change log
2024-03-15	1.0.5.0	CH Lee	Change fonts Change chapter indexes Remove watermark Added description to tables on Chapter 5,6 (configuration guide & functions) Remove change log & Change Chapter 4 title

Table of Contents

1. Overview

This document provides caution or reference when using AUTOSAR platform for CSM use, when setting parameters or designing system. Please refer to the Reference document for details.
The interpretation of the category related to setting is as follows.

- Changeable (C): Items that can be set by the user
- Fixed (F): Items that cannot be changed by the user
- Not Supported (N): Not used

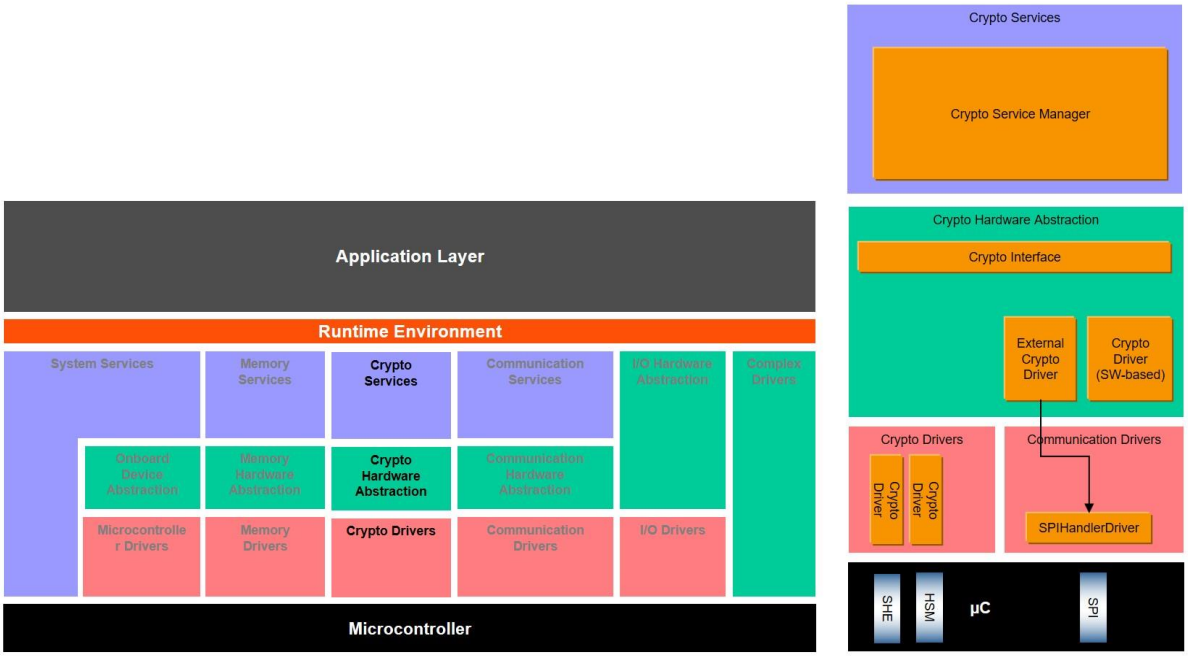
2. Reference

Sl. No.	Title	Version
1	AUTOSAR_SWS_CryptoServiceManager.pdf	4.4.0
2	AUTOSAR_SWS_CryptoInterface.pdf	4.4.0
3	AUTOSAR_SWS_CryptoDriver.pdf	4.4.0
4	AUTOSAR_SWS_DefaultErrorTracer.pdf	4.4.0

3. AUTOSAR System

3.1 Overview of Software Layers

The CSM-related layered architecture of the AUTOSAR platform is as follows.



3.2 AUTOSAR Crypto Stack

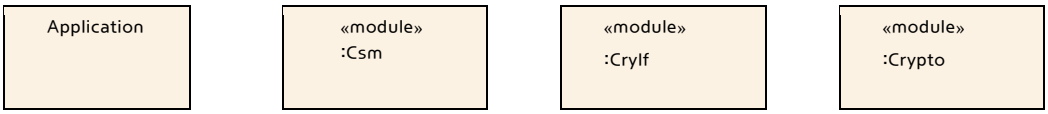
CSM is a service that provides cryptography functionality, based on a crypto driver which relies on a software library or on a hardware module. Also, mixed setups with multiple crypto drivers are possible. The CSM accesses the different CryptoDrivers over the CryIf.

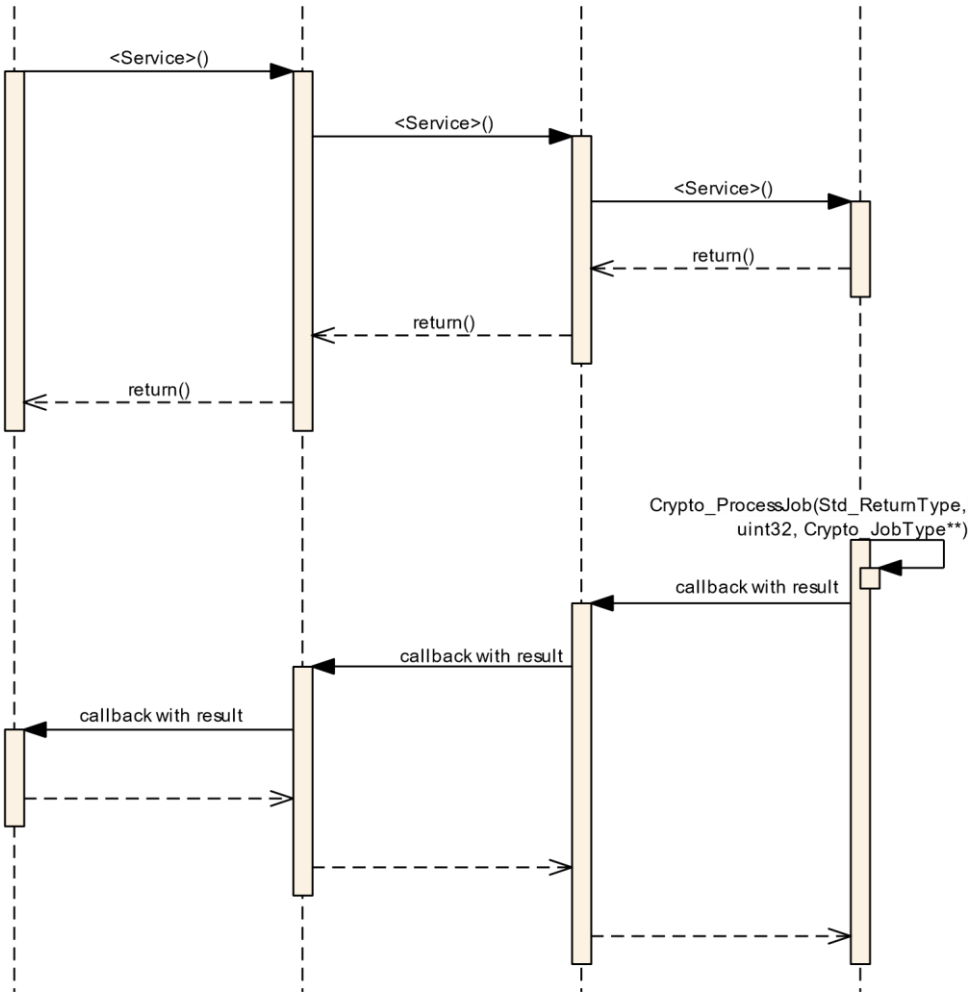
3.2.1 Sequence Diagrams

The following sequence diagrams concentrate on the interaction between the CSM module and software components respectively the ECU state manager.

3.2.1.1 Asynchronous Calls

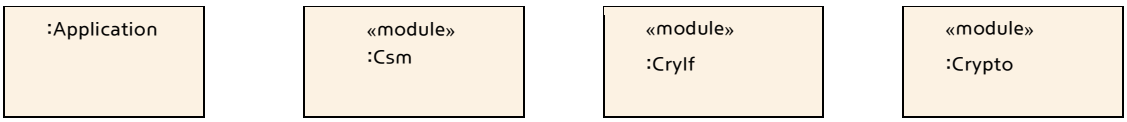
The following diagram (Sequence diagram for asynchronous call) shows a sample sequence of function calls for a request performed asynchronously. The result of the asynchronous function can be accessed after an asynchronous notification (invocation of the configured callback function).

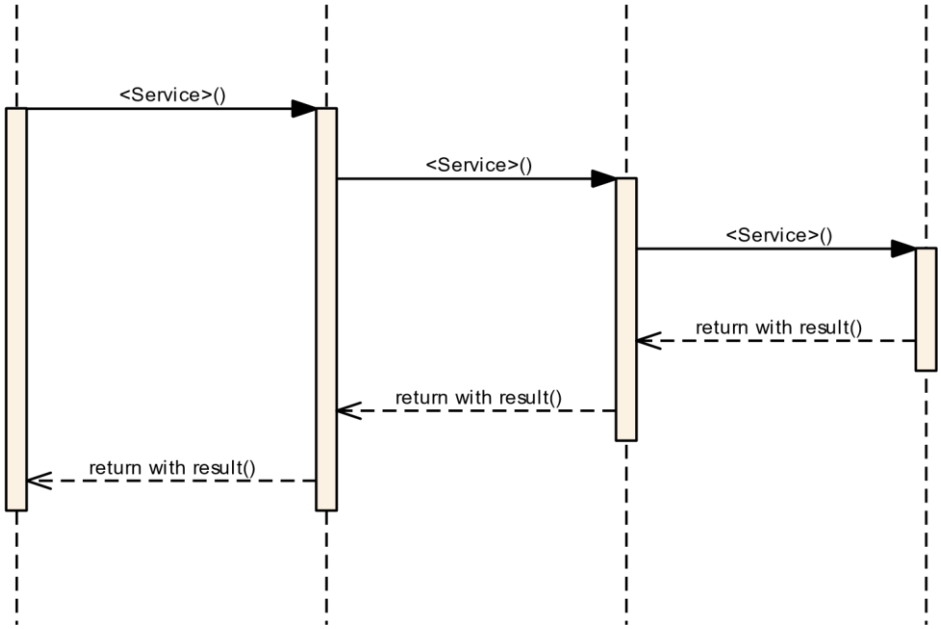




3.2.1.2 Synchronous Calls

The following diagram (Sequence diagram for synchronous calls) shows a sample sequence of function calls with the scheduler for a request performed synchronously.




<Service>()
<Service>()
<Service>()
return with result()
return with result()
return with result()

4. Limitations and Deviations

4.1 Limitations

- Some type definitions of CSM start with the Prefix "CRYPTO_" which will violate SRS_BSW_00305. This will be harmonized in release 4.3.1. Nevertheless due to the constraint [constr_1050] part 1 the ports are still consider to be compatible.

- CRYPTO_ALGOFAM_DRBG, CRYPTO_ALGOFAM_FIPS186, CRYPTO_ALGOFAM_PADDING_PKCS7, CRYPTO_ALGOFAM_PADDING_ONEWITHZEROS in Crypto_AlgorithmFamilyType is not supported by algorithm family configuration. Unsupported configuration can be replaced by CRYPTO_ALGOFAM_CUSTOM.

- CRYPTO_ALGOFAM_3DES, CRYPTO_ALGOFAM_AES, CRYPTO_ALGOFAM_ CHACHA, CRYPTO_ALGOFAM_RSA, CRYPTO_ALGOFAM_ED25519, CRYPTO_ALGOFAM_ BRAINPOOL, CRYPTO_ALGOFAM_ECCNIST, CRYPTO_ALGOFAM_RNG, CRYPTO_ALGOFAM_SIPHASH, CRYPTO_ALGOFAM_ECCANSI, CRYPTO_ALGOFAM_ ECCSEC, CRYPTO_ALGOFAM_DRBG, CRYPTO_ALGOFAM_FIPS186, CRYPTO_ALGOFAM_ PADDING_PKCS7, CRYPTO_ALGOFAM_PADDING_ONEWITHZEROS, CRYPTO_ALGOFAM_PBKDF2, CRYPTO_ALGOFAM_KDFX963, CRYPTO_ALGOFAM_DH in Crypto_AlgorithmFamilyType is not supported by secondary algorithm family configuration. Unsupported configuration can be replaced by CRYPTO_ALGOFAM_CUSTOM.

- CRYPTO_ALGOMODE_PXXXR1 in Crypto_AlgorithmModeType is not supported by algorithm mode configuration. Unsupported configuration can be replaced by CRYPTO_ALGOMODE_CUSTOM.

4.2 Deviation

- Csm_DataPtr should be replaced by Csm_KeyDataType_{Crypto} in SWS_Csm_01905 client server interface of KeyExchangeCalcPubVal, KeyExchangeCalcSecret, RandomSeed.

- Not support Csm_CertificateParse and Csm_CertificateVerify.

- Job processing order in case there are more than 1 queue are configured:

  In case there are 3 configured Csm Queues:
  Queue_0 contains 3 jobs: {job_1(priority 1), job_2(priority 2), job_3(priority 3)}
  Queue_1 contains 2 jobs: {job_5(priority 5), job_6(priority 6)}
  Queue_2 contains 1 job: {job_7(priority 7)}
  When Csm_MainFunction is called.
  1st call of Mainfunction:
  [Queue_0]job_3 > [Queue_1]job_6 > [Queue_2]job_7
  2nd call of Mainfunction:
  > [Queue_0]job_2 > [Queue_1]job_5

3rd call of Mainfunction:
> [Queue_0]job_1.

- AUTOSAR 4.4.0 and upper version do not including the X448 (KeyExchange) and ED448 (EDDSA) Primitive. So, our Csm use this ED448 Primitive similar to ED25519 and ECCNIST.
  * CRYPTO_ALGOFAM_ED448 : **0xEE**, CRYPTO_ALGOFAM_X448 : **0xEF**

5. Configuration Guide

5.1 Csm module

5.1.1 CsmGeneral

Container for common configuration options.

Parameter Name	Value	Category
CsmDevErrorDetect	true / false / unset	C
CsmMainFunctionPeriod	0..Inf	C
CsmVersionInfoApi	true / false / unset	C
CsmUseDeprecated	true / false / unset	C
CsmAsymPrivateKeyMaxLength	1..4294967295	C
CsmAsymPublicKeyMaxLength	1..4294967295	C
CsmUserIncludeFiles	User Defined	C
CsmInputDataNoValidation	true / false / unset	C

1) CsmDevErrorDetect
- Switches the development error detection and notification on or off.
  - true: detection and notification is enabled.
  - false: detection and notification is disabled.

2) CsmMainFunctionPeriod
- Specifies the period of main function Csm_MainFunction in seconds.

3) CsmVersionInfoApi
- Pre-processor switch to enable and disable availability of the API Csm_GetVersionInfo().
  - true: API Csm_GetVersionInfo() is available.
  - false: API Csm_GetVersionInfo() is not available.

4) CsmUseDeprecated
- Decides if the deprecated interfaces shall be used (Backwards combatibility).
  - true: use deprecated interfaces.
  - false: use normal interfaces.

5) CsmAsymPrivateKeyMaxLength
- Maximum length in bytes of an asymmetric public key for all algorithm.

6) CsmAsymPublicKeyMaxLength
- Maximum length in bytes of an asymmetric key for all algorithm.

7) CsmUserIncludeFiles
- Name of the header file(s) to be included by the Csm module like header of user custom algorithms. (Ex: User_header) The inclusion syntax is inserted in generated code(Csm_Cfg.h).

8) CsmInputDataNoValidation
- Pre-processor switch to enable and disable validation of input data on API Csm_PrvValidateInputPtr(). This does not meet the AUTOSAR Spec.
  - true: Validation of input data is disable.
    - A. If input data is NULL_PTR and data length is 0, function return E_OK.
    - B. If input data is NULL_PTR and data length is not 0, function return E_NOT_OK.
  - false: Validation of input data is enable.

5.1.2 CsmJobs

Container for configuration of CSM jobs.

Parameter Name	Value	Category
CsmJobId	0.. 4294967295	C
CsmJobInterfaceUsePort	CRYPTO_USE_FNC / CRYPTO_USE_PORT / CRYPTO_USE_PORT_OPTIMIZED	C
CsmJobPrimitiveCallbackUpdateNotification	true / false	C
CsmJobPriority	0..4294967295	C
CsmProcessingMode	CRYPTO_PROCESSING_ASYNC / CRYPTO_PROCESSING_SYNC	C
CsmInOutRedirectionRef	CsmInOutRedirection	C
CsmJobKeyRef	CsmKey	C
CsmJobPrimitiveCallbackRef	CsmCallback	C
CsmJobPrimitiveRef	CsmAEADDecrypt / CsmAEADEncrypt / CsmDecrypt / CsmEncrypt / CsmHash / CsmJobKeyDerive / CsmJobKeyExchangeCalcPubVal / CsmJobKeyExchangeCalcSecret / CsmJobKeyGenerate / CsmJobKeySetValid / CsmJobRandomSeed / CsmMacGenerate / CsmMacVerify / CsmRandomGenerate / CsmSignatureGenerate / CsmSignatureVerify	C
CsmJobQueueRef	CsmQueue	C

1) CsmJobId

- Identifier of the CSM job. The set of actually configured identifiers shall be consecutive and gapless.

2) CsmJobInterfaceUsePort

- Enable(CRYPTO_USE_PORT) or disable(CRYPTO_USE_FNC) RTE interfaces for job.
  - CRYPTO_USE_FNC: Port is not used.
  - CRYPTO_USE_PORT: Port is used.
  - CRYPTO_USE_PORT_OPTIMIZED: DATA_REFERENCE is used. If the CRYPTO_OPERATIONMODE_FINISH bit is set in job-> jobPrimitiveInputOutput.mode and CsmProcessingMode is set to CRYPTO_PROCESSING_ASYNC, the CSM shall trigger CallbackNotification service.

3) CsmJobPrimitiveCallbackUpdateNotification

- This parameter indicates, whether the callback function shall be called, if the UPDATE operation has been finished.
  - true: If the CRYPTO_OPERATIONMODE_UPDATE bit is set in job->jobPrimitiveInputOutput.mode and the corresponding, the Csm_CallbackNotification shall call the configured callback function.
  - false: Csm_CallbackNotification is not call the callback function.

4) CsmJobPriority
- Priority of the job. The higher the value, the higher the job's priority.

5) CsmProcessingMode
- Determines how the interface shall be used for that job.
  - CRYPTO_PROCESSING_ASYNC: The job is processed as asynchronos. It returns without processing the job.
  - CRYPTO_PROCESSING_SYNC: The job is processed as synchronous. It returns with the result. The caller will be notified by the corresponding callback.

6) CsmInOutRedirectionRef
- The input and/or output data of a job can be re-directed to a key element. Which input and output value to which key and its key element is re-directed shall be statically configured at compile time and shall not be changed at runtime.
  The structure Crypto_JobRedirectionInfoType contains information which key elements shall be used for redirection. Refers 5.1.6 for a description of the parameters of CsmInOutRedirection structures.

7) CsmJobKeyRef
- This parameter refers to the key which shall be used for the CsmPrimitive. It's possible to use a CsmKey for different jobs.

8) CsmJobPrimitiveCallbackRef
- This parameter refers to the used CsmCallback.

9) CsmJobPrimitiveRef
- This parameter refers to the used CsmPrimitive. Different jobs may refer to one CsmPrimitive. The referred CsmPrimitive provides detailed information on the actual cryptographic routine.

10) CsmJobQueueRef
- This parameter refers to the queue. The queue is used if the underlying crypto driver object is busy. The queue refers also to the channel which is used.

5.1.3 CsmKeys

Container for CSM key configurations.

Parameter Name	Value	Category
CsmKeyId	0.. 4294967295	C
CsmKeyUsePort	true / false	C
CsmKeyRef	CryIfKey	C

1) CsmKeyId
- Identifier of the CsmKey. The set of actually configured identifiers shall be consecutive and gapless.

2) CsmKeyUsePort
- Enable or disable to use RTE interface for this key.
  - true: RTE interfaces used for this key
  - false: No RTE interfaces used for this key<

3) CsmKeyRef

- This parameter refers to the used CryIfKey. The underlying CryIfKey refers to a specific CryptoKey in the Crypto Driver.

5.1.4 CsmPrimitives

It shall be possible to create several configurations for each cryptographic primitive. One configuration per job per primitive is possible.

5.1.4.1 CsmHash

Container for Hash Configurations. The container name serves as a symbolic name for the identifier of a key configuration.

Parameter Name	Value	Category
CsmHashAlgorithmFamily	CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 / CRYPTO_ALGOFAM_BLAKE_2s_256 / CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 / CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256	C

Parameter Name	Value	Category
CsmHashAlgorithmFamilyCustom	User Defined	C
CsmHashAlgorithmMode	CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_NOT_SET	C
CsmHashAlgorithmModeCustom	User Defined	C
CsmHashAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_NOT_SET	C
CsmHashAlgorithmSecondaryFamilyCustom	User Defined	C
CsmHashDataMaxLength	1..4294967295	C
CsmHashResultLength	1.. 4294967295	C

1) CsmHashAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmHashAlgorithmFamilyCustom
- This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmHashAlgorithmFamily.

3) CsmHashAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

4) CsmHashAlgorithmModeCustom
- Name of the custom primitive mode.

5) CsmHashAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

6) CsmHashAlgorithmSecondaryFamilyCustom
- This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmHashAlgorithmSecondaryFamily.

7) CsmHashDataMaxLength
- Max size of the input data length in bytes

8) CsmHashResultLength
- Size of the output hash length in bytes

5.1.4.2 CsmMacGenerate

Configurations of MacGenerate primitives. The container name serves as a symbolic name for the identifier of a MAC generation interface.

Parameter Name	Value	Category
CsmMacGenerateAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 /	C

Parameter Name	Value	Category
	CRYPTO_ALGOFAM_BLAKE_2s_256 / CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_RNG / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 / CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256 / CRYPTO_ALGOFAM_SIPHASH	
CsmMacGenerateAlgorithmFamilyCustom	User Defined	C
CsmMacGenerateAlgorithmKeyLength	1.. 4294967295	C
CsmMacGenerateAlgorithmMode	CRYPTO_ALGOMODE_CMAC / CRYPTO_ALGOMODE_CTRDRBG / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_GMAC / CRYPTO_ALGOMODE_HMAC / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_SIPHASH_2_4 / CRYPTO_ALGOMODE_SIPHASH_4_8	C
CsmMacGenerateAlgorithmModeCustom	User Defined	C
CsmMacGenerateAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOMODE_CUSTOM	C
CsmMacGenerateAlgorithmSecondaryFamilyCustom	User Defined	C
CsmMacGenerateDataMaxLength	1.. 4294967295	C
CsmMacGenerateResultLength	1.. 4294967295	C

1) CsmMacGenerateAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmMacGenerateAlgorithmFamilyCustom
- This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmMacGenerateAlgorithmFamily

3) CsmMacGenerateAlgorithmKeyLength
- Size of the MAC key in bytes

4) CsmMacGenerateAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

5) CsmMacGenerateAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service, if CRYPTO_ALGOMODE_CUSTOM is set as CsmMacGenerateAlgorithmMode.

6) CsmMacGenerateAlgorithmSecondaryFamily
- Determines the secondary algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

7) CsmMacGenerateAlgorithmSecondaryFamilyCustom
- This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmHashAlgorithmSecondaryFamilyCustom.

8) CsmMacGenerateDataMaxLength
- Max size of the input data length in bytes

9) CsmMacGenerateResultLength
- Size of the output MAC length in bytes

5.1.4.3 CsmMacVerify

Configurations of MacVerify primitives. The container name serves as a symbolic name for the identifier of a MAC generation interface.

Parameter Name	Value	Category
CsmMacVerifyAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 / CRYPTO_ALGOFAM_BLAKE_2s_256 / CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_RNG / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 / CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256 / CRYPTO_ALGOFAM_SIPHASH	C
CsmMacVerifyAlgorithmFamilyCustom	User Defined	C
CsmMacVerifyAlgorithmKeyLength	1..4294967295	C
CsmMacVerifyAlgorithmMode	CRYPTO_ALGOMODE_CMAC / CRYPTO_ALGOMODE_CTRDRBG / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_GMAC / CRYPTO_ALGOMODE_HMAC / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_SIPHASH_2_4 / CRYPTO_ALGOMODE_SIPHASH_4_8	C

Parameter Name	Value	Category
CsmMacVerifyAlgorithmModeCustom	User Defined	C
CsmMacVerifyAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOMODE_CUSTOM	C
CsmMacVerifyAlgorithmSecondaryFamilyCustom	User Defined	C
CsmMacVerifyCompareLength	1..4294967295	C
CsmMacVerifyDataMaxLength	1..4294967295	C

1) CsmMacVerifyAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmMacVerifyAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service.

3) CsmMacVerifyAlgorithmKeyLength
- Size of the MAC key in bytes

4) CsmMacVerifyAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

5) CsmMacVerifyAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

6) CsmMacVerifyAlgorithmSecondaryFamily
- Determines the secondary algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

7) CsmMacVerifyAlgorithmSecondaryFamilyCustom
- This is the second the name of the custom algorithm, if CRYPTO_ALGOFAM_CUSTOM is set as CsmMacVerifyAlgorithmSecondaryFamily.

8) CsmMacVerifyCompareLength.
- Size of the input MAC length, that shall be verified, in BITS.

9) CsmMacVerifyDataMaxLength
- Max size of the input data length, for whichs MAC shall be verified, in bytes.

5.1.4.4 CsmEncrypt

Configurations of Encryption primitives. The container name serves as a symbolic name for the identifier of an encryption interface.

Parameter Name	Value	Category
CsmEncryptAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_ECIES / CRYPTO_ALGOFAM_RSA	C
CsmEncryptAlgorithmFamilyCustom	User Defined	C
CsmEncryptAlgorithmKeyLength	1..4294967295	C
CsmEncryptAlgorithmMode	CRYPTO_ALGOMODE_12ROUNDS /	C

Parameter Name	Value	Category
	CRYPTO_ALGOMODE_20ROUNDS / CRYPTO_ALGOMODE_8ROUNDS / CRYPTO_ALGOMODE_CBC / CRYPTO_ALGOMODE_CFB / CRYPTO_ALGOMODE_CTR / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_ECB / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_OFB / CRYPTO_ALGOMODE_RSAES_OAEP / CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5 / CRYPTO_ALGOMODE_XTS	
CsmEncryptAlgorithmModeCustom	User Defined	C
CsmEncryptAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_NOT_SET	C
CsmEncryptAlgorithmSecondaryFamily Custom	User Defined	C
CsmEncryptDataMaxLength	1..4294967295	C
CsmEncryptResultMaxLength	1..4294967295	C

1) CsmEncryptAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmEncryptAlgorithmFamilyCustom
- This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmEncryptAlgorithmFamily.

3) CsmEncryptAlgorithmKeyLength
- Size of the encryption key in bytes

4) CsmEncryptAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

5) CsmEncryptAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

6) CsmEncryptAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

7) CsmEncryptAlgorithmSecondaryFamilyCustom.
- Name of the custom secondary algorithm family used for the crypto service

8) CsmEncryptDataMaxLength
- Max size of the input plaintext length in bytes

9) CsmEncryptResultMaxLength
- Max size of the output cipher length in bytes

5.1.4.5 CsmDecrypt

Configurations of Decryption primitives. The container name serves as a symbolic name for the identifier of an decryption interface.

Parameter Name	Value	Category
CsmDecryptAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_ECIES / CRYPTO_ALGOFAM_RSA	C
CsmDecryptAlgorithmFamilyCustom	User Defined	C
CsmDecryptAlgorithmKeyLength	1..4294967295	C
CsmDecryptAlgorithmMode	CRYPTO_ALGOMODE_12ROUNDS / CRYPTO_ALGOMODE_20ROUNDS / CRYPTO_ALGOMODE_8ROUNDS / CRYPTO_ALGOMODE_CBC / CRYPTO_ALGOMODE_CFB / CRYPTO_ALGOMODE_CTR / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_ECB / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_OFB / CRYPTO_ALGOMODE_RSAES_OAEP / CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5 / CRYPTO_ALGOMODE_XTS	C
CsmDecryptAlgorithmModeCustom	User Defined	C
CsmDecryptAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_NOT_SET	C
CsmDecryptAlgorithmSecondaryFamily Custom	User Defined	C
CsmDecryptDataMaxLength	1..4294967295	C
CsmDecryptResultMaxLength	1..4294967295	C

1) CsmDecryptAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmDecryptAlgorithmFamilyCustom
- Name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmDecryptAlgorithmFamily.

3) CsmDecryptAlgorithmKeyLength
- Size of the encryption key in bytes

4) CsmDecryptAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

5) CsmDecryptAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

6) CsmDecryptAlgorithmSecondaryFamily
- Determines the secondary algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

7) CsmDecryptAlgorithmSecondaryFamilyCustom
- Name of the custom secondary algorithm family used for the crypto service.

8) CsmDecryptDataMaxLength
- Max size of the input ciphertext length in bytes

9) CsmDecryptResultMaxLength
- Max size of the output plaintext length in bytes

5.1.4.6 CsmAEADEncrypt

Configuration of AEAD encryption primitives. The container name serves as a symbolic name for the identifier of an AEAD encryption interface.

Parameter Name	Value	Category
CsmAEADEncryptAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_EEA3	C
CsmAEADEncryptAlgorithmFamilyCustom	User Defined	C
CsmAEADEncryptAlgorithmKeyLength	1..4294967295	C
CsmAEADEncryptAlgorithmMode	CRYPTO_ALGOMODE_20ROUNDS / CRYPTO_ALGOMODE_12ROUNDS / CRYPTO_ALGOMODE_8ROUNDS / CRYPTO_ALGOMODE_GCM / CRYPTO_ALGOMODE_CUSTOM	C
CsmAEADEncryptAlgorithmModeCustom	User Defined	C
CsmAEADEncryptAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_POLY1305 / CRYPTO_ALGOFAM_CUSTOM	C
CsmAEADEncryptAlgorithmSecondaryFamilyCustom	User Defined	C
CsmAEADEncryptAssociatedDataMaxLength	1..4294967295	C
CsmAEADEncryptCiphertextMaxLength	1..4294967295	C
CsmAEADEncryptPlaintextMaxLength	1..4294967295	C
CsmAEADEncryptTagLength	1..4294967295	C
CsmAEADEncryptKeyRef	CsmKey	C
CsmAEADEncryptQueueRef	CsmQueue	C

1) CsmAEADEncryptAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmAEADEncryptAlgorithmFamilyCustom
- This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADEncryptAlgorithmFamily.

3) CsmAEADEncryptAlgorithmKeyLength
- Size of the AEAD encryption key in bytes

4) CsmAEADEncryptAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

5) CsmAEADEncryptAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

6) CsmAEADEncryptAlgorithmSecondaryFamily

- Defines the secondary family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

7) CsmAEADEncryptAlgorithmSecondaryFamilyCustom
- This is the name of the custom secondary algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADEncryptAlgorithmFamily.

8) CsmAEADEncryptAssociatedDataMaxLength
- Max size of the input associated data length in bytes

9) CsmAEADEncryptCiphertextMaxLength
- Max size of the output ciphertext length in bytes

10) CsmAEADEncryptPlaintextMaxLength
- Max size of the input plaintext length in bytes

11) CsmAEADEncryptTagLength
- Size of the output Tag length in bytes

12) CsmAEADEncryptKeyRef
- This parameter refers to the key used for that encryption primitive.

13) CsmAEADEncryptQueueRef
- This parameter refers to the queue used for that encryption primitive.

5.1.4.7 CsmAEADDecrypt

Configuration of AEAD decryption primitives. The container name serves as a symbolic name for the identifier of an AEAD decryption interface.

Parameter Name	Value	Category
CsmAEADDecryptAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_EEA3	C
CsmAEADDecryptAlgorithmFamilyCustom	User Defined	C
CsmAEADDecryptAlgorithmKeyLength	1..4294967295	C
CsmAEADDecryptAlgorithmMode	CRYPTO_ALGOMODE_20ROUNDS / CRYPTO_ALGOMODE_12ROUNDS / CRYPTO_ALGOMODE_8ROUNDS / CRYPTO_ALGOMODE_GCM / CRYPTO_ALGOMODE_CUSTOM	C
CsmAEADDecryptAlgorithmModeCustom	User Defined	C
CsmAEADDecryptAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_POLY1305 / CRYPTO_ALGOFAM_CUSTOM	C
CsmAEADDecryptAlgorithmSecondaryFamilyCustom	User Defined	
CsmAEADDecryptAssociatedDataMaxLength	1..4294967295	C
CsmAEADDecryptCiphertextMaxLength	1..4294967295	C
CsmAEADDecryptPlaintextMaxLength	1..4294967295	C
CsmAEADDecryptTagLength	1..4294967295	C
CsmAEADDecryptKeyRef	CsmKey	C

Parameter Name	Value	Category
CsmAEADDecryptQueueRef	CsmQueue	C

1) CsmAEADDecryptAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmAEADDecryptAlgorithmFamilyCustom
- This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADDecryptAlgorithmFamily.

3) CsmAEADDecryptAlgorithmKeyLength
- Size of the AEAD decryption key in bytes

4) CsmAEADDecryptAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

5) CsmAEADDecryptAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

6) CsmAEADDecryptAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

7) CsmAEADDecryptAlgorithmSecondaryFamilyCustom
- This is the name of the custom secondary algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADDecryptAlgorithmFamily.

8) CsmAEADDecryptAssociatedDataMaxLength
- Max size of the input associated data length in bytes

9) CsmAEADDecryptCiphertextMaxLength
- Max size of the input ciphertext in bytes

10) CsmAEADDecryptPlaintextMaxLength
- Size of the output plaintext length in bytes

11) CsmAEADDecryptTagLength
- Size of the input Tag length in BITS

12) CsmAEADDecryptKeyRef
- This parameter refers to the key used for that decryption primitive.

13) CsmAEADDecryptQueueRef
- This parameter refers to the queue used for that decryption primitive.

5.1.4.8 CsmSignatureGenerate

Configurations of SignatureGenerate primitives. The container name serves as a symbolic name for the identifier of signature generation interface.

Parameter Name	Value	Category
CsmSignatureGenerateAlgorithmFamily	CRYPTO_ALGOFAM_BRAINPOOL / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_ECCNIST / CRYPTO_ALGOFAM_ED25519 / CRYPTO_ALGOFAM_RSA /	C

Parameter Name	Value	Category
	CRYPTO_ALGOFAM_ED448	
CsmSignatureGenerateAlgorithmFamilyCustom	User Defined	C
CsmSignatureGenerateAlgorithmMode	CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5 / CRYPTO_ALGOMODE_RSASSA_PSS	C
CsmSignatureGenerateAlgorithmModeCustom	User Defined	C
CsmSignatureGenerateAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 / CRYPTO_ALGOFAM_BLAKE_2s_256 / CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 / CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256 / CRYPTO_ALGOFAM_NOT_SET	C
CsmSignatureGenerateAlgorithmSecondaryFamilyCustom	User Defined	C
CsmSignatureGenerateDataMaxLength	1..4294967295	C
CsmSignatureGenerateKeyLength	1..4294967295	C
CsmSignatureGenerateResultLength	1..4294967295	C

1) CsmSignatureGenerateAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.
- Below is vendor specific define: CRYPTO_ALGOFAM_ED448(0xEE) DSA with Curve448

2) CsmSignatureGenerateAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureGenerateAlgorithmFamily.

3) CsmSignatureGenerateAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

4) CsmSignatureGenerateAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

5) CsmSignatureGenerateAlgorithmSecondaryFamily

- Determines the algorithm mode used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

6) CsmSignatureGenerateAlgorithmSecondaryFamilyCustom
- Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmSignatureGenerateAlgorithmSecondaryFamily.

7) CsmSignatureGenerateDataMaxLength
- Size of the input data length in bytes

8) CsmSignatureGenerateKeyLength
- Size of the signature generate key in bytes.

9) CsmSignatureGenerateResultLength
- Size of the output signature length in bytes

5.1.4.9 CsmSignatureVerify

Configurations of SignatureVerify primitives. The container name serves as a symbolic name for the identifier of signature verification interface.

Parameter Name	Value	Category
CsmSignatureVerifyAlgorithmFamily	CRYPTO_ALGOFAM_BRAINPOOL / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_ECCNIST / CRYPTO_ALGOFAM_ED25519 / CRYPTO_ALGOFAM_RSA / CRYPTO_ALGOFAM_ED448	C
CsmSignatureVerifyAlgorithmFamilyCustom	User Defined	C
CsmSignatureVerifyAlgorithmMode	CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5 / CRYPTO_ALGOMODE_RSASSA_PSS	C
CsmSignatureVerifyAlgorithmModeCustom	User Defined	C
CsmSignatureVerifyAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 / CRYPTO_ALGOFAM_BLAKE_2s_256 / CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 /	C

Parameter Name	Value	Category
	CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256 / CRYPTO_ALGOFAM_NOT_SET	
CsmSignatureVerifyAlgorithmSecondaryFamilyCustom	User Defined	C
CsmSignatureVerifyCompareLength	1..4294967295	C
CsmSignatureVerifyDataMaxLength	1..4294967295	C
CsmSignatureVerifyKeyLength	1..4294967295	C

1) CsmSignatureVerifyAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.
- Below is vendor specific define: CRYPTO_ALGOFAM_ED448(0xEE) Verify signature with Curve448
2) CsmSignatureVerifyAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureVerifyAlgorithmFamily.
3) CsmSignatureVerifyAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.
4) CsmSignatureVerifyAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service
5) CsmSignatureVerifyAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.
6) CsmSignatureVerifyAlgorithmSecondaryFamilyCustom
- Name of the custom secondary algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureVerifyAlgorithmFamily.
7) CsmSignatureVerifyCompareLength
- Number of the least significant bytes of the signature, for which the verification shall be calculated.
8) CsmSignatureVerifyDataMaxLength
- Max size of the input data, for which the signature shall be verified, in bytes.
9) CsmSignatureVerifyKeyLength
- Size of the signature verify key in bytes

5.1.4.10 CsmRandomGenerate

Configurations of RandomGenerate primitives. The container name serves as a symbolic name for the identifier of a random generator configuration.

Parameter Name	Value	Category
CsmRandomGenerateAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 / CRYPTO_ALGOFAM_BLAKE_2s_256 /	C

Parameter Name	Value	Category
	CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_RNG / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 / CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256 / CRYPTO_ALGOFAM_SIPHASH	
CsmRandomGenerateAlgorithmFamilyCustom	User Defined	C
CsmRandomGenerateAlgorithmMode	CRYPTO_ALGOMODE_CMAC / CRYPTO_ALGOMODE_CTRDRBG / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_GMAC / CRYPTO_ALGOMODE_HMAC / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_SIPHASH_2_4 / CRYPTO_ALGOMODE_SIPHASH_4_8	C
CsmRandomGenerateAlgorithmModeCustom	User Defined	C
CsmRandomGenerateAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_NOT_SET	C
CsmRandomGenerateAlgorithmSecondaryFamilyCustom	User Defined	C
CsmRandomGenerateResultLength	1..4294967295	C

1) CsmRandomGenerateAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmRandomGenerateAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmRandomAlgorithmFamily

3) CsmRandomGenerateAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

4) CsmRandomGenerateAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmRandomGenerateAlgorithmFamily.

5) CsmRandomGenerateAlgorithmSecondaryFamily

- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

6) CsmRandomGenerateAlgorithmSecondaryFamilyCustom
- Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmRandomAlgorithmSecondaryFamily.

7) CsmRandomGenerateResultLength
- Size of the random generate key in bytes

5.1.4.11 CsmJobKeySetValid

Configurations of KeySetValid primitives. The container name serves as a symbolic name for the identifier of a key configuration.

Parameter Name	Value	Category
CsmJobKeySetValidAlgorithmFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobKeySetValidAlgorithmFamilyCustom	User Defined	C
CsmJobKeySetValidAlgorithmMode	CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_CUSTOM	C
CsmJobKeySetValidAlgorithmModeCustom	User Defined	C
CsmJobKeySetValidAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobKeySetValidAlgorithmSecondaryFamilyCustom	User Defined	C

1) CsmJobKeySetValidAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmJobKeySetValidAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service.

3) CsmJobKeySetValidAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

4) CsmJobKeySetValidAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

5) CsmJobKeySetValidAlgorithmSecondaryFamily
- Name of the custom secondary algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

6) CsmJobKeySetValidAlgorithmSecondaryFamilyCustom
- Name of the custom secondary algorithm family used for the crypto service.

5.1.4.12 CsmJobRandomSeed

Configurations of RandomSeed primitives. The container name serves as a symbolic name for the

identifier of a random seed configuration.

Parameter Name	Value	Category
CsmJobRandomSeedAlgorithmFamilyCustom	User Defined	C
CsmJobRandomSeedAlgorithmMode	CRYPTO_ALGOMODE_CMAC / CRYPTO_ALGOMODE_CTRDRBG / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_GMAC / CRYPTO_ALGOMODE_HMAC / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_SIPHASH_2_4 / CRYPTO_ALGOMODE_SIPHASH_4_8	C
CsmJobRandomSeedAlgorithmModeCustom	User Defined	C
CsmJobRandomSeedAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobRandomSeedAlgorithmSecondaryFamilyCustom	User Defined	C
CsmRandomSeedAlgorithmFamily	CRYPTO_ALGOFAM_3DES / CRYPTO_ALGOFAM_AES / CRYPTO_ALGOFAM_BLAKE_1_256 / CRYPTO_ALGOFAM_BLAKE_1_512 / CRYPTO_ALGOFAM_BLAKE_2s_256 / CRYPTO_ALGOFAM_BLAKE_2s_512 / CRYPTO_ALGOFAM_CHACHA / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_RIPEMD160 / CRYPTO_ALGOFAM_RNG / CRYPTO_ALGOFAM_SHA1 / CRYPTO_ALGOFAM_SHA2_224 / CRYPTO_ALGOFAM_SHA2_256 / CRYPTO_ALGOFAM_SHA2_384 / CRYPTO_ALGOFAM_SHA2_512 / CRYPTO_ALGOFAM_SHA2_512_224 / CRYPTO_ALGOFAM_SHA2_512_256 / CRYPTO_ALGOFAM_SHA3_224 / CRYPTO_ALGOFAM_SHA3_256 / CRYPTO_ALGOFAM_SHA3_384 / CRYPTO_ALGOFAM_SHA3_512 / CRYPTO_ALGOFAM_SHAKE128 / CRYPTO_ALGOFAM_SHAKE256	C

1) CsmJobRandomSeedAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

2) CsmJobRandomSeedAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

3) CsmJobRandomSeedAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service. This is the name of the custom

algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

4) CsmJobRandomSeedAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

5) CsmJobRandomSeedAlgorithmSecondaryFamilyCustom
- Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

6) CsmRandomSeedAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

5.1.4.13 CsmJobKeyDerive

Configurations of KeyDerive primitives. The container name serves as a symbolic name for the identifier of a key derive configuration.

Parameter Name	Value	Category
CsmJobKeyDeriveAlgorithmFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_HKDF / CRYPTO_ALGOFAM_KDFX963 / CRYPTO_ALGOFAM_PBKDF2	C
CsmJobKeyDeriveAlgorithmMode	CRYPTO_ALGOMODE_CMAC / CRYPTO_ALGOMODE_CTRDRBG / CRYPTO_ALGOMODE_CUSTOM / CRYPTO_ALGOMODE_GMAC / CRYPTO_ALGOMODE_HMAC / CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_SIPHASH_2_4 / CRYPTO_ALGOMODE_SIPHASH_4_8	C
CsmJobKeyDeriveAlgorithmModeCustom	User Defined	C
CsmJobKeyDeriveAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobKeyDeriveAlgorithmFamilyCustom	User Defined	C
CsmJobKeyDeriveAlgorithmSecondaryFamilyCustom	User Defined	C

1) CsmJobKeyDeriveAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmJobKeyDeriveAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

3) CsmJobKeyDeriveAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

4) CsmJobKeyDeriveAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

5) CsmJobKeyDeriveAlgorithmFamilyCustom
- Name of the custom algorithm mode used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

6) CsmJobKeyDeriveAlgorithmSecondaryFamilyCustom
- Name of the custom algorithm mode used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

5.1.4.14 CsmJobKeyGenerate

Configurations of KeyGenerate primitives. The container name serves as a symbolic name for the identifier of a key generate configuration.

Parameter Name	Value	Category
CsmJobKeyGenerateAlgorithmFamily	CRYPTO_ALGOFAM_X448 / CRYPTO_ALGOFAM_ED448 / CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_ECCANSI / CRYPTO_ALGOFAM_ECCNIST / CRYPTO_ALGOFAM_ECCSEC / CRYPTO_ALGOFAM_ECDH / CRYPTO_ALGOFAM_ED25519 / CRYPTO_ALGOFAM_X25519	C
CsmJobKeyGenerateAlgorithmFamilyCustom	User Defined	C
CsmJobKeyGenerateAlgorithmMode	CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_CUSTOM	C
CsmJobKeyGenerateAlgorithmModeCustom	User Defined	C
CsmJobKeyGenerateAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobKeyGenerateAlgorithmSecondaryFamilyCustom	User Defined	C

1) CsmJobKeyGenerateAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.
  Below are vendor specific defines:
  - CRYPTO_ALGOFAM_ED448(0xEE): Curve448 public key that is used for signature verifying.
  - CRYPTO_ALGOFAM_X448(0xEF): Curve448 public key that is used for Key Exchange.

2) CsmJobKeyGenerateAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

3) CsmJobKeyGenerateAlgorithmModeCustom
- Name of the custom algorithm mode used for the crypto service.

4) CsmJobKeyGenerateAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

5.1.4.15 CsmJobKeyExchangeCalcPubVal

Configurations of KeyExchangeCalcPubVal primitives. The container name serves as a symbolic name for the identifier of a key configuration.

Parameter Name	Value	Category
CsmJobKeyExchangeCalcPubValAlgorithmFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_DH / CRYPTO_ALGOFAM_RSA	C
CsmJobKeyExchangeCalcPubValAlgorithmFamilyCustom	User Defined	C
CsmJobKeyExchangeCalcPubValAlgorithmMode	CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_CUSTOM	C
CsmJobKeyExchangeCalcPubValAlgorithmModeCustom	User Defined	C
CsmJobKeyExchangeCalcPubValAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobKeyExchangeCalcPubValAlgorithmSecondaryFamilyCustom	User Defined	C

1) CsmJobKeyExchangeCalcPubValAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmJobKeyExchangeCalcPubValAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

3) CsmJobKeyExchangeCalcPubValAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

4) CsmJobKeyExchangeCalcPubValAlgorithmModeCustom
- Name of the custom primitive mode.

5) CsmJobKeyExchangeCalcPubValAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

6) CsmJobKeyExchangeCalcPubValAlgorithmSecondaryFamilyCustom
- This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

5.1.4.16 CsmJobKeyExchangeCalcSecret

Configurations of KeyExchangeCalcSecret primitives. The container name serves as a symbolic name for the identifier of a JobKeyExchangeCalcSecret configuration.

Parameter Name	Value	Category
CsmJobKeyExchangeCalcSecretAlgorithmFamily	CRYPTO_ALGOFAM_CUSTOM / CRYPTO_ALGOFAM_DH / CRYPTO_ALGOFAM_RSA	C
CsmJobKeyExchangeCalcSecretAlgorithmFamilyCustom	User Defined	C

Parameter Name	Value	Category
CsmJobKeyExchangeCalcSecretAlgorithmMode	CRYPTO_ALGOMODE_NOT_SET / CRYPTO_ALGOMODE_CUSTOM	C
CsmJobKeyExchangeCalcSecretAlgorithmModeCustom	User Defined	C
CsmJobKeyExchangeCalcSecretAlgorithmSecondaryFamily	CRYPTO_ALGOFAM_NOT_SET / CRYPTO_ALGOFAM_CUSTOM	C
CsmJobKeyExchangeCalcSecretAlgorithmSecondaryFamilyCustom	User Defined	C

1) CsmJobKeyExchangeCalcSecretAlgorithmFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

2) CsmJobKeyExchangeCalcSecretAlgorithmFamilyCustom
- Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

3) CsmJobKeyExchangeCalcSecretAlgorithmMode
- Determines the algorithm mode used for the crypto service. Refer 6.1.4 Crypto_AlgorithmModeType.

4) CsmJobKeyExchangeCalcSecretAlgorithmModeCustom
- Name of the custom primitive mode.

5) CsmJobKeyExchangeCalcSecretAlgorithmSecondaryFamily
- Determines the algorithm family used for the crypto service. Refer 6.1.3 Crypto_AlgorithmFamilyType.

6) CsmJobKeyExchangeCalcSecretAlgorithmSecondaryFamilyCustom
- This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used.

5.1.5 CsmQueues

Container for CSM queue configurations. The CsmQueues shall sort the jobs according to the configured job's priority.

Parameter Name	Value	Category
CsmQueueSize	1..4294967295	C
CsmChannelRef	CryIfChannel	C

1) CsmQueueSize
- Maximum Size of the CsmQueue. If jobs cannot be processed by the underlying hardware since the hardware is busy, the jobs stay in the prioritized queue. If the queue is full, the next job will be rejected. This defines in generated header file.

2) CsmChannelRef
- Refers to the underlying Crypto Interface channel. The channel is path to process job in Queue.

5.1.6 CsmInOutRedirections

Configuration for CSM redirection configurations. A redirection let a CSM job use a specific key

element as input or/and output.

Parameter Name	Value	Category
CsmInputKeyElementId	0..4294967295	C
CsmOutputKeyElementId	0..4294967295	C
CsmSecondaryInputKeyElementId	0..4294967295	C
CsmSecondaryOutputKeyElementId	0..4294967295	C
CsmTertiaryInputKeyElementId	0..4294967295	C
CsmInputKeyRef	CsmKey	C
CsmOutputKeyRef	CsmKey	C
CsmSecondaryInputKeyRef	CsmKey	C
CsmSecondaryOutputKeyRef	CsmKey	C
CsmTertiaryInputKeyRef	CsmKey	C

1) CsmInputKeyElementId
- Identifier of the key element used as input. This value indicate length of inputPtr. If Bit#0 (least significant bit) of redirectionConfig is set, this must indicate element that is used for input buffer.

2) CsmOutputKeyElementId
- Identifier of the key element used as output. If Bit#4 of redirectionConfig is set, this redirect to outputPtr.

3) CsmSecondaryInputKeyElementId
- Identifier of the key element used as secondary input. If Bit#1 of redirectionConfig is set, this redirect to secondary InputBuffer.

4) CsmSecondaryOutputKeyElementId
- Identifier of the key element used as secondary output. If Bit#5 of redirectionConfig is set, this redirect to secondary OutputBuffer.

5) CsmTertiaryInputKeyElementId
- Identifier of the key element used as tertiary input.

6) CsmInputKeyRef
- This parameter refers to the key used as input. This indicate inputPtr. If Bit#0 (least significant bit) of redirectionConfig is set, this must indicate element that is used for input buffer.

7) CsmOutputKeyRef
- This parameter refers to the key used as output. If Bit#4 of redirectionConfig is set, this redirect to outputPtr.

8) CsmSecondaryInputKeyRef
- This parameter refers to the key used as secondary input. If Bit#1 of redirectionConfig is set, this redirect to secondary InputBuffer.

9) CsmSecondaryOutputKeyRef
- This parameter refers to the key used as secondary output. If Bit#5 of redirectionConfig is set, this redirect to secondary OutputBuffer.

10) CsmTertiaryInputKeyRef
- This parameter refers to the key used as tertiary input.

5.1.7 CsmCallbacks

Container for callback function configurations

Parameter Name	Value	Category
CsmCallbackFunc	User Defined	C
CsmCallbackId	0..4294967295	C

1) CsmCallbackFunc
- Callback function to be called if an asynchronous operation has finished. The corresponding job has to be configured to be processed asynchronously.

2) CsmCallbackId
- Identifier of the callback function. The set of actually configured identifiers shall be consecutive and gapless.

6. Application Programming Interface (API)

6.1 Type Definitions

6.1.1 Extension to Std_ReturnType

Range	CRYPTO_E_BUSY	0x02	The service request failed because the service is still busy
	CRYPTO_E_SMALL_BUFFER	0x03	The service request failed because the provided buffer is too small to store the result.
	CRYPTO_E_ENTROPY_ EXHAUSTED	0x04	The service request failed because the entropy of the random number generator is exhausted
	CRYPTO_E_QUEUE_FULL	0x05	The service request failed because the queue is full.
	CRYPTO_E_KEY_ READ_FAIL	0x06	The service request failed because read access was denied
	CRYPTO_E_KEY_ WRITE_FAIL	0x07	The service request failed because the writing access failed
	CRYPTO_E_KEY_NOT_ AVAILABLE	0x08	The service request failed because the key is not available
	CRYPTO_E_KEY_NOT_ VALID	0x09	The service request failed because the key is invalid.
	CRYPTO_E_KEY_SIZE_ MISMATCH	0x0A	The service request failed because the key size does not match.
	CRYPTO_E_COUNTER_OVERFLOW	0x0B	The service request failed because the counter is overflowed.
	CRYPTO_E_JOB_ CANCELED	0x0C	The service request failed because the Job has been canceled.
	CRYPTO_E_KEY_ EMPTY	0x0D	The service request failed because of uninitialized source key element.
Description	Overlaid return value of Std_ReturnType for Crypto stack.		
Available via	Csm.h		

6.1.2 Csm_ConfigType

Name	Csm_ConfigType

Kind	Structure	
Elements	implementation specific	
	Type	--
	Comment	The content of the configuration data structure is implementation specific.
Description	Configuration data structure of Csm module	
Available via	Csm.h	

6.1.3 Crypto_AlgorithmFamilyType

Name	Crypto_AlgorithmFamilyType		
Kind	Enumeration		
Range	CRYPTO_ALGOFAM_NOT_SET	0x00	Algorithm family is not set
	CRYPTO_ALGOFAM_SHA1	0x01	SHA1 hash
	CRYPTO_ALGOFAM_SHA2_224	0x02	SHA2-224 hash
	CRYPTO_ALGOFAM_SHA2_256	0x03	SHA2-256 hash
	CRYPTO_ALGOFAM_SHA2_384	0x04	SHA2-384 hash
	CRYPTO_ALGOFAM_SHA2_512	0x05	SHA2-512 hash
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	SHA2-512/224 hash
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	SHA2-512/256 hash
	CRYPTO_ALGOFAM_SHA3_224	0x08	SHA3-224 hash
	CRYPTO_ALGOFAM_SHA3_256	0x09	SHA3-256 hash

	CRYPTO_ALGOFAM_ SHA3_384	0x0a	SHA3-384 hash
	CRYPTO_ALGOFAM_ SHA3_512	0x0b	SHA3-512 hash
	CRYPTO_ALGOFAM_ SHAKE128	0x0c	SHAKE128 hash
	CRYPTO_ALGOFAM_ SHAKE256	0x0d	SHAKE256 hash
	CRYPTO_ALGOFAM_ RIPEMD160	0x0e	RIPEMD hash
	CRYPTO_ALGOFAM_ BLAKE_1_256	0x0f	BLAKE-1-256 hash
	CRYPTO_ALGOFAM_ BLAKE_1_512	0x10	BLAKE-1-512 hash
	CRYPTO_ALGOFAM_ BLAKE_2s_256	0x11	BLAKE-2s-256 hash
	CRYPTO_ALGOFAM_ BLAKE_2s_512	0x12	BLAKE-2s-512 hash
	CRYPTO_ALGOFAM_3DES	0x13	3DES cipher
	CRYPTO_ALGOFAM_AES	0x14	AES cipher
	CRYPTO_ALGOFAM_ CHACHA	0x15	ChaCha cipher
	CRYPTO_ALGOFAM_RSA	0x16	RSA cipher
	CRYPTO_ALGOFAM_ ED25519	0x17	ED22518 elliptic curve
	CRYPTO_ALGOFAM_ BRAINPOOL	0x18	Brainpool elliptic curve
	CRYPTO_ALGOFAM_ ECCNIST	0x19	NIST ECC elliptic curves
	CRYPTO_ALGOFAM_RNG	0x1b	Random Number Generator
	CRYPTO_ALGOFAM_ SIPHASH	0x1c	SipHash
	CRYPTO_ALGOFAM_ ECCANSI	0x1e	Elliptic curve according to ANSI X9.62
	CRYPTO_ALGOFAM_ ECCSEC	0x1f	Elliptic curve according to SECG
	CRYPTO_ALGOFAM_DRBG	0x20	Random number generator according to NIST SP800-90A

	CRYPTO_ALGOFAM_ FIPS186	0x21	Random number generator according to FIPS 186.
	CRYPTO_ALGOFAM_ PADDING_PKCS7	0x22	Cipher padding according to PKCS.7
	CRYPTO_ALGOFAM_ PADDING_ONEWITHZEROS	0x23	Cipher padding mode. Fill/verify data with 0, but first bit after the data is 1. Eg. "DATA" & 0x80 & 0x00...
	CRYPTO_ALGOFAM_PBKDF2	0x24	Password-Based Key Derivation Function 2
	CRYPTO_ALGOFAM_ KDFX963	0x25	ANSI X9.63 Public Key Cryptography
	CRYPTO_ALGOFAM_DH	0x26	Diffie-Hellman
	CRYPTO_ALGOFAM_ CUSTOM	0xff	Custom algorithm family
Description	Enumeration of the algorithm family.		
Available via	Csm.h		

6.1.4 Crypto_AlgorithmModeType

Name	Crypto_AlgorithmModeType		
Kind	Enumeration		
Range	CRYPTO_ALGOMODE_NOT_SET	0x00	Algorithm key is not set
	CRYPTO_ALGOMODE_ECB	0x01	Blockmode: Electronic Code Book
	CRYPTO_ALGOMODE_CBC	0x02	Blockmode: Cipher Block Chaining
	CRYPTO_ALGOMODE_CFB	0x03	Blockmode: Cipher Feedback Mode
	CRYPTO_ALGOMODE_OFB	0x04	Blockmode: Output Feedback Mode
	CRYPTO_ALGOMODE_CTR	0x05	Blockmode: Counter Mode
	CRYPTO_ALGOMODE_GCM	0x06	Blockmode: Galois/Counter Mode
	CRYPTO_ALGOMODE_XTS	0x07	XOR-encryption-based tweaked-codebook mode with ciphertext stealing
	CRYPTO_ALGOMODE_RSAES_ OAEP	0x08	RSA Optimal Asymmetric Encryption Padding

	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5	0x09	RSA encryption/decryption with PKCS#1 v1.5 padding
	CRYPTO_ALGOMODE_RSASSA_PSS	0x0a	RSA Probabilistic Signature Scheme
	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5	0x0b	RSA signature with PKCS#1 v1.5
	CRYPTO_ALGOMODE_8ROUNDS	0x0c	8 rounds (e.g. ChaCha8)
	CRYPTO_ALGOMODE_12ROUNDS	0x0d	12 rounds (e.g. ChaCha12)
	CRYPTO_ALGOMODE_20ROUNDS	0x0e	20 rounds (e.g. ChaCha20)
	CRYPTO_ALGOMODE_HMAC	0x0f	Hashed-based MAC
	CRYPTO_ALGOMODE_CMAC	0x10	Cipher-based MAC
	CRYPTO_ALGOMODE_GMAC	0x11	Galois MAC
	CRYPTO_ALGOMODE_CTRDRBG	0x12	Counter-based Deterministic Random Bit Generator
	CRYPTO_ALGOMODE_SIPHASH_2_4	0x13	Siphash-2-4
	CRYPTO_ALGOMODE_SIPHASH_4_8	0x14	Siphash-4-8
	CRYPTO_ALGOMODE_PXXXR1	0x15	ANSI R1 Curve
	CRYPTO_ALGOMODE_CUSTOM	0xff	Custom algorithm mode
	CRYPTO_ALGOMODE_CCM	0xef	Blockmode: Counter with Cipher Block Chaining-Message Authentication Code Mode
Description	Enumeration of the algorithm mode		
Available via	Csm.h		

6.1.5 Crypto_InputOutputRedirectionConfigType

Name	Crypto_InputOutputRedirectionConfigType

Kind	Enumeration		
Range	CRYPTO_REDIRECT_CONFIG_PRIMARY_INPUT	0x01	--
	CRYPTO_REDIRECT_CONFIG_SECONDARY_INPUT	0x02	--
	CRYPTO_REDIRECT_CONFIG_TERTIARY_INPUT	0x04	--
	CRYPTO_REDIRECT_CONFIG_PRIMARY_OUTPUT	0x10	--
	CRYPTO_REDIRECT_CONFIG_SECONDARY_OUTPUT	0x20	--
Description	Defines which of the input/output parameters are re-directed to a key element. The values can be combined to define a bit field.		
Available via	Csm.h		

6.1.6 Crypto_JobType

Name	Crypto_JobType	
Kind	Structure	
Elements	jobId	
	Type	uint32
	Comment	Identifier for the job structure.
	jobState	
	Type	Crypto_JobStateType
	Comment	Determines the current job state.
	jobPrimitiveInputOutput	
	Type	Crypto_JobPrimitiveInputOutputType
	Comment	Structure containing input and output information depending on the job and the crypto primitive.
	jobPrimitiveInfo	
	Type	const Crypto_JobPrimitiveInfoType*
	Comment	Pointer to a structure containing further information which depends on the job and the crypto primitive.

	jobInfo	
	Type	const Crypto_JobInfoType*
	Comment	Pointer to a structure containing further information which depends on the job and the crypto primitive.
	cryptoKeyId	
	Type	uint32
	Comment	Identifier of the Crypto Driver key. The identifier shall be written by the Crypto Interface.
	jobRedirectionInfoRef	
	Type	Crypto_JobRedirectionInfoType*
	Comment	Pointer to a structure containing further information on the usage of keys as input and output for jobs.
	targetCryptoKeyId	
	Type	uint32
	Comment	Target identifier of the Crypto Driver key. The identifier shall be written by the Crypto Interface.
Description	Structure which contains further information, which depends on the job and the crypto primitive.	
Available via	Csm.h	

6.1.7 Crypto_JobStateType

Name	Crypto_JobStateType		
Kind	Enumeration		
Range	CRYPTO_ JOBSTATE_IDLE	0x00	Job is in the state "idle". This state is reached after Csm_ Init() or when the "Finish" state is finished.
	CRYPTO_ JOBSTATE_ ACTIVE	0x01	Job is in the state "active". There was already some input or there are intermediate results. This state is reached, when the "update" or "start" operation finishes.
Description	Enumeration of the current job state.		

Available via	Csm.h

6.1.8 Crypto_JobPrimitiveInputOutputType

Name	Crypto_JobPrimitiveInputOutputType	
Kind	Structure	
lements	inputPtr	
	Type	const uint8*
	Comment	Pointer to the input data.
	inputLength	
	Type	uint32
	Comment	Contains the input length in bytes.
	secondaryInputPtr	
	Type	const uint8*
	Comment	Pointer to the secondary input data (for MacVerify, SignatureVerify).
	secondaryInputLength	
	Type	uint32
	Comment	Contains the secondary input length in bits or bytes, depending on the requested service.
	tertiaryInputPtr	
	Type	const uint8*
	Comment	Pointer to the tertiary input data (for MacVerify, SignatureVerify).
	tertiaryInputLength	
	Type	uint32
	Comment	Contains the tertiary input length in bytes.
	outputPtr	

Type	uint8*
Comment	Pointer to the output data.
outputLengthPtr	
Type	uint32*
Comment	Holds a pointer to a memory location containing the output length in bytes.
secondaryOutputPtr	
Type	uint8*
Comment	Pointer to the secondary output data.
secondaryOutputLengthPtr	
Type	uint32*
Comment	Holds a pointer to a memory location containing the secondary output length in bytes.
Input64	
Type	uint64
Comment	versatile input parameter.
verifyPtr	
Type	Crypto_VerifyResultType*
Comment	Output pointer to a memory location holding a Crypto_VerifyResultType
mode	
Type	Crypto_OperationModeType
Comment	Indicator of the mode(s)/operation(s) to be performed
cryIfKeyId	
Type	uint32
Comment	Holds the CryIf key id for key operation services.

	targetCryIfKeyId	
	Type	uint32
	Comment	Holds the target CryIf key id for key operation services.
Description	Structure which contains input and output information depending on the job and the crypto primitive.	
Available via	Csm.h	

6.1.9 Crypto_JobInfoType

Name	Crypto_JobInfoType	
Kind	Structure	
Elements	jobId	
	Type	const uint32
	Comment	The family of the algorithm
	jobPriority	
	Type	const uint32
	Comment	Specifies the importance of the job (the higher, the more important).
Description	Structure which contains job information (job ID and job priority).	
Available via	Csm.h	

6.1.10 Crypto_JobPrimitiveInfoType

Name	Crypto_JobPrimitiveInfoType	
Kind	Structure	
Elements	callbackId	
	Type	uint32
	Comment	Internal identifier of the callback function, to be called by Csm, if the configured service is finished.
	primitiveInfo	

	Type	const Crypto_PrimitiveInfoType*
	Comment	Pointer to a structure containing further configuration of the crypto primitives
	cryIfKeyId	
	Type	uint32
	Comment	Identifier of the CryIf key.
	processingType	
	Type	Crypto_ProcessingType
	Comment	Determines the synchronous or asynchronous behavior.
	callbackUpdateNotification	
	Type	boolean
	Comment	Indicates, whether the callback function shall be called, if the UPDATE operation has finished.
Description	Structure which contains further information, which depends on the job and the crypto primitive.	
Available via	Csm.h	

6.1.11 Crypto_ServiceInfoType

Name	Crypto_ServiceInfoType		
Range	CRYPTO_HASH	0x00	Hash Service
	CRYPTO_MACGENERATE	0x01	MacGenerate Service
	CRYPTO_MACVERIFY	0x02	MacVerify Service
	CRYPTO_ENCRYPT	0x03	Encrypt Service
	CRYPTO_DECRYPT	0x04	Decrypt Service
	CRYPTO_AEADENCRYPT	0x05	AEADEncrypt Service
	CRYPTO_AEADDECRYPT	0x06	AEADDecrypt Service
	CRYPTO_SIGNATUREGENERATE	0x07	SignatureGenerate Service

	CRYPTO_SIGNATUREVERIFY	0x08	SignatureVerify Service
	CRYPTO_RANDOMGENERATE	0x0B	RandomGenerate Service
	CRYPTO_RANDOMSEED	0x0C	RandomSeed Service
	CRYPTO_KEYGENERATE	0x0D	KeyGenerate Service
	CRYPTO_KEYDERIVE	0x0E	KeyDerive Service
	CRYPTO_ KEYEXCHANGECALCPUBVAL	0x0F	KeyExchangeCalcPubVal Service
	CRYPTO_ KEYEXCHANGECALCSECRET	0x10	KeyExchangeCalcSecret Service
	CRYPTO_CERTIFICATEPARSE	0x11	CertificiateParse Service
	CRYPTO_CERTIFICATEVERIFY	0x12	CertificateVerify Service
	CRYPTO_KEYSETVALID	0x13	KeySetValid Service
Description	Enumeration of the kind of the service.		
Available via	Csm.h		

6.1.12 Crypto_JobRedirectionInfoType

Name	Crypto_JobRedirectionInfoType	
Kind	Structure	
Elements	redirectionConfig	
	Type	uint8
	Comment	Bit structure which indicates which buffer shall be redirected to a key element. Values from Crypto_InputOutputRedirectionConfigType can be used and combined with unary OR operation.
	inputKeyId	
	Type	uint32
	Comment	Identifier of the key which shall be used as input

inputKeyElementId	
Type	uint32
Comment	Identifier of the key element which shall be used as input
secondaryInputKeyId	
Type	uint32
Comment	Identifier of the key which shall be used as secondary input
secondaryInputKeyElementId	
Type	uint32
Comment	Identifier of the key element which shall be used as secondary input
tertiaryInputKeyId	
Type	uint32
Comment	Identifier of the key which shall be used as tertiary input
tertiaryInputKeyElementId	
Type	uint32
Comment	Identifier of the key element which shall be used as tertiary input
outputKeyId	
Type	uint32
Comment	Identifier of the key which shall be used as output
outputKeyElementId	
Type	uint32
Comment	Identifier of the key element which shall be used as output
secondaryOutputKeyId	
Type	uint32
Comment	Identifier of the key which shall be used as secondary output

	secondaryOutputKeyElementId	
	Type	uint32
	Comment	Identifier of the key element which shall be used as secondary output
Description	Structure which holds the identifiers of the keys and key elements which shall be used as input and output for a job and a bit structure which indicates which buffers shall be redirected to those key elements.	
Available via	Csm.h	

6.1.13 Crypto_AlgorithmInfoType

Name	Crypto_AlgorithmInfoType	
Kind	Structure	
Elements	family	
	Type	Crypto_AlgorithmFamilyType
	Comment	The family of the algorithm
	secondaryFamily	
	Type	Crypto_AlgorithmFamilyType
	Comment	The secondary family of the algorithm
	keyLength	
	Type	uint32
	Comment	The key length in bits to be used with that algorithm
	mode	
	Type	Crypto_AlgorithmModeType
	Comment	The operation mode to be used with that algorithm
Description	Structure which determines the exact algorithm. Note, not every algorithm needs to specify all fields. AUTOSAR shall only allow valid combinations.	
Available via	Csm.h	

6.1.14 Crypto_ProcessingType

Name	Crypto_ProcessingType		
Kind	Enumeration		
Range	CRYPTO_PROCESSING_ASYNC	0x00	Asynchronous job processing
	CRYPTO_PROCESSING_SYNC	0x01	Synchronous job processing
Description	Enumeration of the processing type.		
Available via	Csm.h		

6.1.15 Crypto_PrimitiveInfoType

Name	Crypto_PrimitiveInfoType	
Kind	Structure	
Elements	resultLength	
	Type	const uint32
	Comment	Contains the result length in bytes.
	service	
	Type	const Crypto_ServiceInfoType
	Comment	Contains the enum of the used service, e.g. Encrypt
	algorithm	
	Type	const Crypto_AlgorithmInfoType
	Comment	Contains the information of the used algorithm
Description	Structure which contains basic information about the crypto primitive.	
Available via	Csm.h	

6.1.16 Csm_ConfigIdType

Name	Csm_ConfigIdType
Kind	Type

Derived from	uint16		
Range	0..65535	--	--
Description	Identification of a CSM service configuration via a numeric identifier, that is unique within a service. The name of a CSM service configuration, i.e. the name of the container Csm_<Service>Config, shall serve as a symbolic name for this parameter		
Available via	Csm.h		

6.2 Macro Constants

None

6.3 Functions

6.3.1 General Interface

6.3.1.1 Csm_Init

Service Name	Csm_Init	
Syntax	void Csm_Init (const Csm_ConfigType* configPtr)	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	configPtr	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the CSM module.	
Configuration Dependency	None	

Available via	Csm.h

The Configuration pointer configPtr is currently not used and shall therefore be set null pointer value.

6.3.1.2 Csm_GetVersionInfo

Service Name	Csm_GetVersionInfo	
Syntax	void Csm_GetVersionInfo (Std_VersionInfoType* versioninfo)	
Service ID [hex]	0x3b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module.	
Configuration Dependency	CsmVersionInfoApi should be set true.	
Available via	Csm.h	

6.3.2 Hash Interface

A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the hash value, such that an accidental or intentional change to the data will change the hash value. Main properties of hash functions are that it is infeasible to find a message that has a given hash or to find two different messages with the same hash.

6.3.2.1 Csm_Hash

Service Name	Csm_Hash

Syntax	Std_ReturnType Csm_Hash (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr)	
Service ID [hex]	0x5d	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Contains the pointer to the data for which the hash shall be computed.
	dataLength	Contains the number of bytes to be hashed.
Parameters (inout)	resultLengthPtr	Holds a pointer to the memory location in which the output length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Contains the pointer to the data where the hash value shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed
Description	Uses the given data to perform the hash calculation and stores the hash.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmHash container.	
Available via	Csm.h	

6.3.3 MAC interface

A message authentication code (MAC) is a short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC. The MAC value protects both a message's data integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

6.3.3.1 Csm_MacGenerate

Service Name	Csm_MacGenerate	
Syntax	Std_ReturnType Csm_MacGenerate (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* macPtr, uint32* macLengthPtr)	
Service ID [hex]	0x60	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Contains the pointer to the data for which the MAC shall be computed.
	dataLength	Contains the number of bytes to be hashed.
Parameters (inout)	macLengthPtr	Holds a pointer to the memory location in which the output length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer provided by macPtr. When the request has finished, the actual length of the returned MAC shall be stored.
Parameters (out)	macPtr	Contains the pointer to the data where the MAC shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	

Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Csm Job refer to CsmPrimitive that has CsmMacGenerate container.
Available via	Csm.h

6.3.3.2 Csm_MacVerify

Service Name	Csm_MacVerify	
Syntax	Std_ReturnType Csm_MacVerify (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, const uint8* macPtr, const uint32 macLength, Crypto_VerifyResultType* verifyPtr)	
Service ID [hex]	0x61	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Indicates which operation mode(s) to perform.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Holds a pointer to the data for which the MAC shall be verified.
	dataLength	Contains the number of data bytes for which the MAC shall be verified.
	macPtr	Holds a pointer to the MAC to be verified.
	macLength	Contains the MAC length in BITS to be verified.
Parameters (inout)	None	
Parameters (out)	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Verifies the given MAC by comparing if the MAC is generated with the given data.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmMacVerify container.	
Available via	Csm.h	

6.3.4 Cipher Interface

The cipher interfaces can be used for symmetrical and asymmetrical encryption or decryption. Furthermore, it is also possible to use these interfaces for compression and decompression, respectively.

6.3.4.1 Csm_Encrypt

Service Name	Csm_Encrypt	
Syntax	Std_ReturnType Csm_Encrypt (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr)	
Service ID [hex]	0x5e	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Contains the pointer to the data to be encrypted.

	data Length	Contains the number of bytes to encrypt.
Parameters (inout)	resultLengthPtr	Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Contains the pointer to the data where the encrypted data shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmEncrypt container.	
Available via	Csm.h	

In the case of block ciphers, it shall be possible to pass a dataLength which is not a multiple of the corresponding block size. The underlying Crypto Driver is responsible for handling these input data.

6.3.4.2 Csm_Decrypt

Service Name	Csm_Decrypt
Syntax	Std_ReturnType Csm_Decrypt (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr)
Service ID [hex]	0x5f
Sync/Async	Asynchronous or Synchronous, depending on the job configuration

Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Contains the pointer to the data to be decrypted.
	dataLength	Contains the number of bytes to decrypt.
Parameters (inout)	resultLengthPtr	Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Contains the pointer to the memory location where the decrypted data shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Decrypts the given encrypted data and store the decrypted plaintext in the memory location pointed by the result pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Csm Job refer to CsmPrimitive that has CsmDecrypt container.	
Available via	Csm.h	

6.3.5 Authenticated Encryption with Associated Data (AEAD) Interface

AEAD (also known as Authenticated Encryption) is a block cipher mode of operation which also allows integrity checks (e.g. AES-GCM).

6.3.5.1 Csm_AEADEncrypt

Service Name	Csm_AEADEncrypt

Syntax	Std_ReturnType Csm_AEADEncrypt (uint32 jobId, Crypto_OperationModeType mode, const uint8* plaintextPtr, uint32 plaintextLength, const uint8* associatedDataPtr, uint32 associatedDataLength, uint8* ciphertextPtr, uint32* ciphertextLengthPtr, uint8* tagPtr, uint32* tagLengthPtr)	
Service ID [hex]	0x62	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	plaintextPtr	Contains the pointer to the data to be encrypted.
	plaintextLength	Contains the number of bytes to encrypt.
	associatedDataPtr	Contains the pointer to the associated data.
	associatedDataLength	Contains the number of bytes of the associated data.
Parameters (inout)	ciphertextLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the ciphertext is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
	tagLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the Tag is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	ciphertextPtr	Contains the pointer to the data where the encrypted data shall be stored.

	tagPtr	Contains the pointer to the data where the Tag shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Uses the given input data to perform a AEAD encryption and stores the ciphertext and the MAC in the memory locations pointed by the ciphertext pointer and Tag pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Cs mJob refer to CsmPrimitive that has CsmAEADEncrypt container.	
Available via	Csm.h	

6.3.5.2 Csm_AEADDecrypt

Service Name	Csm_AEADDecrypt	
Syntax	Std_ReturnType Csm_AEADDecrypt (uint32 jobId, Crypto_OperationModeType mode, const uint8* ciphertextPtr, uint32 ciphertextLength, const uint8* associatedDataPtr, uint32 associatedDataLength, const uint8* tagPtr, uint32 tagLength, uint8* plaintextPtr, uint32* plaintextLengthPtr, Crypto_VerifyResultType* verifyPtr)	
Service ID [hex]	0x63	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters	jobId	Holds the identifier of the job using the CSM service.

(in)	mode	Indicates which operation mode(s) to perform.
	ciphertextPtr	Contains the pointer to the data to be decrypted.
	ciphertextLength	Contains the number of bytes to decrypt.
	associatedDataPtr	Contains the pointer to the associated data.
	associatedDataLength	Contains the length in bytes of the associated data.
	tagPtr	Contains the pointer to the Tag to be verified.
	tagLength	Contains the length in bytes of the Tag to be verified.
Parameters (inout)	plaintextLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the plaintext is stored. On calling this function, this parameter shall contain the size of the buffer provided by plaintextPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	plaintextPtr	Contains the pointer to the data where the decrypted data shall be stored.
	verifyPtr	Contains the pointer to the result of the verification.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Uses the given data to perform an AEAD encryption and stores the ciphertext and the MAC in the memory locations pointed by the ciphertext pointer and Tag pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Cs mJob refer to CsmPrimitive that has CsmAEADDecrypt container.	
Available via	Csm.h	

6.3.6 Signature Interface

A digital signature is a type of asymmetric cryptography. Digital signatures are equivalent to

traditional handwritten signatures in many respects.
Digital signatures can be used to authenticate the source of messages as well as to prove integrity of signed messages. If a message is digitally signed, any change in the message after signature will invalidate the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature.

6.3.6.1 Csm_SignatureGenerate

Service Name	Csm_SignatureGenerate	
Syntax	Std_ReturnType Csm_SignatureGenerate (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr)	
Service ID [hex]	0x76	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Contains the pointer to the data to be signed.
	dataLength	Contains the number of bytes to sign.
Parameters (inout)	resultLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the signature is stored. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Contains the pointer to the data where the signature shall be stored.

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Cs mJob refer to CsmPrimitive that has CsmSignatureGenerate container.	
Available via	Csm.h	

6.3.6.2 Csm_SignatureVerify

Service Name	Csm_SignatureVerify	
Syntax	Std_ReturnType Csm_SignatureVerify (uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, const uint8* signaturePtr, uint32 signatureLength, Crypto_VerifyResultType* verifyPtr)	
Service ID [hex]	0x64	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	mode	The Crypto_JobInfoType job with the corresponding jobId shall be modified in the following way:
	dataPtr	Contains the pointer to the data to be verified.
	dataLength	Contains the number of data bytes.
	signaturePtr	Holds a pointer to the signature to be verified.
	signatureLength	Contains the signature length in bytes.

Parameters (inout)	None	
Parameters (out)	verifyPtr	Holds a pointer to the memory location, which will hold the result of the signature verification.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Verifies the given MAC by comparing if the signature is generated with the given data.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmSignatureVerify container.	
Available via	Csm.h	

6.3.7 Random Interface

The random interface provides generation of random numbers. A random number can be generated either by a physical device (true random number generator), or by computational algorithms (pseudo random number generator). The randomness of pseudo random number generators can be increased by an appropriate selection of the seed.

6.3.7.1 Csm_RandomGenerate

Service Name	Csm_RandomGenerate	
Syntax	Std_ReturnType Csm_RandomGenerate (uint32 jobId, uint8* resultPtr, uint32* resultLengthPtr)	
Service ID [hex]	0x72	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.

Parameters (inout)	resultLengthPtr	Holds a pointer to the memory location in which the result length in bytes is stored. On calling this function, this parameter shall contain the number of random bytes, which shall be stored to the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Holds a pointer to the memory location which will hold the result of the random number generation.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_ENTROPY_EXHAUSTED: Request failed, entropy of random number generator is exhausted
Description	Generate a random number and stores it in the memory location pointed by the result pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Cs mJob refer to CsmPrimitive that has CsmRandomGenerate container.	
Available via	Csm.h	

To generate a random number, no streaming approach is necessary. The interface Csm_RandomGenerate can be called arbitrarily often to generate multiple random numbers.

6.3.8 Key Management Interface

The following interfaces are used for key management. Basically, a key contains of one or more key elements. A key element can be part of multiple keys. For example, this allows to derive a key element from a password with one keyId, and to use this derived key element for encryption with another keyId.

Note: If the actual key element to be modified is directly mapped to flash memory, there could be a bigger delay when calling the key management functions (synchronous operation).

6.3.8.1 Key Setting Interface

1) Csm_KeyElementSet

Service Name	Csm_KeyElementSet
Syntax	Std_ReturnType Csm_KeyElementSet (uint32 keyId, uint32 keyElementId, const uint8* keyPtr, uint32 keyLength)
Service ID [hex]	0x78

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	keyId	Holds the identifier of the key for which a new material shall be set.
	keyElementId	Holds the identifier of the key element to be written.
	keyPtr	Holds the pointer to the key element bytes to be processed.
	keyLength	Contains the number of key element bytes.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_WRITE_FAIL: Request failed because write access was denied CRYPTO_E_KEY_NOT_AVAILABLE: Request failed because the key is not available CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element size does not match size of provided data
Description	Sets the given key element bytes to the key identified by keyId.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

2) Csm_KeySetValid

Service Name	Csm_KeySetValid
Syntax	Std_ReturnType Csm_KeySetValid (uint32 keyId)
Service ID [hex]	0x67
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (in)	keyId	Holds the identifier of the key for which a new material shall be validated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy
Description	Sets the key state of the key identified by keyId to valid.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

6.3.8.2 Key Extraction Interface

1) Csm_KeyElementGet

Service Name	Csm_KeyElementGet	
Syntax	Std_ReturnType Csm_KeyElementGet (uint32 keyId, uint32 keyElementId, uint8* keyPtr, uint32* keyLengthPtr)	
Service ID [hex]	0x68	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	keyId	Holds the identifier of the key from which a key element shall be extracted.
	key ElementId	Holds the identifier of the key element to be extracted.
Parameters (inout)	keyLengthPtr	Holds a pointer to the memory location in which the output buffer length in bytes is stored. On calling this function, this parameter shall contain the buffer length in bytes of the keyPtr. When the request has finished, the actual size of the written input bytes shall be stored.

Parameters (out)	keyPtr	Holds the pointer to the memory location where the key shall be copied to.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed because read access was denied CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Retrieves the key element bytes from a specific key element of the key identified by the keyId and stores the key element in the memory location pointed by the key pointer.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

The underlying Crypto Driver has to decide if and how the key element bytes are extracted.

6.3.8.3 Key Copying Interface

1) Csm_KeyElementCopy

Service Name	Csm_KeyElementCopy	
Syntax	Std_ReturnType Csm_KeyElementCopy (const uint32 keyId, const uint32 keyElementId, const uint32 targetKeyId, const uint32 targetKeyElementId)	
Service ID [hex]	0x71	
Sync/Async	Synchronous	
Reentrancy	Reentrant, but not for the same keyId	
Parameters (in)	keyId	Holds the identifier of the key whose key element shall be the source element.
	keyElementId	Holds the identifier of the key element which shall be the source for the copy operation.

	targetKeyId	Holds the identifier of the key whose key element shall be the destination element.
	targetKeyElementId	Holds the identifier of the key element which shall be the destination for the copy operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	This function shall copy a key elements from one key to a target key.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

2) Csm_KeyCopy

Service Name	Csm_KeyCopy	
Syntax	Std_ReturnType Csm_KeyCopy (const uint32 keyId, const uint32 targetKeyId)	
Service ID [hex]	0x73	
Sync/Async	Synchronous	
Reentrancy	Reentrant, but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key whose key element shall be the source element.

	targetKeyId	Holds the identifier of the key whose key element shall be the destination element.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	This function shall copy all key elements from the source key to a target key.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

3) Csm_KeyElementCopyPartial

Service Name	Csm_KeyElementCopyPartial	
Syntax	Std_ReturnType Csm_KeyElementCopyPartial (uint32 keyId, uint32 keyElementId, uint32 keyElementSourceOffset, uint32 keyElementTargetOffset, uint32 keyElementCopyLength, uint32 targetKeyId, uint32 targetKeyElementId)	
Service ID [hex]	0x79	
Sync/Async	Synchronous	
Reentrancy	Reentrant, but not for the same keyId	
Parameters (in)	keyId	Holds the identifier of the key whose key element shall be the source element for copy operation.

	keyElementId	Holds the identifier of the key element which shall be the source for the copy operation.
	keyElementSourceOffset	This is the offset of the source key element indicating the start index of the copy operation.
	keyElementTargetOffset	This is the offset of the destination key element indicating the start index of the copy operation.
	keyElementCopyLength	Specifies the number of bytes that shall be copied.
	targetKeyId	Holds the identifier of the key whose key element shall be the destination element.
	targetKeyElementId	Holds the identifier of the key element which shall be the destination for the copy operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Copies a key element to another key element in the same crypto driver. The keyElementSourceOffset and keyElementCopyLength allows to copy just a part of the source key element into the destination. The offset into the target key is also specified with this function.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

Note: A Concatenation of partial keys into one key element is possible by calling Csm_KeyElementCopyPartial() multiple times and adjusting keyElementTargetOffset properly.

6.3.8.4 Key Generation interface

The key generation interface is used to generate a key into the key element CRYPTO_KE_KEYGENERATE_KEY according to the algorithm defined in the key element CRYPTO_KE_KEYGENERATE_ALGORITHM. The key will be generated from the random value that is located in the key element CRYPTO_KE_KEYGENERATE_SEED. The random value can be generated, for example, with the function Csm_RandomGenerate() and must be stored in CRYPTO_KE_KEYGENERATE_SEED before the key generation is triggered. It is important to check the quality of the randomness and its entropy of the seed, which depends on the used hardware, and software of a stack. The randomness has a major impact on the quality of the generated key material.

The key element with the id=CRYPTO_KE_KEYGENERATE_ALGORITHM contains a type from "Crypto_AlgorithmFamilyType", e.g. CRYPTO_ALGOFAM_AES, CRYPTO_ALGOFAM_RSA or CRYPTO_ALGOFAM_ED25519, that allows to generate an adequate key. As a counter example, the algorithm family type CRYPTO_ALGOFAM_SHA2_256 is not adequate because it provides no hint what key shall be generated.

For the key element CRYPTO_KE_KEYGENERATE_KEY the key element configuration item CryptoKeyElement/CryptoKeyElementFormat indicates the format of the generated key.

1) Csm_RandomSeed

Service Name	Csm_RandomSeed	
Syntax	Std_ReturnType Csm_RandomSeed (uint32 keyId, const uint8* seedPtr, uint32 seedLength)	
Service ID [hex]	0x69	
Sync/Async	Synchronous	
Reentrancy	Reentrant, but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key for which a new seed shall be generated.
	seedPtr	Holds a pointer to the memory location which contains the data to feed the seed.
	seedLength	Contains the length of the seed in bytes.
Parameters (inout)	None	
Parameters (out)	None	

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid"
Description	Feeds the key element CRYPTO_KE_RANDOM_SEED with a random seed.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

2) Csm_KeyGenerate

Service Name	Csm_KeyGenerate	
Syntax	Std_ReturnType Csm_KeyGenerate (uint32 keyId)	
Service ID [hex]	0x6a	
Sync/Async	Synchronous	
Reentrancy	Reentrant but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key for which a new material shall be keyId generated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Generates new key material and store it in the key identified by keyId.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

6.3.8.5 Key Derivation Interface

In cryptography, a key derivation function (or KDF) is a function, which derives one or more secret

keys from a secret value and/or other known information such as a passphrase or cryptographic key.
Specification of input keys that are protected by hardware means can be achieved by using the Csm_KeyDeriveKey interface.

1) Csm_KeyDerive

Service Name	Csm_KeyDerive	
Syntax	Std_ReturnType Csm_KeyDerive (uint32 keyId, uint32 targetKeyId)	
Service ID [hex]	0x6b	
Sync/Async	Synchronous	
Reentrancy	Reentrant, but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key which is used for key derivation.
	targetKeyId	Holds the identifier of the key which is used to store the derived key.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Derives a new key by using the key elements in the given key identified by the keyId. The given key contains the key elements for the password and salt. The derived key is stored in the key element with the id 1 of the key identified by targetKeyId.	
Configuration Dependency	At least one CsmKey container should be configured.	

Available via	Csm.h

6.3.8.6 Key Exchange Interface

Two users that each have a private secret can use a key exchange protocol to obtain a common secret, e.g. a key for a symmetric-key algorithm, without telling each other their private secret and without any listener being able to obtain the common secret or their private secrets.

The functions Csm_KeyExchangeCalcPubVal() / Csm_JobKeyExchangeCalcPubVal() and Csm_KeyExchangeCalcSecret() / Csm_JobKeyExchangeCalcSecret() are used to support Diffie-Hellman (DH) key exchange. This allows two partners, Alice and Bob, to generate private and public key material, to exchange public parts so that both parties can generate at the end a common shared secret. This shared secret can further be used, e.g. for symmetric data operation such as data encryption or MAC generation. The public and private key material can either be based on prime based large number as it is used with RSA or on elliptic curve (so-called elliptic-curve Diffie-Hellman).

The CSM key exchange functions require a key with key elements according to [SWS_Csm_01022], in the line of Crypto Service "Key Exchange". The key elements CRYPTO_KE_KEYEXCHANGE_BASE, CRYPTO_KE_KEYEXCHANGE_PRIVKEY and CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY are used to hold the public/private key material. These values can either be pre-defined and set by Csm_KeyElementSet() followed by Csm_KeySetValid() or generated. For example, these key values can be generated by Csm_KeyGenerate() and then copied with Csm_KeyElementCopy() to the corresponding key elements, followed by a call to Csm_KeySetValid().
In a first step, Alice will call Csm_KeyExchangeCalcPubVal() / Csm_JobKeyExchangeCalcPubVal() and send the results to Bob (exchanged data may need to be signed and/or encrypted depending on the protocol). It should be noted, that if KeyExchangeCalcPubVal is called but no valid key material exists (key is not valid or essential key elements have length=0), the function shall generate the necessary key material and continue as normal. If needed, Bob will put received key material from Alice into the corresponding key elements. He will also call Csm_KeyExchangeCalcPubVal() to generate his shared value that needs to be sent to Alice. Afterwards, Bob can call Csm_KeyExchangeCalcSecret() to generate the common secret. This value will be placed into the key element CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE. When Alice receives the public value from Bob, it will call KeyExchangeCalcSecret() and provides the value from Bob in the parameter of the function. The common shared secret will be generated by this function into the key element CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE. Depending on the algorithm, Bob needs to send key material to Alice to allow her to generate the common shared secret.

The key element CRYPTO_KE_KEYEXCHANGE_ALGORITHM specifies the Diffie-Hellman algorithm. The key element value is of type Crypto_AlgorithmFamily, for example CRYPTO_ALGOFAM_DH (for modulo based DH) or CRYPTO_ALGOFAM_ED25519 (for ECDH(E)). Additional elliptic curve parameter can be specified with the additional key element CRYPTO_KE_KEYEXCHANGE_CURVE.

The other key elements have the following meaning:

	DH(E)	ECDH(E)
CRYPTO_KE_KEYEXCHANGE_BASE	Modulo	Generator point
CRYPTO_KE_KEYEXCHANGE_PRIVKEY	Local exponent	Private key

CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY	Generator	Public key

1) Csm_KeyExchangeCalcPubVal

Service Name	Csm_KeyExchangeCalcPubVal	
Syntax	Std_ReturnType Csm_KeyExchangeCalcPubVal (uint32 keyId, uint8* publicValuePtr, uint32* publicValueLengthPtr)	
Service ID [hex]	0x6c	
Sync/Async	Synchronous	
Reentrancy	Reentrant but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
Parameters (inout)	publicValueLengthPtr	Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	publicValuePtr	Contains the pointer to the data where the public value shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Calculates the public value of the current user for the key exchange and stores the public key in the memory location pointed by the public value pointer.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

2) Csm_KeyExchangeCalcSecret

Service Name	Csm_KeyExchangeCalcSecret	
Syntax	Std_ReturnType Csm_KeyExchangeCalcSecret (uint32 keyId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength)	
Service ID [hex]	0x6d	
Sync/Async	Synchronous	
Reentrancy	Reentrant but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
	partnerPublicValuePtr	Holds the pointer to the memory location which contains the partner's public value.
	partnerPublicValueLength	Contains the length of the partner's public value in bytes.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Calculates the shared secret key for the key exchange with the key material of the key identified by the keyId and the partner public key. The shared secret key is stored as a key element in the same key.	
Configuration Dependency	At least one CsmKey container should be configured.	
Available via	Csm.h	

6.3.9 Cryptographic Primitives and Schemes

The keyId configured in the Job is only used to determine which driver objects needs to be used for the specific JobKeyPrimitive operation.

6.3.9.1 Csm_JobKeySetValid

Service Name	Csm_JobKeySetValid	
Syntax	Std_ReturnType Csm_JobKeySetValid (uint32 jobId, uint32 keyId)	
Service ID [hex]	0x7a	
Sync/Async	Sync or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	keyId	Holds the identifier of the key for which a new material shall be validated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy
Description	Stores the key if necessary and sets the key state of the key identified by keyId to valid.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmJobKeySetValid container.	
Available via	Csm.h	

6.3.9.2 Csm_JobRandomSeed

Service Name	Csm_JobRandomSeed
Syntax	Std_ReturnType Csm_JobRandomSeed (uint32 jobId, uint32 keyId, const uint8* seedPtr, uint32 seedLength)

Service ID [hex]	0x7b	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	keyId	Holds the identifier of the key for which a new seed shall be generated.
	seedPtr	Holds a pointer to the memory location which contains the data to feed the seed.
	seedLength	Contains the length of the seed in bytes.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid"
Description	This function shall dispatch the random seed function to the configured crypto driver object.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmJobRandomSeed container.	
Available via	Csm.h	

Note: The provided key Id(s) shall be transformed from CsmKeyId's to CryIfKeyId's.

6.3.9.3 Csm_JobKeyGenerate

Service Name	Csm_JobKeyGenerate
Syntax	Std_ReturnType Csm_JobKeyGenerate (uint32 jobId, uint32 keyId)
Service ID [hex]	0x7c
Sync/Async	Asynchronous or Synchronous, depending on the job configuration

Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	keyId	Holds the identifier of the key for which a new material shall be generated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy
		CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Generates new key material and stores it in the key identified by keyId.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the CsmJob refer to CsmPrimitive that has CsmJobKeyGenerate container.	
Available via	Csm.h	

Note: The provided key Id(s) shall be transformed from CsmKeyId's to CrylfKeyId's.

6.3.9.4 Csm_JobKeyDerive

Service Name	Csm_JobKeyDerive	
Syntax	Std_ReturnType Csm_JobKeyDerive (uint32 jobId, uint32 keyId, uint32 targetKeyId)	
Service ID [hex]	0x7d	
Sync/Async	Asynchronous or Synchronous, depending on the job configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	keyId	Holds the identifier of the key which is used for key derivation.

	targetKeyId	Holds the identifier of the key which is used to store the derived key.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible CRYPTO_E_KEY_EMPTY: Request failed because of
		uninitialized source key element
Description	Derives a new key by using the key elements in the given key identified by the keyId. The given key contains the key elements for the password and salt. The derived key is stored in the key element with the id 1 of the key identified by targetKeyId.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Csm Job refer to CsmPrimitive that has CsmJobKeyDerive container.	
Available via	Csm.h	

Note: The provided key Id(s) shall be transformed from CsmKeyId's to CrylfKeyId's.

6.3.9.5 Csm_JobKeyExchangeCalcPubVal

Service Name	Csm_JobKeyExchangeCalcPubVal
Syntax	Std_ReturnType Csm_JobKeyExchangeCalcPubVal (uint32 jobId, uint32 keyId, uint8* publicValuePtr, uint32* publicValueLengthPtr)
Service ID [hex]	0x7e
Sync/Async	Asynchronous or Synchronous, depending on the job configuration

Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
Parameters (inout)	publicValueLengthPtr	Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	publicValuePtr	Contains the pointer to the data where the public value shall be stored.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Calculates the public value of the current user for the key exchange and stores the public key in the memory location pointed by the public value pointer.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Cs mJob refer to CsmPrimitive that has CsmJobKeyExchangeCalcPubval container.	
Available via	Csm.h	

Note: The provided key Id(s) shall be transformed from CsmKeyId's to CrylfKeyId's.

6.3.9.6 Csm_JobKeyExchangeCalcSecret

Service Name	Csm_JobKeyExchangeCalcSecret
Syntax	Std_ReturnType Csm_JobKeyExchangeCalcSecret (uint32 jobId, uint32 keyId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength)
Service ID [hex]	0x7f
Sync/Async	Asynchronous or Synchronous, depending on the job configuration

Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
	partnerPublicValuePtr	Holds the pointer to the memory location which contains the partner's public value.
	partnerPublicValueLength	Contains the length of the partner's public value in bytes.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Calculates the shared secret key for the key exchange with the key material of the key identified by the keyId and the partner public key. The shared secret key is stored as a key element in the same key.	
Configuration Dependency	At least one CsmJob container should be configured and Primitive Ref of the Csm Job refer to CsmPrimitive that has CsmJobKeyExchangeCalcSecret container.	
Available via	Csm.h	

Note: The provided key Id(s) shall be transformed from CsmKeyId's to CryIfKeyId's.

6.3.10 Job Cancellation Interface

6.3.10.1 Csm_CancelJob

Service Name	Csm_CancelJob
Syntax	Std_ReturnType Csm_CancelJob (uint32 job, Crypto_OperationModeType mode)
Service ID [hex]	0x6f

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	job	Holds the identifier of the job to be canceled
	mode	Not used, just for interface compatibility provided.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful. Job removed from any queue and potentially from crypto driver hardware. E_NOT_OK: Request failed CRYPTO_E_JOB_CANCELED: Immediate cancelation not possible. The cancelation will be done at next suitable processing step and notified via a negative job's closing callback.
Description	Cancels the job processing from asynchronous or streaming jobs.	
Configuration Dependency	At least one CsmJob container should be configured and the CsmJob Processing Mode should set to CRYPTO_PROCESSING_ASYNC.	
Available via	Csm.h	

Note: In case the crypto driver does not support an instant cancelation of the job, the application need to wait for the job's closing callback to free the buffers. The crypto driver could potentially still write to the output buffer(s).

6.3.11 **Callback Notifications**

6.3.11.1 Csm_CallbackNotification

Service Name	Csm_CallbackNotification
Syntax	void Csm_CallbackNotification (Crypto_JobType* job, Crypto_ResultType result)
Service ID [hex]	0x70
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (in)	job	Holds a pointer to the job, which has finished.
	result	Contains the result of the cryptographic operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notifies the CSM that a job has finished. This function is used by the underlying layer (CRYIF). The function name itself is derived from "{CsmJob/CsmJobPrimitive CallbackRef}/CsmCallbackFunc".	
Configuration Dependency	At least one CsmJob container should be configured and the CsmJob Processing Mode should set to CRYPTO_PROCESSING_ASYNC.	
Available via	Csm.h	

6.3.12 Scheduled functions

6.3.12.1 Csm_MainFunction

Service Name	Csm_MainFunction
Syntax	void Csm_MainFunction (void)
Service ID [hex]	0x01
Description	API to be called cyclically to process the requested jobs. The Csm_MainFunction shall check the queues for jobs to pass to the underlying CRYIF.
Configuration Dependency	None
Available via	SchM_Csm.h

6.4 Expected Interfaces

Service Name	<Csm_ApplicationCallbackNotification>

Syntax	void <Csm_ApplicationCallbackNotification> (const Crypto_JobType* job, Crypto_ResultType result)	
Service ID [hex]	0x80	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	job	JobID of the operation that caused the callback
	result	Contains the result of the cryptographic operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	CSM notifies the application that a job has finished. The function name is configurable. The function name itself is derived from "{CsmJob/CsmJobPrimitiveCallbackRef}/CsmCallbackFunc"	
Configuration Dependency	None	
Available via	Csm.h	

7. Generator

7.1 Generator Option

Options	Description
-G,--Generation	Symbolic parameters to be used for fore generation (skip validation).
-H,--Help	Display this help message.
-I,--Input <I>	ECU description file path of the module for which generation tool need to run.
-L,--Log	Symbolic parameters to be used for generation error log.
-M,--Module <M>	Specify module name and version to be generated code for.
-O,--Output <O>	Project-relative path to location where the generated code is to be placed.
-T,--Top_path <T>	Symbolic parameters to be used for set path of module.
-V,--Validate	Symbolic parameters to be used for invoking validation checks.

7.2 Generator Error Message

7.2.1 Csm

7.2.1.1 Error Messages

1) ERR110001: Csm Job ID of "</AUTRON/Csm/CsmJobs/CsmJob.shortname>" is not consecutive and gapless.

CsmJobId configuration shall be consecutive and gapless, start from 0, then 1, 2, 3, etc.

2) ERR110002: Csm Key ID of "</AUTRON/Csm/CsmKeys/CsmKey.shortname>" is not consecutive and gapless.

CsmKeyId configuration shall be consecutive and gapless, start from 0, then 1, 2, 3, etc.

3) ERR110003: Pair of CsmInputKeyRef and CsmInputKeyElementId must be configured together.

It shall form a key ID – key element ID pair.

4) ERR110004: Pair of key "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmOutputKeyRef.shortname>" and key element id "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmOutputKeyElementId.shortname>" must be configured together.

It shall form a key ID – key element ID pair.

5) ERR110005: Pair of key "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmSecondaryInputKeyRef.shortname>" and key element id "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmSecondaryInputKeyElementId.shortname>" must be configured together.

It shall form a key ID – key element ID pair.

6) ERR110006: Pair of key "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmSecondaryOutputKeyRef.shortname>" and key element id "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmSecondaryOutputKeyElementId.shortname>" must be configured together.

It shall form a key ID – key element ID pair.

7) ERR110007: [ErrorId]: Pair of key

"</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmTertiaryInputKeyRef.shortname>" and key element id "</AUTRON/Csm/CsmInOutRedirections/CsmInOutRedirection/CsmTertiaryInputKeyElementId.shortname>" must be configured together

It shall form a key ID – key element ID pair.

8) ERR110008: [ErrorId]: CsmCallbackId of "</AUTRON/Csm/CsmCallbacks/CsmCallback.shortname" is not consecutive and gapless

CsmCallbackId configuration shall be consecutive and gapless, start from 0, then 1, 2, 3, etc.

9) ERR110009: No application callback reference (CsmJobPrimitiveCallbackRef) configured for asynchronous job "</AUTRON/Csm/CsmJobs/CsmJob.shortname>"

When a job is configured with CRYPTO_USE_FNC and CRYPTO_PROCESSING_ASYNC, user shall configure callback function for that job, this callback function will be called when job is finished.

10) ERR110010: <parameterName> is not configured for "</AUTRON/CsmPrimitives/*.shortName>" container

parameterName = CsmHashDataMaxLength | CsmMacGenerateDataMaxLength ...

e.g: CsmHashDataMaxLength is not configured for "CsmHashPrimitive" container

11) ERR110011: CsmCallbackId and (or) CsmCallbackFunc is not configured for "</AUTRON/Csm/CsmCallbacks/CsmCallback.shortname"

If CsmCallback is used, CsmCallbackId and CsmCallbackFunc shall be configured.

12) ERR110012: The CsmCallbackFunc name "</AUTRON/Csm/CsmCallbacks/CsmCallback/CsmCallbackFunc.value>" of "</AUTRON/Csm/CsmCallbacks/CsmCallback.shortname" has already been defined. Please choose a different name for CsmCallbackFunc

This error occurs when user configures duplicated callback function names, it shall lead to linking error.

13) ERR110013: CsmJobPrimitiveRef of "</AUTRON/Csm/CsmJobs/CsmJob.shortname>" should be linked to a sub-container of "<refs(/AUTRON/Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef).shortname>"

14) ERR110014: "</AUTRON/Csm/CsmPrimitives/sub-container.shortname>" container of "</AUTRON/Csm/CsmPrimitives.shortname>" and "</AUTRON/Csm/CsmPrimitives/sub-container.shortname>" container of "</AUTRON/Csm/CsmPrimitives.shortname>" have the same name "/AUTRON/Csm/CsmPrimitives/sub-container.shortname>"

For all the sub-containers of CsmPrimitives, they should have different shortname.

15) ERR110015: error_1 = [ErrorId]: CsmJob("</AUTRON/Csm/CsmJobs/CsmJob.shortname>"): CsmJobInterfaceUsePort("CRYPTO_USE_PORT") should be used with CsmProcessingMode("CRYPTO_PROCESSING_SYNC")

error_2 = [ErrorId]: CsmJob("</AUTRON/Csm/CsmJobs/CsmJob.shortname>"): CsmJobInterfaceUsePort("CRYPTO_USE_PORT_OPTIMIZED") should be used with CsmProcessingMode("CRYPTO_PROCESSING_ASYNC")

16) ERR110016: The value configured for parameter MODULE-ID in container BSW-MODULE-DESCRIPTION in provided MDT file is not correct. Module ID of Csm must be 110.

If value of ModuleId in file BSWMDT is not equal with the ModuleId of Csm.

17) ERR110017: The value configured for parameter VENDOR-ID in container BSW-IMPLEMENTATION in provided MDT file is not correct. Vendor ID of Csm must be 76.

If value of VendorId in file BSWMDT is not equal with the VendorId of Csm

18) ERR110018: The parameter <Parameter Name> in the container <Container Name> should be configured.

If any of the mandatory configuration parameters mentioned below is not configured in ECU Configuration Description File. Refer to table below:

Container Name	Parameter Name
BSW-IMPLEMENTATION	AR-RELEASE-VERSION
	VENDOR-ID
	SW-VERSION
BSW-MODULE-DESCRIPTION	MODULE-ID

19) ERR110019: The value configured parameter <Parameter Name> in the container <Container Name> is incorrect. It should be 1.0.0 for example.

If the parameters <Parameter Name> is not configured as per the pattern. Refer to table below:

Parameter Name	Container Name	Pattern	Example
SW-VERSION	BSW-IMPLEMENTATION	[0-9]+.[0-9]+.[0-9]+	1.0.0

20) ERR110020: AUTOSAR RELEASE VERSION <configured_version> is configured for the parameter <AR-RELEASE-VERSION> in provided MDT file is not correct. AUTOSAR RELEASE VERSION should be 4.4.0.

If the value of the element AR-RELEASE-VERSION present in file BSWMDT is configured other than 4.4.0. Refer to table below:

Var Name	Value
ar_release_major_version	4
ar_release_minor_version	4
ar_release_revision_version	0
sw_major_version	1
sw_minor_version	0
sw_patch_version	0
vendor_id	76
module_id	110

7.2.1.2 Warning Messages

8. Appendix

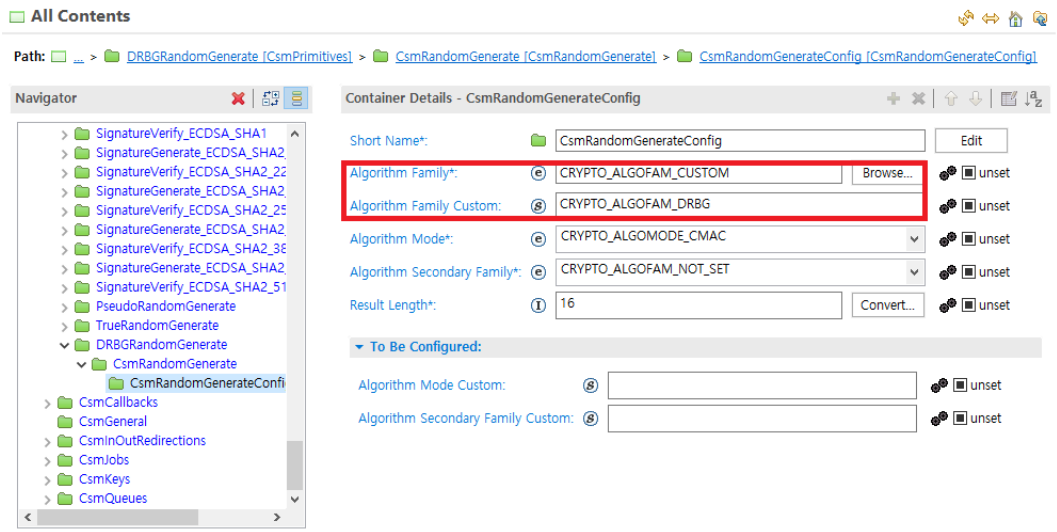
All change of Swcd show below:

- Remove operation to list of Client Server Interface and CancelJob:
  Remove mode parameter from port definition in chapter 8.5.5.2 and 8.5.5.3.
  Usage for Custom Algorithm/Mode in PrimitiveConfig:

Example of CUSTOM configuration in CsmPrimitives:

- If Algorithm Family, Algorithm Mode, Algorithm Secondary Family is CRYPTO_ALGOFAM_CUSTOM/CRYPTO_ALGOMODE_CUSTOM, and Algorithm(mode)customref is set with string, Generator will generate that string as configuration. User should define that string by include they header file in CsmGeneral which contain the define of macro so that crypto driver can understand.
- If Algorithm Family, Algorithm Mode, Algorithm Secondary Family is CRYPTO_ALGOFAM_CUSTOM/CRYPTO_ALGOMODE_CUSTOM, and Algorithm(mode)customref is blank (not set), Generator will generate CRYPTO_ALGOFAM_CUSTOM/CRYPTO_ALGOMODE_CUSTOM as configuration.
- For example: Using CRYPTO_ALGOFAM_CUSTOM
  Click Browse button and config CRYPTO_ALGOFAM_CUSTOM.

  Write Algorithm Family Custom name.



Add User Include Files string that same Algorithm Family Custom name.